Sanitizable Cross-domain Access Control with Policy-driven Dynamic Authorization

Jianfei Sun, Guowen Xu, Hongwei Li, *Fellow, IEEE*, Tianwei Zhang, Cong Wu, Xuehuan Yang, and Robert H. Deng, *Fellow, IEEE*

Abstract—The increasing demand for secure and efficient data sharing has underscored the importance of developing robust cryptographic schemes. However, many existing endeavors have overlooked the following critical issues: (1) unauthorized access resulting from malicious information leakage by senders; (2) absence of constraints on write and read permissions for participants; (3) and inflexibility of strategies to dynamically designate ciphertexts to multiple recipients. In this paper, we present SCPA, a cross-domain access control scheme imbued with sanitization features and propelled by policy-driven dynamic authorization, tailored for cloud-based data sharing. This scheme not only facilitates access controls, including regulations for no-read and no-write stipulations, governing the data permissible for senders to transmit and recipients to acquire but also enables the dynamic sharing of a data ciphertext subset with additional recipients beyond the originally sanctioned ones. We also provide comprehensive security proofs rigorously indicating the security of the invented SCPA. Moreover, to assess the efficacy of our SCPA, we undertake thorough theoretical and experimental analyses, showcasing its feasibility and superior performance.

Index Terms—Cross-domain, dynamic, unauthorized access, effectiveness.

1 Introduction

Cloud services have revolutionized the way organizations or individuals operate, providing flexibility, scalability, and cost-efficiency. However, this shift to the cloud has raised significant concerns regarding data security and privacy [1], [2], [3]. This is because entrusting sensitive data to third-party providers raises concerns about unauthorized access, data breaches, and compliance with privacy regulations [4], [5], [6]. As organizations or individuals navigate the cloud landscape, addressing security and privacy concerns is crucial to protect their data and maintain the trust of their stakeholders. To combat these concerns, some cryptographic techniques (such as identity/attribute-based encryption) have emerged as commonly utilized solutions for encrypting sensitive data. Specifically, data senders typically encrypt the data and entrust the encoded data to clouds, ensuring that only authorized recipients decrypt and access the data.

1.1 Security and Efficiency Concerns

Despite the fact that many cryptographic efforts enable tackling some security and privacy concerns, the state-of-the-art solutions still require to be enhanced in terms of security and privacy aspects stated as follows.

(1) **Failure to resist unauthorized access resulting from malicious information leakage by the sender.** To ensure data confidentiality, a prevalent approach involves encrypting the data prior to its upload to the cloud. The encryption is complemented by access controls that restrict data access solely to authorized users. While these access controls effectively specify the intended recipients permitted to access the data, they are insufficient to prevent a malicious sender from purposefully leaking confidential information through steganographic techniques to unauthorized recipients. For instance, the secret information could be concealed within the randomness of a ciphertext, retrievable only by recipients with malicious intent. This illicit practice ultimately leads to unauthorized access. *Most current data-sharing scenarios usually assume that the sender is completely honest, but rarely consider that the sender is not completely trustworthy, e.g.*, some unauthorized access probably occurs due to malicious information leakage from the sender.

One intuitive solution is to mandate the sender's utilization of digital signatures or blockchain technology to ensure the authenticity of the data. However, this method solely ensures the unforgeability of the ciphertext, and regrettably, it falls short in preventing the embedding of the secret within the randomness of said ciphertext. Another possible approach is that there is a need for a third party to censor the malicious behavior of a sender in avoiding illegal authorization. For instance, steganalysis tools (*e.g.*, software applications or algorithms) can be employed to detect hidden messages or data in avoiding illegal authorization. Although this method can help identify any steganographic techniques used to hide information, it is computationally intensive, requiring significant processing power and time to analyze data thoroughly. This can be a limitation when dealing with large volumes of data or real-time monitoring scenarios. Therefore, how to efficiently resist unauthorized access resulting from malicious information leakage by the sender remains a crucial challenge that needs to be handled.

Jianfei Sun, Guowen Xu and Hongwei Li are with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, P.R. China. (email:sjf215.uestc@gmail.com, {guowen.xu,hongweili}@uestc.edu.cn). Guowen Xu is the corresponding author.
Tianwei Zhang and Cong Wu, Xuehuan Yang are with the School of Computer Science and Engineering, Nanyang Technological University, Singapore; (email: {tianwei.zhang, congwu}@ntu.edu.sg and s190113@e.ntu.edu.sg).
Robert. H. Deng is with the School of Computing and Information Systems, Singapore Management University, Singapore; (email: robertdeng@smu.edu.sg).

(2) **Lack of cross-domain No-write and No-read rules to restrict write and read permissions for participants.** Considering practical data-sharing scenarios, users are typically assigned different security or clearance levels, such as public, secret, and top-secret. These security levels play a pivotal role in determining not only which messages a user is permitted to receive and read (*i.e.,* no read-up principle, indicating that the messages categorized as confidential or top-secret are inaccessible to users with public clearance), but also which messages the user is allowed to write-and-send. (*i.e.,* no write-down principle, stating that the messages designated as confidential or public cannot be written and transmitted by users possessing top-secret clearance). Failure to enforce such rules can lead to unauthorized access, data manipulation, and integrity compromise.

A straightforward solution is to employ access control encryption (ACE) [30], [31], [32]. This approach is broadly acknowledged as a mechanism for conferring unique access privileges upon diverse users, encompassing both the messages they are sanctioned to receive and those for which they hold the authorization to transmit. By implementing ACE, users can effectively control and manage the flow of information, ensuring that only designated recipients have access to specific messages while senders are restricted from sending appropriate messages according to their permissions. *However, existing standard ACE solutions have flexible problems with cross-domain authorization problems,* (See more details in Section 2.2) *i.e.,* existing conventional approaches solely leverage a single trusted third party to govern all participant keys originating from disparate domains. Apparently, this setting is unreasonable since it is difficult to conceive of a scenario in cross-domain systems where users from separate systems are jointly managed by a single key-granting authority. Besides, it is also hard to imagine that the access authorization in cross-domain environments may be dictated by the counterpart rather than being determined by the participants themselves, *e.g.*, in the collaboration between companies A and B, the access control should be determined by company A, specifying that their CEO can only write messages to the CEO of company B, rather than allowing company B to control who the CEO of company A can write to. Hence, how to realize cross-domain write and read permissions for participants from distinct domains is an urgent challenge.

(3) **Deficiency in inflexibility of strategies to dynamically designate ciphertexts to multiple recipients.** In practical data-sharing scenarios, outsourced data are frequently allowed to be shared dynamically with multiple recipients, *i.e.*, the data can be also accessible by other recipients beyond those already authorized. A trivial solution is to decrypt and re-encrypt the data for each new recipient. Nevertheless, this methodology displays a notable inefficiency, as it mandates the data sender to repeatedly encrypt identical data and necessitates the server to maintain numerous copies of the designated data. To address this inefficiency and facilitate a more streamlined approach to dynamic data sharing, proxy re-encryption technology enables the transformation of ciphertext from an original recipient to an additional designated recipient, thereby allowing the seamless transfer of data access privileges to recipients beyond initially authorized.

However, existing proxy re-encryption solutions have the following issues (See more details in Section 2.1): i) one limitation is that the sender can only share encrypted data with one recipient instead of multiple recipients at a time; ii) they commonly suffer from either impracticality or inefficiency as they are limited to an ”all-or-nothing” ciphertext conversion mechanism, meaning that it allows either the sharing of all data or none at all; iii) these solutions exhibit inflexibility since only one condition instead of multiple conditions in the re-encryption key can be specified, thus being incapable of supporting a subset of ciphertext sharing bound with the multi-conditional setting. *In other words, no solution to handle the flexibility issue of dynamically designating a subset of ciphertexts with the multi-conditional setting to multiple recipients has been proposed yet.* Consequently, an additional challenge lies in how to flexibly and dynamically designate a subset of ciphertexts to multiple recipients beyond those previously authorized.

1.2 Existing Solutions and Technical Challenges

Limitations of existing solutions: To the best of our knowledge, there is currently no purely-cryptographic methodology that can simultaneously well handle the above challenges. As mentioned earlier, digital signatures, *e.g.*, identity/attribute-based signatures (IBS/ABS), enable embedding a sender’s encryption key into the ciphertext, so as to realize data authenticity; access control encryption (ACE) permits a sender to utilize his/her encryption keys that are associated with specific access control policies to the data for realizing the enforcement of no-read and no-write rules via a sanitizer (*i.e.*, a trusted third party); Conditional proxy re-encryption (CPRE) empowers a data sender to enforce a condition in the re-encryption key, enabling the proxy to convert only ciphertexts satisfying the condition. *While these cryptographic techniques may be leveraged to address the aforementioned challenges, their applicability is primarily limited to specific contexts.*

Concisely, identity/attribute-based signature (IBS/ABS) has the capability to ensure data authenticity, but cannot prevent the malicious behavior of a corrupt sender, (e.g., deliberately embedding some secret) unless it is combined with non-interactive zero-knowledge (NIZK) proof (as it enables the sender (prover) proves the validity of the statement to another party (verifier)), otherwise, the digital signature cannot actually satisfy the requirement of (1). Besides, this methodology cannot cater to the requirements of (2) & (3); Indeed, both no-read and no-write rules can be realized by the enforcement of access control via ACE, however, existing ACE solutions either require a sanitization key distributed to the sanitizer for sanitization, either are incapable of support cross-domain property. In addition, the requirements of (1) & (3) cannot be satisfied with these methods since existing ACE solutions support single functions and have not been extended, making them infeasible for more complex applicable scenarios; CPRE enables the selective transformation of ciphertext from one recipient to another recipient based on predefined conditions, including identity-based CPRE (IB-CPRE) or fine-grained IB-CPRE (FIB-CPRE). However, existing CPRE solutions are either inflexible since only one condition instead of multiple

conditions in the re-encryption key can be specified, or impractical due to the failure to support many recipients. Moreover, most CPRE solutions do not fit the requirements of (1) & (2) since in these solutions not only the sender is assumed to be fully trusted, but also cross-domain No-write and No-read rules are not considered. In summary, to date, there is no purely-cryptographic methodology that can simultaneously well-tackle the above challenges of (1)-(3). Please note that the above techniques did not handle the cross-domain authorization issue.

Potential solutions & technical challenges: Intuitively, the integration of the aforementioned technologies with the NIZK technology can have the potential to address the security and efficiency concerns highlighted in Section 2.1 except cross-domain authorization issue. To resolve the above issues of (1)-(3), we first need to handle cross-domain authorization issues. Fortunately, realizing cross-domain authorization is not difficult by establishing two authorization authorities to manage the keys of the sender and receiver. Next, the methodologies mentioned in the following are fused with the NIZK technique to address the remaining challenges. Specifically, the most natural solution would be to introduce NIZK, IB-CPRE into ACE or apply NIZK, ACE to FIB-CPRE.

However, the seamless technological integration of these two types of possible solutions to build a sanitizable cross-domain access control scheme with policy-driven dynamic authorization is not trivial but complicated due to the following reasons: constructing such a scheme is not simply to unite these technologies together. (1) For the integration of NIZK, IB-CPRE to ACE, NIZK may not be compatible with ACE and IB-CPRE encryption schemes due to its complexities, compatibility issues, trust assumptions and performance trade-offs. Besides, even though the NIZK can be successfully incorporated into an ACE, overcoming the setting of a single condition in the re-encryption key is not easy since each condition may require a separate cryptographic key, which leads to additional overhead and potential security risks (*e.g.*, collusion attacks); (2) For incorporation of NIZK, ACE to FIB-CPRE, apart from facing the same NIZK-compatible challenges as that in (1), most existing ACE designs mainly rely on indistinguishability obfuscation or lattice-based assumption, which makes their integration challenging since most FIB-CPRE schemes are constructed on group-based assumption.

1.3 Our Contributions

In this paper, a sanitizable cross-domain access control scheme with policy-driven dynamic authorization (SCPA) was designed for cloud-based data sharing. The key innovations include: we observe that applying the linear secret sharing technique and PRE to ACE can be exploited to realize the multi-conditional dynamic access control encryption. By setting two different authorities to manage respectively participants in multi-conditional dynamic access control encryption, the cross-domain authorization issue can be addressed. Besides, with the NIZK technology, the tamper-proof data of a sender can be guaranteed for preventing the sender from tampering with the original data. The principal contributions manifest as follows:

- *Malicious unauthorized access resistance*: To prevent the corrupt sender from deliberately leaking some secrets leading to illegal access authorization, our SCPA introduces a sanitizer with the NIZK technology to censor this malicious behavior and ensure tamper-proof data sharing.
- *Dynamical and flexible data Sharing*. To facilitate flexibility and dynamicity during data sharing, apart from enabling senders to share their data with a designated group of recipients, our SCPA also allows any initial recipient to forward the same data to another set of recipients.
- *Cross-domain No-read and No-write rules*. To enforce cross-domain write and read permissions for participants from distinct domains, our SCPA enables one-to-many access control to the messages the recipients are allowed to receive and the messages the senders are allowed to send.
- *Policy-based multi-conditional setting*. To realize sharing a subset of ciphertext bound with a collection of conditions, our SCPA provides support for the sender to formulate an access policy to create an authorization token, which can be used for converting any initial ciphertext matching the access policy into a new ciphertext, such that the underlying data could be accessed by additional recipients beyond those granted originally.

In addition, the security proofs of SCPA are rigorously formalized to prove its security. We also conduct comprehensive theoretical and experimental analysis on our SCPA scheme and make comparisons with other relevant schemes. The results of our experiments showcase the practicality of our methodology, validating its effectiveness in data-sharing scenarios.

2 Related Work

2.1 Proxy Re-encryption Based Data Sharing

To facilitate secure data sharing, conventional encryption techniques are employed to safeguard data confidentiality. As more flexible data-sharing methodologies, broadcast encryption (BE) [7], [8] and attribute-based encryption (ABE) [9], [10] are introduced. In a BE, a data sender is allowed to encode a message to a collection of data recipients simultaneously, with access only granted to those whose identity is listed in the access list. In an ABE, a data sender can assign a fine-grained access strategy to the data, allowing only users whose attributes match the access strategy to access the encoded message. Although the two methods mentioned above are effective for securely sharing data with multiple recipients, they have a limitation when it comes to sharing encrypted data with new recipients beyond those initially specified. To mitigate this challenge, a technique called proxy re-encryption (PRE) was proposed. This technique allows a proxy to transform the original ciphertext into a new one that can be decrypted by a different set of recipients. To date, varieties of PRE schemes have been formulated including identity-based PRE (IB-PRE), conditional PRE (CPRE), and fine-grained PRE (FG-PRE).

Identity-based PRE: Green *et al.* [11] for the first time introduced the PRE into identity-based encryption (IBE) to

propose an IBPRE for achieving dynamical multi-recipients' data sharing while mitigating issues related to public key certificate management. To block the collusion attacks from the proxy and the recipients granted to access the re-encrypted data, a collusion-resistant IBPRE was proposed by Zhang *et al.* [12] to secure data sharing. To enhance the versatile functionality of IBPRE, Shao *et al.* [13] suggested a multi-time IBPRE scheme that enables the re-encryption of ciphertext multiple times, *i.e.*, it allows a legitimate recipient to further share the re-encrypted ciphertext with others. To facilitate secure data sharing with a group of recipients, Ge *et al.* [14] put forward an identity-based broadcast PRE (IB-BPRE) scheme, which realizes the ciphertext conversion from a ciphertext for one set of recipients into a new ciphertext for a different group. To enhance the security, Sun *et al.* [15] further lowered the security assumption and proposed a verifiable IB-BPRE, which realizes the secure ciphertext transformation even if the proxy is malicious. Very recently, a novel IBPRE was proposed by Deng *et al.* [16] to transform a single-recipient ciphertext into a multiple-recipient broadcast ciphertext. While IBPRE can render a feasible solution to dynamically sharing encrypted data, the proxy is authorized too much power in the re-encryption, *i.e.*, the proxy has the ability to convert all of a data owner's ciphertexts instead of just a subset that the data owner may intend to share (*i.e.*, all or nothing).

Conditional PRE: To remedy the drawback of the IBPRE, Weng *et al.* [17] formulated a conditional PRE (CPRE), in which a data owner enforces a condition in the re-encryption key such that only a subset of ciphertexts satisfying the condition can be converted by the proxy. To further optimize the efficiency of Weng *et al.*'s work [17], Vivek *et al.* [18] proposed an efficient CPRE by reducing the number of bilinear pairing operations. Subsequently, Chu *et al.* [19] raised a conditional broadcast PRE, which supports a multi-recipient ciphertext in the context of CPRE. By integrating IBE and CPRE, Shao *et al.* [20] formalized and put forward the concept of identity-based CPRE (IB-CPRE). Afterward, Yin *et al.* [21] suggested an identity-based broadcast CPRE (IB-BCPRE), which offers enhanced security and efficiency. With IB-BCPRE, ciphertexts can only be shared once, thus ensuring data inaccessibility for unauthorized parties. Then, Liang *et al.* [22] invented a multi-time IB-BCPRE that permits the proxy to further transformed the re-encrypted ciphertext. Xu *et al.* [23] devised an innovative IB-BCPRE scheme that enables the proxy to perform multiple re-encodings of a broadcast ciphertext, thereby enhancing its functionality. Although the above CPRE schemes enable data owners to pick a subset of ciphertexts to share, these works only permit data owners to *specify one condition* in the re-encryption key for ciphertext conversion, thus leading to the inflexibility of data sharing.

Fine-grained PRE: To improve the flexibility for secure data sharing, Fang *et al.* [24] put forward a fuzzy CPRE, in which a re-encryption key and a ciphertext are respectively bound with a set of conditions. Subsequently, Yang *et al.* [25] invented a ciphertext-policy CPRE, where the generated ciphertext is related to an access tree and the re-encryption operation can be conducted by the proxy if the access tree is satisfied. Ge *et al.* [26] proposed a fine-grained IB-CPRE (FIB-CPRE) scheme, where each ciphertext is associated with a set of conditions that must be satisfied for access to be granted, and each re-encryption key is labeled with an access tree indicating which types of ciphertexts the proxy can convert. Although Ge *et al.*'s work realizes flexible data sharing, it is limited to single-recipient data sharing. To overcome the flaw of single-recipient sharing, Ge *et al.* [27] also proposed a novel fine-grained identity-based broadcast re-encryption scheme. Although the security Ge *et al.* claimed was comprehensively analyzed, the security as described in [28] was informally verified since neither mathematical proofs nor threat model is formulated. Huang *et al.* [29] proposed a new fine-grained IB-BCPRE, in which a user labeled with a set of conditions enables re-encryption key generation for the same condition set, which limits the flexibility of data sharing since a user cannot self-decide a set of conditions in the re-encryption key generation. Very recently, Deng *et al.* [28] proposed a provably secure FIB-CPRE that simultaneously supports fine-grained, dynamical, multiple-recipient sharing of encrypted data.

To summarize, although the above PRE schemes realize dynamical ciphertext transformation, they suffer from either inefficiency or impracticality. Moreover, they all assume that the sender is fully-trusted but in reality the senders are incompletely honest.

2.2 Access Control Encryption Based Data Sharing

To better control the information flow of data sharing, Damgard *et al.* proposed an access control encryption (ACE) scheme [30], which provides a mechanism for implementing fine-grained access control over shared data by allowing different levels of access for different users. This includes not only controlling which messages a user is allowed to receive (*i.e.*, no read-up rule) but also which messages they are permitted to send (*i.e.*, no write-down rule). This level of control either enables ensuring that only authorized users can access sensitive information *even though the sender may maliciously embed some secret* or realizes that the information remains secure even if it is shared among multiple parties with varying levels of trust or access privileges. However, in [30], it requires the sanitizer to keep using a private sanitization key for sanitization. Kim *et al.* also proposed an ACE scheme [31], which supports expressive access control policies and was proven secure in the standard model. However, this ACE still suffers from the same defect as that in [30]. Afterward, Fuchsbauer *et al.* [32] invented a pairing-based ACE scheme for equality policy, in which the recipient is the sender since the designed ACE is symmetric-key encryption. However, the above works are all based on a trusted third party to govern all the keys of participants. Very recently, Wang *et al.* [33] for the first time proposed a novel standard ACE, which overcomes the drawback of requiring a private sanitization key for sanitization, however, this scheme only features limited functionality, resulting in the lack of feasibility for real applications. Overall, since existing ACE works either need a private sanitization key for sanitization or lack flexible data sharing functionality for further practical deployment, it is necessary to design a more flexible ACE scheme, such as supporting dynamical data sharing.

Summary of related works: For ease of our motivation presentation, we summarize the characteristic comparisons

TABLE 1: Property-wise Comparisons among Related Data Sharing Schemes

Type of scheme	No-read rule	No-write rule	Multiple recipient sharing	Multi-conditional re-encryption	Fine-grained delegation	Malicious authorization resistance	Cross-domain
IBPRE [12], [13]	✓	✗	✗	✗	✗	✗	✗
IB-BPRE [14], [15], [16]	✓	✗	✓	✗	✗	✗	✗
IB-CPRE [17], [18], [20]	✓	✗	✗	✗	✗	✗	✗
IB-BCPRE [19], [21], [22], [23]	✓	✗	✓	✗	✗	✗	✗
FIB-CPRE [24], [25], [26], [27]	✓	✗	✓	✗	✓	✗	✗
FIB-BCPRE [28], [29]	✓	✗	✓	✓	✓	✗	✗
ACE [30], [31], [32]	✓	✓	✓	✗	✗	✓	✗
Cross-domain ACE [33]	✓	✓	✓	✗	✗	✓	✓
DA-ACE	✓	✓	✓	✓	✓	✓	✓

Note: "✓" indicates that the scheme supports this property; "✗" signifies that the scheme does not feature property.

of existing related works in TABLE. 1. Specifically, *No-read rule* ensures the message that the recipients are allowed can receive and read. *No-write rule* guarantees that the message that the senders are allowed to send and write to. *Multiple recipient sharing* indicates that the message can be shared with additional recipients beyond those designated previously. *Multi-conditional re-encryption* supports multi-condition ciphertext transformation. *Fine-grained delegation* means flexible re-encryption operations. *Malicious authorization resistance* states that some malicious behavior of a sender leading to invalid authorization can be hindered. *Cross-domain* implies that the senders and recipients are managed by distinct domains instead of one authority.

From TABLE. 1, it is straightforward to see that only the works [28], [29] and our framework can realize multi-conditional and fine-grained delegation; only the works [30], [31], [32], [33] and our work can support no-read and no-write rules as well as be also immune to malicious authorization due to the malicious behaviors of the senders; all works except [12], [13], [17], [18], [20] can realize dynamical multi-recipient data sharing. To summarize, only SCPA simultaneously provides the following properties: no-read & no-write rules, multiple recipient data sharing, fine-grained multi-conditional re-encryption, cross-domain, and malicious authorization resistance.

3 Preliminaries and Building Block

In this section, we present the basic concepts and foundational components, which encompass notations, hardness assumptions, structure-preserving digital signatures (SPS), non-interactive zero-knowledge proofs (NIZK), linear secret sharing scheme (LSSS), SCPA definitions along with the associated security games.

3.1 Notation

The following notations used in our paper are concluded in TABLE 2.

TABLE 2: Notations

Notation	Description
$\mathcal{H}, \mathcal{H}_0$	Hash functions
$\mathcal{S}$	Identity set
$\mathcal{L}$	The set of conditions
$\nvDash$ / $\vDash$	Mismatch/Match
ℓ_{max}	The maximum number of system users
$\mathbb{A}$	The access policy regarding conditions
π	The proof of zero-knowledge protocol
$\mathsf{id} \rightarrow \mathcal{S}\|\mathbb{A}$	Conversion from identity id to set $\mathcal{S}$ under policy $\mathbb{A}$

3.2 Hardness Assumption (GDDHE)

GDDHE Assumption: Let $(g, g^{\alpha}, \ldots, g^{\alpha^{t-1}}, g^{s\alpha f(\alpha)}, g^{\alpha f(\alpha)}\mu, \mu^{\alpha}, \ldots, \mu^{\alpha^{2n}}, \mu^{1/g(\alpha)}, \mu^{f(\alpha)/g(\alpha)}, \mu^{sg(\alpha)}, \mathcal{Z})$ be a tuple representing the **G**eneral **D**ecisional Bilinear **D**iffie-**H**ellman **E**xponent (GDDHE) assumption. Here, g and μ serve as generators for multiplicative cyclic groups $\mathbb{G}_0$ and $\mathbb{G}_1$ respectively and the functions f and g have two coprime polynomials with pairwise distinct roots, of respective orders t and n, while α, s, γ are random elements of $\mathbb{Z}_p$. For any adversary $\mathcal{A}$, the task of distinguishing $\mathcal{Z} = e(g,\mu)^{sf(\alpha)}$ and $\mathcal{Z} = \mathcal{Z}_1$ is difficult, where $\mathcal{Z}_1$ is a random element of the group $\mathbb{G}_2$. Note that we additionally need group elements $\mu^{f(\alpha)/g(\alpha)}, \mu^{sg(\alpha)}$ for our reduction. We do not introduce a new name as there is a lack of a naming convention for these assumptions.

In the following, we state the functions f and g are unitary polynomials but are not mandatory issues. The specific notations are as follows: $f(x) = \prod_{i=1}^{t}(x + x_i)$, $g(x) = \prod_{i=t+1}^{n}(x + x_i)$, $f_i(x) = f(x)/(x + x_i)$ for $i \in [1, t]$, $g_i(x) = g(x)/(x + x_i)$ for $i \in [t + 1, t + n]$.

3.3 Digital Signature

A standard digital signature scheme is formed of three algorithms: ($\mathcal{DS}$.**KeyGen**, $\mathcal{DS}$.**Sign**, $\mathcal{DS}$.**Vfy**). Given $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathcal{DS}.\mathbf{KeyGen}(\lambda)$, a user can sign on a message via $\sigma \leftarrow \mathcal{DS}.\mathbf{Sign}(m, \mathsf{sk})$ so as to satisfy $\mathcal{DS}.\mathbf{Vfy}(m, \mathsf{vk}, \sigma) = 1$. $\mathcal{DS}$ is forgeable under chosen plaintext attacks if no adversary can return a valid message/signature pair (m^*, σ^*), in which m^* never issues as a signing-oracle query, which outputs $\mathcal{DS}.\mathbf{Sign}(m, \mathsf{sk})$. In this manuscript, we mainly use a special digital signature scheme, which is referred to as a structure-preserving signature (SPS) scheme [34]. In more detail, the following algorithms of SPS are described.

- $\mathcal{DS}.\mathbf{KeyGen}(\lambda)$: With the input security parameter 1^{λ}, it picks a bilinear group $\mathcal{BG} = (p, \mathbb{G}_0, \mathbb{G}_1, \mathbb{G}_2, e, g, \mathsf{D})$, where g and D are respective generators of groups $\mathbb{G}_0$ and $\mathbb{G}_1$ with its prime order

p, and outputs a global public parameter $\mathsf{gpk} = (\mu, \mathcal{BG})$, where $\mu \in \mathbb{G}_1$. It also chooses $\mathsf{Y} \in \mathbb{G}_0$, $\tau \in \mathbb{Z}_p$, computes $\mathsf{V} = \mu^{\tau}$, generates a signing key τ and a verification key $\mathsf{vk} = (\mathsf{Y}, \mathsf{V}, \mathsf{gpk})$. It finally outputs its secret key $\mathsf{sk} = (\tau, \mathsf{gpk})$.

- $\mathcal{DS}.\mathbf{Sign}(\mathsf{sk}, \mathsf{A})$: To sign on $\mathsf{A} \in \mathbb{G}_1$, it selects $\beta \in \mathbb{Z}_p$ and calculates $\mathsf{R} = \mathsf{D}^{\beta}, \mathsf{S} = \mathsf{A}^{\frac{\tau}{\beta}}\mathsf{Y}^{\frac{1}{\beta}}, \mathsf{T} = \mathsf{S}^{\frac{\tau}{\beta}}g^{\frac{1}{\beta}}, \mathsf{RT} = g^{1/\tau}$, which forms a signature $\sigma = (\mathsf{R}, \mathsf{S}, \mathsf{T})$ on A and a signature randomization token $\mathsf{RT} = g^{\frac{1}{\beta}}$. It finally produces a signature σ and RT.
- $\mathcal{DS}.\mathbf{Rand}(\sigma, \mathsf{RT})$: To randomize σ, it randomly chooses τ', it outputs $\sigma' = (\mathsf{R} = \mathsf{R}^{1/\tau'}, \mathsf{S} = \mathsf{S}^{\tau'}, \mathsf{T}' = \mathsf{T}^{\tau'^2}\mathsf{RT}^{\tau'(1-\tau')})$
- $\mathcal{DS}.\mathbf{Vfy}(\mathsf{vk}, \mathsf{A}, \sigma)$: For the signed message A and its signature $\sigma = (\mathsf{R}, \mathsf{S}, \mathsf{T})$, it returns 1 if $e(\mathsf{S}, \mathsf{R})e(\mathsf{A}, \mathsf{V}^{-1}) = e(\mathsf{Y}, \mathsf{D})$, $e(\mathsf{T}, \mathsf{R})e(\mathsf{S}, \mathsf{V}^{-1}) = e(\mathsf{g}, \mathsf{D})$.

3.4 Non-interactive Zero-knowledge Proof (NIZK)

In the context of a language L and a relation R, the statement $x \in L$ holds if and only if there exists a witness w satisfying $(x, w) \in R$. A NIZK system for R is comprised of the following algorithms: $\mathcal{Z}.\mathbf{Gen}$, $\mathcal{Z}.\mathbf{Prove}$, $\mathcal{Z}.\mathbf{Vfy}$. A NIZK system [38] is correct, perfectly sound, knowledge extractable, and zero-knowledge if the following properties hold.

- *Correctness*: For all adversaries $\mathcal{A}$,

$$\mathsf{Pr}\left\{\begin{array}{rcl} \mathcal{Z}.\mathbf{Gen}(\lambda, L) \to \mathsf{crs}; & \textit{if} & (x,w) \in R, \\ \mathcal{A}(\mathsf{crs}) \to (x, w); & : & \textit{then} \quad \mathcal{Z}.\mathbf{Vfy}(\\ \mathcal{Z}.\mathbf{Prove}(\mathsf{crs}, x, w) \to \pi; & & \mathsf{crs}, x, \pi) = 1 \end{array}\right\} \simeq 1.$$

- *Soundness*: For all adversaries $\mathcal{A}$,

$$\mathsf{Pr}\left\{\begin{array}{rcl} \mathcal{Z}.\mathbf{Gen}(\lambda, L) \to \mathsf{crs}; & \textit{if} & x \notin L, \textit{then} \\ \mathcal{A}(\mathsf{crs}) \to (x, \pi); & & \mathcal{Z}.\mathbf{Vfy}(\mathsf{crs}, x, \pi) = 1 \end{array}\right\} \simeq 1.$$

- *Knowledge Extraction*: There is an existing knowledge extractor, which is an algorithm pair $(\mathcal{E}_1, \mathcal{E}_2)$ holding the following properties:
  (1) For all adversaries $\mathcal{A}$,

$$\mathsf{Pr}\left\{ \mathcal{Z}.\mathbf{Gen}(\lambda, L) \to \mathsf{crs} : \mathcal{A}(\mathsf{crs}) \right\} \simeq \mathsf{Pr}\left\{ \mathcal{E}_1(\lambda, L) \to \mathsf{crs} : \mathcal{A}(\mathsf{crs}) \right\}.$$

  (2) For all adversaries $\mathcal{A}$,

$$\mathsf{Pr}\left\{\begin{array}{rcl} \mathsf{crs} \to \mathcal{E}_1(\lambda, L); & \textit{if} & \mathcal{Z}.\mathbf{Vfy}(\mathsf{crs}, \\ (x, w) \to \mathcal{A}(\mathsf{crs}); & : & x, \pi) = 0 \\ w \to \mathcal{E}_1(\mathsf{crs}, \tau, x, w); & & \mathsf{or} \quad (x, w) \in R \end{array}\right\} \simeq 1.$$

- *Zero-knowledge*: There is an existing knowledge extractor, which is an algorithm pair $(\mathcal{S}_1, \mathcal{S}_2)$ holding the following property:

$$\mathsf{Pr}\left\{ \mathsf{crs} \to \mathcal{Z}.\mathbf{Gen}(\lambda, L) : \mathcal{A}^{\mathcal{Z}.\mathbf{Prove}(\mathsf{crs}, \ldots)}(\mathsf{crs}) \right\}$$
$$\simeq \mathsf{Pr}\left\{ \mathsf{crs} \to \mathcal{S}_1(\lambda, L) : \mathcal{A}^{\mathcal{S}'(\mathsf{crs}, \tau, \ldots)}(\mathsf{crs}) \right\},$$

  where $\mathcal{S}'(\mathsf{crs}, \tau, x, w) = \mathcal{S}_2(\mathsf{crs}, x, w)$ if $(x, w) \in R$; otherwise, it returns an abortion symbol $\perp$.

3.5 Linear Secret Sharing Scheme (LSSS)

An LSSS scheme Π over a set of parties P is termed linear if it features the subsequent properties: 1) The shares allocated to each individual party construct a vector over $\mathbb{Z}_p$. 2) There exists a matrix A, referred to as the sharing-generating matrix for Π, consisting of ℓ rows and m columns. For every $k = 1, \ldots, \ell$, the k-th row of matrix A is associated with a party $\rho(k)$, where ρ is a mapping function from the domain $(1, \ldots, \ell)$ to the set P. Whenever the column vector $\vec{v} = (\alpha, y_2, \ldots, y_m)$ is given, with α being the secret to be shared and $y_2, \ldots, y_m$ being random elements from $\mathbb{Z}_p$, then $\lambda_i = A_i\vec{v}$ represents the share intended for the party $\rho(i)$. Based on the knowledge of [28], each LSSS is equipped with the capability of reconstruction. To elaborate, in the context of an LSSS (A, ρ) for an access structure $\mathbb{A}$ and an authorized set S_a, there inevitably exist constants $\beta_k \in \mathbb{Z}_p$ that enables the recovery of the secret α. *In our approach, we refer to a group of condition sets as an access structure. When a specific set of conditions aligns with an access structure, we designate it as part of the authorized list.*

3.6 Definition of our SCPA

- **Global Setup** (1^{λ}): With the input security parameter 1^{λ}, it outputs a global public parameter gpk.
- **Receiver-Authority Setup** (gpk): With the input global public parameter gpk, it (**RASetup**) outputs its public parameter $\mathsf{pk_{ra}}$ and master secret key $\mathsf{msk_{ra}}$.
- **Sender-Authority Setup** (gpk, $\mathsf{pk_{ra}}$): With the input gpk and $\mathsf{pk_{ra}}$, it (**SASetup**) outputs its public parameter $\mathsf{pk_{sa}}$ and master secret key $\mathsf{msk_{sa}}$.
- **Encryption-Key Registration**($\mathcal{S}$, $\mathsf{msk_{sa}}$, $\mathsf{pk_{sa}}$): With the input an identity set $\mathcal{S}$ that is permitted to write to, it (**EKGen**) produces an encryption key $\mathsf{ek_{id}}_j$.
- **Decryption-Key Registration** ($\mathsf{msk_{ra}}$, id_{j^*}): With the input $\mathsf{msk_{ra}}$ and an identity id_{j^*}, it (**DKGen**) generates a decryption key $\mathsf{sk_{id}}_{j^*}$ for the identity id_{j^*}.
- **Encryption** (gpk, $\mathsf{pk_{sa}}$, m, $\mathsf{ek_{id}}_j$, $\mathcal{L}$): With the input gpk, $\mathsf{pk_{sa}}$, an encryption key $\mathsf{ek_{id}}_j$, a cleartext m with its decryption labeling a set $\mathcal{L}$ of conditions, it (**Enc**) outputs an unsanitized ciphertext ct.
- **Sanitization** (ct): With the input unsanitized ciphertext ct, it (**San**) returns a sanitized ciphertext ct^*.
- **Delegation** ($\mathsf{pk_{ra}}$, $\mathcal{S}'$, $\mathsf{sk_{id}}$, $\mathbb{A}$): With the input $\mathsf{pk_{ra}}$, a new identity set $\mathcal{S}'$, a secret key $\mathsf{sk_{id}}$ ($\mathsf{id}_k \in \mathcal{S}$) and an access policy $\mathbb{A}$ including n conditions, it (**Del**) generates the delegation key $\mathsf{dk_{id}}_{\to \mathcal{S}'|\mathbb{A}}$.
- **Re-Encryption** (ct^*, $\mathcal{S}$, $\mathcal{L}$, $\mathsf{dk_{id}}_{\to \mathcal{S}'|\mathbb{A}}$): With the input sanitized ciphertext ct^*, the identity list $\mathcal{S}$, the conditions $\mathcal{L}$, the delegation key $\mathsf{dk_{id}}_{\to \mathcal{S}'|\mathbb{A}}$, it (**Re-Enc**) outputs a new re-encrypted ciphertext ct'.
- **Decryption** ($\mathsf{pk_{ra}}$, $\mathsf{sk_{id}}$, $\mathcal{S}/\mathcal{S}'$, $\mathsf{ct}^*/\mathsf{ct}'$): With the input $\mathsf{pk_{ra}}$, the secret key $\mathsf{sk_{id}}$, the ciphertext $\mathsf{ct}^*/\mathsf{ct}'$ and the identity list $\mathcal{S}/\mathcal{S}'$, it (**Dec**) can recover the plaintext m with corresponding secret key $\mathsf{sk_{id}}$.

TABLE 3: Security games for No-read and No-write rules

No-read **Game Definition**	No-write **Game Definition**	**Oracle Definition**
(1) $\mathsf{pk} \rightarrow \mathbf{Setup}(\lambda)$; $(\mathsf{msk}_{\mathsf{ra}}, \mathsf{mpk}_{\mathsf{ra}}) \leftarrow \mathbf{RASetup}()$; $(\mathsf{msk}_{\mathsf{sa}}, \mathsf{mpk}_{\mathsf{sa}}) \leftarrow \mathbf{RASetup}(\mathsf{mpk}_{\mathsf{ra}})$; Initialize an empty set $\mathcal{I}_R$. (2) $(m_0, m_1, \mathsf{id}, \mathsf{id}_1) \leftarrow$ $\mathcal{A}^{\mathcal{O}_W(\text{“}II\text{”},.),\mathcal{O}_R(.),\mathcal{O}_E(.,.)}(\mathsf{mpk}_{\mathsf{sa}}, \mathsf{mpk}_{\mathsf{ra}})$. (3) $b \leftarrow \{0,1\}$. (4) $\mathsf{ek}_{\mathsf{id}_b} \leftarrow \mathbf{EKGen}(\mathsf{mpk}_{\mathsf{sa}}, P, \mathsf{id}_b)$; $\mathsf{ct}^* \leftarrow \mathbf{Enc}(\mathsf{ek}_{\mathsf{id}_b}, m_b)$ or $\mathsf{ct}^{*\prime} \leftarrow \mathbf{Re\text{-}Enc}(\mathsf{ct}^*)$.. (5) $b' \leftarrow \mathcal{A}^{\mathcal{O}_W(.,.),\mathcal{O}_R(.),\mathcal{O}_E(.,.)/\mathcal{O}_{RE}(.,.)}(\mathsf{ct}^*/\mathsf{ct}^{*\prime})$.	(1) $\mathsf{pk} \rightarrow \mathbf{Setup}(\lambda)$ $(\mathsf{msk}_{\mathsf{ra}}, \mathsf{mpk}_{\mathsf{ra}}) \leftarrow \mathbf{RASetup}()$; $(\mathsf{msk}_{\mathsf{sa}}, \mathsf{mpk}_{\mathsf{sa}}) \leftarrow \mathbf{SASetup}(\mathsf{mpk}_{\mathsf{ra}})$; Initialize two empty sets $\mathcal{I}_R$ and $\mathcal{I}_W$. (2) $(\mathsf{ct}_0, \mathsf{id}^*) \leftarrow$ $\mathcal{A}^{\mathcal{O}_W(\text{“}I\text{”},.),\mathcal{O}_R(.),\mathcal{O}_{SE}(.,.)(\mathsf{mpk}_{\mathsf{sa}},\mathsf{mpk}_{\mathsf{ra}})}$. (3) $m^* \leftarrow \mathbb{G}_2$, $b \leftarrow \{0,1\}$. (4) $\mathsf{ek}_{\mathsf{id}^*} \leftarrow \mathbf{EKGen}(\mathsf{mpk}_{\mathsf{sa}}, P, \mathsf{id}^*)$; $\mathsf{ct}_1 \leftarrow \mathbf{Enc}(\mathsf{ek}_{\mathsf{id}^*}, m_b)$. (5) $b' \leftarrow \mathcal{A}^{\mathcal{O}_W(.,.),\mathcal{O}_R(.),\mathcal{O}_{SE}(.,.)}(\mathbf{San}(\mathsf{ct}_b))$.	$\mathcal{O}_W(\mathsf{phase}, \mathsf{id}_i)$: 1) If (phase=“I”), add id_i into $\mathcal{I}_W$. 2) Return $t \leftarrow \mathbf{EKGen}(\mathsf{mpk}_{\mathsf{sa}}, P, \mathsf{id}_i)$. $\mathcal{O}_R(\mathsf{id}_j)$: 1) Add id_j into $\mathcal{I}_R$. 2) Return $t \leftarrow \mathbf{DKGen}(\mathsf{mpk}_{\mathsf{sa}}, P, \mathsf{id}_i)$ $\mathcal{O}_E(\mathsf{id}_i, m)$: 1) $\mathsf{ek}_{\mathsf{id}_i} \leftarrow \mathbf{EKGen}(\mathsf{mpk}_{\mathsf{sa}}, P, \mathsf{id}_i)$. 2) Return $\mathsf{ct} \leftarrow \mathbf{Enc}(\mathsf{ek}_{\mathsf{id}_i}, m)$ $\mathcal{O}_{RE}(\mathsf{id}_i, m)$: 1) $\mathsf{ek}_{\mathsf{id}_i} \leftarrow \mathbf{EKGen}(\mathsf{mpk}_{\mathsf{sa}}, P, \mathsf{id}_i)$. 2) Return $\mathsf{ct}' \leftarrow \mathbf{Re\text{-}Enc}(\mathbf{Enc}(\mathsf{ek}_{\mathsf{id}_i}, m))$. $\mathcal{O}_{SE}(\mathsf{id}_i, m)$: 1) $\mathsf{ek}_{\mathsf{id}_i} \leftarrow \mathbf{EKGen}(\mathsf{mpk}_{\mathsf{sa}}, P, \mathsf{id}_i)$. 2) Return $\mathsf{ct} \leftarrow \mathbf{Enc}(\mathbf{San}(\mathbf{Enc}(\mathsf{ek}_{\mathsf{id}_i}, m)))$.

3.7 Security Game Definitions

Definition 1 (CPA Security). Our SCPA can be securely immune to chosen plaintext attacks (CPA) if the advantage of $\mathcal{A}$ interacting with $\mathcal{C}$ in winning the following game is negligible.

- **Init**: An identity list $\mathcal{S}^* = \{\mathsf{id}_1^*, \ldots, \mathsf{id}_\ell^*\}$ that is allowed to write to and a set $\mathcal{L}^* = (\mathcal{L}_1^*, \ldots, \mathcal{L}_n^*)$ of conditions are picked and then sent to $\mathcal{C}$.
- **Setup**: $\mathcal{C}$ makes queries to the various setup algorithms to get the globe public key gpk, the sender/receiver-authority public key $\mathsf{pk}_{\mathsf{sa/ra}}$ to $\mathcal{A}$ and the master sender/receiver-authority secret key $\mathsf{msk}_{\mathsf{sa/ra}}$. $\mathcal{C}$ then gives gpk, $\mathsf{pk}_{\mathsf{sa/ra}}$ to $\mathcal{A}$.
- **Phases 1 & 2**: $\mathcal{A}$ makes the following queries to $\mathcal{C}$:
  −**Encryption-Key Registration** (id): If $\mathsf{id}_i \in \mathcal{S}^*$, $\mathcal{C}$ aborts. Otherwise, $\mathcal{C}$ performs the **Encryption-Key Registration** algorithm to invent the encryption key $\mathsf{sk}_{\mathsf{id}}$.
  −**Decryption-Key Registration** (id): If $\mathsf{id} \in \mathcal{S}^*$, $\mathcal{C}$ aborts. Otherwise, it runs **Decryption-Key Registration** algorithm to produce the decryption key $\mathsf{sk}_{\mathsf{id}}$ and returns it to $\mathcal{A}$.
  −**Delegation** ($\mathsf{id} \rightarrow \mathcal{S}'|\mathbb{A}$): The delegation key queries for an identity id, an identity set $\mathcal{S}^*$, and an access policy $\mathbb{A}$ are made by $\mathcal{A}$, a delegation key $\mathsf{dk}_{\mathsf{id}\rightarrow\mathcal{S}'|\mathbb{A}}$ is returned. If $\mathsf{id} \in \mathcal{S}^*, \mathcal{L}^* \in \mathbb{A}$ and there exists a tuple $(\mathsf{id}' \in \mathcal{S}', \mathsf{sk}_{\mathsf{id}'})$ in $\mathcal{T}_{\mathsf{sk}}$, $\mathcal{C}$ aborts; otherwise, it runs **Delegation** algorithm to produce $\mathsf{dk}_{\mathsf{id}\rightarrow\mathcal{S}'|\mathbb{A}}$ and returns it to $\mathcal{A}$.
- **Challenge**: $\mathcal{A}$ submits two picked equal-length messages m_0 and m_1 to challenger $\mathcal{C}$, $\mathcal{C}$ flips a coin $\zeta \in \{0,1\}$, encrypts m_b under $\mathcal{L}^*, \mathcal{S}^*$ via **Encryption** & **Sanitization** algorithms and gives the sanitized ciphertext to $\mathcal{A}$.
- **Guess**: $\mathcal{A}$ submits its guess $\zeta' \in \{0,1\}$ and $\mathcal{C}$ returns the same bit.

In this CPA security game, the confidentiality of delegation keys is not required to securely store, in other words, the delegation keys can be known to all adversarial attackers because the delegation key queries can be made by the adversary according to their own choice. Data privacy can be ensured as long as the legitimate secret key is securely stored even though the delegation key is leaked.

Definition 2 (No-read Rule). As depicted in TABLE. 3, the No-read rule between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ is satisfied if the advantage of $\mathcal{A}$ to win the No-read rule game is negligible. In this game, if $b = b'$, $\mathsf{id}_0, \mathsf{id}_1 \in \{0,1\}^*$ and for all queries id_j to $\mathcal{O}_R$ holding $\mathsf{P}(\mathsf{id}_0, \mathsf{id}_j) = \mathsf{P}(\mathsf{id}_1, \mathsf{id}_j) = 0$ ($\forall \mathsf{id}_j \in \mathcal{I}_R$), we say $\mathcal{A}$ wins this game.

The secret key queries for any identity cannot be made by $\mathcal{A}$, where the identity is entitled to read from id_0 or id_1. In this game, there is no restriction on the encryption key queries and the ciphertexts returned from the encryption oracle are unsanitized. As well, the challenge ciphertexts are also unsanitized. Since our SCPA does not require the sanitization key, $\mathcal{A}$ is allowed to compromise the sanitizer.

Definition 3 (No-write Rule). As depicted in Fig. 3, the No-write rule between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ is satisfied if the advantage of $\mathcal{A}$ to win the No-write rule game is negligible. In this game, if $b = b'$, $\mathsf{id}^* \in \mathcal{I}_W$, $\forall \mathsf{id}_i \in \mathcal{I}_W$, $\forall \mathsf{id}_j \in \mathcal{I}_R$, $\mathsf{P}(\mathsf{id}_i, \mathsf{id}_j) = 0$ and $\mathbf{San}(\mathsf{ct}) \neq \perp$, we say $\mathcal{A}$ wins this game.

In the initial game phase, $\mathcal{A}$ is granted the ability to request encryption keys before the challenge is issued. It's pivotal to emphasize that the ciphertext ct_0 formed by $\mathcal{A}$ originates solely from those distinct keys. Moreover, $\mathcal{A}$ submits the identity id^*. Following this, the challenger generates a ciphertext ct_1 by encrypting an arbitrary message m^* using the key tied to id^*. The challenge ciphertext given to $\mathcal{A}$ is either a sanitization of ct_0 or ct_1. Importantly, $\mathcal{A}$ is precluded from obtaining a decryption key for the identity slated to receive all encryption keys, including id^*. By upholding this constraint, $\mathcal{A}$ can't discern any data from the inception of ct_0, ensuring the ciphertext remains devoid of unintended information.

4 System Overview

This section mainly illustrates the system architecture, the threat model, and our design objectives.

4.1 System Architecture

The system architecture is shown in Fig 1, where there are five kinds of entities involved: authority, data sender, sanitizer, cloud server, and data recipient. The detailed responsibility of each entity is described as follows: (1)

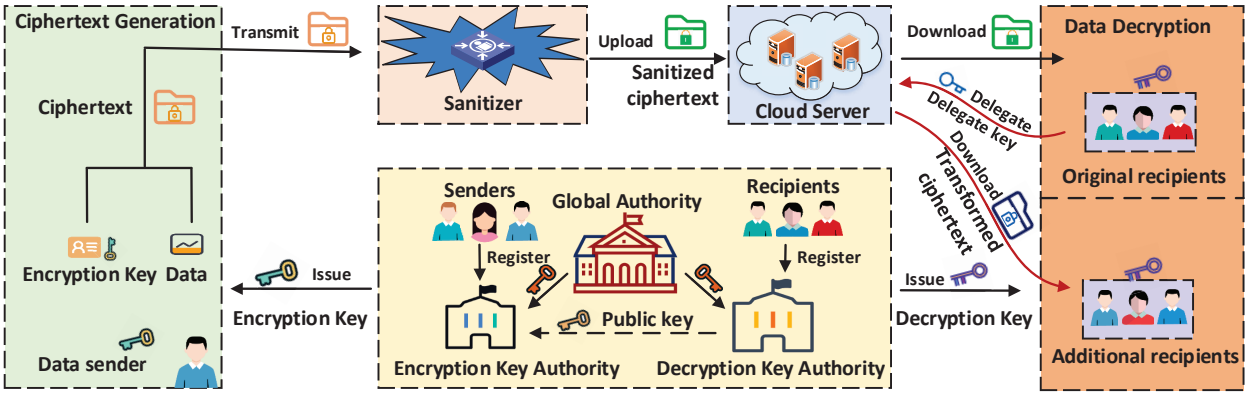


Fig. 1: System Architecture of SCPA

the *Authority* in our system is actually divided into three kinds of authority: global authority, encryption key authority, and decryption key authority. The whole global public parameter is maintained by the global authority for the whole system. The encryption key authority working as an encryption key generation center is responsible for issuing a unique encryption key for each sender who registers into the system and managing its produced public key. The decryption key authority acting as a private key generation center takes charge of maintaining its own public key as well as producing a unique secret key for each recipient who also registers into the system. (2) The *data sender* performs data encryption operation on his/her owned plain data to produce the corresponding ciphertext and sends it to the sanitizer. (3) The sanitizer takes the responsibility to ensure the authenticity of the ciphertext and transform the original ciphertext into a sanitized ciphertext. *Such a sanitization implementation on ciphertext aims to hinder illegal data access resulting from some malicious behaviors of malicious senders.* (4) The *cloud server* provides infinite cloud storage resources for ciphertext storage and responds to the downloading requests of any recipients. Besides, it also receives the ciphertext conversion request from an original recipient and then dynamically transforms the sanitized ciphertext into a re-encrypted one for sharing with additional recipients. (5) The *data recipient* in the system has two categories: original data recipients and additional recipients. These two kinds of recipients can freely download the cipher data that they are interested in from the cloud server. Prior to decoding the ciphertext, the recipients must register into the system to acquire the valid secret key from the decryption key authority. For a valid original recipient, the sanitized ciphertext can be decoded and accessed if he/she holds the legitimate decryption key. For a valid additional authorized recipient, the re-encrypted ciphertext can be deciphered and accessed if he/she has the valid decryption key.

4.2 Threat Model and Design Objectives

Without the loss of generality, we assume that various authorities and the sanitizer are fully trusted entities who neither collude with any other entities nor suffer from any compromise, ensuring their faithful execution of the corresponding algorithms [40], [42]. The sender can be either an honest or malicious entity, *i.e.*, both honest and malicious senders can perform the encryption operation on the cleartext while *malicious* senders may engage in potentially harmful behaviors during the encryption process. The cloud server is a semi-reliable entity, which honestly performs the tasks but attempts to learn some information from the data ciphertext regardless of the original or re-encrypted one. For the recipients, the authorized recipient is an honest entity, who can share neither his/her authorized secret nor decrypted plaintext with other entities, and the invalid recipient who desires to decode and access the plain data is malicious. In our model, we consider the attackers (including honest-but-curious cloud servers and unauthorized recipients) may engage in various malicious activities to try to gain access to the plaintext data despite not possessing legitimate authorization. Besides, the honest-but-curious cloud server can also learn nothing from the delegated keys. *It is worth noting that our threat model considers the case of a malicious sender, while it is in most data-sharing scenarios commonly considered a fully-trusted one.* Correspondingly, the design objectives of our SCPA are as follows:

- *Confidentiality of data.* The encrypted data can only be successfully recovered and accessed by the recipient possessing legitimate decryption keys. That is to say, any adversary, including the cloud server and unauthorized recipients, will be unable to access the encrypted data without the correct decryption keys.
- *Malicious unauthorized access resistance.* The malicious behavior of a sender deliberately leaking some secrets leading to illegal access authorization will be blocked. In other words, any ciphertext via a sanitizer will be sanitized to ensure the legitimate access of only authorized users.
- *No-read and No-write Rules*. This objective is to ensure that only intended receivers should have access to any information about the message. Furthermore, any group of (corrupt) senders (within the identity list $\mathcal{S}$) should not be able to transfer any information to any group of (corrupt) recipients (within the identity list $\mathcal{R}$) unless at least one of the senders in $\mathcal{S}$ is authorized to communicate with at least one of the

recipients in $\mathcal{R}$.

5 Our SCPA

5.1 An Overview of Our SCPA

As we all know, proxy re-encryption technology enables the transformation of ciphertext from an original recipient to an additional designated recipient, thereby allowing the seamless transfer of data access privileges to recipients beyond initially authorized. Indeed, applying attribute-based encryption [39] to proxy re-encryption can solve the inflexibility of strategies to dynamically designate ciphertexts to multiple recipients. However, existing proxy re-encryption solutions commonly suffer from either impracticality or inefficiency as they are limited to an "all-or-nothing" ciphertext conversion mechanism. Besides, they usually exhibit inflexibility since only one condition instead of multiple conditions in the re-encryption key can be specified, thus being incapable of supporting a subset of ciphertext sharing bound with the multi-conditional setting.

Our SCPA seems to be a simple combination of cross-domain ACE and PRE but actually is not a trivial combination. Our motivation for our SCPA is to devise a sanitizable cross-domain access control mechanism with policy-driven dynamic authorization based on identity-based cross-domain ACE. The most challenge in devising such a scheme is how to handle the flexibility issue of dynamically designating a subset of ciphertexts with the multi-conditional setting to multiple recipients in the identity-based setting, *i.e.*, realizing the "all-or-nothing" ciphertext conversion to a selective subset of ciphertext, which is different from the traditional PRE incorporated into cross-domain ACE. Specifically, in our SCPA, we devised a novel approach for applying a linear secret sharing scheme to the secret key of a delegator according to access policy, thus realizing a subset of ciphertext sharing bound with the multi-conditional setting. Compared to previous PRE solutions failing to realize access policy in the delegation key for flexible sharing, we overcame the challenge of the fact that the master secret key is unknown to the delegator and addressed the issue of splitting the master secret key into shares and assigning each share to a condition involved in the access policy.

5.2 Concrete Construction of SCPA

- **Global Setup**: With the input security parameter 1^λ, it picks a bilinear group $\mathcal{BG} = (p, \mathbb{G}_0, \mathbb{G}_1, \mathbb{G}_2, e, g, \mathsf{D})$, where g and D are respective generators of groups $\mathbb{G}_0$ and $\mathbb{G}_1$ with its prime order p, and outputs a global public parameter $\mathsf{gpk} = \mathcal{BG}$.
- **Receiver-Authority Setup**: With the input global public parameter gpk, it picks $\alpha, \varphi \in \mathbb{Z}_p$, $\mu, u, h, v, w \in \mathbb{G}_1$ and calculates $\rho = g^\varphi$, $g_i = g^{\alpha^i}$, $v_i = v^{\alpha^i}$, $w_i = w^{\alpha^i}$, $\hat{\rho} = e(\rho, \mu)$. Besides, it also chooses two hash functions: $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_p$ and $\mathcal{H}_0 : \mathbb{G}_T \to \mathbb{G}_0$. It finally outputs its public parameter $\mathsf{pk_{ra}} = (\mu, u, h, v, w, \hat{\rho}, \{g_i, v_i, w_i\}_{i\in[1,\ell_{\max}]}, \mathcal{H}, \mathcal{H}_0)$ and master secret key $\mathsf{msk_{ra}} = (\mu^\varphi, \alpha)$, where $\ell_{\max}$ denotes the maximum number of users.
- **Sender-Authority Setup**: With the input gpk and $\mathsf{pk_{ra}}$, it first chooses $\mathsf{Y} \in \mathbb{G}_0$, $\tau \in \mathbb{Z}_p$, computes $\mathsf{V} = \mu^\tau$, generates a signing key τ and a verification key $\mathsf{vk} = (\mathsf{Y}, \mathsf{V}, \mathsf{gpk})$. Besides, for the NIZK scheme, it produces its common reference string crs. It finally outputs its public parameter $\mathsf{pk_{sa}} = (\mathsf{vk}, \mathsf{crs})$ and master secret key $\mathsf{msk_{sa}} = \tau$.
- **Encryption-Key Registration**: With the input an identity set $\mathcal{S}$ that is permitted to write to, it first parses $\mathcal{S} = (\mathsf{id}_1, \ldots, \mathsf{id}_\ell)$, where $\ell \le \ell_{\max}$ and computes $\mathsf{A} = g^{\prod_{i\in\mathcal{S}}(\alpha+\mathcal{H}(\mathsf{id}_i))}$. In addition, it selects $\beta \in \mathbb{Z}_p$ and calculates $\mathsf{R} = \mathsf{D}^\beta$, $\mathsf{S} = \mathsf{A}^{\frac{\tau}{\beta}}\mathsf{Y}^{\frac{1}{\beta}}$, $\mathsf{T} = \mathsf{S}^{\frac{\tau}{\beta}} g^{\frac{1}{\beta}}$, which forms a signature $\sigma = (\mathsf{R}, \mathsf{S}, \mathsf{T})$ on A and a signature randomization token $\mathsf{RT} = g^{\frac{1}{\beta}}$. It finally produces an encryption key $\mathsf{ek}_{\mathsf{id}_j} = (\mathsf{A}, \sigma)$.
- **Decryption-Key Registration**: With the input $\mathsf{msk_{ra}} = (\mu^\varphi, \alpha)$ and an identity id_{j^*}, it generates a decryption key $\mathsf{sk}_{\mathsf{id}_{j^*}} = \mu^{\varphi/(\alpha+\mathcal{H}(\mathsf{id}_{j^*}))}$ for the identity id_{j^*}.
- **Encryption**: With the input gpk, $\mathsf{pk_{sa}}$, an encryption key $\mathsf{ek}_{\mathsf{id}_j}$, a cleartext m with its decryption labeling a set $\mathcal{L} = \{\mathcal{L}_1, \ldots, \mathcal{L}_n\}$ of conditions, it performs the following steps:
(1) It uniformly chooses $s \in \mathbb{Z}_p$ and counts $\mathsf{C}_1 = \rho^s$, $\mathsf{C}_2 = \mathsf{A}^s$, $\mathsf{C}_3 = v^{s\prod_{\mathsf{id}_i\in\mathcal{S}}\frac{\alpha+\mathcal{H}(\mathsf{id}_j)}{\mathcal{H}(\mathsf{id}_i)}}$, $\mathsf{C}_4 = \hat{\rho}^s \cdot m$. Additionally, for each condition $\mathcal{L}_k$ in $\mathcal{L}$, it randomly chooses $r_k \in \mathbb{Z}_p$ and conducts the following calculations: $\mathsf{C}_{5,k} = g^{r_k}$, $\mathsf{C}_{6,k} = (u^{\mathcal{H}(\mathcal{L}_k)}h)^{r_k} \cdot w^{-s\prod_{\mathsf{id}_i\in\mathcal{S}}(\alpha+\mathcal{H}(\mathsf{id}_i))}$. Note that $v^\alpha, \ldots, v^{\alpha^n}$ can be used to compute $v^{\prod_{\mathsf{id}_i\in\mathcal{S}}(\alpha+\mathcal{H}(\mathsf{id}_i))}$.
(2) It also selects $s', r'_k \in \mathbb{Z}_p$ and calculates "Sanitizing items" as $(c_1, c_2, c_3, c_4) = (\rho^{s'}, \mathsf{A}^{s'}, v^{s'\prod_{\mathsf{id}_i\in\mathcal{S}}\frac{\alpha+\mathcal{H}(\mathsf{id}_j)}{\mathcal{H}(\mathsf{id}_i)}}, \hat{\rho}^{s'})$ and $(c_{5,k}, c_{6,k}) = (g^{r'_k}, (u^{\mathcal{H}(\mathcal{L}_k)}h)^{r'_k} \cdot w^{-s'\prod_{\mathsf{id}_i\in\mathcal{S}}(\alpha+\mathcal{H}(\mathsf{id}_i))})$.
(3) It next generates a NIZK proof π as follows for $\mathsf{A} = g^{\prod_{i\in\mathcal{S}}(\alpha+\mathcal{H}(\mathsf{id}_i))}$, signature $\sigma = (\mathsf{R}, \mathsf{S}, \mathsf{T})$ and randomness (s, r_k, s', r'_k):

$$\mathsf{PoK}\left\{\Phi : \begin{array}{c} e(\mathsf{S},\mathsf{R})e(\mathsf{A},\mathsf{V}^{-1}) = e(\mathsf{Y},\mathsf{D}) \wedge \\ e(\mathsf{T},\mathsf{R})e(\mathsf{S},\mathsf{V}^{-1}) = e(\mathsf{g},\mathsf{D}) \wedge \\ \mathsf{C}_1 = \rho^s \wedge \mathsf{c}_1 = \rho^{s'} \wedge \mathsf{C}_2 = \mathsf{A}^s \wedge \\ \mathsf{c}_2 = \mathsf{A}^{s'} \wedge \mathsf{C}_3 = v^{s\prod_{i\in\mathcal{S}}\frac{\alpha+\mathcal{H}(\mathsf{id}_j)}{\mathcal{H}(\mathsf{id}_i)}} \wedge \\ \mathsf{c}_3 = v^{s'\prod_{\mathsf{id}_i\in\mathcal{S}}\frac{\alpha+\mathcal{H}(\mathsf{id}_j)}{\mathcal{H}(\mathsf{id}_i)}} \wedge \mathsf{C}_4 = \hat{\rho}^s \cdot m \\ \wedge \mathsf{c}_4 = \hat{\rho}^{s'} \wedge \mathsf{C}_{5,k} = g^{r_k} \wedge \mathsf{c}_{5,k} = g^{r'_k} \wedge \\ \mathsf{C}_{6,k} = (u^{\mathcal{H}(\mathcal{L}_k)}h)^{r_k} \cdot w^{-s\prod_{\mathsf{id}_i\in\mathcal{S}}(\alpha+\mathcal{H}(\mathsf{id}_i))} \wedge \\ c_{6,k} = (u^{\mathcal{H}(\mathcal{L}_k)}h)^{r'_k} \cdot w^{-s'\prod_{\mathsf{id}_i\in\mathcal{S}}(\alpha+\mathcal{H}(\mathsf{id}_i))} \end{array}\right\}$$

by utilizing Schnorr's style proof with Shamir heuristics, where $\Phi = (\mathsf{A}, \sigma, m, s, r_k, s', r'_k)$. Moreover, the **Rand**() algorithm presented in the SPS system [34] can be used to randomize σ into $(\mathsf{R}', \mathsf{S}', \mathsf{T}')$. Note that the NIZK proof for our PoK can be found in Supplementary Material B.
(4) It finally outputs an unsanitized ciphertext $\mathsf{ct} = ((\mathsf{C}_1, \mathsf{C}_2, \mathsf{C}_3, \mathsf{C}_4, \mathsf{C}_{5,k}, \mathsf{C}_{6,k}, c_1, c_2, c_3, c_4, c_{5,k}, c_{6,k}), \pi)$.

- **Sanitization**: With the input unsanitized ciphertext $\mathsf{ct} = (\mathsf{c}, \pi)$, it first performs the verification of proof π and if the verification gives 0, it aborts and outputs $\perp$; otherwise, it chooses $r \in \mathbb{Z}_p$ and returns a sanitized ciphertext $\mathsf{ct}^* = (\mathsf{C}_1 \cdot c_1^r, \mathsf{C}_2 \cdot c_2^r, \mathsf{C}_3 \cdot c_3^r, \mathsf{C}_4 \cdot c_4^r, \mathsf{C}_{5,k} \cdot c_{5,k}^r, \mathsf{C}_{6,k} \cdot c_{6,k}^r) = (\mathsf{C}_1^*, \mathsf{C}_2^*, \mathsf{C}_3^*, \mathsf{C}_4^*, \mathsf{C}_{5,k}^*, \mathsf{C}_{6,k}^*)$.
- **Delegation**: With the input $\mathsf{pk}_{\mathsf{ra}}$, an identity set $\mathcal{S}'$, a secret key $\mathsf{sk}_{\mathsf{id}}$ ($\mathsf{id}_k \in \mathcal{S}$) and an access policy including n conditions, it conducts the following procedures to generate the delegation key $\mathsf{dk}_{\mathsf{id}\to\mathcal{S}'|\mathbb{A}}$:
(1) It first chooses $\eta, s^* \in \mathbb{Z}_p$ and computes $\mathsf{d}_0 = g^\eta \cdot \mathcal{H}_1(e(\rho,\mu)^{s^*})$, $\mathsf{d}_1 = g^{s^* \prod_{i \in \mathcal{S}'}(\alpha + \mathcal{H}(\mathsf{id}_i'))}$, $\mathsf{d}_2 = \rho^{s^*}$.
(2) It next creates an LSSS (A, ϱ) for $\mathbb{A}$, where A is $l \times \kappa$ matrix and ϱ is the function mapping each row of A to a condition, and uniformly picks $x_2, x_3, \ldots, x_\kappa \in \mathbb{Z}_p$ to form a vector $\vec{\eta} = (\eta, x_2, x_3, \ldots, x_\kappa)$. For the q-the row $A_q = (A_{q,1}, \ldots, A_{q,\kappa})$ of A, it computes a share of η as $\beta_q = A_q \cdot \vec{\eta}$ for each $q \in l$. Besides, it also picks $x_2', x_3', \ldots, x_\kappa' \in \mathbb{Z}_p$ and calculates $\mathsf{sk}_{\mathsf{id}}^{A_{q,1}} \cdot \mu^{x_2' A_{q,2} + \ldots + x_{q,\kappa}' A_{q,\kappa}}$. If $\mathsf{sk}_{\mathsf{id}} = \mu^x$ for an unknown $x \in \mathbb{Z}_p$, then the above formulas can be denoted as $\mathsf{sk}_{\mathsf{id}}^{A_{q,1}} \cdot \mu^{x_2' A_{q,2} + \ldots + x_{q,\kappa}' A_{q,\kappa}} = \mu^{A_q \vec{x}}$, where $\vec{x} = (x, x_2', x_3', \ldots, x_\kappa')$. For ease of expression, $\chi_q = A_q \vec{x}$. It continues to choose $t_q \in \mathbb{Z}_p$ for $q \in [1, l]$ and calculates $\mathsf{d}_{3,q} = \mu^{\chi_q} \cdot v^{\beta_q / \mathcal{H}(\mathsf{id})} \cdot w^{t_q}$, $\mathsf{d}_{4,q} = (u^{\varrho(q)} h)^{-t_q}$, $\mathsf{d}_{5,q} = g^{-t_q}$.
(3) It outputs the delegation key $\mathsf{dk}_{\mathsf{id}\to\mathcal{S}'|\mathbb{A}} = (\mathsf{d}_0, \mathsf{d}_1, \mathsf{d}_2, \mathsf{d}_{3,q}, \mathsf{d}_{4,q}, \mathsf{d}_{5,q})$, where $q \in [1, l]$.
- **Re-Encryption**: With the input sanitized ciphertext ct^*, the identity list $\mathcal{S}$, the conditions $\mathcal{L}$, the delegation key $\mathsf{dk}_{\mathsf{id}\to\mathcal{S}'|\mathbb{A}}$, it aborts and outputs $\perp$ if $\mathcal{L} \not\models \mathbb{A}$ holds. Otherwise, due to the property of reconstruction of LSSS, it can find $\lambda_q \in \mathbb{Z}_p$ such that $\sum_{q \in \Omega} \beta_q \lambda_q = \eta$ and $\sum_{q \in \Omega} \chi_q \lambda_q = x$, where $\Omega = \{q : \varrho(q) = \mathcal{H}(\mathcal{L}_j)\}$ for $\mathcal{L}_j \in \mathcal{L} \subseteq [1, l]$. It next carries out the following steps:
(1) For each $q \in \Omega$, it calculates $\mathsf{W}_q = e(\mathsf{d}_{3,q}, \mathsf{C}_2^*) \cdot e(\mathsf{d}_{4,q}, \mathsf{C}_{5,k}^*) e(\mathsf{d}_{6,q}, \mathsf{C}_{6,k}^*)$.
(2) It next computes $\mathsf{W} = [\prod_{q \in \Omega} \mathsf{W}_q^{\lambda_q} \cdot e(g^\Delta, \mathsf{C}_1^{*-1})]^{\frac{1}{\prod_{\mathsf{id}_i \in \mathcal{S}^*} \mathcal{H}(\mathsf{id}_i)}}$, where $\Delta = \prod_{\mathsf{id}_i \in \mathcal{S}^*}(\alpha + \mathcal{H}(\mathsf{id}_i)) - \prod_{\mathsf{id}_i \in \mathcal{S}^*} \mathcal{H}(\mathsf{id}_i)$ and $\mathcal{S}^* = \mathcal{S} \setminus \mathsf{id}$.
(3) It finally sets and outputs $\mathsf{ct}' = (\mathsf{C}_1', \mathsf{C}_2', \mathsf{C}_3', \mathsf{C}_4', \mathsf{C}_5')$, where $\mathsf{C}_1' = \mathsf{d}_2$, $\mathsf{C}_2' = \mathsf{d}_1$, $\mathsf{C}_3' = \mathsf{C}_3^*$, $\mathsf{C}_4' = \mathsf{C}_4^* / \mathsf{W}$ and $\mathsf{C}_5' = \mathsf{d}_0$.
- **Decryption**: With the input $\mathsf{pk}_{\mathsf{ra}}$, the secret key $\mathsf{sk}_{\mathsf{id}}$, the ciphertext $\mathsf{ct}^*/\mathsf{ct}'$ and the identity list $\mathcal{S}/\mathcal{S}'$, it can recover the plaintext with corresponding $\mathsf{sk}_{\mathsf{id}}$.
(1) For the ciphertext ct^* and the secret key $\mathsf{sk}_{\mathsf{id}_{j^*}}$, where $\mathsf{id}_{j^*} \in \mathcal{S}$, it computes $m = \mathsf{C}_4^* \cdot [e(\mathsf{C}_1^*, \mu^\Delta) \cdot e(\mathsf{C}_2^*, \mathsf{sk}_{\mathsf{id}_{j^*}}^{-1})]^{\frac{1}{\prod_{\mathsf{id}_i \in \mathcal{S}^*} \mathcal{H}(\mathsf{id}_i)}}$, where $\Delta = \prod_{\mathsf{id}_i \in \mathcal{S}^*}(\alpha + \mathcal{H}(\mathsf{id}_i)) - \prod_{\mathsf{id}_i \in \mathcal{S}^*} \mathcal{H}(\mathsf{id}_i)$ and $\mathcal{S}^* = \mathcal{S} \setminus \mathsf{id}_{j^*}$.
(2) For the ciphertext $\mathsf{ct}' = (\mathsf{C}_1', \mathsf{C}_2', \mathsf{C}_3', \mathsf{C}_4', \mathsf{C}_5')$ and the secret key $\mathsf{sk}_{\mathsf{id}_{j^*}'}$, where $\mathsf{id}_{j^*}' \in \mathcal{S}'$, it first computes $\mathsf{X} = [e(\mathsf{C}_1'^{-1}, \mu^{\Delta'}) \cdot e(\mathsf{C}_2', \mathsf{sk}_{\mathsf{id}_{j^*}'})]^{\frac{1}{\prod_{\mathsf{id}_i \in \mathcal{S}^{*\prime}} \mathcal{H}(\mathsf{id}_i)}}$, where $\Delta' = \prod_{\mathsf{id}_i \in \mathcal{S}^{*\prime}}(\alpha + \mathcal{H}(\mathsf{id}_i)) - \prod_{\mathsf{id}_i \in \mathcal{S}^{*\prime}} \mathcal{H}(\mathsf{id}_i)$ and $\mathcal{S}^{*\prime} = \mathcal{S}' \setminus \mathsf{id}_{j^*}'$. It then calculates $g^\eta = \mathsf{C}_5' / \mathcal{H}_1(\mathsf{X})$. It finally computes $m = \mathsf{C}_4' \cdot e(g^\eta, \mathsf{C}_3')$.

Remark: In our SCPA, the Sender-Authority Setup algorithm's use of the receiver authority's public key is a crucial step to facilitate secure and efficient cross-domain encryption. It ensures that messages encrypted by the sender are compatible with the decryption capabilities of the receiver, without compromising the independence of the respective authorities managing the senders and recipients.

Besides, the identity-based matchmaking encryption (IB-ME) [41] cannot completely solve but only partially solve the following problems described in our paper: (1) how to resist unauthorized access resulting from malicious information leakage by the sender; (2) how to design a cross-domain No-write and No-read rules to restrict write and read permissions for participants; (3) how to solve the inflexibility of strategies to dynamically designate ciphertexts to multiple recipients. As we all know, IB-ME can indeed solve the challenge (1) since both the sender and the receiver (each with its own identity) can specify policies (i.e., identity) the other party must satisfy in order for the message to be revealed. However, the challenges (2 &3) cannot be well-resolved with the IB-ME. Specifically, for the challenge (2), only one sender's and one receiver's identity are embedded in a ciphertext in the IB-ME, which basically determines one-to-one sharing mode while the motivation of our SCPA is intended for one-to-many sharing mode; For the third challenge, the existing IB-ME does not consider realizing the seamless transfer of data access privileges to recipients beyond initially authorized, which is the important property for dynamical data sharing scenarios.

6 Correctness and Security Analysis

In this section, we provide a comprehensive exposition of the proofs pertaining to our SCPA scheme, which specifically includes the rigorous proofs of SCPA correctness, CPA security, and No-read and No-write rules.

6.1 Correctness of Our SCPA

Theorem 1. Regardless of whether the ciphertext is the original sanitized ciphertext or the transformed ciphertext, a user, if he/she owns the authorized secret key, can successfully recover the plaintext hidden in the ciphertext.

Proof 1. In the decryption process, there are two kinds of ciphertext including sanitized ciphertext and transformed ciphertext to be decoded with legitimate secret keys. To be more specific, the following derivation process can prove the correctness of this theorem.

- For the valid sanitized ciphertext $\mathsf{ct}^* = ((\mathsf{C}_1^*, \mathsf{C}_2^*, \mathsf{C}_3^*, \mathsf{C}_4^*, \mathsf{C}_{5,k}^*, \mathsf{C}_{6,k}^*), \mathcal{S})$, a user with his/her secret key $\mathsf{sk}_{\mathsf{id}_i^*} = \mu^{\varphi/(\alpha + \mathcal{H}(\mathsf{id}_{j^*}))}$ can perform the following computation if $\mathsf{id}_{i^*} \in \mathcal{S}$:

$$\begin{aligned} m &= \mathsf{C}_4^* \cdot [e(\mathsf{C}_1^*, \mu^\Delta) \cdot e(\mathsf{C}_2^*, \mathsf{sk}_{\mathsf{id}_{j^*}}^{-1})]^{\frac{1}{\prod_{\mathsf{id}_i \in \mathcal{S}^*} \mathcal{H}(\mathsf{id}_i)}} \\ &= m \cdot \hat{\rho}^{\hat{s}} \cdot [e(\rho^{\hat{s}}, \mu^{\prod_{\mathsf{id}_i \in \mathcal{S}^*}(\alpha + \mathcal{H}(\mathsf{id}_i)) - \prod_{\mathsf{id}_i \in \mathcal{S}^*} \mathcal{H}(\mathsf{id}_i)}) \cdot \\ &\quad e(g^{\hat{s} \prod_{\mathsf{id}_i \in \mathcal{S}}(\alpha + \mathcal{H}(\mathsf{id}_i))}, \mu^{-\varphi/(\alpha + \mathcal{H}(\mathsf{id}_{i^*}))})]^{\frac{1}{\prod_{\mathsf{id}_i \in \mathcal{S}^*} \mathcal{H}(\mathsf{id}_i)}} \\ &= m \cdot e(g, \mu)^{\varphi \hat{s}} \cdot e(g, \mu)^{-\varphi \hat{s}}, \end{aligned}$$

where $\hat{s} = s + rs'$, $\mathcal{S}^* = \mathcal{S} \setminus \mathsf{id}_{i^*}$, and $\Delta = \prod_{\mathsf{id}_i \in \mathcal{S}^*}(\alpha + \mathcal{H}(\mathsf{id}_i)) - \prod_{\mathsf{id}_i \in \mathcal{S}^*} \mathcal{H}(\mathsf{id}_i)^{\frac{1}{\prod_{\mathsf{id}_i \in \mathcal{S}^*} \mathcal{H}(\mathsf{id}_i)}}$.

- Before proving the correctness of decrypting the transformed ciphertext, we first show the correctness of the re-encryption algorithm. Specifically, we first calculate the following equations:

$$\begin{aligned}
\mathsf{W}_q &= e(\mathsf{d}_{3,q}, \mathsf{C}_2^*) \cdot e(\mathsf{d}_{4,q}, \mathsf{C}_{5,k}^*) e(\mathsf{d}_{6,q}, \mathsf{C}_{6,k}^*) \\
&= e(\mu^{\chi_q} \cdot v^{\beta_q/\mathcal{H}(\mathsf{id}_i)} \cdot w^{t_q}, g^{(s+rs')\prod_{\mathsf{id}_i \in \mathcal{S}}(\alpha+\mathcal{H}(\mathsf{id}_i))}) \\
&\cdot e((u^{\varrho(q)}h)^{-t_q}, g^{r_k+rr'_k}) \cdot e(g^{-t_q}, (u^{\mathcal{H}(\mathcal{L}_k)}h)^{r_k+rr'_k} \cdot \\
&w^{-(s+rs')\prod_{\mathsf{id}_i \in \mathcal{S}}(\alpha+\mathcal{H}(\mathsf{id}_i))}) \\
&= e(\mu^{\chi_q} \cdot v^{\beta_q/\mathcal{H}(\mathsf{id}_i)}, g^{(s+rs')\prod_{\mathsf{id}_i \in \mathcal{S}}(\alpha+\mathcal{H}(\mathsf{id}_i))}), \\
\mathsf{W} &= [\prod_{q\in\Omega} \mathsf{W}_q^{\lambda_q} \cdot e(g^{\Delta}, \mathsf{C}_1^{*-1})]^{\frac{1}{\prod_{\mathsf{id}_i \in \mathcal{S}^*} \mathcal{H}(\mathsf{id}_i)}} \\
&= [e(\mu^x \cdot v^{\eta/\mathcal{H}(\mathsf{id}_i)}, g^{(s+rs')\prod_{\mathsf{id}_i \in \mathcal{S}}(\alpha+\mathcal{H}(\mathsf{id}_i))}) \cdot \\
&e(g^{\prod_{\mathsf{id}_i \in \mathcal{S}^*}(\alpha+\mathcal{H}(\mathsf{id}_i)) - \prod_{\mathsf{id}_i \in \mathcal{S}^*} \mathcal{H}(\mathsf{id}_i)}, \rho^{-(s+rs')})]^{\frac{1}{\prod_{\mathsf{id}_i \in \mathcal{S}^*} \mathcal{H}(\mathsf{id}_i)}} \\
&= e(v^{\eta}, g^{(s+rs')\prod_{\mathsf{id}_i \in \mathcal{S}} \frac{\alpha+\mathcal{H}(\mathsf{id}_i)}{\mathcal{H}(\mathsf{id}_i)}}) \cdot e(g, \rho)^{s+rs'}, \\
\mathsf{C}'_4 &= \mathsf{C}_4^*/\mathsf{W} = m \cdot e(v^{-\eta}, g^{(s+rs')\prod_{\mathsf{id}_i \in \mathcal{S}} \frac{\alpha+\mathcal{H}(\mathsf{id}_i)}{\mathcal{H}(\mathsf{id}_i)}}),
\end{aligned}$$

where $x = \sum_{q\in\Omega} \lambda_q \chi_q$, $\eta = \sum_{q\in\Omega} \lambda_q \beta_q$, and $\mathcal{S}^* = \mathcal{S} \setminus \mathsf{id}_i$.
For the transformed ciphertext $\mathsf{ct}^* = (\mathsf{C}'_1, \mathsf{C}'_2, \mathsf{C}'_3, \mathsf{C}'_4, \mathsf{C}'_5), \mathcal{S}')$, a user with his/her secret key $\mathsf{sk}_{\mathsf{id}'_{i^*}}$ ($\mathsf{id}'_{i^*} \in \mathcal{S}'$) performs the following computation:

$$\begin{aligned}
\mathsf{X} &= [e(\mathsf{C}_1'^{-1}, \mu^{\Delta'}) \cdot e(\mathsf{C}'_2, \mathsf{sk}_{\mathsf{id}'_{i^*}})]^{\frac{1}{\prod_{\mathsf{id}_i \in \mathcal{S}^{*'}} \mathcal{H}(\mathsf{id}_i)}} \\
&= [e(\rho^{-s^*}, \mu^{\prod_{\mathsf{id}_i \in \mathcal{S}^{*'}}(\alpha+\mathcal{H}(\mathsf{id}_i)) - \prod_{\mathsf{id}_i \in \mathcal{S}^{*'}} \mathcal{H}(\mathsf{id}_i)}) \cdot \\
&e(g^{s^* \prod_{\mathsf{id}_i \in \mathcal{S}'}(\alpha+\mathcal{H}(\mathsf{id}_i))}, \mu^{\varphi/(\alpha+\mathcal{H}(\mathsf{id}'_{i^*}))})]^{\frac{1}{\prod_{\mathsf{id}_i \in \mathcal{S}^{*'}} \mathcal{H}(\mathsf{id}_i)}} \\
&= e(g, \mu)^{\varphi s^*},
\end{aligned}$$

where $\mathcal{S}^{*'} = \mathcal{S}' \setminus \mathsf{id}'_{i^*}$ and $\Delta' = \prod_{\mathsf{id}_i \in \mathcal{S}^{*'}}(\alpha+\mathcal{H}(\mathsf{id}_i)) - \prod_{\mathsf{id}_i \in \mathcal{S}^{*'}} \mathcal{H}(\mathsf{id}_i)$.
The user can calculate $g^{\eta} = \mathsf{C}'_5/\mathcal{H}_1(\mathsf{X})$ and finally recover $m = \mathsf{C}'_4 \cdot e(g^{\eta}, \mathsf{C}'_3)$.

6.2 Proof of CPA-security of Our SCPA

Theorem 2. Assuming the variant GDDHE assumption holds, our SCPA is also secure in the random oracle model if the IBBE scheme [7] is CPA-secure and the structure-preserving signature scheme [34] is unforgeability-secure. In other words, neither the cloud nor unauthorized users without legitimate secret keys can successfully decipher the original sanitized ciphertexts or any re-encrypted ones.

Proof 2. It is worth noting that the construction of our SCPA is built on an IBBE scheme [7], which preserves the ciphertext and decryption structure of the IBBE scheme, thus leading to the fact that the sanitized ciphertext in our SCPA contains an IBBE ciphertext and the secret key is almost identical to the secret key of IBBE. Furthermore, the additional components in the sanitized ciphertext and the delegation key can be simulated without requiring the master secret key. Hence, the adversary's successful attacks against our SCPA can be exploited to break the security of IBBE [7]. Since the security of IBBE [7] has been notoriously proved by the theorem 1 [7], hence no adversary can break the security of our SCPA. It is nothing that the GDDHE tuple $(g, g^{\alpha}, \ldots, g^{\alpha^{t-1}}, g^{\alpha f(\alpha)}, g^{s\alpha f(\alpha)}, \mu, \mu^{\alpha}, \ldots, \mu^{\alpha^n}, \mu^{1/g(\alpha)}, \mu^{f(\alpha)/g(\alpha)}, \mu^{sg(\alpha)})$ including a black box parameter of IBBE to produce the SCPA's parameters, the task of distinguishing $\mathcal{Z} = e(g, \mu)^{sf(\alpha)}$ and $\mathcal{Z} = \mathcal{Z}_1$ is difficult, where $\mathcal{Z}_1$ is a random element of the group $\mathbb{G}_2$. To be more specific, the formal security proof is shown as follows:

- **Init**: An identity list $\mathcal{S}^* = \{\mathsf{id}_1^*, \ldots, \mathsf{id}_\ell^*\}$ that is allowed to write to and a set $\mathcal{L}^* = (\mathcal{L}_1^*, \ldots, \mathcal{L}_n^*)$ of conditions are picked by $\mathcal{A}$ and then sent to $\mathcal{C}$. In this phase, $\mathcal{C}$ needs to initialize two empty tables ($\mathcal{T}_{\mathsf{sk}}$ and $\mathcal{T}_{\mathsf{dk}}$) for storing the results of secret key queries and delegation key queries.
- **Setup**: $\mathcal{C}$ makes queries to the setup algorithm of IBBE [7] to get the public key $\mathsf{pk}_{\mathsf{IBBE}} = (g, g^{\alpha}, \ldots, g^{\alpha^{t-1}}, g^{\alpha f(\alpha)}, \mu, \mu^{\alpha}, \ldots, \mu^{\alpha^n})$. It then picks $\varphi, \gamma_1, \gamma'_1 \in \mathbb{Z}_p$ and computes $\rho = g^{\varphi}$, $v = \mu^{\gamma_1}$, $w = \mu^{\gamma'_1}$, $v_i = \mu^{\gamma_1 \alpha^i}$, $w_i = \mu^{\gamma'_1 \alpha^i}$. Besides, it also selects $u, h \in \mathbb{G}_2$ and two hash functions: $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_p$ and $\mathcal{H}_0 : \mathbb{G}_T \to \mathbb{G}_0$ as random oracles. Finally, it returns the public key $\mathsf{pk}_{\mathsf{ra}} = (\mu, u, h, v, w, \{g_i, v_i, w_i\}_{i\in[1,\ell_{\max}]}, \mathcal{H}, \mathcal{H}_0)$ to $\mathcal{A}$ and the master secret key $\mathsf{msk}_{\mathsf{ra}} = (\mu^{\varphi}, \alpha)$ is unknown to $\mathcal{C}$. For generating the public key and master secret key for sender-authority, it first chooses $\mathsf{Y} \in \mathbb{G}_0$, $\tau \in \mathbb{Z}_p$, computes $\mathsf{V} = \mu^{\tau}$, generates a signing key τ and a verification key $\mathsf{vk} = (\mathsf{Y}, \mathsf{V}, \mathsf{gpk})$. Besides, for the NIZK scheme, it produces its common reference string crs and finally sends its public parameter $\mathsf{pk}_{\mathsf{sa}} = (\mathsf{vk}, \mathsf{crs})$ to $\mathcal{A}$.
  For the two hash functions $\mathcal{H}, \mathcal{H}_0$, two tables $\mathcal{T}_{\mathcal{H}}$ and $\mathcal{T}_{\mathcal{H}_0}$ should be prepared by $\mathcal{C}$. For the query of $\mathcal{H}$ on $\mathsf{id} \in \{0,1\}^*$ or $\mathcal{L}$, if there exists a tuple $(\mathsf{id}/\mathcal{L}, \Theta)$, return Θ; otherwise, randomly select $\Theta \in \mathbb{Z}_p$, record it in $\mathcal{T}_{\mathcal{H}}$ and return Θ. For the query of $\mathcal{T}_{\mathcal{H}_0}$ on Ω, if there exists a tuple (Ω, Φ), return Φ; otherwise, randomly select $\Phi \in \mathbb{G}_0$, record it in $\mathcal{T}_{\mathcal{H}_0}$ and return Φ.
- **Phases 1 & 2**: $\mathcal{A}$ delivers the following queries to $\mathcal{C}$:
  −**Encryption-Key Registration** (id): The encryption key queries are made by $\mathcal{A}$. If $\mathsf{id}_i \in \mathcal{S}^*$, $\mathcal{C}$ aborts. Otherwise, $\mathcal{C}$ performs the encryption key generation algorithm of the signature scheme [34] to invent the encryption key $\mathsf{sk}_{\mathsf{id}}$.
  −**Decryption-Key Registration** (id): The decryption key queries are made by $\mathcal{A}$. If $\mathsf{id} \in \mathcal{S}^*$, $\mathcal{C}$ aborts. If there is a tuple $(\mathsf{id}', \mathcal{S}', \mathbb{A}, *)$ in $\mathcal{T}_{\mathsf{dk}}$ where $\mathsf{id}' \in \mathcal{S}^*$, $\mathcal{L}^* \in \mathbb{A}$ and $\mathsf{id} \in \mathcal{S}'$, $\mathcal{C}$ aborts. If there is a tuple $(\mathsf{id}, \mathsf{sk}_{\mathsf{id}})$ in $\mathcal{T}_{\mathsf{sk}}$, it returns $\mathsf{sk}_{\mathsf{id}}$; otherwise, $\mathcal{C}$ forwards the query of id to the key generation algorithm of IBBE [7] to capture the decryption key $\mathsf{sk}_{\mathsf{id}}$.
  −**Delegation** ($\mathsf{id} \to \mathcal{S}'|\mathbb{A}$): The delegation key queries for an identity id, an identity set $\mathcal{S}^*$, and an access policy $\mathbb{A}$ are made by $\mathcal{A}$, a delegation key $\mathsf{dk}_{\mathsf{id}\to\mathcal{S}'|\mathbb{A}}$ is returned. If $\mathsf{id} \in \mathcal{S}^*, \mathcal{L}^* \in \mathbb{A}$ and there exists a tuple $(\mathsf{id}' \in \mathcal{S}', \mathsf{sk}_{\mathsf{id}'})$ in $\mathcal{T}_{\mathsf{sk}}$, $\mathcal{C}$ aborts. If there exists a tuple $(\mathsf{id}', \mathcal{S}', \mathbb{A}, \mathsf{dk}_{\mathsf{id}\to\mathcal{S}'|\mathbb{A}})$ in $\mathcal{T}_{\mathsf{dk}}$, $\mathcal{C}$

returns $\mathsf{dk}_{\mathsf{id}\to\mathcal{S}'|\mathbb{A}}$; otherwise, the following cases are considered:

1) $\mathsf{id} \notin \mathcal{S}^*$: For this case, if there is no tuple $(\mathsf{id}, \mathsf{sk}_{\mathsf{id}})$ in $\mathcal{T}_{\mathsf{sk}}$, $\mathcal{C}$ first makes the query of id to the key generation algorithm of IBBE to create a decryption key $\mathsf{sk}_{\mathsf{id}}$. Next, it uses the generated $\mathsf{sk}_{\mathsf{id}}$ to produce $\mathsf{dk}_{\mathsf{id}\to\mathcal{S}'|\mathbb{A}}$ as that in Delegation algorithm since the master secret key is not required for this algorithm, thus returning a well-formed $\mathsf{dk}_{\mathsf{id}\to\mathcal{S}'|\mathbb{A}}$.

2) $\mathsf{id} \in \mathcal{S}^*$: For this case, the query of id to the key generation algorithm of IBBE to create a decryption key $\mathsf{sk}_{\mathsf{id}}$ can not be allowed since $\mathsf{id} \in \mathcal{S}^*$ is the prohibited query defined in the security definition of the IBBE scheme. Without holding $\mathsf{sk}_{\mathsf{id}}$, a well-formed $\mathsf{dk}_{\mathsf{id}\to\mathcal{S}'|\mathbb{A}}$ cannot be invented. To enable a random delegation key, $\mathcal{C}$ first creates an LSSS (A, ϱ) for $\mathbb{A}$, where A is $l \times \kappa$ matrix and ϱ is the function mapping each row of A to a condition, and uniformly picks $x_2, x_3, \ldots, x_\kappa \in \mathbb{Z}_p$ to form a vector $\vec{\eta} = (\eta, x_2, x_3, \ldots, x_\kappa)$. For the q-the row $A_q = (A_{q,1}, \ldots, A_{q,\kappa})$ of A, it computes a share of η as $\beta_q = A_q \cdot \vec{\eta}$ for each $q \in l$. Besides, it also picks $\mathcal{R} \in \mathbb{G}_1$, $s^*, x'_2, x'_3, \ldots, x'_\kappa \in \mathbb{Z}_p$ and counts $\mathsf{d}_0 = g^{\eta}\cdot\mathcal{H}_1(e(\rho,\mu)^{s^*})$, $\mathsf{d}_1 = g^{s^*\prod_{\mathsf{id}'_i\in\mathcal{S}'}(\alpha+\mathcal{H}(\mathsf{id}'_i))}$, $\mathsf{d}_2 = \rho^{s^*}$. It continues to choose $t_q \in \mathbb{Z}_p$ for $q \in [1, l]$ and calculate $\mathsf{d}_{3,q} = \mathcal{R}^{A_{q,1}} \cdot \mu^{x'_2 A_{q,2}+\ldots+x'_{q,\kappa}A_{q,\kappa}} \cdot v^{\beta_q/\mathcal{H}(\mathsf{id})} \cdot w^{t_q}$, $\mathsf{d}_{4,q} = (u^{\varrho(q)}h)^{-t_q}$, $\mathsf{d}_{5,q} = g^{-t_q}$. Ultimately, $\mathcal{C}$ returns $\mathsf{dk}_{\mathsf{id}\to\mathcal{S}'|\mathbb{A}} = (\mathsf{d}_0, \mathsf{d}_1, \mathsf{d}_2, \mathsf{d}_{3,q}, \mathsf{d}_{4,q}, \mathsf{d}_{5,q})$, where $q \in [1, l]$.

- **Challenge**: $\mathcal{A}$ submits two picked equal-length messages m_0 and m_1 and forwards it to the encryption of IBBE scheme [7], which then picks a random $\zeta \in \{0, 1\}$ and produces the ciphertext $\mathsf{C}_4 = m_\zeta \cdot e(\rho,\mu)^s$, $\mathsf{C}_1 = \rho^s$, $\mathsf{C}_2 = g^{s\prod_{i\in\mathcal{S}^*}(\alpha+\mathcal{H}(\mathsf{id}_i))}$. Next, it also computes $\mathsf{C}_3 = (\mu^{s\prod_{i\in\mathcal{S}^*}(\alpha+\mathcal{H}(\mathsf{id}_i))})^{\gamma_1/(\prod_{\mathsf{id}_i\in\mathcal{S}^*}\mathcal{H}(\mathsf{id}_i))} = v^{s\prod_{\mathsf{id}_i\in\mathcal{S}^*}\frac{\alpha+\mathcal{H}(\mathsf{id}_i)}{\mathcal{H}(\mathsf{id}_i)}}$ due to the fact that γ_1 in $v = \mu^{\gamma_1}$ is selected by $\mathcal{C}$. As well, γ'_1 in $w = \mu^{\gamma'_1}$ is also known to $\mathcal{C}$ and $\mathcal{C}$ also selects $r_k \in \mathbb{Z}_p$ to computes $\mathsf{C}_{5,k} = g^{r_k}$, $\mathsf{C}_{6,k} = (u^{\mathcal{H}(\mathcal{L}_k)}h)^{r_k} \cdot \mu^{-s\gamma'_1\prod_{\mathsf{id}_i\in\mathcal{S}^*}(\alpha+\mathcal{H}(\mathsf{id}_i))} = (u^{\mathcal{H}(\mathcal{L}_k)}h)^{r_k} \cdot w^{-s\prod_{\mathsf{id}_i\in\mathcal{S}^*}(\alpha+\mathcal{H}(\mathsf{id}_i))}$. It finally outputs a ciphertext $\mathsf{ct} = (\mathsf{C}_1, \mathsf{C}_2, \mathsf{C}_3, \mathsf{C}_4, \mathsf{C}_{5,k}, \mathsf{C}_{6,k})$. Please note that here ct is just partial ciphertexts of the original ciphertexts since the other parts can be also simulated with the same way. Furthermore, ct is a well-formed ciphertext. Here, we omit the generation of the sanitized ciphertext since it can be easily produced based on the re-randomization for the challenge ciphertext.
- **Guess**: $\mathcal{A}$ submits its guess $\zeta' \in \{0, 1\}$ and $\mathcal{C}$ returns the same bit.

From the above simulation, it is easily observed that the security game of our SCPA is perfected simulated except that a random delegation key that is not a well-formed one is created by $\mathcal{C}$. In the following, we will prove a random delegation key $\mathsf{dk}_{\mathsf{id}\to\mathcal{S}'|\mathbb{A}}$ is indistinguishable from a well-formed one. To prove this, we discuss the following two cases:

- *Case I*: $\mathsf{id} \in \mathcal{S}^*$ and there is no tuple $(\mathsf{id}' \in \mathcal{S}', \mathsf{sk}_{\mathsf{id}'})$ in $\mathcal{T}_{\mathsf{sk}}$. In this case, we state that $\mathcal{A}$ can differentiate the random delegation key from the well-formed one with the probability no more than $\mathcal{A}dv_{\mathcal{A}}^{\mathsf{IBBE\text{-}security}}$. Due to the fact that $\mathcal{R}$ is randomly selected, hence there is an unknown value y satisfying $\mathcal{R} = \mathsf{sk}_{\mathsf{id}} \cdot v^{y/\mathcal{H}(\mathsf{id}_i))}$. Thus, we find $\mathsf{d}_{3,q} = \mu^{\chi_q} \cdot v^{yA_{q,1}/\mathcal{H}(\mathsf{id})} \cdot v^{\beta_q/\mathcal{H}(\mathsf{id})} \cdot w^{t_q} = \mu^{\chi_q} \cdot v^{\beta'_q/\mathcal{H}(\mathsf{id})} \cdot w^{t_q}$, where $\beta'_q = A_q(\eta + y, x_2, x_3, \ldots, x_\kappa)$ is the share of $\eta' = \eta + y$ for q-th row. For the delegation key $\mathsf{d}_0 = g^{\eta}\cdot\mathcal{H}_1(e(\rho,\mu)^{s^*})$, $\mathsf{d}_1 = g^{s^*\prod_{\mathsf{id}'_i\in\mathcal{S}'}(\alpha+\mathcal{H}(\mathsf{id}'_i))}$, $\mathsf{d}_2 = \rho^{s^*}$, these are the actual components of the IBBE for g^{η}. On the other hand, the well-formed one for the above components should involve the IBBE ciphertexts $\mathsf{d}'_0 = g^{\eta'} \cdot \mathcal{H}_1(e(\rho,\mu)^{s^{*\prime}})$, $\mathsf{d}'_1 = g^{s^{*\prime}\prod_{\mathsf{id}'_i\in\mathcal{S}'}(\alpha+\mathcal{H}(\mathsf{id}'_i))}$, $\mathsf{d}'_2 = \rho^{s^{*\prime}}$ for $g^{\eta'}$. In this case, since $\mathcal{A}$ has no secret key $\mathsf{sk}_{\mathsf{id}'}$ for $\mathsf{id}' \in \mathcal{S}'$, therefore $\mathcal{A}$ cannot distinguish $(\mathsf{d}_0, \mathsf{d}_1, \mathsf{d}_2)$ and $(\mathsf{d}'_0, \mathsf{d}'_1, \mathsf{d}'_2)$. However, if $\mathcal{A}$ can differentiate $(\mathsf{d}_0, \mathsf{d}_1, \mathsf{d}_2)$ and $(\mathsf{d}'_0, \mathsf{d}'_1, \mathsf{d}'_2)$, the security of the IBBE scheme is actually breached. Since the probability of breaking the security of the IBBE scheme is at most $\mathcal{A}dv_{\mathcal{A}}^{\mathsf{IBBE\text{-}security}}$, therefore, the probability of distinguishing $(\mathsf{d}_0, \mathsf{d}_1, \mathsf{d}_2)$ from $(\mathsf{d}'_0, \mathsf{d}'_1, \mathsf{d}'_2)$ is also at most $\mathcal{A}dv_{\mathcal{A}}^{\mathsf{IBBE\text{-}security}}$.
- *Case II*: $\mathsf{id} \in \mathcal{S}^*$, $\mathcal{L} \notin \mathbb{A}$ and there is a tuple $(\mathsf{id}' \in \mathcal{S}', \mathsf{sk}_{\mathsf{id}'})$ in $\mathcal{T}_{\mathsf{sk}}$. In this case, we also prove that the probability of $\mathcal{A}$ in differentiating the random delegation key from the well-formed one is no more than $\mathcal{A}dv_{\mathcal{A}}^{\mathsf{IBBE}}$. Specifically, $\mathcal{A}$ has the secret key $\mathsf{sk}_{\mathsf{id}'}$ for $\mathsf{id}' \in \mathcal{S}'$, then it can get g^{η} from $(\mathsf{d}_0, \mathsf{d}_1, \mathsf{d}_2)$ of the random $\mathsf{dk}_{\mathsf{id}\to\mathcal{S}'|\mathbb{A}} = (\mathsf{d}_0, \mathsf{d}_1, \mathsf{d}_2, \mathsf{d}_{3,q}, \mathsf{d}_{4,q}, \mathsf{d}_{5,q})$. The only difference between the random one and the well-formed one is $\mathsf{d}_{3,q}$. In more detail, the real decryption key $\mathsf{sk}_{\mathsf{id}}$ should be utilized for the well-formed delegation key generation while the random delegation key should be created with the random $\mathcal{R}$ that replaces $\mathsf{sk}_{\mathsf{id}}$. Note that in this case $\mathcal{A}$ has no secret key $\mathsf{sk}_{\mathsf{id}}$ for $\mathsf{id} \in \mathcal{S}^*$. If the random delegation key can be discerned from a well-formed one, then it means that $\mathsf{sk}_{\mathsf{id}}$ and $\mathcal{R}$ can be distinguished. For ease of clearly proving, we let $\mathcal{A}_{\mathsf{dif}}(E, F)$ denote the event that $\mathcal{A}$ can differentiate E from F. Since $\mathcal{A}$ can compute $g^{f(\alpha)}$ with GDDHE instance, then we can derive the following result: $\mathcal{A}_{\mathsf{dif}}(\mathsf{sk}_{\mathsf{id}}, \mathcal{R}) \Rightarrow \mathcal{A}_{\mathsf{dif}}(e(\mathsf{sk}_{\mathsf{id}}, g^{f(\alpha)}), e(\mathcal{R}, g^{f(\alpha)})) \Rightarrow \mathcal{A}_{\mathsf{dif}}(e(\rho,\mu), e(\mathcal{R}, g^{f(\alpha)}))$. Here we assume $\mathcal{R} = \mu^{sg(\alpha)} \cdot \mu^t$ for unknown $t \in \mathbb{Z}_p$, we can proceed the above equation as follows: $\mathcal{A}_{\mathsf{dif}}(e(\rho,\mu), e(\mathcal{R}, g^{f(\alpha)})) \Rightarrow \mathcal{A}_{\mathsf{dif}}(e(\rho,\mu), e(\mu^{sg(\alpha)}\mu^t, g^{f(\alpha)})) \Rightarrow \mathcal{A}_{\mathsf{dif}}(e(\rho,\mu)e(g,\mu^{sg(\alpha)f(\alpha)})^{-1}, e(\mu^t, g^{f(\alpha)})) \Rightarrow \mathcal{A}_{\mathsf{dif}}(e(g^{g(\alpha)}, \mu^{sf(\alpha)})^{-1}, e(\rho,\mu)^{-1}e(\mu^t, g^{f(\alpha)})) \Rightarrow \mathcal{A}_{\mathsf{dif}}(e(g^s, \mu^{f(\alpha)})^{-1}, e(g,\mu)^{(tf(\alpha)-\varphi}/g(\alpha)))$ $\Rightarrow \mathcal{A}_{\mathsf{dif}}(e(g,\mu)^{-sf(\alpha)}, e(g,\mu)^{t'})$. Here $t' = (tf(\alpha) - \varphi)/g(\alpha)$.

From the above illustrations, we can learn that $\mathcal{A}$ knows $(g, g^{\alpha}, \ldots, g^{\alpha^{t-1}}, g^{\alpha f(\alpha)}, g^{s\alpha f(\alpha)}, \mu, \mu^{\alpha}, \ldots, \mu^{\alpha^n}, \mu^{1/g(\alpha)}, \mu^{f(\alpha)/g(\alpha)}, \mu^{sg(\alpha)}), \mathcal{A}_{\mathsf{dif}}(e(g,\mu)^{-sf(\alpha)}, e(g,\mu)^{t'})$ means that $\mathcal{A}$ can solve the hardness problem of GDDHE by differentiating $e(g,\mu)^{t'}$ from $e(g,\mu)^{-sf(\alpha)}$. That is to say, the advantage

TABLE 4: Computation cost comparisons of our SCPA with related schemes

Scheme	Costs at authority side		Costs at client side			Costs at server side	
	Setup	Registration	Encrypt	Delegation	Decrypt-I ‖ Decrypt-II	Re-Encryption	Sanitization
DZQ+ [28]	$(3n+3)e_0+p$	e_0	$(3\ell+3m+5)e_0+e_1$	$(\ell+6m+3)e_0$	$2p+(\ell+1)e_0+e_1 \parallel 3p+(\ell+1)e_0+e_1$	$(3m+1)p+(3m+\ell)e_0+e_1$	NA
XJW+ [23]	$(4n+3)e_0+p$	e_0	$2\ell+4e_0+e_1+p$	$(\ell+6)e_0+e_1$	$2p+(\ell+1)e_0+e_1 \parallel 3p+(\ell+1)e_0+e_1$	$2p+\ell e_0+e_1$	NA
WC [33]	$(2n+3)e_0+p$	e_0	$8e_0+4e_1+6p$	NA	$2p+(\ell+1)e_0+e_1 \parallel$ NA	NA	$3e_0$
Ours	$(3n+3)e_0+p$	e_0	$(4\ell+12m+12)e_0+2e_1+6p$	$(\ell+6m+3)e_0$	$2p+(\ell+1)e_0+e_1 \parallel 3p+(\ell+1)e_0+e_1$	$(3m+1)p+(3m+\ell)e_0+e_1$	$(14m+16)e_0+4e_1+6p$

TABLE 5: Storage cost comparisons of our SCPA with related schemes

Scheme	Costs at client side			Costs at server side	
	pp storage	sk storage	dk storage	Original ct storage	Transformed ct storage
XJW+ [23]	$(3n+5)\|\mathbb{G}_0\|+\|\mathbb{G}_2\|$	$\|\mathbb{G}_0\|$	$4\|\mathbb{G}_0\|$	$(2\ell+3)\|\mathbb{G}_0\|+\|\mathbb{G}_2\|$	$4\|\mathbb{G}_0\|+\|\mathbb{G}_2\|$
DZQ+ [28]	$(3n+5)\|\mathbb{G}_0\|+\|\mathbb{G}_2\|$	$\|\mathbb{G}_0\|$	$(3m+3)\|\mathbb{G}_0\|$	$3\|\mathbb{G}_0\|+2m\|\mathbb{G}_1\|+\|\mathbb{G}_2\|$	$4\|\mathbb{G}_0\|+\|\mathbb{G}_2\|$
WC [33]	$(2n+4)\|\mathbb{G}_0\|+\|\mathbb{G}_2\|$	$\|\mathbb{G}_0\|$	NA	$4\|\mathbb{G}_0\|+2\|\mathbb{G}_2\|$	NA
Ours	$(3n+5)\|\mathbb{G}_0\|+\|\mathbb{G}_2\|$	$\|\mathbb{G}_0\|$	$(3m+3)\|\mathbb{G}_0\|$	$12\|\mathbb{G}_0\|+8m\|\mathbb{G}_1\|+4\|\mathbb{G}_2\|+(2m+2)\|\mathbb{Z}_p\|$	$3\|\mathbb{G}_0\|+\|\mathbb{G}_1\|+\|\mathbb{G}_2\|$

of $\mathcal{A}$ in distinguishing the random delegation key from a well-formed one is no more than $\mathcal{A}dv_{\mathcal{A}}^{\mathsf{GDDHE}}$. Since the IBBE [7] has been proven secure ($\mathcal{A}dv_{\mathcal{A}}^{\mathsf{GDDHE}} = \mathcal{A}dv_{\mathcal{A}}^{\mathsf{IBBE\text{-}security}}$), it indicates that the advantage of $\mathcal{A}$ in distinguishing the random delegation key from a well-formed one is negligible. Hence, any adversary cannot breach the security of our SCPA with some advantage.

6.3 Proofs of No-read and No-write Rules for Our SCPA

Theorem 3. The no-read rule can be satisfied with our SCPA if the NIZK system used in our SCPA is zero-knowledgeable and our SCPA is IND-CPA secure.

Theorem 4. The no-write rule can be satisfied with our SCPA if the NIZK system used in our SCPA is zero-knowledge, our SCPA is IND-CPA secure and the signature is unforgeable.

Proof: For the security proofs of ***Theorems* 3 & 4**, here we omit them due to the limited space. The readers can refer to the **Supplementary Material** for more details.

7 Performance Evaluation

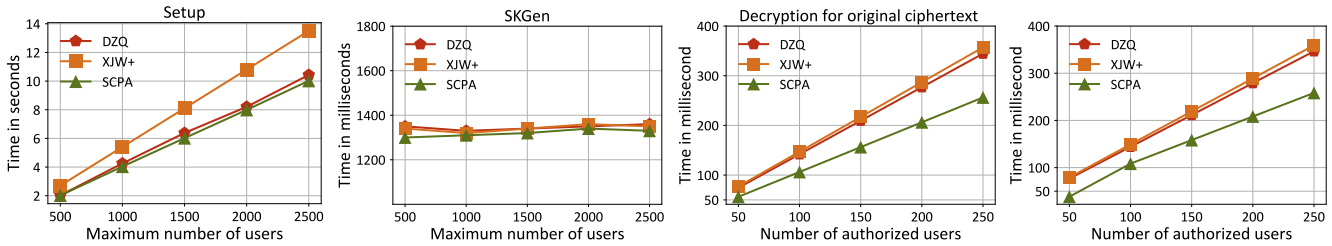
This section showcases the performance evaluation via specific theoretical analysis and experimental evaluation to indicate the practicability of our SCPA.

7.1 Theoretical Analysis

TABLEs 4 and 5 present a comprehensive analysis of computation and storage overheads across various works [23], [28], [33] in the context of dynamic multi-recipient data sharing due to the fact that these works are the forefront and state-of-the-art solutions. The tables contain details about various time-consuming calculations (i.e., exponentiation operation and bilinear pairings) and storage sizes involved in the comparisons. In TABLE 4, the running time required for certain cryptographic operations is provided, *e.g.*, p denotes the time taken to perform one bilinear pairing operation; e_0 indicates the time taken for a single exponentiation computation in $\mathbb{G}_0$; e_1 means the time taken for a single exponentiation computation in $\mathbb{G}_2$; Let n, ℓ, m be the maximum number of recipients in the system, the number of authorized recipients and the number of conditions specified in ciphertext, respectively. In TABLE 5, the storage costs of single-group elements in different groups are presented. *e.g.*, $|\mathbb{G}_0|$, $|\mathbb{G}_1|$, $\mathbb{G}_2$ denote the storage cost of a single group element in respective $\mathbb{G}_0$, $\mathbb{G}_1$, $\mathbb{G}_2$. Here, "NA" implies that the function of the scheme is not applicable.

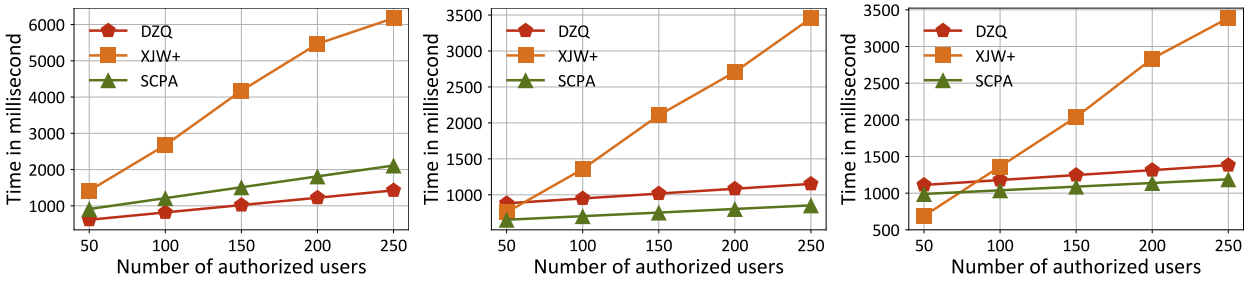
As indicated in TABLE 4, we readily observe that the computation costs of running the Setup algorithm to initialize system parameters in each scheme increase linearly with the maximum number of system users. Furthermore, the computation costs of performing the Registration algorithm to create decryption keys in each scheme remain constant. We also summarize that the calculation costs of executing the Encryption in DZQ+ [28] and ours are incremental with the number of conditions and authorized users, and the computation cost of implementing decryption of all schemes only grows linearly with the number of authorized users (regardless of the number of conditions). It is worth noting that the TABLE 4 only presents the original calculation costs of executing the Encryption, Delegation and Re-encryption of both XJW+ [23] and WC [33], since XJW+ [23], WC [33] fails to support the conditional sharing functionality. If realizing the same property as that of DZQ+ [28] and ours, then in fact the calculation costs of executing the Encryption, Delegation and Re-encryption of both XJW+ [23] and WC [33] also increase linearly with the number of conditions or authorized users. From TABLE 4, we can also observe that the computation cost of implementing each algorithm of DZQ+ [28] except for Encryption is almost identical to that of our SCPA since our SCPA enhances the functionality of DZQ+ [28] without incurring any additional computation overheads. Besides, the encryption computation cost in our SCPA is higher than that of DZQ+ [28] since the NIZK proof to be performed is used for realizing sanitization.

As evidenced by TABLE 5, it becomes apparent that in each comparison scheme, the required storage costs for running the Setup algorithm to produce public parameters follow linear relationships with the maximum number of system users. Similarly, the needed storage costs for execut-



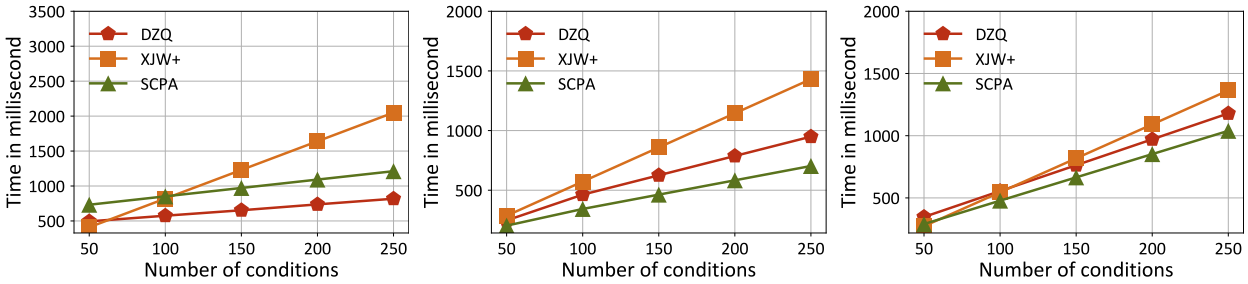
(a) Setup (b) SKGen (c) Dec-I (d) Dec-II

Fig. 2: Running time for Setup, SKGen, Decryption algorithms



(a) Enc (b) Del (c) Re-Enc

Fig. 3: Running time for Encryption, Delegation and Re-encryption algorithms with the number of authorized users



(a) Enc (b) Del (c) Re-Enc

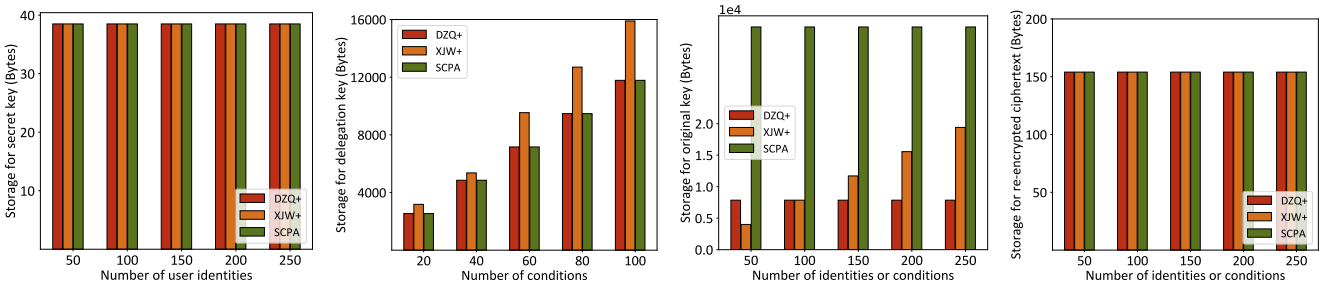
Fig. 4: Running time for Encryption, Delegation and Re-encryption algorithms with the number of conditions

ing Secret Key Registration (SKGen) to create the decryption key remain constant across all works. However, it's worth noting that the storage overhead of generating the delegation key in DZQ+ [28] and our SCPA grows linearly with the number of conditions. A few notable observations can be made regarding the required storage costs for performing Encryption and Re-encryption. The original-ciphertext storage costs in all schemes, except in WC [33], either grow with the number of (system) recipients or increase with the number of conditions. On the other hand, the transformed-ciphertext storage costs remain constant across all works. In general, smaller storage costs for storing ciphertext and decryption keys imply more efficient decryption. Notably, in our SCPA, we have achieved lower storage costs for storing decryption keys and transformed-ciphertext, resulting in a more efficient decryption process.

In summary, our SCPA offers favorable calculation and storage costs compared to other works. From TABLE 1, it is also evident that our SCPA possesses some more desired properties than the other works. In other words, our SCPA achieves satisfactory performance while incorporating the desired features.

7.2 Experimental Analysis

In our experimental simulations, we have opted to include the works [23], [28] for comparative analysis with our SCPA. This selection is based on the fact that these experimental endeavors share a common foundation of dynamic multiple-recipient data sharing and aim to achieve functionalities closely resembling those of our SCPA. The experimental performance evaluation utilized Python 3.6.13 and relied on Charm 0.43, PBC-0.5.14 library, and OpenSSL-1.1.1. Simulations were executed on a laptop equipped with an Intel Core i9-9900K CPU @ 3.6GHz * 16 and 32GB RAM, running 64-bit Ubuntu 18.04.5 LTS to represent cloud servers. Additionally, a Raspberry Pi 4 Model B device with Broadcom BCM 2711, Quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz and 2GB RAM, running Raspbian, was used to simulate a mobile user. For the implementation, 128-bit AES keys were employed to encode real data, specifically medical images from https://www.smir.ch/BRATS/Start2015, using a modified AES algorithm [35].



(a) Secret key (sk) size (b) Delegation key (dk) size (c) Original ciphertext (ct) size (d) Re-encrypted ciphertext size

Fig. 5: Storage consumption of secret key (sk), delegation key (dk) and ciphertext (ct)

Fig. 2 exhibits the running time comparisons for Setup, SKGen (secret key registration) & Decryption algorithms of [23], [28] and ours. As presented in Fig. 2(a), we can find that the running time for Setup algorithm of each work grows linearly with the maximum number of system users. From Fig. 2(b), it can be seen that the running time for SKGen algorithm of each scheme is almost smaller-constant. Regarding the running time of Decryption on the original ciphertext or re-encrypted ciphertext (i.e., transformed ciphertext) shown in Figs. 2(c) & 2(d), it is straightforward to see that the running time for decrypting the ciphertext also increase linearly with the number of authorized users. In addition, although TABLE 4 shows that the computation cost for executing Setup, SKGen (secret key registration) & Decryption algorithms of each work is almost the same, we can also find from Fig. 2 that our SCPA has slightly better performance than the other works since the Type-III pairing curve used in our SCPA is recognized to be faster and safety than Type-I pairing curve utilized in other works [36].

Figs. 3 & 4 illustrate the running time of Encryption, Delegation, Re-encryption algorithms with respect to the number of authorized users and the number of conditions, respectively. As shown in Fig. 3(a), the running time of executing the Encryption algorithm in each work is found to increase linearly with the number of authorized recipients and Fig. 4(a) demonstrates that the running time of executing the Encryption algorithm in each scheme increases linearly with the number of conditions. For Figs. 3(b) & 4(b) and Figs. 3(c) & 4(c), the same conclusion as that depicted in Fig. 3(a) & 4(a) can be obtained, *i.e.*, the running time of executing Re-encryption or delegation algorithms in each work follows linear relationships with the number of authorized users or conditions. Besides, it can be also found that our SCPA has slightly more satisfactory computation efficiency than the other works in terms of Del and Re-Enc algorithms due to the fast Type-I pairing curve utilized.

Fig. 5 presents the storage cost comparisons of conducting SKGen, Delegation, Encryption and Re-encryption to produce corresponding sk, dk, original ct and re-encrypted ct. From Fig. 5(a), it's concluded that our SCPA requires the same storage resource for storing the secret key as the DZQ+ [28] and XJW+ [23]. From Fig. 5(b), we can see that our SCPA requires the same storage costs for keeping the delegation key and original ciphertext as DZQ+ [28] but requires fewer storage costs than XJW+ [23]. From Fig. 5(c), our storage costs in our SCPA are higher than that in the other two schemes since the NIZK proof is deployed in the encryption algorithm of our SCPA. As revealed from Fig. 5(d), it is easily concluded that our SCPA requires the same storage resource for storing re-encrypted ciphertext as the DZQ+ [28] and XJW+ [23]. In addition, from Fig. 5, we observe that the storage costs for executing SKGen and Re-encryption to store sk and re-encrypted ct in all works are always constant regardless of the number of users and conditions, while the storage costs for executing Delegation and Encryption to store dk and original ct increase with the number of users or conditions.

To summarize, since our SCPA has relatively lower costs regardless of computation and storage costs and enables more satisfactory features, our SCPA is more appropriate for real-world applications.

8 Conclusion

In this paper, we for the first time suggested a sanitizable cross-domain access control scheme with policy-driven dynamic authorization (SCPA), which solves the practical issues the existing data sharing schemes rarely considered, including illegal authorization caused by malicious behavior, cross-domain read-and-write permissions for participants, inflexible strategies for dynamical ciphertext sharing with multiple recipients. Apart from allowing access controls for no-read and no-write rules for regulating the data that the sender is allowed to send and which that the recipients are allowed to receive, our SCPA also enables dynamic sharing of a subset of data ciphertext with additional receivers beyond those originally authorized. Besides, we presented comprehensively strict security proofs to demonstrate the security and featured properties of our SCPA. The practicability and effectiveness of our SCPA are also showcased via the performance evaluation. In future work, our proposal will be extended to design such a scheme that assumes the sanitizer to be a malicious entity instead of a fully trusted one. To address this issue, a potential solution is to exploit the Trusted Execution Environment (TEE) tools to replace the execution of the Sanitization algorithm, protecting it from tampering with potentially malicious sanitizers. Additionally, we are also interested in exploring more practical functionalities based on our SCPA, such as privilege revocation, etc. The potential solution to handle this is to introduce a revocation list to ensure the invalid of revocable users.

ACKNOWLEDGEMENT

This research / project is supported by the National Research Foundation, Singapore, and Cyber Security Agency of Singapore under its National Cybersecurity R&D Programme and CyberSG R&D Cyber Research Programme Office. Any opinions, findings and conclusions or recommendations expressed in these materials are those of the author(s) and do not reflect the views of National Research Foundation, Singapore, Cyber Security Agency of Singapore as well as CyberSG R&D Programme Office, Singapore.

REFERENCES

[1] Y. Guo, C. Zhang, C. Wang, et al., "Towards public verifiable and forward-privacy encrypted search by using blockchain", *IEEE TDSC,* vol. 20, no. 3, pp. 2111-2126, 2023.
[2] C. Fang, N. Nazari, B. Omidi, et al., "Heteroscore: Evaluating and mitigating cloud security threats brought by heterogeneity", *Network and Distributed System Security (NDSS) Symposium*, 2023.
[3] A. Mehta, M. Alzayat, R. D. Viti, et al., "Pacer: Comprehensive Network Side-Channel Mitigation in the Cloud", *USENIX Security 2023*, pp. 2819–2838, 2023.
[4] S. Xu, J. Ning, Y. Li, et al., "A secure EMR sharing system with tamper resistance and expressive access control", *IEEE TDSC,* vol. 20, no. 12023, pp. 53-67, 2023.
[5] C. Lai, Y. Ma, R. Lu, et al., "A novel authentication scheme supporting multiple user access for 5G and beyond", *IEEE TDSC,* vol. 20, no. 4, pp. 2970-2987, 2023.
[6] Q. Mei, M. Yang, J. Chen, et al., "Expressive Data Sharing and Self-Controlled Fine-Grained Data Deletion in Cloud-Assisted IoT", *IEEE TDSC,* vo. 20, no. 3, pp. 2625-2640, 2023.
[7] R. Sakai, J. Furukawa, "Identity-based broadcast encryption", *Cryptology ePrint Archive,* 2007.
[8] H. Wee, "Optimal broadcast encryption and CP-ABE from evasive lattice assumptions", *EUROCRYPT'22*, LNSC 13276, pp. 217-241, 2022.
[9] C. Ge, Z. Liu, W. Susilo, et al., "Attribute-Based Encryption with Reliable Outsourced Decryption in Cloud Computing Using Smart Contract", *IEEE TDSC,* DOI: 10.1109/TDSC.2023.3265932, 2023.
[10] S.Agrawal, M. Chase, "FAME: fast attribute-based message encryption", *CCS'17*, pp. 665-682, 2017.
[11] M. Green, G. Ateniese, "Identity-based proxy re-encryption", *ACNS'07*, LNSC 4521, pp. 288-306, 2007.
[12] L. Zhang, H. Ma, Z. Liu, et al., "'Security analysis and improvement of a collusion-resistant identity-based proxy re-encryption scheme", *BWCCA–2016*, vol. 2, pp. 839-846, 2017.
[13] J. Shao, Z. Cao, "Multi-use unidirectional identity-based proxy re-encryption from hierarchical identity-based encryption", *Information Sciences,* vol. 206, pp. 83-95, 2012.
[14] C. Ge, Z. Liu, J. Xia, et al., "Revocable identity-based broadcast proxy re-encryption for data sharing in clouds", *IEEE TDSC,* vol. 18, no. 3, pp. 1214-1226, 2019.
[15] J. Sun, G. Xu, T. Zhang, et al., "Verifiable, Fair and Privacy-Preserving Broadcast Authorization for Flexible Data Sharing in Clouds", *IEEE TIFS*, vol. 18, pp. 683-698, 2022.
[16] H. Deng, Z. Qin, Q. Wu, et al., "Identity-based encryption transformation for flexible sharing of encrypted data in public cloud", *IEEE TIFS*, vol. 15, pp. 3168-3180, 2020.
[17] J. Weng, R. H. Deng, X. Ding, et al., "Conditional proxy re-encryption secure against chosen-ciphertext attack", *CCS 2009,* pp. 322-332, 2009.
[18] S. S. Vivek, D. S. Sharmila, V. Radhakishan, et al., "Conditional proxy re-encryption-a more efficient construction", *CNSA'11*, LNSC 196, pp. 502-512, 2011.
[19] C. K. Chu, J. Weng, S. S. Chow, et al., "Conditional proxy broadcast re-encryption", *Information Security and Privacy: 14th Australasian Conference, ACISP 2009*, LNSC 5594, pp. 327-342, 2009.
[20] J. Shao, G. Wei, Y. Ling, et al., "Identity-based conditional proxy re-encryption", *IEEE ICC*, IEEE, pp. 1-5, 2011.
[21] S. Yin, H. Li, L. Teng, "A Novel Proxy Re-encryption Scheme Based on Identity Property and Stateless Broadcast Encryption Under Cloud Environment", *Int. J. Netw. Secur.*, vol. 21, no. 5, pp. 797-803, 2019.
[22] K. Liang, C. K. Chu, X. Tan, et al., "Chosen-ciphertext secure multi-hop identity-based conditional proxy re-encryption with constant-size ciphertexts", *TCS,* vol. 539, pp. 87-105, 2014.
[23] P. Xu, T. Jiao, Q. Wu, et al., "Conditional identity-based broadcast proxy re-encryption and its application to cloud email", *IEEE Transactions on Computers,* vol. 65, no. 1, pp. 66-79, 2015.
[24] L. M. Fang, J. D. Wang, C. Ge, et al., "Fuzzy conditional proxy re-encryption", *SCIS,* vol. 56, pp. 1-13, 2013.
[25] Y. Yang, H. Lu, J. Weng, et al., "Fine-grained conditional proxy re-encryption and application", *ProvSec,* LNSC 8782, pp. 206-222, 2014.
[26] C. Ge, W. Susilo, J. Wang, et al., "Identity-based conditional proxy re-encryption with fine grain policy", *Computer Standards & Interfaces,* vol. 52, pp. 1-9, 2017.
[27] C. Ge, L. Zhou, J. Xia, et al., "A secure fine-grained identity-based proxy broadcast re-encryption scheme for micro-video subscribing system in clouds", *SocialSec 2019,* pp. 139-151, 2019.
[28] H. Deng, J. Zhang, Z. Qin, et al., "Policy-based broadcast access authorization for flexible data sharing in clouds", *IEEE TDSC*, vol. 19, no. 5, pp. 3024-3037, 2021.
[29] Q. Huang, Y. Yang, J. Fu, "PRECISE: Identity-based private data sharing with conditional proxy re-encryption in online social networks", *FGCS,* vol. 86, pp. 1523-1533, 2018.
[30] I. Damgard, H. Haagh, C. Orlandi, "Access control encryption: Enforcing information flow with cryptography", *Theory of Cryptography (TCC'16)*, LNSC 9986, pp. 547-576, 2016.
[31] S. Kim, D. J. Wu, "Access control encryption for general policies from standard assumptions", *ASIACRYPT'17*, LNSC 10624, pp. 471-501, 2017.
[32] G. Fuchsbauer, R. Gay, L. Kowalczyk, et al., "Access control encryption for equality, comparison, and more", *PKC'17*, LNSC 10175, pp. 88-118, 2017.
[33] X. Wang, S. S. Chow, "Cross-domain access control encryption: arbitrary-policy, constant-size, efficient", *IEEE S & P'21*, IEEE, pp. 748-761, 2021.
[34] M. Abe, J. Groth, M. Ohkubo, et al., "Unified, minimal and selectively randomizable structure-preserving signatures", *TCC*, pp. 688-712, 2014.
[35] M. Zeghid, M. Machhout, L. Khriji, et al., "A modified AES based algorithm for image encryption", *IJCSE,* vol. 1, no. 1, 70-75, 2007, (https://github.com/JHUISI/charm).
[36] S. Agrawal, M. Chase, "FAME: fast attribute-based message encryption", *CCS 2017*, pp. 665-682, 2017.
[37] F. Bao, R. H. Deng, H. Zhu. "Variations of Diffie-Hellman problem", *International Conference on Information and Communications Security*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2003: 301-312.
[38] J. Groth. "On the size of pairing-based non-interactive arguments", *Advances in Cryptology–EUROCRYPT 2016,* Springer Berlin Heidelberg, 2016: 305-326.
[39] Y. Miao, F. Li, X. Li, et al., "Verifiable outsourced attribute-based encryption scheme for cloud-assisted mobile e-health system", *IEEE TDSC*, DOI: 10.1109/TDSC.2023.3292129, 2023.
[40] Q. Huang, C. Wang, L. Chen, "Secure and fine-grained flow control for subscription-based data services in cloud-edge computing", *IEEE TSC*, vol. 16, no. 3, pp. 2165-2177, 2023.
[41] G. Ateniese, D. Francati, D. Nunez, et al., "Match me if you can: matchmaking encryption and its applications", *Journal of Cryptology,* vol. 34, pp. 1-50, 2021.
[42] Q. Huang, L. Chen, C. Wang, "A parallel secure flow control framework for private data sharing in mobile edge cloud", *IEEE TPDS,* vol. 33, no. 12, pp. 4638-4653, 2022.

Jianfei Sun is currently a research fellow at the School of Computer Science and Engineering, Nanyang Technological University. He will join SMU under the supervision of Prof. Robert Deng and Prof. Guomin Yang after Nov 2023. He has published many papers on IEEE TDSC, IEEE TIFS, IEEE TII, IEEE TCC, IEEE TVT, IEEE IoTJ, Inf. Sci, IEEE Systems, etc. His research interests include applied cryptography, network security, and IoT security. He is currently an Associate Editor of IEEE Transactions on Industrial Informatics.

Guowen Xu is currently a Professor at School of Computer Science and Engineering, University of Electronic Science and Technology of China. He received his Ph.D. degree in 2020 from the University of Electronic Science and Technology of China. He has published a wealth of papers in reputable conferences/journals, including ACM CCS, IEEE S&P, NeurIPS, ECCV, IEEE TIFS, TDSC, ASIACCS, ACSAC, ESORICS, etc. He is the recipient of the Best Paper Award of the 26th IEEE International Conference on Parallel and Distributed Systems (ICPADS 2020), the Best Student Paper Award of the Sichuan Province Computer Federation (SCF 2019), the Student Conference Award of IEEE International Conference on Computer Communications (INFOCOM 2020), and the Distinguished Reviewer of ACM Transactions on the Web. His research interests include applied cryptography and privacy-preserving Deep Learning.

Hongwei Li (M'12-SM'18-F'23) is currently the Head and a Professor at Department of Information Security, School of Computer Science and Engineering, University of Electronic Science and Technology of China. He received the Ph.D. degree from University of Electronic Science and Technology of China in June 2008. He worked as a Postdoctoral Fellow at the University of Waterloo from October 2011 to October 2012. His research interests include network security and applied cryptography. He is a Fellow of IEEE, the Distinguished Lecturer of IEEE Vehicular Technology Society.

Tianwei Zhang is an assistant professor at School of Computer Science and Engineering, at Nanyang Technological University. His research focuses on computer system security. He is particularly interested in security threats and defenses in machine learning systems, autonomous systems, computer architecture and distributed systems. He received his Bachelor's degree at Peking University in 2011, and the Ph.D degree at Princeton University in 2017. He is serving on the editorial boards of IEEE Transactions on Circuits and Systems for Video Technology, ACM Transactions on Sensor Networks.

Cong Wu is currently a research fellow at School of Computer Science and Engineering, Nanyang Technological University, Singapore. He received Ph.D. degree at School of Cyber Science and Engineering, Wuhan University in 2022. His research interests include security of intelligent systems and Web3 security. His research outcomes have appeared in USENIX Security, ACM CCS, IEEE TDSC, TIFS.

Xuehuan Yang received his Bachelor's degree from Nanyang Technological University. Now he was a Ph.D in School of Computer Science and Engineering, at Nanyang Technological University. His research focuses on software engineering and secure autonomous vehicle technology.

Robert H. Deng (F'16) is AXA Chair Professor of Cybersecurity, Director of the Secure Mobile Centre, and Deputy Dean for Faculty & Research, School of Computing and Information Systems, Singapore Management University (SMU). His research interests are in the areas of data security and privacy, network security, and applied cryptography. He received the Outstanding University Researcher Award from National University of Singapore, Lee Kuan Yew Fellowship for Research Excellence from SMU, and Asia-Pacific Information Security Leadership Achievements Community Service Star from International Information Systems Security Certification Consortium. He serves/served on the editorial boards of ACM Transactions on Privacy and Security, IEEE Security & Privacy, IEEE Transactions on Dependable and Secure Computing, IEEE Transactions on Information Forensics and Security, Journal of Computer Science and Technology, and Steering Committee Chair of the ACM Asia Conference on Computer and Communications Security. He is a Fellow of IEEE and Fellow of Academy of Engineering Singapore.