ESMAC: Efficient and Secure Multi-Owner Access Control With TEE in Multi-Level Data Processing

Dan Liu, Zheng Yan, *Senior Member, IEEE*, Wenxiu Ding, *Member, IEEE*, Yuxuan Cai, Yaxing Chen, and Zhiguo Wan, *Member, IEEE*

Abstract—Traditional data access control schemes only prevent unauthorized access to private data with a single owner. They are not suitable for application in a Multi-Level Data Processing (MLDP) scenario, where data are processed by a series of parties who also insert new data. Hence, the accumulated dataset should be protected through access control handled by hierarchically-structured parties who are at least partial data owners in MLDP. Existing multi-owner access control schemes mainly focus on controlling access to co-owned data of multiple entities with the equal ownership, but seldom investigates how to apply access control in MLDP. In this paper, we base the off-the-shelf Trusted Execution Environment (TEE), Intel SGX, to propose an Efficient and Secure Multi-owner Access Control scheme (ESMAC) for access authorization in MLDP. Moreover, to prevent unauthorized data disclosure by non-root data owners aiming to gain extra profits, we further introduce undercover polices to supervise their behaviors. Specifically, we design a data protection scheme based on game theory to decide the payoffs and punishments of honest and dishonest data owners, which motivates data owners to behave honestly when claiming ownership over data. Through comprehensive security analysis and performance evaluation, we demonstrate ESMAC's security and effectiveness.

Index Terms—SGX, cloud computing, multi-owner access control, multi-level data processing, game theory

1 Introduction

With increasing collaboration among data processors, data that is processed by multiple parties tend to be jointly owned by them. Each processor contributes different parts of a final data processing result, which mostly involves accumulating newly inserted information to improve its freshness and value. Herein, we consider a Multi-Level Data Processing (MLDP) scenario, in which a number of data owners provide their private data during different processing phases by conducting further data analysis. In more detail, a first-level data owner has authority over its own data (e.g., medical data) and can permit one or more second-level owners (i.e., its authorized users) to perform deep analysis on the data with newly inserted data. Furthermore, the processing results may also be outsourced to authorized users in the next level to facilitate deep processing. In such a case, data undergo several processing phases, resulting in a dataset with in-depth information. For example, Health Clinical Center (HCC) A holds some medical data and trains to produce a model $\mathbb{A}$. HCC B may share its own dataset to optimize the accuracy of this model and obtain a new model $\mathbb{B}$. If HCC C wants to access the model $\mathbb{B}$, this access should be granted by both HCC A and B. In addition, if HCC $\tilde{B}$ has some conflicts of interest with HCC B, it may request the training model from HCC A rather than HCC B and then optimize it through its own dataset to gain a new model $\mathbb{B}$. Some entities may further insert their own datasets and improve the training model for their benefit. In such an MLDP case, access to the training model in each level should be tightly controlled, because all involved owners contribute their personal data to the final training model.

However, the fact that data are contributed by multiple owners and processed through several phases in MLDP makes access control extremely complicated. For example, a third-level data processing result is gained by analyzing the data from all related data owners in the upper three levels. Hence, accessing the processing results should require authorization from all owners involved. However, the hierarchical structure of both the data ownership and the data in question following multi-phase processing introduces several challenges associated with flexibly controlling the access.

First, the hierarchical ownership structure makes key management more difficult for data owners aiming to control

- *Dan Liu and Zheng Yan are with the State Key Laboratory on Integrated Services Networks, School of Cyber Engineering, Xidian University, Xi'an 710071, China. E-mail: 15596173220@163.com, zyan@xidian.edu.cn.*
- *Wenxiu Ding and Yuxuan Cai are with the School of Cyber Engineering, Xidian University, Xi'an 710071, China. E-mail: wxding@xidian.edu.cn, 21151213618@stu.xidian.edu.cn.*
- *Yaxing Chen is with the School of Computer Science, Northwestern Polytechnical University, Xi'an, Shaanxi 710072, China. E-mail: yxchen@nwpu.edu.cn.*
- *Zhiguo Wan is with the Zhejiang Laboratory, Hangzhou, Zhejiang 311121, China. E-mail: wanzhiguo@zhejianglab.com.*

Manuscript received 31 December 2021; revised 6 October 2022; accepted 15 October 2022. Date of publication 20 October 2022; date of current version 1 September 2023.
The work was supported in part by the National Natural Science Foundation of China under Grants 61802293 and 62072351, in part by the Fundamental Research Funds for the Central Universities under Grant XJS211503, in part by the National Postdoctoral Program for Innovative Talents under Grant BX20180238, in part by the Open Research Project of Zhejiang Lab under Grant 2021PD0AB01, and in part by the 111 Project under Grant B16037. (Corresponding author: Wenxiu Ding.)
Digital Object Identifier no. 10.1109/TDSC.2022.3215977

data access in MLDP. Traditional data access control schemes are based on cryptographic primitives [1], [2] (such as Attribute Based Encryption (ABE) [3], [4], Proxy Re-Encryption (PRE) [5], etc.), which depend on a single owner or authority issuing keys to authorized users. These existing schemes are not suitable for MLDP. Some efficient schemes [6], [7] offer flexible multi-user access control over encrypted data based on a Trusted Execution Environment (TEE). Intel Software Guard Extensions (SGX) [8] defines a protected memory region on the hardware referred to as an Enclave. While the Enclave has the full power to decrypt and access the data, it is also vulnerable to attacks launched by a malicious host program, which poses a new threat to key management. Hence, for supporting data access control in MLDP, key management becomes a critical issue in TEE-based schemes.

Second, dynamic authorization managed by multiple owners is not easy to offer in the MLDP context. Since the data in each level are co-owned by all owners in the upper levels, data leakage in the current level would violate the privacy of related owners in the upper levels. Thomas et al. [9] demonstrated the inadvertent leakage of sensitive data caused by the absence of joint control over co-owned data. Hence, the authorization jointly managed by multiple owners becomes a significant issue. While some works in the field of online social networks have attempted to solve this issue [10], [11], [12], [13], [14], [15], [16], [17], they concentrate on coordinating access control policies over plaintext among multiple owners. Therefore, they cannot be applied to controlling access over co-owned data in MLDP. Moreover, the ownership changes dynamically in MLDP, which increases the difficulty of multi-owner authorization.

Third, data leakage caused by non-root data owners (i.e., the owners that are not located in the first level of the hierarchical ownership structure) is an inevitable obstacle in multi-level data processing, which could damage the benefits of the data owners in upper levels. Each non-root data owner who also actually consumes the data from the owners in the upper levels, should gain access rights from all involved owners. If a non-root owner negates the contribution of upper-level owners and claims full ownership of the co-owned or partially owned data, this will definitely violate other owners' benefits. However, the question of how to effectively avert dishonest ownership claims of non-root data owners remains an open one which seriously affects the trustworthiness of data access control in MLDP.

In this paper, we employ Intel SGX to design an efficient and secure multi-owner access control scheme, named ESMAC, to control access to co-owned data in MLDP. ESMAC contains three sub-schemes: namely, a Key Management (KM) scheme, Multi-Owner Access Control (MOAC) scheme, and Data Protection (DP) scheme.

In order to support the hierarchical structure of ownerships, the KM scheme distinguishes the authority of root owners and non-root owners, then applies secret sharing to split the master key in order to pre-generate key components for users. For offering dynamic authorization managed by multiple owners in MLDP, the MOAC scheme utilizes SGX Enclaves to handle secure data analysis, but also takes into account data access control managed by the data owners inside SGX. It realizes secure integration of data analysis and data access. To prevent data leakage caused by non-root data owners, we further design the DP scheme by introducing undercover polices to monitor involved users. It also applies game theory to provide a rational setting of payoff and punishment, which can motivate non-root owners to act honestly and improve the reliability of the MOAC scheme.

Briefly, in ESMAC, data are outsourced to the cloud for analysis and sharing. When the SGX is deployed at the cloud server, the data can be decrypted and then processed in plaintext form after authorization has been obtained from the related owners. To guarantee the ownerships of the involved multiple owners, we employ shared secrets to allow these related data owners to jointly control data access and key revocation. Meanwhile, the Enclave can retrieve secret keys for authorized users based on the secret components issued by the related owners. To prevent owners from claiming ownership dishonestly, their behaviors are monitored and challenged by undercover polices based on game theory.

Accordingly, the main contributions of this work can be summarized as follows:

- We design a secure and efficient SGX-based KM scheme , which allows hierarchical data owners to jointly control their co-owned data in MLDP.
- Based on the KM scheme, we propose the MOAC scheme to support flexible access control over multi-level processed data; this approach offers secure and privacy-preserving data processing and guarantees the security of both the original and processed data in each data processing level.
- We further propose the DP scheme based on game theory to restrict data owners to follow the design of MOAC by introducing undercover police to monitor the behaviors of non-root data owners, as well as to provide proper settings of payoff and punishment for honest and dishonest owners respectively.
- We analyze the security and computational complexity of ESMAC and carry out experiments to demonstrate its efficiency through comparisons with existing works.

The remainder of this paper is organized as follows. Section 2 briefly introduces related works. Section 3 outlines the system model, security model, and design goals of ESMAC. Section 4 provides the design details of ESMAC, which is followed by security analysis and performance evaluation in Section 5. In Section 6, we conduct extensive experiments to test ESMAC's performance based on a proof-of-concept simulation. Finally, we conclude our work in Section 7.

2 Related Work

To support flexible access control over shared data, several schemes have been proposed based on various techniques. We divide the existing related works into two categories according to the number of data owners of shared data: traditional single-owner access control and multi-owner access control. In the former scheme, access policies are specified by a single data owner, while in the latter, access policies are jointly specified by multiple data owners. A data user

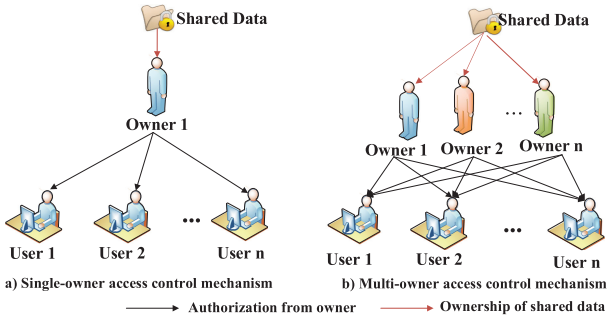


Fig. 1. Single-owner and Multi-owner access control scenarios.

can access the data if and only if it obtains authorizations from all related data owners. The difference between the two types of data access control is illustrated in Fig. 1.

2.1 Traditional Single-Owner Access Control

To date, numerous schemes have been proposed to realize single-owner access control over cloud data. There are two main varieties of schemes. The first type is cryptography-based solutions. For example, PRE is used to manage encrypted data sharing in the cloud by re-encrypting ciphertext for specified authorized users [5]; however, it does not support fine-grained access control. To solve this issue, Attribute-based Access Control (ABAC) [3], [4] is proposed, which associates access policy with the attributes of data users. Unfortunately, ABAC suffers from high computational overhead due to the costly bilinear pairing operations. Furthermore, both ABAC and PRE require the presence of a trusted entity to manage key generation. oGBAC[18] and Crypt-DAC[19] also achieve flexible access control by setting policies, but can only support single-owner data sharing.

The other research line is hardware-based solutions, where a TEE (e.g., SGX, Trustzone [20]) is adopted to facilitate data sharing [21], [22], [23], [24] with the expectation of simultaneously reducing computational overhead and enhancing security. Contiu et al. [21] took advantage of SGX to design a secure and efficient access control scheme based on identity-based broadcast encryption (IBBE). Compared with a traditional IBBE scheme [25], moves away from dependence on a trusted third-party and improves computation and storage efficiency. Similarly, Fan et al. [26] proposed a secure Ciphertext-Policy ABE (CP-ABE) scheme based on Trustzone for mobile applications to solve their data access problems. However, this method suffers from a serious performance problem caused by the limited computational resources on mobile devices. Ning et al. [27] combined SGX and CP-ABE to control cloud data sharing. Specifically, their scheme can control both data access and download requests, which enables it to resist Distributed Denial of Service (DDoS)/Economic Denial of Sustainability (EDos) attacks.

All the approaches outlined above concentrate on the access control over data storage, but ignore the control of access to processed data results. Moreover, although our previous works realized flexible access control over privacy-preserving data processing [28] [29], neither can support multi-level data processing.

2.2 Secure Multi-Owner Access Control

Multi-owner authorization was first designed in online social networks [9], [10], [11], [12], [13], [15], where shared data are published in users' personal spaces and controlled by direct system settings. Hence, such authorization schemes focus on how to coordinate the access policies of multiple data owners during data publishing. Due to the diversity of application scenarios, the schemes listed above are infeasible for authorizing access to co-owned data outsourced in the form of ciphertext.

Some schemes [1], [2], [30] have been proposed to support multi-authorization of outsourced data in ciphertext. Layouni et al. [2] proposed a pairing-based multi-owner access control scheme for private information retrieval, in which a record in a database is co-owned by a fixed number of parties. However, sensitive data are stored in a remote server that is responsible for data encryption, thus data are disclosed to the server and the privacy cannot be protected. Miao et al. [30] designed a keyword search scheme based on CP-ABE, in which data are jointly managed by multiple data owners, and a data user can only retrieve data by obtaining the authorization keys from all data owners. Similarly, Huang et al. [1] designed a fine-grained multi-owner access control (MOCC) scheme for conditional dissemination in the cloud computing context that employs attribute-based conditional PRE (CPRE), which allows each data owner to append its own access policies to the ciphertext. However, both [30] and [1] involve an abundance of complicated operations, resulting in low efficiency. To reduce overhead, QShield [7] was proposed to support secure SQL queries from multiple data users by applying SGX. However, this approach is incompatible with MLDP, which requires dynamic multi-level data processing and multi-owner access control.

In summary, no existing work has yet achieved multi-owner access control over hierarchical data processing results. ESMAC is thus the first scheme to offer secure and flexible data access in MLDP.

3 Problem Statement

3.1 System Model

To improve readability, Table 1 lists the main notations used in this paper. Before outlining the details of the system model, we first introduce an MLDP model, shown in Fig. 2. Notably, $O_{i,j}$ denotes the jth data owner in the ith level, while $U_{i,j,k}$ indicates the kth authorized user of $O_{i,j}$. In this figure, we assume that each owner has n data users — the maximum number of data users for each owner. Raw data are first provided by $O_{1,1}$, who generates an authorization key for each of its user. Then the data user (e.g., $U_{1,1,2}$) can access the original data and insert its newly generated data (e.g., data processing results) to facilitate deeper and more complete analysis. $U_{1,1,2}$ can also choose to store the data processing result in the cloud and become a second-level owner (e.g., $O_{2,1}$). Another user (e.g., $U_{2,1,1}$) can access the data of $O_{2,1}$ if and only if $U_{2,1,1}$ obtains authorizations from the upper-level owners of this data (e.g., both $O_{2,1}$ and $O_{1,1}$). Similarly, $U_{2,1,1}$ can process the data and become a third-level owner (e.g., $O_{3,1}$). In other words, to access the processed data, a user must first be authorized by all upper-level

TABLE 1
NOTATIONS

Symbol	Description
g	Public system generator;
n	The number of authorization keys generated by a data owner;
$O_{i,j}$	The jth data owner in the ith level;
$U_{i,j,k}$	The kth authorized user of $O_{i,j}$;
$ID_{o_{i,j}}$	The identity of $O_{i,j}$;
$ID_{i,j,k}$	The identity of $U_{i,j,k}$;
$m_{i,j}$	The private data of $O_{i,j}$;
$sk_{i,j}$	The symmetric key of $O_{i,j}$ for encryption;
$ID_{m_{i,j}}$	The identity of $m_{i,j}$;
$CT_{m_{i,j}}$	The encrypted data of $m_{i,j}$ under $sk_{i,j}$;
$q_{i,j}$	The defined first-order polynomial of $O_{i,j}$;
$\mathcal{H}(\cdot)$	The hash function;
$tk_{i,j,k}$	The authorization token of $O_{i,j}$ for $U_{i,j,k}$;
$TK_{i,j}$	The authorization token set of $O_{i,j}$ for its users;
E_{App}	The predefined SGX Enclave program;
$AK_{i,j,k}$	The authorization key issued by $O_{i,j}$ for user $U_{i,j,k}$, which is one part of the authorization token $tk_{i,j,k}$
$sk_{a_{i,j}}$	The master secret share generated by owners;
$sk_{e_{i,j}}$	The key components shared with E_{App} by $O_{i,j}$, which equals to $\{sk_{a_{i,j}}, \{\{T_{i,j,k}, ID_{m_{i,j}}\} \vert 1 \le k \le n\}\}$
TK^u_{i,j_i,k_i}	The authorization token set for an ith level user issued by $\{O_{1,j_1}, \ldots, O_{(i-1),j_{(i-1)}}\}$;
$sk^u_{e_{i,j}}$	The partial secret share set related to TK^u_{i,j_i,k_i};
pol	The access policy file defined by an owner;
λ	The conditional probability of encountering an undercover police when $O_{i,j}$ was bribed;
V	The value of the shared data of $O_{i,j}$ to user;
Q_v	The fee paid by U_{v,j_v,k_v} to its upper owner O_{v,j_v};
P	The payoff gained by a user to bribe $O_{i,j}$;
$W_{o_{i,j}}$	The punishment for a cheating $O_{i,j}$;
n_p	The penalty degree.

related data owners. Subsequently, the user can further process the accessed data together with its own data and share the newly formed data, thereby becoming a new data owner.

Fig. 3 illustrates the system model of ESMAC, which comprises four types of system entities.

1) *Cloud Application Server (CAS):* CAS is an SGX-enabled cloud computing platform with data storage, which stores encrypted data and deploys application service programs to support user data processing and control data access. In more detail when an authorized user wants to process shared data, a host program in CAS loads the encrypted data from the untrusted part into Enclave. Next, SGX rebuilds the encryption key according to the authorization keys from multiple owners, then decrypts the data. Subsequently, SGX processes the data according to the user's request. If the user wants to store the processing result and becomes a next-level owner, SGX generates an encryption key and a number of authorization keys for it, then stores the encrypted data into the storage part of CAS.
2) *Data owner:* A data owner is a subscriber to CAS for data processing and storage. It also manages the authorization of its own data.

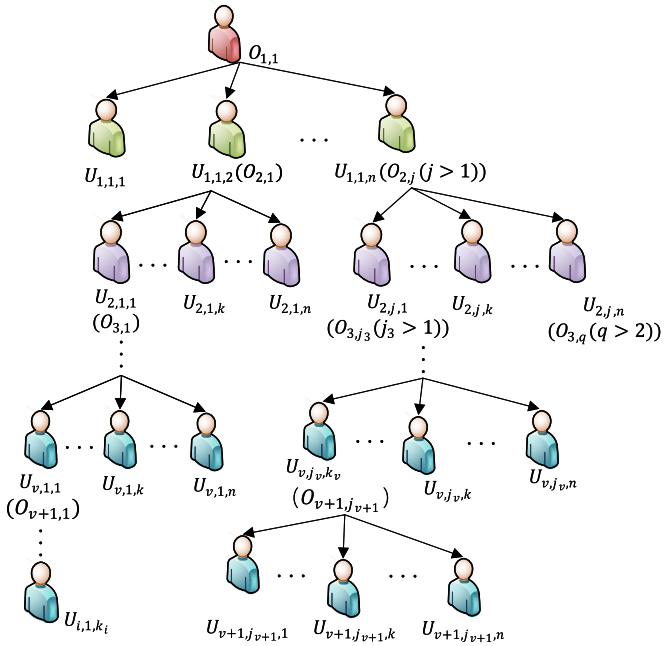


Fig. 2. Relationships between data owners in an MLDP model.

3) *Data user:* Data users are data consumers. A data user can request an authorization key from a data owner. In more detail, a data user can request to process the shared data belonging to the data owner in the $ith(i \ge 1)$ level from CAS when it obtains all authorization keys from related data owners located in the upper data processing levels. After data processing, the data user can choose to store the processed results in CAS and then becomes an $(i+1)th$-level data owner.

3.2 Security Model

In our target system, a first-level data owner is honest and holds its own original data. It generates an encryption key for encrypting the original data and authorization keys that

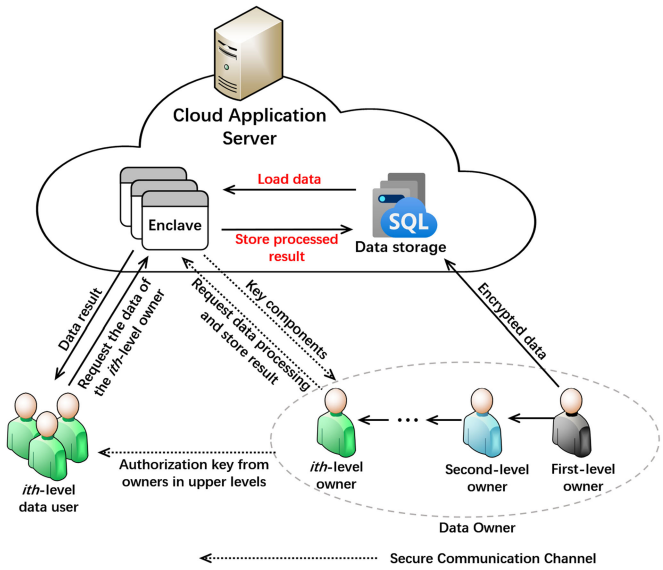


Fig. 3. ESMAC system model.

permit its users to access the original data. Non-root data owners, serving as data users, are rational but curious. They will honestly follow the design of ESMAC as a non-root owner; however, they may dishonestly claim to be a root data owner of co-owned data, and share these data with other users to gain more profits. Obviously, they will select the strategy that gains them the highest profits. That is, if leaking data will bring them higher profits, they may damage the interests of other data owners and leak the data selfishly. Since a data user can become a non-root owner, the security assumptions applicable to a non-root data owner also apply to a data user. In CAS, server resources are divided into trusted and untrusted parts. We assume that the Enclaves based on SGX hardware are attested, faithfully execute ESMAC operations, and output the desired processing results. As an untrusted part, the data storage component is also considered honest but curious: it executes the ESMAC scheme to load and store shared data, but will try to obtain sensitive data in order to extra profits. In addition, we assume that secure communication channels are established for communications between the involved entities. When the Enclave crashes or is unexpectedly terminated, the user and owner only lose the intermediate data they are currently working on, but the data stored outside the Enclave is not affected; the root owner simply reconstructs the Enclave and reperforms the previous task.

Notes: Intel SGX is vulnerable to side-channel attacks, but many countermeasures have been proposed to solve this issue [31], [32], [33]. In addition, we have conducted a deep overview [34] on this issue and have identified some potential avenues of future work to improve the security of Intel SGX. In this work, we directly take SGX as a technique capable of providing a secure memory region without addressing the issue of side-channel attacks. Aside from the side-channel attack issue, several Enclaves need to jointly complete a complex functionality, which can lead to incorrect processing results. In our previous work [35], a verifiable computation based on SGX is designed that can check the correctness of computation across multiple Enclaves in SGX.

3.3 Design Goals

ESMAC aims to achieve efficient and secure multi-owner access control over multi-level processed data, especially those data gained after multiple rounds of processing. ESMAC holds the following design goals:

- *Confidentiality of both data and secret key:* When sensitive data belonging to an owner are outsourced to the cloud for storage or processing, the data must be well protected from being tampered with or compromised.

Definition 1. *We say that ESMAC achieves confidentiality if for any Probabilistic Polynomial Time (PPT) attacker $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}$ such that for any PPT environment ϵ, we have*

$$EXEC_{ESMAC,\mathcal{A},\epsilon} \approx EXEC_{DBDH,\mathcal{B},\epsilon}$$

That is, the difficulty of breaking ESMAC is equivalent to the difficulty of breaking Decisional–Bilinear Diffie-Hellman (DBDH).

- *Trusted data processing:* Data processing must be secure, reliable, and trustworthy, such that data are processed correctly and can be protected from any unauthorized entity.
- *Secure data access controlled by multiple owners:* When data have been processed by multiple data owners, a user's authorization must be determined by all involved data owners. In other words, only once the user has obtained the key shares granted by the top-i level owners can it access and process the corresponding ith-level data jointly owned by these parties.
- *Efficiency:* The computational overhead of ESMAC should be low enough to support its practical deployment.

4 ESMAC Scheme for MLDP

In this section, we first present some preliminaries, followed by the design of the ESMAC scheme, which consists of three sub-schemes: the KM scheme, MOAC scheme, and DP scheme.

4.1 Preliminaries

4.1.1 Bilinear Pairing

Let $\mathbb{G}_1, \mathbb{G}_2$ be a cyclic group with prime p, while g is a generator of $\mathbb{G}_1$. A bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is defined, if it satisfies the following features:

- Bilinearity: For $\forall a, b \in \mathbb{Z}_p, \forall u, v \in \mathbb{G}_1, e(u^a, v^b) = e(u, v)^{ab}$;
- Nondegeneracy: $e(g, g) \neq 1$.

4.1.2 Decisional Bilinear Diffie–Hellman Assumption

The security of our proposed key management scheme is based on the DBDH assumption. Its specific definitions are as follows:

Let a, b, c, z be four random numbers selected from $\mathbb{Z}_p$ while g is a generator of $\mathbb{G}_1$. The DBDH assumption is that there is no polynomial-time probability algorithm $\mathcal{B}$ that can distinguish tuples $(A = g^a, B = g^b, C = g^c, e(g, g)^{abc})$ from $(A = g^a, B = g^b, C = g^c, e(g, g)^z)$ with more than a negligible advantage ε. The advantages of $\mathcal{B}$ in solving the above problem can be described by the following mathematical expression:

$$\left| \begin{array}{c} Pr\left[\mathcal{B}\left(A, B, C, e(g, g)^{abc}\right) = 0\right] \\ -Pr[\mathcal{B}(A, B, C, e(g, g)^z) = 0] \end{array} \right| \leq \varepsilon \tag{1}$$

4.1.3 Intel SGX

Intel SGX is a set of instruction extensions of the Intel x86-64 architecture that enable an application to create a cryptographically protected memory region, called an Enclave, which protects the confidentiality and integrity of the data and codes running inside [8]. Memory isolation is a main functionality of SGX. When codes are running in the Enclave, programs such as OS, Virtual Machine Monitors and other privileged software cannot access the data; once an Enclave process exits, the codes and data are encrypted and stored in a hard disk. Communications between an Enclave and the

untrusted memory areas (utilized by its host program) are achieved through predefined Ecall/Ocall functional interfaces. Notably, a call to an Enclave is referred to as an Ecall, while an Ocall allows Enclave codes to call untrusted functions outside. SGX also supports two auxiliary functions: remote attestation and storage sealing [36]. Remote attestation enables a distant entity to verify whether an Enclave instance has been created in a genuine Intel SGX-enabled hardware platform and check the integrity of the code and data inside the Enclave. Along with remote attestation, a secure communication channel between an Enclave and its attester can also be established, through which a user's private data is securely provisioned to the Enclave for processing. Storage sealing provides an encryption service for a host program using a hardware-based encryption key to securely cache Enclave data outside.

4.1.4 Shamir t-out-of-n' Threshold Secret Sharing

Assume that n' denotes the number of parties in an access structure. Let p^{ξ} be the size of the domain of secrets, where p is a prime-power that is bigger than n'. Let $s \in \mathbb{F}_{p^{\xi}}$ be a secret. A dealer chooses $t-1$ unique random elements $\{r_1, r_2, \ldots, r_{t-1}\}$ in $\mathbb{F}_{p^{\xi}}$. Then, *secret polynomial* is defined as follows:

$$q(x) = r_{t-1}x^{t-1} + r_{t-2}x^{t-2} + \ldots + r_1 x + s \tag{2}$$

Observe that $q(0) = s$. The dealer gives a piece $q(i)$ to party $P_i (i \in \mathbb{F}_{p^{\xi}})$, where $q(i)$ is a linear combination of the random inputs r_j and the secret s. Now, each set of at least t cardinalities can reconstruct $q(x)$ by means of Lagrange interpolation. That is, the parties $\{P_{i_1}, \ldots, P_{i_t}\}$ holding the pieces $\{s_{i_1}, s_{i_2}, \ldots, s_{i_t}\}$ can compute $q(x)$ as follows:

$$q(x) = \sum_{j=1}^{t} s_{i_j} \prod_{i_j \in I, i_d \neq i_j} \frac{i_d - x}{i_d - i_j} \tag{3}$$

where set $I = i_1, i_2, \ldots, i_t$. The secret s is recovered by substituting 0 for x in Eq. (3). *The Lagrange coefficient* $\Delta_{i_d,I}$ for i_d and I can be defined as follows:

$$\Delta_{i_d,I} = \prod_{i_j \in I, i_d \neq i_j} \frac{i_d - x}{i_d - i_j} \tag{4}$$

4.1.5 Game Theory and Information Security

Game theory offers a mathematical framework for studying conflict and cooperation between two or more rational decision-makers [37]. In recent years, the combination of game theory and information security has attracted increasing research interest in academia [38]. As an example, game theory can help address the issue of collusion between participants and service providers in Secure Multi-party Computation (SMC) based on secret sharing [39].

A game model consists of three basic components: player, strategy, and utility. Each player has its own strategy set, and can only choose one strategy in the set in each game. An action taken by a player may affect any or all other players. The strategies chosen by all players constitute a strategy combination. Under certain strategy combinations, the benefit obtained by a player is referred to as the utility of the player, and the strategy combination that maximizes the utilities of all players, each of whom has no incentive to deviate from his chosen strategy after considering an opponent's choice, is called a Nash Equilibrium (NE). In this paper, we use game theory to model the interactions between owners and accordingly data users, and accordingly design a data protection scheme that motivates non-root owners to behave honestly and follow the design of ESMAC.

4.2 KM Scheme

To clarify the different roles of root owners and non-root owners in an MLDP scenario, we design the KM scheme, which consists of two key generation algorithms for root and non-root owners, respectively.

4.2.1 Key Generation Algorithm of Root Owner

In an MLDP scenario, the root owner who originally provides the data, has the full control over its data, and can grant access rights to others by itself. Root owner uses Algorithm 1 to generate its own key parameters.

Algorithm 1. $KG_{RO}(1^{\tau}, n)$

Require: $1^{\tau}, n$;
Ensure: $sk_{1,1}, sk_{e_{1,1}}, TK_{1,1}$;
1: **define** $\mathbb{G}_1$ and $\mathbb{G}_2$ by prime order p and $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_2$
2: randomly select $\sigma_{1,1}, \beta_{1,1}$ from $\mathbb{Z}_p$
3: compute $sk_{a_{1,1}} = e(g,g)^{q_{1,1}(1)}$
4: compute $T_{1,1,k} = g^{r_{1,1,k}}$
5: compute $AK_{1,1,k} = g^{q_{1,1}(1)/r_{1,1,k}}$
6: **return** $sk_{1,1}$;
7: $sk_{e_{1,1}} = \{sk_{a_{1,1}}, \{\{T_{1,1,k}, ID_{m_{1,1}}\} | 1 \leq k \leq n\}\}$;
8: $TK_{1,1} = \{tk_{1,1,k} = \{AK_{1,1,k}, ID_{O_{1,1}}\} | 1 \leq k \leq n\}$;

$KG_{RO}(1^{\tau}, n) \rightarrow \{sk_{1,1}, sk_{e_{1,1}}, TK_{1,1}\}$: This algorithm is executed by the root owner $O_{1,1}$. It takes a security parameter τ and a non-zero positive integer n as input, and outputs the following secret key components for potential data users: a symmetric encryption key $sk_{1,1}$, a secret share $sk_{e_{1,1}}$ as, and authorization tokens $TK_{1,1}$.

In this algorithm, $O_{1,1}$ first defines two cyclic groups $\mathbb{G}_1, \mathbb{G}_2$ of prime order p and a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_2$. Next, $O_{1,1}$ leverages the Shamir threshold secret sharing scheme to generate the above keys, as follows:

- $O_{1,1}$ chooses two random numbers $\sigma_{1,1}, \beta_{1,1} \in \mathbb{Z}_p$ and defines a first-order polynomial $q_{1,1}(x) = \sigma_{1,1}x + \beta_{1,1}$, then computes $sk_{1,1} = e(g,g)^{\beta_{1,1}}$.
- For each authorized user $U_{1,1,k} (1 \leq k \leq n)$, $O_{1,1}$ uniformly chooses a random number $r_{1,1,k}$ from $\mathbb{Z}_p$. Then, it computes the following:

$$sk_{a_{1,1}} = e(g,g)^{q_{1,1}(1)} \tag{5}$$

$$T_{1,1,k} = g^{r_{1,1,k}} \tag{6}$$

$$AK_{1,1,k} = g^{\frac{q_{1,1}(2)}{r_{1,1,k}}} \tag{7}$$

Finally, it outputs a secret share $sk_{e_{1,1}}$ and all authorization tokens $TK_{1,1}$ for granting user access rights as shown in Algorithm 1. $ID_{m_{1,1}}$ in $sk_{e_{1,1}}$ indicates the index of the outsourced data from owner $O_{1,1}$, while other parts will be used to construct the symmetric encryption key. Among the tokens, $AK_{1,1,k}$ is used for key reconstruction, while $ID_{O_{1,1}}$ is the identity of the owner who holds the data. IDs in tokens can help Enclave search for secret shares $sk_{e_{i,j}}$ and recover the symmetric key through combination with key components $AK_{i,j,k}$.

4.2.2 Key Generation Algorithm for Non-Root Owners

As stated in the MDLP model, a non-root owner should first be granted access by its upper-level owners, and then be able to jointly control the access over accumulated dataset. Thus, the key generation algorithm for non-root owners is designed in Algorithm 2:

$KG_{NRO}(TK^u_{(i-1),j_{(i-1)},k_{(i-1)}}, sk^u_{e_{(i-1),j_{(i-1)}}}) \rightarrow$ $\{sk_{i,j}, sk_{e_{i,j}}, TK_{i,j}\}$: Different from KG_{RO}, this algorithm is executed by an Enclave. It takes all authorization tokens and related secret shares from $\{O_{1,j_1}, \ldots, O_{(i-1),j_{(i-1)}}\}$ in upper levels as input. And it outputs: 1) a symmetric encryption key $sk_{i,j}$, along with a secret key share $sk_{e_{i,j}}$ for non-root owner $O_{i,j}$ (i.e., $U_{(i-1),j_{(i-1)},k_{(i-1)}}$); 2) authorization tokens $TK_{i,j}$ for non-root data owner $O_{i,j}$ in the ith ($i > 1$) level.

First, its symmetric encryption key can then be constructed as follows:

Suppose shared data $m_{(i-1),j_{(i-1)}}$ are processed by a number of owners $\{O_{1,j_1}, \ldots, O_{i,j_i} | j_1 = 1, j_i = j\}$. $O_{v,j_v} (1 \le v \le i)$ is the $k_{(v-1)}th$ user of $O_{v-1,j_{(v-1)}}$, where $AK_{(v-1),j_{(v-1)},k_{(v-1)}}$ is the authorization key of O_{v,j_v} from $O_{(v-1),j_(v-1)}$. Since O_{v,j_v} obtains all authorization keys $\{AK_{1,j_1,k_1}, \ldots, AK_{(v-1),j_{(v-1)}, k_{(v-1)}}\}$, the secret $e(g,g)^{q_{(v-1),j_{(v-1)}}(0)} (1 \le v \le i)$ of the $j_{(v-1)}th$ data owner $O_{(v-1),j_{(v-1)}}$ in the $(v-1)th$ level can be recovered easily, as shown below:

$$
\begin{aligned}
& e(g,g)^{q_{(v-1),j_{(v-1)}}(0)} \\
&= \frac{\left(sk_{a_{(v-1),j_{(v-1)}}}\right)^2}{e(T_{(v-1),j_{(v-1)},k_{(v-1)}}, AK_{(v-1),j_{(v-1)},k_{(v-1)}})} \\
&= \frac{e(g,g)^{2q_{(v-1),j_{(v-1)}}(1)}}{e\left(g^{r_{(v-1),j_{(v-1)},k_{(v-1)}}}, g^{\frac{q_{(v-1),j_{(v-1)}}(2)}{r_{(v-1),j_{(v-1)},k_{(v-1)}}}}\right)} \\
&= \frac{e(g,g)^{2\sigma_{(v-1),j_{(v-1)}}+2\beta_{(v-1),j_{(v-1)}}}}{e(g,g)^{2\sigma_{(v-1),j_{(v-1)}}+\beta_{(v-1),j_{(v-1)}}}} \\
&= \begin{cases} e(g,g)^{\beta_{1,1}} & if\ v=1 \\ e(g,g)^{\mathcal{H}(AK_{(v-2),j_{(v-2)},k_{v-2}})} & otherwise \end{cases}
\end{aligned} \tag{8}
$$

The secret key $sk_{i,j}$ for $O_{i,j}$ can be generated by the following equation:

$$
\begin{aligned}
sk_{i,j} &= sk_{(i-1),j_{(i-1)}} e(g,g)^{\mathcal{H}\left(AK_{(i-1),j_{(i-1)},k_{(i-1)}}\right)} \\
&= e(g,g)^{\beta_{1,1}+\sum_{v=2}^{i-1}\mathcal{H}(AK_{v,j_v,k_v})}
\end{aligned} \tag{9}
$$

Second, $O_{i,j}$ generates the authorization token $tk_{i,j,k}$ for its next-level user $U_{i,j,k}$ according to the following steps:

- $O_{i,j}$ randomly selects a number $\sigma_{i,j} \in Z_p$ and defines a first-order polynomial $q_{i,j}(x) = \sigma_{i,j}x + \mathcal{H}(AK_{(i-1),j_{(i-1)},k_{(i-1)}})$, where $q_{i,j}(0) = \mathcal{H}(AK_{(i-1),j_{(i-1)},k_{(i-1)}})$.
- For each user $U_{i,j,k} (1 \le k \le n)$, $O_{i,j}$ uniformly chooses a number $r_{i,j,k}$ at random from $\mathbb{Z}_p$. Subsequently, $O_{i,j}$ computes:

$$sk_{a_{i,j}} = e(g,g)^{q_{i,j}(1)} \tag{10}$$

$$T_{i,j,k} = g^{r_{i,j,k}} \tag{11}$$

$$AK_{i,j,k} = g^{\frac{q_{i,j}(2)}{r_{i,j,k}}} \tag{12}$$

Finally, this algorithm outputs all key components of $O_{i,j}$: encryption key $sk_{i,j}$, secret share $sk_{e_{i,j}}$ and authorization tokens $TK_{i,j}$.

In both key generation algorithms, the secret share $sk_{e_{i,j}}$ will be shared with E_{App}, while the authorized tokens $TK_{i,j}$ will be issued to its authorized users. Hence, this approach can protect the original data from unauthorized users and even the SGX.

Algorithm 2. $KG_{NRO}(TK_{i,j,k}, sk_e)$

Input: $TK^u_{(i-1),j_{(i-1)},k_{(i-1)}}, sk^u_{e_{(i-1),j_{(i-1)}}}$;

Output: $sk_{i,j}, sk_{e_{i,j}}, TK_{i,j}$;

set $res = 1$
set $tem = 1$
for $v = i-1$ to 1 do:
  if $v = 1$:
    compute $tem = e(g,g)^{q_{v,j_v}}$;
    $= e(g,g)^{\beta_{1,1}}$
  else:
    compute $tem = e(g,g)^{q_{v,j_v}}$;
    $= e(g,g)^{\mathcal{H}(AK_{(v-1),j_{(v-1)},k_{(v-1)}})}$;
  $res = res \cdot tem$;
end for
set $sk_{i-1,j_{(i-1)}} = res$
$= e(g,g)^{\beta_{1,1}+\sum_{v=2}^{i-2}\mathcal{H}(AK_{v,j_v,k_v})}$;
generate $sk_{i,j} = sk_{i-1,j_{(i-1)}} e(g,g)^{\mathcal{H}(AK_{(i-1),j_{(i-1)},k_{(i-1)}})}$;
$= e(g,g)^{\beta_{1,1}+\sum_{v=2}^{i-1}\mathcal{H}(AK_{v,j_v,k_v})}$;
randomly select $\sigma_{i,j}$ from Z_p;
define $q_{i,j}(x) = \sigma_{i,j}x + \mathcal{H}(AK_{(i-1),j_{(i-1)},k_{(i-1)}})$;
for $k = 1$ to n do:
  randomly select $r_{i,j,k}$ from $\mathbb{Z}_p$
  compute $sk_{a_{i,j}} = e(g,g)^{q_{i,j}(1)}$;
  compute $T_{i,j,k} = g^{r_{i,j,k}}$;
  compute $AK_{i,j,k} = g^{\frac{q_{i,j}(2)}{r_{i,j,k}}}$
end for
return $sk_{i,j}$;
  $sk_{e_{i,j}} = \{sk_{a_{i,j}}, \{\{T_{i,j,k}, ID_{m_{i,j}}\} | 1 \le k \le n\}\}$;
  $TK_{i,j} = \{tk_{i,j,k} = \{AK_{i,j,k}, ID_{O_{i,j}}\} | 1 \le k \le n\}$;

4.3 MOAC Scheme

A MOAC scheme is designed to provide flexible and dynamic access control over co-owned data based on the key management schemes proposed above.

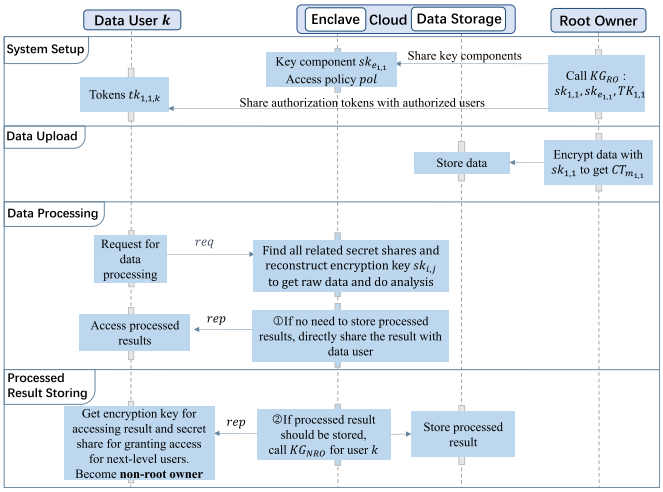


Fig. 4. Overview of MOAC scheme procedure.

4.3.1 Brief Overview of MOAC Scheme Procedure

In this subsection, we provide a brief description of the MOAC scheme, which comprises of four steps: System Setup, Data Upload, Data Processing, and Processed Results Storing. We take the root owner as an example, and provide an overview of the MOAC scheme procedure in Fig. 4.

1) $SysSetup(1^\tau, n) \rightarrow \{sk_{1,1}, sk_{e_{1,1}}, TK_{1,1}\}$: Root owners call KG_{RO} to generate its key parameters $sk_{1,1}, sk_{e_{1,1}}$, $TK_{1,1}$. The symmetric key $sk_{1,1}$ is used for encrypting outsourced data, while the key components $sk_{e_{1,1}}$ and tokens $TK_{1,1}$ will be shared with the Enclave and data users respectively.
2) $DataUpload(sk_{1,1}, m_{1,1}) \rightarrow \{CT_{m_{1,1}}\}$: When uploading data, the root owners encrypt its dataset with $sk_{1,1}$; this can only be decrypted with the recovered symmetric key based on the key parameters from both the Enclave and authorized users. Finally, a data packet that contains the IDs of the data owner and message together with the encrypted outsourced data will be uploaded to the cloud.
3) $DataProc(sk_{e_{1,1}}, tk_{1,1,k}) \rightarrow \{m_{2,2}\}$: In data processing, upon receiving the request from authorized users, Enclave will collect the tokens $tk_{1,1,k}$ and key components $sk_{e_{1,1}}$ to recover the symmetric key. If the data user has no need to store the processing result, then Enclave simply replies to the data user k with the processing results $m_{2,2}$.
4) $DataStor(m_{2,2}, sk_{e_{1,1}}, tk_{1,1,k}) \rightarrow \{res, sk_{2,j_k}, TK_{2,j_k}\}$: If data user k also wants to be a next-level data owner, then Enclave conducts analysis over the cumulative dataset and then calls KG_{NRO} to generate the basic parameters for data user k. In addition, it will store the processing results res and cumulative dataset in cloud storage.

4.3.2 Details of MOAC Scheme

In this subsection, a more detailed introduction to the MOAC scheme is given. The four steps are each outlined in more detail below.

Step 1 (System Setup @$O_{1,1}$, E_{App}): When the MOAC scheme is first executed, $O_{1,1}$ calls the algorithm $KG_{RO}(1^\lambda, n) \rightarrow \{sk_{1,1}, sk_{e_{1,1}}, TK_{1,1}\}$ to generate its key components. Then, $O_{1,1}$ creates an access control list pol, initialized as $NULL$. The CAS creates an Enclave instance E_{App} through a well-developed program. $O_{1,1}$ issues a remote attestation with CAS to ensure that E_{App} has been established in a trusted hardware environment. Through remote attestation, a secure communication channel between $O_{1,1}$ and E_{App} is established. Then, $O_{1,1}$ sends $sk_{e_{1,1}}$ and the access control list pol to E_{App} through the secure channel. For the kth user $U_{1,1,k}$, $O_{1,1}$ selects a unique authorization token $tk_{1,1,k} = \{AK_{1,1,k}, ID_{O_{1,1}}\}$ and securely delivers it to its authorized user by applying a secure communication protocol. All tokens and requests should be transferred securely. To further prevent the leakage of tokens from legitimate users, encrypted tokens can be used that contains the IDs of the owner and authorized user. If some owner wants to share a token with its authorized user, it can encrypt the authorized key with the symmetric key that has been shared with Enclave to obtain a token $Enc_K(tk_{i,j,k}, ID_{i,j,k})$. In this way, each token is bounded to each authorized user and can only be decrypted by Enclave for further key recovery.

Step 2 (Data Upload @$O_{i,j}$): With the generated symmetric key $sk_{i,j}(i > 0, j > 0)$, $O_{i,j}$ can encrypt its private data $m_{i,j}$ via Advanced Encryption Standard (AES) to get the ciphertext $CT_{m_{i,j}} = Enc(m_{i,j}, sk_{i,j})$, then upload $\{ID_{O_{i,j}}, ID_{m_{i,j}}, CT_{m_{i,j}}\}$ to the cloud. In this data packet, $ID_{O_{i,j}}$ and $ID_{m_{i,j}}$ denote the identity of the data owner and stored data respectively, while $CT_{m_{i,j}}$ is the outsourced ciphertext.

Step 3 (Data Processing @$O_{i,j}$, E_{App}): In this phase, an authorized user u issues a data processing request req to E_{App} to process the shared data of $O_{i,j}(i \geq 1)$, where $req = \{cd, ID_{O_{i,j}}, ID_{m_{i,j}}, TK^u_{i,j_i,k_i}, \delta\}$. In this data packet, req includes the identity of the processing command cd, the identity of data owner $ID_{O_{i,j}}$, a set of requested data for analysis, a set of tokens $TK^u_{i,j_i,k_i} = \{tk_{v,j_v,k_v} | 1 \leq v \leq i, j_1 = 1, j_i = j, k_i = k\}$ from upper-level owners, and a Boolean value δ. If δ is true, the data processing result res will be stored in the data storage part of CAS to facilitate the next-level access, and the user u becomes the $(i+1)th$ level data owner $O_{(i+1),j_{(i+1)}}$. Meanwhile, $O_{(i+1),j_{(i+1)}}$ can authorize other users to access the results. Otherwise, res are simply returned to the user $U_{i,j,k}$ but not stored. When E_{App} receives req, it processes the request as follows:

1) **Key Recovery**: $KR(sk^u_{e_{i,j}}, T_{v,j_v,k_v}) \rightarrow \{sk_{i,j}\}$. Algorithm 3 is executed by E_{App}. It takes the secret share $sk^u_{e_{i,j}} = \{sk_{e_{1,j_1}}, \ldots, sk_{e_{i,j_i}}\}$ of all involved data owners as input, and outputs symmetric encryption key. First, it retrieves $sk^u_{e_{i,j}}$ to obtain $\{sk_{a_{v,j_v}} | 1 \leq v \leq i\}$ and $\{T_{v,j_v,k_v} | 1 \leq v \leq i\}$. Next, it recovers $\{e(g,g)^{q_{v,j_v}(0)} | 1 \leq v \leq i\}$ as follows:

$$e(g,g)^{q_{v,j_v}(0)} = \frac{(e(g,g)^{q_{v,j_v}(1)})^2}{e(AK_{v,j_v,k_v}, T_{v,j_v,k_v})} = \begin{cases} sk_{1,1} & if\ v = 1 \\ e(g,g)^{\mathcal{H}(AK_{(v-1),j_{(v-1)},k_{(v-1)}})} & otherwise \end{cases} \quad (13)$$

The secret keys $\{sk_{v,j_v} | 1 \leq v \leq i\}$ of all owners $\{O_{1,j_1}, \ldots, O_{i,j_i}\}$ can be reconstructed as follows:

$$sk_{v,j_v} = \begin{cases} sk_{1,1} & if\ v = 1 \\ sk_{(v-1),j_{(v-1)}} e(g,g)^{\mathcal{H}(AK_{(v-1),j_{(v-1)},k_{(v-1)}})} & otherwise \end{cases} \tag{14}$$

When $v = i$, the encryption key $sk_{i,j}$ can be recovered.

2) *Data Decryption*: $Dec(CT_{m_{i,j}}, sk_{i,j}) \rightarrow \{m_{i,j}\}$. E_{App} checks the access control list pol and decrypts the authorized ciphertext $CT_{m_{i,j}}$ with the recovered key $sk_{i,j}$ to obtain the raw data $m_{i,j}$.
3) *Data Processing*: $DPr(m_{i,j}) \rightarrow \{m_{i+1,j_{(i+1)}}\}$. E_{App} performs computational operations over $m_{i,j}$ according to the cd in the user's request req, then stores the hierarchical dataset as $m_{i+1,j_{(i+1)}}$ in CAS.

Algorithm 3. $KR(sk^u_{e_{i,j}}, T_{v,j_v,k_v})$

Input: $sk^u_{e_{i,j}}, T_{v,j_v,k_v}(1 \le v \le n)$;
Output: $sk_{i,j}$;

set $res = 1$
set $tem = 1$
for $v = i$ to 1 do:
  if $v = 1$:
    compute $tem = e(g,g)^{q_{v,j_v}}$;
    $= e(g,g)^{\beta_{1,1}}$
    $sk_{1,1}$
  else:
    compute $tem = e(g,g)^{q_{v,j_v}}$;
    $= e(g,g)^{\mathcal{H}(AK_{(v-1),j_{(v-1)},k_{(v-1)}})}$;
    $res = tem \cdot res$;
end for
generate $sk_{i,j} = res$;
$= e(g,g)^{\beta_{1,1}+\sum_{v=2}^{i-1}\mathcal{H}(AK_{v,j_v,k_v})}$;
return $sk_{i,j}$;

Step 4 (Processing Results Storing @E_{App},Cloud): In this phase, the data user inserts its personal data for analysis and also wants be a new data owner in the hierarchical dataset. If δ in req is true, E_{App} invokes $KG_{NRO}(TK^u_{i,j_i,k_i}, sk^u_{e_{i,j}}) \rightarrow \{sk_{(i+1),j_{(i+1)}}, sk_{e_{(i+1),j_{(i+1)}}}, TK_{(i+1),j_{(i+1)}}\}$ to generate the symmetric encryption key for $O_{(i+1),j_{(i+1)}}$ and authorization tokens for next-level data users. It then delivers the corresponding response $rep = \{res, sk_{(i+1),j_{(i+1)}}, TK_{(i+1),j_{(i+1)}}\}$ via the established secure channel, where $res = m_{(i+1),j_{(i+1)}}$. To securely store results, it encrypts res with $sk_{(i+1),j_{(i+1)}}$ and stores the packet $\{CT_{m_{(i+1),j_{(i+1)}}}, ID_{O_{(i+1),j_{(i+1)}}}, ID_{m_{(i+1),j_{(i+1)}}}\}$ in CAS. Then, user u can become the $(i+1)th$-level data owner $O_{(i+1),j_{(i+1)}}$, which can choose a unique authorization token $tk_{(i+1),j_{(i+1)},k}$, in $TK_{(i+1),j_{(i+1)}}$ for user u of $O_{(i+1),j_{(i+1)}}$ and distribute it via a secure channel. Next, $O_{(i+1),j_{(i+1)}}$ sends a request to E_{App} to update pol, which should be approved by its upper-level owners. Otherwise, E_{App} simply returns res to the authorized data user securely.

Moreover, MOAC can also realize user delegation and revocation by combining the schemes proposed herein with the "heartbeats" protocol presented in our previous work [40], in which the data owner sends a periodic heartbeat signal to maintain the normal service of Enclave. Since the trusted code is called by an untrusted host program in SGX, a compromised host program may block the user delegation or revocation instruction in order to prevent the corresponding Enclave function from being invoked. Herein, data owners can create a dedicated thread to send a "heartbeat" signal to maintain the freshness of E_{App}. A user u gains access to the shared data of $O_{i,j}$, until it obtains the secret tokens of all data owners related to the data and E_{App} is active. When an owner wishes to revoke an authorized user, it sends a revocation instruction with a heartbeat emission to delete the related key components in E_{App}. In this way, it can revoke the access rights of specified users.

4.4 DP Scheme

Adopting a different perspective, we introduce game theory to design the DP scheme, which motivates the owners to claim their ownership rights honestly and discourages non-root data owners from engaging in data leakage behaviors. It also provides the payoff and punishment settings in the system, which work as an incentive mechanism for the MLDP scenario.

4.4.1 Data Leakage Problem

In the MOAC scheme, when an authorized user requests data processing, the scheme requires the user (such as, a non-root owner $O_{i,j}(i > 1)$) to honestly declare in req whether to store the data processing result. However, the non-root owner is rational, and may thus selfishly disclose the obtained shared data to obtain extra benefits. Assume that the data have been processed by $\{O_{1,j_1}, \ldots, O_{v,j_v}, \ldots, O_{i,j_i} | j_1 = 1, j_i = j\}$ and an authorized user $U_{i,j,k}$ wants to obtain the ith-level data processing result $m_{i,j}$ from $O_{i,j}$. Moreover, the user $U_{i,j,k}$ needs to pay $Q_v(1 \le v \le i)$ for each owner $O_{v,j_v}(1 \le v \le i)$ who contributing their personal data in the hierarchical dataset. Thus, $U_{i,j,k}$ pays a total of $\sum_{v=1}^{i} Q_v$ to acquire all authorized tokens TK^u_{i,j_i,k_i}. Since $O_{i,j}$ has obtained $m_{i,j}$, $U_{i,j,k}$ may choose to bribe $O_{i,j}$ to obtain $m_{i,j}$. If $U_{i,j,k}$ gives $P(Q_i < P < \sum_{v=1}^{i} Q_v)$ to $O_{i,j}$ privately, $O_{i,j}$ may choose to leak $m_{i,j}$ or upload the data as a root owner for a higher payoff. That is, when $U_{i,j,k}$ requests data processing results from owners in the upper levels, $O_{i,j}$ chooses not to store the data processing results, and instead privately uploads $m_{i,j}$ to CAS as a root data owner; Subsequently, $O_{i,j}$ becomes the sole party capable of authorizing users to access the uploaded data. In this way, the upper-level data owners lose their control over $m_{i,j}$ and their interests are damaged. Therefore, effective measures must be taken to prevent non-root data owners from leaking data.

At present, there are two methods in place that attempt to solve this issue. One is data watermarking, which embeds the relevant information of data owners into data during processing. When a malicious non-root owner leaks the co-owned data to others, or uploads them while purporting to be a root data owner, it is possible to check against predefined watermarks to determine whether the owner has acted honestly according to a well-designed protocol. However, data with different formats can be watermarked in different ways, and there are various forms of data in MLDP, meaning that data watermarking is not applicable for ESMAC. The other method is to leverage game theory to motivate system entities to act honestly in order to gain high profits across their lifetime.

4.4.2 Game Between O(i,j) and its User

In order to deal with the issue discussed above, we apply game theory to assess the respective pros and cons when a

TABLE 2
Game Between $O_{i,j}$ AND $U_{i,j,k}$

		$O_{i,j}$	
		H	C
$U_{i,j,k}$	H	$(V-\sum_{v=1}^{i} Q_v, Q_i)$	$(0, -W_{O_{i,j}})$
	C	$(V-\sum_{v=1}^{i} Q_v, Q_i)$	$(V-\lambda V-P+\lambda P, -\lambda W_{O_{i,j}} +P-\lambda P)$

Notes: H = Honest C = Cheat

data user acts honestly or dishonestly. Specifically, we leverage undercover polices in [41] to check whether $O_{i,j}$ is engaging in cheating behaviors. In ESMAC, an honest monitoring agency is introduced into our game, in which user u is an undercover police who pretends to offer a bribe to $O_{i,j}$. If $O_{i,j}$ accepts the bribe and intends to disclose the co-owned data to the undercover police, then the agency gives $O_{i,j}$ a certain penalty. Moreover, a cheating owner can be reported by an honest user in return for exemption from payment.

We will next describe the designed game in more detail. First, let λ be the conditional probability of encountering the police when $O_{i,j}$ is bribed. In the game, a user u has two choices: honesty and bribery. $U_{i,j,k}$ can choose to honestly obtain authorization keys from all owners according to the design of the ESMAC scheme, or alternatively, to bribe $O_{i,j}$ to obtain the shared data at a lower cost. $O_{i,j}$ also has two choices: honesty and cheating. $O_{i,j}$ can choose to perform data processing and storage in accordance with the ESMAC scheme honestly, or induce user u to offer a bribe and thereby obtain higher utility. The payoff matrix of the game between $O_{i,j}$ and user u is presented in Table 2. Here, V represents the value of the shared data of $O_{i,j}$ to user u; $Q_v (1 \leq v \leq i)$ indicates the fee paid by user u to O_{v,j_v}; P is the bribe that user u offers to $O_{i,j}$ to decrease his own costs for accessing shared data, where $Q_i < P < \sum_{v=1}^{i} Q_v, V - \sum_{v=1}^{i} Q_v > 0, Q_i \geq 0$. To facilitate better understanding of the payoffs in Table 2, we provide the following explanation:

- If the user is honest and does not bribe $O_{i,j}$ to obtain data, λ is zero, as the undercover police will pretend to bribe $O_{i,j}$ to induce them to leak data. If $O_{i,j}$ is honest, the utilities of $O_{i,j}$ and user u are Q_i and $V - \sum_{v=1}^{i} Q_v$, respectively. Otherwise, the honest user u will report the malicious $O_{i,j}$, and their payoffs will be 0 and $-W_{O_{i,j}}$, respectively.
- If user u attempts to bribe $O_{i,j}$, user u may be an undercover police or a real data user. If $O_{i,j}$ is honest, $O_{i,j}$ will refuse the bribe. At this point, both parties can only execute in accordance with the scheme design; thus, the expected gain of $O_{i,j}$ is

$$\lambda\Big(V-\sum_{v=1}^{i} Q_v\Big) + \Big(1-\lambda\Big)\Big(V-\sum_{v=1}^{i} Q_v\Big) = V-\sum_{v=1}^{i} Q_v \quad (15)$$

  The expected payoff of user u is

$$\lambda * Q_i + (1-\lambda) * Q_i = Q_i \quad (16)$$

  As is evident, their expected payoffs are the same as (honest, honest).
- If $O_{i,j}$ is malicious and accepts a bribe from user u, there are two possible cases. One is that user u is undercover police, and then $O_{i,j}$ will be punished, so the utility of $O_{i,j}$ is $-W_{O_{i,j}}$; the other is that user u is malicious, so the payoff of $O_{i,j}$ is Q. The expected utility of $O_{i,j}$ is

$$\lambda * (-W_{O_{i,j}}) + (1-\lambda) * P = -\lambda W_{O_{i,j}} + P - \lambda P \quad (17)$$

  The expected payoff of user u is

$$\lambda * 0 + (1-\lambda) * (V-P) = V - \lambda V - P + \lambda P \quad (18)$$

When $Q_i > -\lambda W_{O_{i,j}} + P - \lambda P$, namely $\lambda > \frac{P-Q_i}{P+W_{O_{i,j}}}$, no matter how user u chooses, $O_{i,j}$ will choose to perform honestly according to the MOAC design. At this time, honesty is a strongly dominant strategy for $O_{i,j}$. Thus, there are two NE, namely (honest, honest) and (honest, cheat). On the other hand, when $\lambda \leq \frac{P-Q_i}{P+W_{O_{i,j}}}$, (honest, honest) and (cheat, cheat) are clearly the two NE of this game. In either case, (honest, honest) is a NE. As long as there are enough undercover polices or we increase the punishment for the dishonest non-root owners, (cheat, cheat) will not occur. For ease of analysis, we set $W_{O_{i,j}} = n_p \sum_{v=1}^{i} Q_v (n_p \geq 1)$, where n_p denotes the penalty degree. As Q_v is determined by the value of the shared data, hence the variables in $\frac{P-Q_i}{P+n_p \sum_{v=1}^{i} p_v}$ include P and n_p. Let $f(P, n_p) = \frac{P-Q_i}{P+n_p \sum_{v=1}^{i} p_v} (Q_i < P < \sum_{v=1}^{i} Q_v, n_p \geq 1)$, we find the partial derivative of $f(P, n_p)$ about P:

$$\begin{aligned}\frac{\partial f(P,n_p)}{\partial P} &= -\frac{P-Q_i}{\left(P+n_p\sum_{v=1}^{i} Q_v\right)^2} + \frac{1}{P+n_p\sum_{v=1}^{i} Q_v} \\ &= \frac{Q_i + n_p\sum_{v=1}^{i} Q_v}{\left(P+n_p\sum_{v=1}^{i} Q_v\right)^2} > 0\end{aligned} \quad (19)$$

Thus, $f(P, n_p)$ increases as q increases, where $argmax(f(P, n_p)) = f(\sum_{v=1}^{i} Q_v, n_p)$. We have accordingly shown that when $\lambda > f(P, n_p)$, honesty is the dominant strategy of $O_{i,j}$. As long as $\lambda > argmax(f(P, n_p))$ is satisfied in the MOAC, $O_{i,j}$ will always choose to follow the ESMAC design honestly. Through analysis, we can observe that:

$$\begin{aligned}\lambda > argmax(f(P,n_p)) &= \frac{\sum_{v=1}^{i} Q_v - Q_i}{\sum_{v=1}^{i} Q_v + n_p\sum_{v=1}^{i} Q_v} \\ &< \frac{1}{n_p+1}\end{aligned} \quad (20)$$

Obviously, when $\lambda \geq \frac{1}{n_p+1}$, regardless of whether the user u bribes $O_{i,j}$ to illegally obtain data, $O_{i,j}$ will refuse it.

Notably, a non-root data owner may only choose to interact with familiar users while rejecting contacts from strange users (potentially undercover police), and may choose to sell the result on a different platform, which increases the difficulty of levying a fine. In this paper, we do not concentrate on this issue but instead simply focus on the data protection scheme. In the future, we will continue our research to further ensure the ownership security by combining watermark [42] and our proposed DP scheme. For example,

SGX could insert a watermark into the learning model or analyzed data and share them only with authorized users. The watermark can directly mark the ownership and thereby enable the malicious misappropriation to be traced. In addition, the DP scheme is also applied to detect malicious owners and motivate users to act honestly.

5 Security Analysis and Performance Evaluation

5.1 Security Analysis

This section analyzes ESMAC security. First, the DP scheme should be secure and correct, such that all non-root data owners honestly follow the rules of the proposed scheme. Second, the confidentiality of the shared data (including data processing results in each level) should be ensured. In accordance with Theorem 1, all rational non-root data owners will not privately disclose the co-owned data, even if there are few undercover police. Theorems 2 and 3 jointly prove that ESMAC can ensure the confidentiality of shared data in each level by reducing the difficulty of attacking ESMAC to the difficulty of solving the DBDH problem.

Theorem 1. *Even if there are few undercover police, the DP scheme can still ensure that non-root data owners honestly follow the design of MOAC.*

Proof. As mentioned in Section 4.4.2, when $\lambda \geq \frac{1}{n_p+1}$, $O_{i,j}$ will honestly behave according to the principles of MOAC. When the number of undercover police λ is small, we can increase the value of the punishment (i.e., n_p) to satisfy $\lambda \geq \frac{1}{n_p+1}$. □

Theorem 2. *If an attacker $\mathcal{A}$ can decrypt the ciphertext of a root data owner with polynomial time, a simulation algorithm $\mathcal{B}$ can be constructed to break the DBDH problem.*

Proof. Suppose that $\mathcal{A}$ can decrypt the shared data of $O_{1,1}$ with an advantage ε. We construct a simulator $\mathcal{B}$ so that it is capable of winning in a DBDH game with an advantage $\frac{\varepsilon}{2}$. The specific simulation process is as follows:

First, the ESMAC scheme defines two cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_2$ with prime order p, and a bilinear map $e: \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. It then randomly tosses a coin ϑ in some place where B cannot see. If $\vartheta = 0$, it sets $(A, B, C, Z) = (g^a, g^b, g^c, e(g,g)^{abc})$; otherwise, it sets $(A, B, C, Z) = (g^a, g^b, g^c, e(g,g)^z)$, where $a, b, c, z \in \mathbb{Z}_p$.

Init. $\mathcal{B}$ runs $\mathcal{A}$. $\mathcal{A}$ initializes an Enclave instance E_{App}, and inputs a set of authorized users $\alpha = \{ID_{1,1,1}, \ldots, ID_{1,1,n}\}$.

Phase 1. $\mathcal{A}$ requests the authorization key from $\mathcal{B}$ to obtain the shared data of $O_{1,1}$, $\mathcal{B}$ sends $sk_{e_{1,1}}$ or n authorization tokens $TK_{1,1} = \{tk_{1,1,k} | 1 \leq k \leq n\}$ to $\mathcal{A}$.

$\mathcal{B}$ inputs a security parameter τ and a positive integer n, then sets $sk_{1,1} = e(A, B^c) = e(g,g)^{abc}$. Subsequently, $\mathcal{B}$ constructs $q_{1,1}(x)$ in the following two cases:

- If $\mathcal{B}$ sends $sk_{e_{1,1}}$ to $\mathcal{A}$, it constructs $q_{1,1}(x)$ with $q_{1,1}(1) = \lambda_1$ and $q_{1,1}(0) = a$;
- Otherwise, it constructs $q_{1,1}(x)$ with $q_{1,1}(2) = \lambda_2$ and $q_{1,1}(0) = a$, where λ_1, λ_2 are random numbers selected from $\mathbb{Z}_p$.

Let the final polynomial be $Q_{1,1}(x) = bcq_{1,1}(x)$, where $Q_{1,1}(0) = abc$. $\mathcal{B}$ computes $sk_{a_{i,j}} = e(g,g)^{Q_{1,1}}$. Then, it selects n random numbers $\{r_{1,1,k} | 1 \leq k \leq n\}$ from $\mathbb{Z}_p$ and computes $\{R_{1,1,k} = g^{r_{1,1,k}} | 1 \leq k \leq n\}$. Next, $\mathcal{B}$ lets $sk_{e_{1,1}} = \{sk_a, \{\{R_{1,1,1}, ID_{1,1,1}\}, \ldots, \{R_{1,1,n}, ID_{1,1,n}\}\}\}$ and computes $TK_{1,1}$, where $tk_{1,1,k} = \left\{ AK_{1,1,k} = g^{\frac{Q_{1,1}^{(2)}}{r_{1,1,k}}} = B^{\frac{cq_{1,1}^{(2)}}{r_{1,1,k}}},\ ID_{1,1,k},\ ID_{O_{1,1}} \right\}$.

Therefore, $\mathcal{B}$ has the ability to construct secret sharing and authorization tokens like the KG_{RO} algorithm.

Challenge. $\mathcal{A}$ submits two challenge messages $m_{1,1}^0$ and $m_{1,1}^1$ to $\mathcal{B}$, after which $\mathcal{B}$ tosses a fair binary coin ψ. The ciphertext of $m_{1,1}^{\psi}$ according to the result ψ is thus as follows:

$$Enc_{AES}(m_{1,1}^{\psi}, sk_{1,1}) \rightarrow CT_{m_{1,1}^{\psi}}$$

If the coin tossed in first step $\vartheta = 0$, then $sk_{1,1} = Z = e(g,g)^{abc}$. Thus, $m_{1,1}^{\psi}$ is a valid ciphertext. Otherwise, if $\vartheta = 1$, then $sk_{1,1} = Z = e(g,g)^z$. Since z is a random number, the message encrypted with $sk_{1,1}$ is also a random number for $\mathcal{A}$ and does not contain any message of $m_{1,1}^{\psi}$.

Phase 2. $\mathcal{B}$ acts as it did in phase 1.

Guess. $\mathcal{A}$ submits the guess result ψ' for ψ. If $\psi' = \psi$, $\mathcal{B}$ outputs $\mu' = 0$, which indicates that $\mathcal{B}$ gave a valid BDH-tuple; otherwise, it will output $\mu' = 1$, which indicates that it was a random 4-tuple.

When the coin tossed in first step $\vartheta = 1$, $\mathcal{A}$ cannot obtain any information about ψ. Therefore, we have $Pr[\psi' \neq \psi | \vartheta = 1] = \frac{1}{2}$. Since, when $\psi' \neq \psi$, $\mathcal{B}$ outputs $\mu' = 1$, we have $Pr[\mu' = \vartheta | \vartheta = 1] = \frac{1}{2}$. Otherwise, when $\vartheta = 0$, $\mathcal{A}$ sees the ciphertext of $m_{1,1,\psi}$. In this case, the advantage of $\mathcal{A}$ is ε, so $Pr[\psi' = \psi | \vartheta = 1] = \frac{1}{2} + \varepsilon$. Because when $\psi' = \psi$, $\mathcal{B}$ outputs $\mu' = 0$, we have $Pr[\mu' = \vartheta | \vartheta = 0] = \frac{1}{2} + \varepsilon$.

Therefore, the advantage of $\mathcal{B}$ in the DBDH game is as follows: $\frac{1}{2} Pr[\mu' = \vartheta | \vartheta = 1] + \frac{1}{2} Pr[\mu' = \vartheta | \vartheta = 0] - \frac{1}{2} = \frac{\varepsilon}{2}$ □

Theorem 3. *Suppose that the shared data $m_{1,1}$ are processed by the data owners $\{O_{1,j_1}, \ldots, O_{v,j_v}, \ldots, O_{i,j_i}\}$. If the confidentiality of data in the vth$(1 < v < i)$ level is guaranteed in MOAC, an adversary cannot break the ciphertext of the shared data in the $(v+1)$th level.*

Proof. Since the confidentiality of the data in the vth level can be guaranteed, an adversary who wants to obtain the shared data in the vth level must obtain all authorization tokens and secret shares from $\{O_{1,j_1}, \ldots, O_{v,j_v}\}$. Assume that an attacker $\mathcal{A}$ wants to decrypt $CT_{m_{(v+1),j_{(v+1)}}}$. We only need to prove that if $\mathcal{A}$ does not acquire all the authorization tokens and related secret shares, then it cannot crack $CT_{m_{(v+1),j_{(v+1)}}}$. According to the designed key management scheme, we have:

$$sk_{(v+1),j_{(v+1)}} = sk_{v,j_v} e(g,g)^{\mathcal{H}(AK_{v,j_v,k_v})} (v \geq 1)$$

When $\mathcal{A}$ acquires the key components, there are two possible cases:

Case 1: $\mathcal{A}$ obtains all authorization tokens TK_{v,j_v,k_v}^u and secret shares $sk_{e_{v,j_v}}^u$ from $\{O_{1,j_1}, \ldots, O_{v,j_v}\}$, and it knows TK_{v,j_v} or related secret shares $sk_{e_{v,j_v}}$. $\mathcal{A}$ can recover

TABLE 3
Functionality Comparison With Other Existing Schemes

Schemes	Access control type	Security assumptions about all owners	Secure data processing	Owner revocation	User revocation	Key revocation in SGX
MOCC [1]	Multi-owner access control with a fixed number of owners	Trusted	No	No	No	—
MACS [17]	Multi-owner access control with a fixed number of owners	Trusted	No	No	No	—
IBBE-SGX [21]	Single-owner access control	Trusted	—	No	No	
oGBAC [18]	Single-owner access control	Trusted	No	Yes	Yes	—
Crypt-DAC [19]	Single-owner access control	Trusted	Yes	No	Yes	—
ESMAC	Multi-owner access control with a dynamically changed number of owners	Only root data owner is trusted	Yes	Yes	Yes	Yes

Notes: Yes: support No: does not support —: not mention

$sk_{(v+1),j_{(v+1)}}$. If $\mathcal{A}$ wants to pretend to be $U_{(v+2),j_{(v+2)},k_{(v+2)}}$ to obtain $sk_{(v+2),j_{(v+2)}}$ and decrypt $CT_{m_{(v+1),j_{(v+1)}}}$. It also needs to know $e(g,g)^{\mathcal{H}(AK_{(v+1),j_{(v+1)},k_{(v+1)}})}$. Similar to Theorem 2, obtaining $e(g,g)^{\mathcal{H}(AK_{(v+1),j_{(v+1)},k_{(v+1)}})}$ can be reduced to solving the DBDH problem. Therefore, $\mathcal{A}$ cannot decrypt $CT_{m_{(v+1),j_{(v+1)}}}$.

Case 2: $\mathcal{A}$ has not obtained all authorization tokens TK^u_{v,j_v,k_v} and secret shares $sk^u_{e_{v,j_v}}$, but has obtained $TK_{(v+1),j_{(v+1)}}$ and $sk_{e_{(v+1),j_{(v+1)}}}$. $\mathcal{A}$ can recover $e(g,g)^{\mathcal{H}(AK_{(v+1),j_{(v+1)},k_{(v+1)}})}$. Since the MOAC scheme can ensure the security of shared data in the vth level and sk_{v,j_v} cannot be recovered, $\mathcal{A}$ cannot crack the ciphertext. □

5.2 Functionality Comparison

In this subsection, we compare our scheme with existing works that support flexible access control.

First, we compare ESMAC with some traditional multi-authority access control schemes (MOCC [1] and MACS [17]). While both of these schemes implement multi-owner access control for co-owned private data, our scheme has several points of difference. First, our scheme supports access to multi-level data processing results in MLDP that is controlled by multiple owners with a dynamically changing number, while both MOCC and MACS merely focus on multi-owner access control of jointly managed data and the number of owners is fixed. Second, the security model of ESMAC is different. All data owners are considered fully trusted in MOCC and MACS, and a fully trusted server is needed in MACS. In ESMAC, we only trust the root data owner and SGX hardware, which makes it suitable for real-world scenarios.

Second, we also conduct a comparison between our scheme and those that only support a single data owner. IBBE-SGX [21] achieves data access control and supports the joint editing of shared data by utilizing SGX to reduce the computational overhead of an IBBE algorithm. However, this work does not consider the key revocation issue in SGX and only offers single-owner access control. In this scheme, moreover, SGX always retains the master private key, which increases the risk of key leakage. In contrast ,our scheme addresses key revocation at the Enclave by splitting the encryption key of a data owner in MOAC, which ensures that the encryption key is reconstructed and used for decryption if and only if an authorized user sends a data processing request with a valid authorization token. If there is no data processing request, the Enclave does not have the capability to recover the original data, since it holds only one part of the decryption key. oGBAC[18] also achieves a formal group-based access control by adding multiple labels to different entities. However, this scheme does not take the possible forgery of labels into account. For its part, Crypt-DAC[19] achieves dynamic data access control over cloud data. It requires the owner to upload the revocation key and encrypt the file in order to revoke the user. However, as the number of operations increases, the size of the key list will become too large and complicated.

A detailed comparison between the methods described above is presented in Table 3.

5.3 Performance Analysis

From the comparison in Section 5.2, we can observe that MACS [17] depends on a fully trusted server, while IBBE-SGX [21] can only support the analysis over data from a single owner. Hence, in this section, we further analyze the computational complexity of the proposed MOAC and compare it with MOCC [1]. In our analysis, we ignore the cost of data processing and only consider specific complex operations (such as the bilinear pairing operation (BP)), which are much more time-consuming than the data processing of plaintexts with Enclave. Comparisons are conducted from the following four aspects: encryption key generation of data owner, authorization key generation, data encryption, and data decryption. Let N represent the number of data owners. We assume that there is only one path from the root data owner to the last data owner in our scheme, and that an authorized user wants to decrypt the shared data in the Nth level. In addition, we suppose that each data owner generates a maximum number of n authorized keys (i.e., n data users for each owner). The cost of key generation in MOCC includes the cost of generating the master private key and public key during scheme initialization and appending the attribute strategy of non-first-level data owners. In addition, since the data disseminator can only re-encrypt the co-owned data for the access of the data accessor when it

TABLE 4
Computational Overhead Analysis

Phase	Proposed MOAC	MOCC [1]
Encryption key generation of data owner	$1 * BP + 1 * Ha + 1 * MM$	$2N * BP + N(n+6) * MM + 1 * MA + (Nn + A) * Ha$
Authorization key generation	$n * ME + n * MM + 2 * MA$	$(1+S) * MM + 1 * MA + (1+3S) * ME + S * Ha$
Data encryption	$1 * SE$	$1 * SE + 1 * BP + (3N + 9 + A) * ME + (1 + N + n + A) * Ha + 2(N+n) * MA + (5N + 3n + 2A + 7) * MM$
Data acquisition	$N * BP + 1 * Ha + (N-1) * MM + 1 * SE + O(N * n)$	$(2A * N + 2) * BP + (A * N + n + 3) * MM$

Notes: BP = Bilinear pairing operation; Ha = Hash operation; ME = Modular exponentiation operation; MM = Modular multiplication operation; MA = Modular addition operation; SE = Symmetric encryption operation

satisfies the attributes of all data owners, we only count the data disseminator with regards to the generation of authorization keys and data decryption. Herein, A represents the total number of attributes in the access control policy, while S denotes the number of attributes of the authorization key of the disseminator. The comparisons are illustrated in Table 4, while the detailed analysis of the computational cost of our scheme is provided below:

Key generation of data owner. The generation of a data owner's key includes the execution of the encryption key generation. The computational overhead of the root data owner for key generation is $1 * BP$. After recovering the upper owner's sk, the computational cost of encryption key generation for each non-root data owner comprises the costs of the bilinear pairing operation, hash operation, and modular multiplication. The computational cost incurred by the data owner in the $ith(i > 1)$ level to generate an encryption key is $i * BP + 1 * Hash + (i-1) * MM (2 \leq i \leq N)$.

Authorization key generation. Each data owner splits its data encryption key into two parts by applying the Shamir threshold secret sharing scheme, where one part is used to generate the authorization token $TK_{i,j}$, while the other part is sent to E_{App}. The computational overhead required by each data owner is $n * ME + n * MM + 2 * MA$.

Data encryption. Since the data encryption in MOAC applies symmetric encryption (SE), the cost is $1 * SE$.

Data decryption. Data decryption is executed in E_{App}, and its computational cost depends on the number of levels of encrypted data. At the beginning of data decryption, Enclave needs to retrieve the secret shares related to the authorization key to recover the encryption key via Eq. (9); hence, its time computational complexity is $\mathcal{O}(N^2)$. In MOAC's implementation, we used a multi-tree structure to store the secret shares owned by the Enclave, so that its computational complexity is $\mathcal{O}(N * n)$. Moreover, the computational cost of key recovery is $N * BP + 1 * Hash + (N-1) * MM$. To decrypt the data, E_{App} executes a symmetric decryption, which costs $1 * SE$.

6 Experimental Results

In this section, we tested the sub-schemes in ESMAC through a proof-of-concept implementation and benchmarked its performance. In more detail, we demonstrated its efficiency and verified its correctness by simulating the KM scheme, MOAC scheme, and DP scheme. We also evaluated the performance of ESMAC in a real-world scenario, specifically on a real ML model trained on the NIH Pancreas-CT Dataset[43].

In our simulation, we used a laptop with two 11th Gen 64-bit Intel i7-11700 processors at 2.50GHz with 4.0GB RAM. The ESMAC scheme was running on Ubuntu-18.04.6-desktop, and SGX driver version is 2.11 while SGX SDK version 2.12. We used NVIDIA GeForce GTX 1650 SUPER to accelerate the data analysis part of ESMAC. As SGX just supports partial CPU instructions and the trusted code executed in Enclave can only be linked to a static library, we first rewrote an Enclave-safe trusted pairing-based cryptography (PBC) library and its dependency GMP library to support the bilinear pairing operations in an Enclave.

6.1 Performance Evaluation of KM and MOAC

We first conducted experiments to evaluate the efficiency of two sub-schemes: KM scheme and MOAC scheme, especially their computation costs in constrained-users. In our simulation, we chose Type-A 160-bit elliptic curve groups in PBC library and 128-bit AES algorithm in GCM model to perform data encryption and decryption.

6.1.1 Performance Comparison of KM and MOCC in Key Generation

The KM scheme sets up the system and lays the basis for the ESMAC scheme. Hence, we tested our scheme and compared it with MOCC [1]. When a non-root data owner joins, the KM scheme needs to first find the secret shares stored by the upper-level owners in E_{App}, then generate the related key components. Next, E_{App} generates an encryption key and authorization tokens for the owner. Herein, we assume that an equal number of authorization tokens generated by a data owner in each level, denoted by n, and that only one user authorized by each owner becomes the next-level data owner. We tested the key generation time of the data owner in varied levels, from 1 to 10, with different numbers of authorization tokens (n= 20, 50, 100, 200, 300). From Fig. 5, we can observe that the levels of data owners do not affect the key generation time for each owner, while the number of tokens does. Compared with MOCC, when n is less than 100, our sub-scheme KM has higher efficiency. As n increases, the superiority of our method becomes weaker. However, in MOCC, the number of co-owners (which in our scheme are non-root owners) at each level was limited to one. In our scheme, the maximum number of non-root owners per level can reach n.

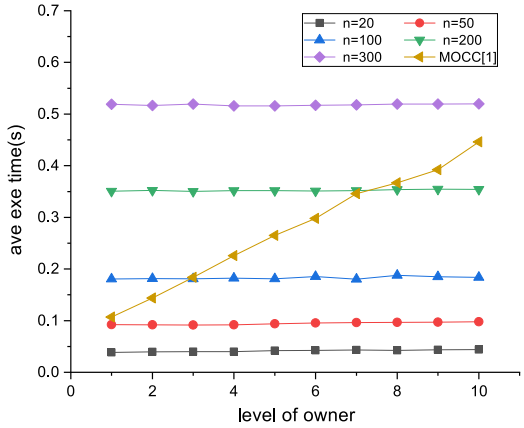


Fig. 5. Key generation time in MOAC.

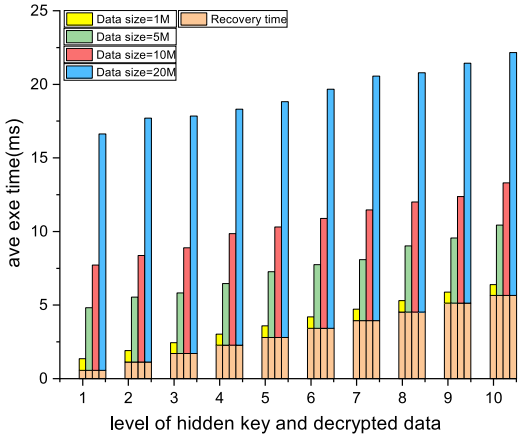


Fig. 7. Key recovery and decryption time in MOAC.

6.1.2 Performance Comparison of MOAC and MOCC in Data Acquisition

We tested the running time to obtain the requested data in MOAC and MOCC, which involves two acquisition steps: key recovery and symmetric decryption. We set the number of owner levels from 1 to 10 with the authorized user number n=10.

As shown in Fig. 6, our MOAC scheme is more efficient than MOCC in data acquisition. As the data size increases, the superiority of MOAC expands. Although the time in MOAC is affected by the number of levels, it still performs much better than MOCC when there are 10 levels.

In Fig. 7, we highlighted the time required for key reconstruction in yellow and also plotted the total time. We can observe that the decryption time increases with the data size, while key recovery time increases with the owner level. Even when the data size reaches 20M and the level is 10, SGX only requires about 20ms for data acquisition and 5ms for key recovery.

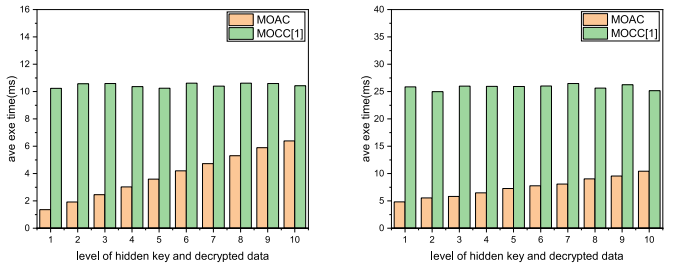
We then presented only the decryption time for SGX and users with different data sizes (1MB, 5MB, 10MB, 20MB) in Fig. 8. While the time cost differs slightly when decryption is executed in different environments, we can observe that our approach is still highly efficient for both SGX and resource-constrained users.

6.1.3 MOAC Decryption in SGX and Client

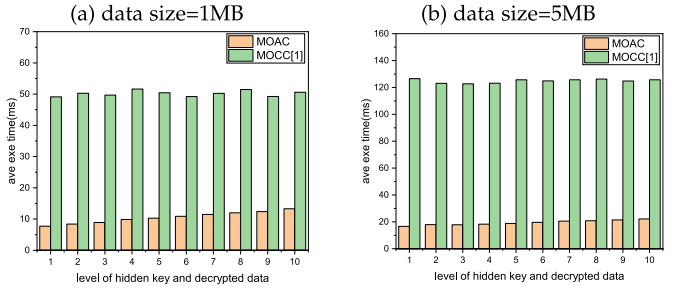
When receiving the user's request instruction, E_{App} first queries the tree structure in Enclave to recover the symmetric decryption key $sk_{i,j}$ and then decrypts the ciphertext. In MOAC, the decryption can be handled by E_{App} or the users. Therefore, we evaluated the time required for SGX to reconstruct $sk_{i,j}$ and decrypt the ciphertext with different sizes of original data (1MB, 5MB, 10MB, 20MB), respectively.

6.2 Simulation of DP Scheme

In this subsection, we used the GameBug simulator [44] to simulate the designed DP scheme. We tested multiple data transactions between the ninth-level data owners and authorized users. Herein, we assume $Q_k = 1 (1 \leq k \leq 9)$, $P = 8$, and $V = 10$. We then set 50% of owners and users to be honest. Let $\lambda = 0.2$, $n_p = 4$, i.e., $\lambda \geq \frac{1}{n_p+1}$. The final result

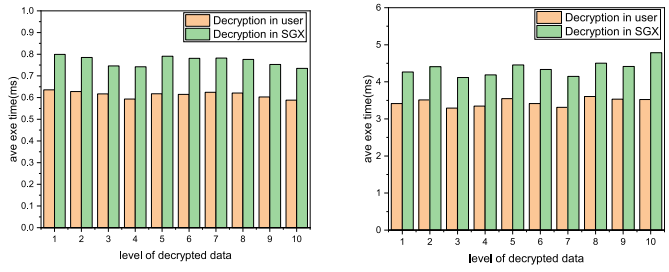


(a) data size=1MB (b) data size=5MB

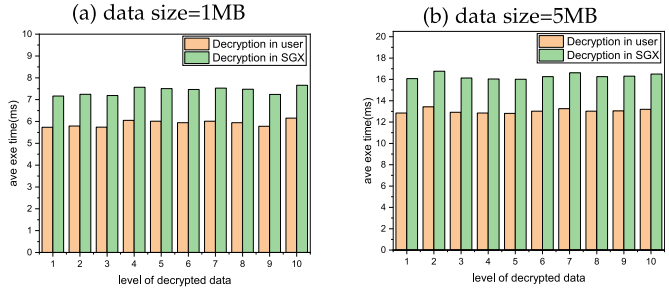


(c) data size=10MB (d) data size=20MB

Fig. 6. Data acquisition time in MOAC and MOCC.



(a) data size=1MB (b) data size=5MB



(c) data size=10MB (d) data size=20MB

Fig. 8. Decryption time of user or SGX in MOAC.

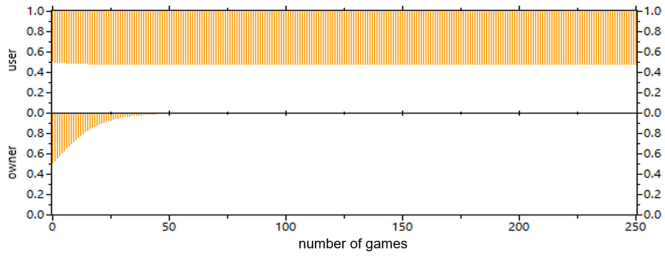


Fig. 9. Owner-user game when $\lambda \geq \frac{1}{n_p+1}$.

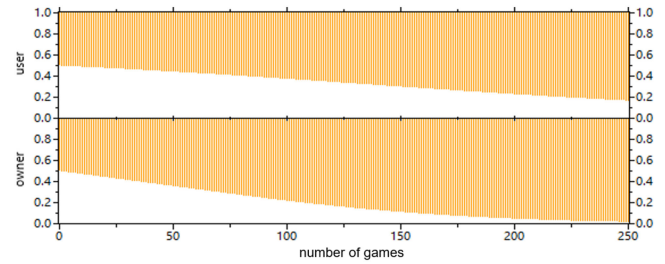
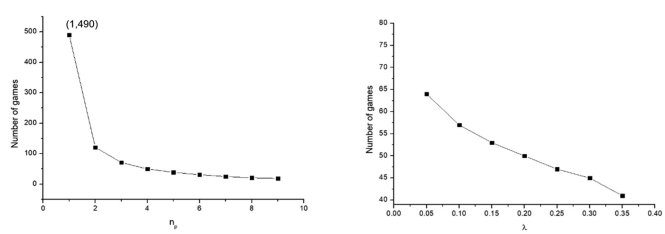


Fig. 10. Owner-user game when $\lambda < \frac{1}{n_p+1}$.

is depicted in Fig. 9, where the part shaded in yellow indicates dishonest entities; moreover, the abscissa denotes the number of games, while the ordinate indicates the population of honest and dishonest entities. From Fig. 9, we can observe that the ratio of honest and dishonest owners tends to be stable when the time of games is about 50. Moreover, with the progress of the game, the ratio of honest and dishonest users eventually tends to 1:1, and the dishonest owners gradually disappear. This occurs because our game has two NE, which does not affect the security of our design since we only need to ensure that owners remain honest. On the other hand, Fig. 10 shows the results when $\lambda = 0.2$ and $n_p = 0$, i.e., $\lambda < \frac{1}{n_p+1}$. We can observe that if cheating data owners are not punished, all data owners and users will eventually choose to be dishonest.

Furthermore, we evaluate the relationship between n_p and the number of games needed for the scheme to be stable when $\lambda = 0.2$ in Fig. 11a. We can find that the larger the value of n_p (i.e., the heavier the penalty), the faster the game converges to become stable. Moreover, Fig. 11b displays the relationship between λ and the number of games required for the scheme to become stable when $n_p = 4$. The larger the value of λ (i.e., the more undercover police), the faster the game reaches a stable status. The results are consistent with the actual situation. In addition, we can observe that when



(a) The relationship between n_p and the number of games

(b) The relationship between λ and the number of games

Fig. 11. Experimental results of data protection scheme.

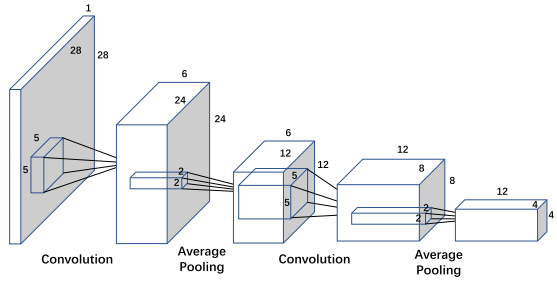


Fig. 12. DeepLearnToolBox CNN.

$\lambda = 0.2$, the scheme rapidly reaches a stable equilibrium when $n_p \geq 4$ (from Fig. 11a). With $n_p = 4$, it quickly reaches stable equilibrium when $\lambda \geq 0.2$ (shown in Fig. 11b). Therefore, $\lambda \geq \frac{1}{n_p+1}$ is a sufficient and unnecessary condition for a non-root owner to act honestly.

6.3 Implementation of the ESMAC

In this subsection, we deployed ESMAC in a real-life environment and evaluated the performance. Specifically, we considered the issue of multi-layer access to healthcare data. When HCC C wants to access the data and model from HCC B in order to train model $\mathbb{C}$, HCC C is required to make an application to HCC A and HCC B to obtain tokens. HCC C then sends a data processing request to CAS. The Enclave decrypts the data using the recovered symmetric key and trains the CNN to obtain model $\mathbb{C}$. We deployed and evaluated ESMAC in this multi-layer healthcare data scenario.

To conduct simulation experiments in real scenarios, we reconstructed a convolutional neural network (CNN) provided by DeepLearnToolbox[45], then trained the CNN model with NIH[43]. For the sampling settings of the data set, we used 784 sample pixels per image. Fig. 12 shows the architecture of our target CNN, featuring one input layer, two convolutional layers, and two pooling layers. Before conducting the simulation, we first obtained the optimal parameter settings through testing. As shown in Fig. 13, the best training results are achieved with the number of epochs set to 400 and a batch size of 4. When running ESMAC with the CNN training, the enclave is running a memory of 194MB, which does not exceed the 256MB limit of the EPC.

6.3.1 *Evaluation of Data Processing in Enclave*

To show the influence of SGX on performance, we further conducted some comparative experiments. First, we tested the execution time of training the same CNNs with the raw NIH dataset[43] in the cloud and inside Enclave, respectively. From Fig. 14, we can see that it takes an additional overhead

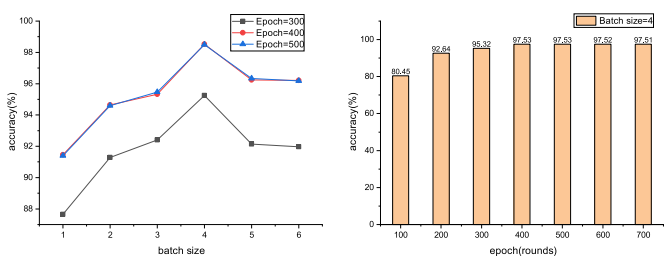


Fig. 13. Parameters setting and accuracy.

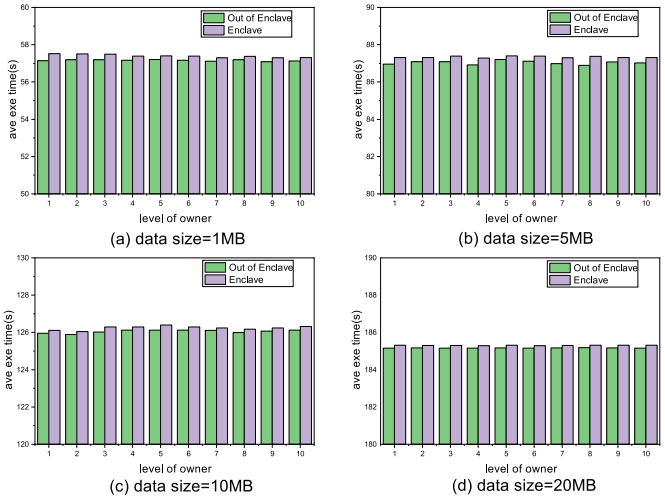


Fig. 14. Time consumption inside and outside the Enclave.

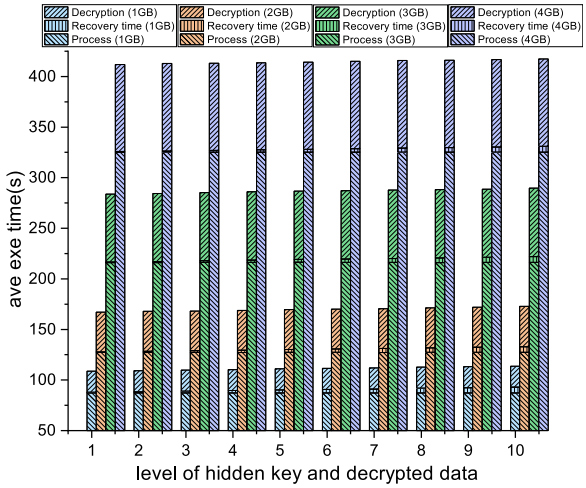


Fig. 16. ESMAC execution time.

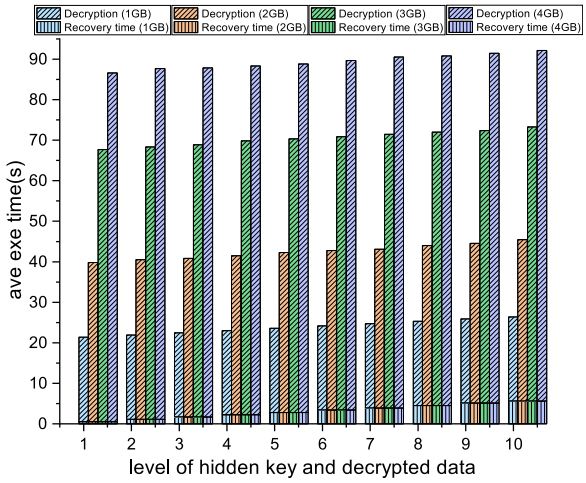


Fig. 17. Recovery and decryption time.

of less than 1% to process data in the SGX Enclave. As the data volume increases from 1GB to 4GB, the impact of the additional overhead incurred due to data processing in the Enclave is reduced. Moreover, our scheme preserves data privacy.

Second, we also conducted a comparison with CryptoNets [46] to show the superiority of the proposed scheme. CryptoNets is a privacy-preserving neural network framework implemented by homomorphic encryption. ESMAC also achieves privacy protection by migrating data to the Enclave to perform data processing in plaintext. From Fig. 15, we can observe that ESMAC is much more efficient and achieves a higher accuracy because CryptoNets incurs some errors when using polynomial to fit the activation function.

6.3.2 Performance Evaluation of ESMAC

In order to show and evaluate the execution of ESMAC more realistically, we deployed ESMAC to evaluate the overall runtime and measure the storage overhead. In the ESMAC scheme, the time required to implement multi-layer data access control management includes three parts: symmetric key recovery, data decryption, and data processing. When a user request is received, the Enclave starts executing ESMAC. The symmetric key $sk_{i,j}$ is first recovered based on the identity information provided by the user and the shared secret components from data owners. The ciphertext $CT_{m_{i,j}}$ is decrypted by $sk_{i,j}$, after which the Enclave performs data processing according to the user's command cd. We evaluated the whole processing overhead with different data sizes and ownership levels in the experimental configuration previously described.

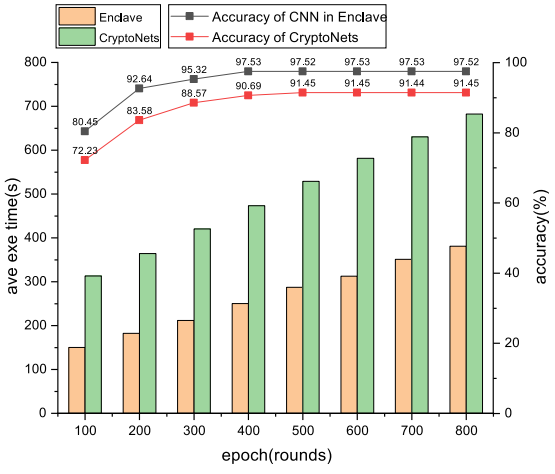


Fig. 15. Performance comparison with CryptoNets.

As shown in Fig. 16, the time spent on data processing and decryption increases with the size of the data, which is irrelevant to ownership level. However, key recovery time is affected by the number of ownership levels: the more complex the structure of ownership, the more time it takes to recover the key. Fig. 17 shows the execution time to recover a key and decrypt data with regard to data size and the level of hidden and decrypted data.

We also tested the storage overhead of the owner side and the cloud side while running ESMAC. Data owners should pre-generate and store the tokens for its potential users, which introduces extra storage overhead. As shown in Fig. 18a, it can be observed that ESMAC does not incur high storage cost to the owner. The cloud side should store datasets and all related access control lists and tokens in the MLDP scenario. As shown in Fig. 18b, we tested the storage cost of the cloud side to store the access control lists and tokens with different numbers of data owners and data

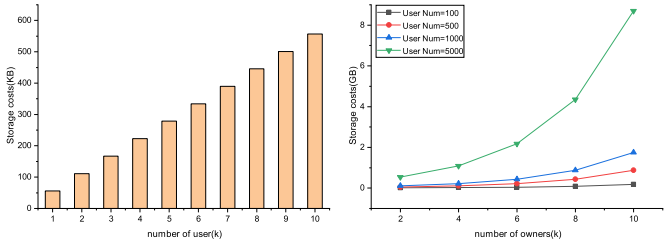


Fig. 18. Storage overhead of ESMAC for access control.

users for each owner, respectively. It can be found that it takes about 9GB to store related parameters when the total number of data users achieves 50k, which is acceptable for the cloud with enormous resources.

7 Conclusion

In this paper, we utilized SGX to design ESMAC, an efficient and secure multi-owner access control scheme to address access authorization over hierarchically structured data co-owned by multiple owners. We designed a novel key management scheme to generate key components for all data owners through key splitting. ESMAC supports trusted multiple rounds of data processing and multi-owner access control over MLDP results in a hierarchical way. Moreover, to discourage non-root data owners from behaving maliciously in pursuit of profit, we further proposed the game theory-DP scheme by introducing undercover polices, who supervise the behaviors of non-root data owners and prevent them from illegally disclosing co-owned data. Security analysis and performance evaluation demonstrate ESMAC's security and efficiency. In the future, we will further investigate how to hide the identity of the undercover police and guarantee the ownership across different platforms in order to enhance identity privacy and improve ESMAC's applicability.

References

[1] Q. Huang, Y. Yang, W. Yue, and Y. He, "Secure data group sharing and conditional dissemination with multi-owner in cloud computing," *IEEE Trans. Cloud Comput.*, vol. 9, no. 4, pp. 1607–1618, Fourth Quarter 2021.

[2] M. Layouni, M. Yoshida, and S. Okamura, "Efficient multi-authorizer accredited symmetrically private information retrieval," in *Proc. Int. Conf. Informat. Commun. Secur.*, 2008, pp. 387–402.

[3] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Secur. Privacy*, 2007, pp. 321–334.

[4] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. 13th ACM Conf. Comput. Commun. Secur.*, 2006, pp. 89–98.

[5] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Trans. Inf. Syst. Secur.*, vol. 9, no. 1, pp. 1–30, 2006.

[6] F. McKeen et al., "Innovative instructions and software model for isolated execution," in *Proc. 2nd Int. Workshop Hardware Architectural Support Secur. Privacy*, 2013, Art. no. 1.

[7] Y. Chen, Q. Zheng, Z. Yan, and D. Liu, "QShield: Protecting outsourced cloud data queries with multi-user access control based on SGX," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 2, pp. 485–499, Feb. 2021.

[8] V. Costan and S. Devadas, "Intel SGX explained," *IACR Cryptol. ePrint Arch.*, vol. 2016, no. 86, pp. 1–118, 2016.

[9] K. Thomas, C. Grier, and D. M. Nicol, "unfriendly: Multi-party privacy risks in social networks," in *Proc. Int. Symp. Privacy Enhancing Technol. Symp.*, 2010, pp. 236–252.

[10] K. Xu, Y. Guo, L. Guo, Y. Fang, and X. Li, "My privacy my decision: Control of photo sharing on online social networks," *IEEE Trans. Dependable Secure Comput.*, vol. 14, no. 2, pp. 199–210, Mar./Apr. 2017.

[11] N. Vishwamitra, Y. Li, K. Wang, H. Hu, K. Caine, and G.-J. Ahn, "Towards PII-based multiparty access control for photo sharing in online social networks," in *Proc. 22nd ACM Symp. Access Control Models Technol.*, 2017, pp. 155–166.

[12] H. Hu and G.-J. Ahn, "Multiparty authorization framework for data sharing in online social networks," in *Proc. IFIP Annu. Conf. Data Appl. Secur. Privacy*, 2011, pp. 29–43.

[13] H. Hu, G.-J. Ahn, and J. Jorgensen, "Multiparty access control for online social networks: Model and mechanisms," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 7, pp. 1614–1627, Jul. 2013.

[14] N. C. Rathore and S. Tripathy, "Collaborative access control model for online social networks," in *Proc. IEEE 6th Int. Conf. Adv. Comput.*, 2016, pp. 19–24.

[15] J. M. Such and N. Criado, "Resolving multi-party privacy conflicts in social media," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 7, pp. 1851–1863, Jul. 2016.

[16] H. Hu, G.-J. Ahn, Z. Zhao, and D. Yang, "Game theoretic analysis of multiparty access control in online social networks," in *Proc. 19th ACM Symp. Access Control Models Technol.*, 2014, pp. 93–102.

[17] M. Layouni, M. Yoshida, and S. Okamura, "Efficient multi-authorizer accredited symmetrically private information retrieval," in *Proc. Int. Conf. Inf. Commun. Secur.*, 2008, pp. 387–402.

[18] D. Hu, C. Hu, Y. Fan, and X. Wu, "oGBAC-A group based access control framework for information sharing in online social networks," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 1, pp. 100–116, Jan./Feb. 2021.

[19] S. Qi and Y. Zheng, "Crypt-DAC: Cryptographically enforced dynamic access control in the cloud," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 2, pp. 765–779, Mar./Apr. 2021.

[20] J. Winter, "Trusted computing building blocks for embedded linux-based ARM trustzone platforms," in *Proc. 3rd ACM Workshop Scalable Trusted Comput.*, 2008, pp. 21–30.

[21] S. Contiu, R. Pires, S. Vaucher, M. Pasin, P. Felber, and L. Réveillère, "IBBE-SGX: Cryptographic group access control using trusted execution environments," in *Proc. IEEE/IFIP 48th Annu. Int. Conf. Dependable Syst. Netw.*, 2018, pp. 207–218.

[22] J. B. Djoko, J. Lange, and A. J. Lee, "Nexus: Practical and secure access control on untrusted storage platforms using client-side SGX," in *Proc. IEEE/IFIP 49th Annu. Int. Conf. Dependable Syst. Netw.*, 2019, pp. 401–413.

[23] Y. Fan, S. Liu, G. Tan, and X. Lin, "CSCAC: One constant-size cpabe access control scheme in trusted execution environment," *Int. J. Comput. Sci. Eng.*, vol. 19, no. 2, pp. 162–168, 2019.

[24] S. Contiu, S. Vaucher, R. Pires, M. Pasin, P. Felber, and L. Réveillère, "Anonymous and confidential file sharing over untrusted clouds," in *Proc. IEEE 38th Symp. Reliable Distrib. Syst.*, 2019, pp. 21–2110.

[25] C. Delerablée, "Identity-based broadcast encryption with constant size ciphertexts and private keys," in *Proc. Int. Conf. Theory Appl. Cryptology Inf. Secur.*, 2007, pp. 200–215.

[26] Y. Fan, S. Liu, G. Tan, and F. Qiao, "Fine-grained access control based on trusted execution environment," *Future Gener. Comput. Syst.*, vol. 109, pp. 551–561, 2020.

[27] J. Ning, X. Huang, W. Susilo, K. Liang, X. Liu, and Y. Zhang, "Dual access control for cloud-based data storage and sharing," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 2, pp. 1036–1048, Feb. 2022.

[28] W. Ding, Z. Yan, and R. H. Deng, "Privacy-preserving data processing with flexible access control," *IEEE Trans. Dependable Secure Comput.*, vol. 17, no. 2, pp. 363–376, Mar./Apr. 2020.

[29] W. Ding et al., "An extended framework of privacy-preserving computation with flexible access control," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 2, pp. 918–930, Jun. 2020.

[30] Y. Miao et al., "Privacy-preserving attribute-based keyword search in shared multi-owner setting," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 3, pp. 1080–1094, May/Jun. 2021.

[31] O. Ohrimenko et al., "Oblivious multi-party machine learning on trusted processors," in *Proc. 25th USENIX Conf. Secur. Symp.*, 2016, pp. 619–636.

[32] D. Kim et al., "SGX-LEGO: Fine-grained SGX controlled-channel attack and its countermeasure," *Comput. Secur.*, vol. 82, pp. 118–139, 2019.

[33] V. Costan, I. Lebedev, and S. Devadas, "Sanctum: Minimal hardware extensions for strong software isolation," in *Proc. 25th USENIX Secur. Symp.*, 2016, pp. 857–874.
[34] S. Fei, Z. Yan, W. Ding, and H. Xie, "Security Vulnerabilities of SGX and Countermeasures: A Survey," *ACM Comput. Surv.s*, vol. 54, no. 6, pp. 1–36, 2021.
[35] W. Ding, W. Sun, Z. Yan, and R. H. Deng, "An efficient and secure scheme of verifiable computation for intel SGX," *CoRR*, vol. abs/2106.14253, 2021. [Online]. Available: https://arxiv.org/abs/2106.14253
[36] I. Anati, S. Gueron, S. Johnson, and V. Scarlata, "Innovative technology for cpu based attestation and sealing," in *Proc. 2nd Int. Workshop Hardware Architectural Support Secur. Privacy*, 2013, Art. no. 7.
[37] R. B. Myerson, *Game Theory*. Cambridge, MA, USA: Harvard Univ. Press, 2013.
[38] A. Fielder, E. Panaousis, P. Malacaria, C. Hankin, and F. Smeraldi, "Game theory meets information security management," in *Proc. IFIP Int. Inf. Secur. Conf.*, 2014, pp. 15–29.
[39] Z. Wang, Y. Luo, and S.-C. Cheung, "Efficient multi-party computation with collusion-deterred secret sharing," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2014, pp. 7401–7405.
[40] Y. Chen, W. Sun, N. Zhang, Q. Zheng, W. Lou, and Y. T. Hou, "Towards efficient fine-grained access control and trustworthy data processing for remote monitoring services in IoT," *IEEE Trans. Inf. Forensics Secur.*, vol. 14, no. 7, pp. 1830–1842, Jul. 2018.
[41] Z. Wang, S.-C. S. Cheung, and Y. Luo, "Information-theoretic secure multi-party computation with collusion deterrence," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 4, pp. 980–995, Apr. 2017.
[42] S. Szyller, B. G. Atli, S. Marchal, and N. Asokan, *DAWN: Dynamic Adversarial Watermarking of Neural Networks*. New York, NY, USA: Association for Computing Machinery, 2021, pp. 4417–4425.
[43] H. R. Roth et al., "Deeporgan: Multi-level deep convolutional networks for automated pancreas segmentation," in *Medical Image Computing and Computer-Assisted Intervention*, N. Navab, J. Hornegger, W. M. Wells, and A. Frangi, Eds. Cham, Switzerland: Springer, 2015, pp. 556–564.
[44] R. Wyttenbach, "Gamebug software," 2011. [Online]. Available: http://ess.nbb.cornell.edu/download.html
[45] R. B. Palm, "Prediction as a candidate for learning deep hierarchical models of data," Master's thesis, 2012. [Online]. Available: http://www.imm.dtu.dk/English.aspx
[46] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 201–210.

Dan Liu received the BEng degree in communication engineering from Jilin University, Changchun, China, in 2017. She is currently working toward the master's degree with the State Key Laboratory on Integrated Services Networks, Xidian University. Her research interests are in data analytics, privacy preservation and edge computing.

Zheng Yan (Senior Member, IEEE) received the BEng degree in electrical engineering and the MEng degree in computer science and engineering from the Xi'an Jiaotong University, Xi'an, China in 1994 and 1997, respectively, and other MEng degree in information security from the National University of Singapore, Singapore, in 2000, and the licentiate of science and the doctor of science in technology in electrical engineering from Helsinki University of Technology, Helsinki, Finland. She is currently a professor with the Xidian University, Xi'an, China. Her research interests are in trust, security, privacy, and security-related data analytics. She serves as a general or program chair for more than 30 international conferences and workshops. She is a steering committee co-chair of IEEE Blockchain international conference. She is also an area/associate editor of many reputable journals, e.g., *IEEE Internet of Things Journal*, *Information Sciences*, *Information Fusion*, *IEEE Network*, etc.

Wenxiu Ding (Member, IEEE) received the BEng degree and PhD degree in information security from Xidian University, Xi'an, China, in 2012 and 2017. She was the research assistant with the School of Information Systems, Singapore Management University from 2015 to 2016. Now, she is an associate professor with the School of Cyber Engineering with Xidian University. Her research interests include privacy preservation, access control and trust management.

Yuxuan Cai received the BEng degree in computer science and technology from Chongqing Technology and Business University, Chongqing, China, in June 2021. He is currently working toward the master degree in the School of Cyber Engineering, Xidian University. His research interests include privacy-preserving Federal Learning, homomorphic encryption, and SGX.

Yaxing Chen received the PhD degree in computer science from Xi'an Jiaotong University, in 2021. He was a visiting scholar with Virginia Tech from 2016 to 2018 and is currently an associate professor with the Northwestern Polytechnical University. His research interests lie on data security and privacy protection, confidential computing, and swarm intelligence, with focus on Enclaved cloud services, optimization between performance and privacy protection.

Zhiguo Wan (Member, IEEE) received the BS degree in computer science from Tsinghua University, in 2002 and the PhD degree from the School of Computing, National University of Singapore, Singapore, in 2007. He was an associate professor with the School of Computer Science and Technology, Shandong University. From 2008 to 2014, he was an assistant professor with the School of Software, Tsinghua University. He was a post-doctoral researcher with the Katholieke University of Leuven, Belgium, from 2006 to 2008. He is currently a principal investigator with Zhejiang Laboratory, Hangzhou, Zhejiang, China. His research interests include security and privacy for blockchain, cloud computing, and intelligent computing.