SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

February 2026, Vol. 69, Iss. 2, 122305:1–122305:17
https://doi.org/10.1007/s11432-025-4571-x

Forward secure bilateral access control for end-cloud collaborative Internet of Things

Keyong HONG[1], Jianfeng MA[1,2], Chuang TIAN[1,2*], Xiangyu WANG[1,2], Dilxat GHOPUR[3], Junwei ZHANG[1,2] & Xinghua LI[1,2]

[1]*College of Cyber Engineering, Xidian University, Xi'an 710126, China*
[2]*State Key Laboratory of Integrated Service Networks (ISN), Xidian University, Xi'an 710071, China*
[3]*College of Computer Science and Technology, Kashi University, Kashgar 844006, China*

Received 19 January 2025/Revised 24 May 2025/Accepted 28 August 2025/Published online 9 January 2026

Abstract In the Internet of Things (IoT) environment, the end-cloud collaborative architecture enables secure sharing and processing of massive data between the terminals and the cloud. It highlights establishing trusted transmission and computing mechanisms to ensure security over the data's full lifecycle. However, privacy issues about illegal access and forgery of sensitive data are increasingly severe during the computation, storage, and distribution process, thus hindering the development of IoT. Although the bilateral access control methods for end-cloud collaboration enforce access control on user privilege and ensure data source credibility through authentication mechanisms, they struggle to balance forward security and communication cost, resulting in limited key updating efficiency and policy flexibility. To address these practical issues, this paper proposes a novel forward secure bilateral access control scheme named FSBiAC for end-cloud collaborative IoT, which draws on the idea of matchmaking encryption and puncturable encryption. Our scheme outsources a significant portion of puncture tasks to reduce local updating overhead, while the bi-directional match between policies and attributes ensures fine-grained bilateral access control. Detailed proofs demonstrate that FSBiAC achieves semantic security under selectively chosen plaintext attacks (IND-sCPA) and existential unforgeability under chosen message attacks (EUF-CMA). Simulation shows that FSBiAC realizes superior computation and storage overhead compared to the previous studies.

Keywords end-cloud collaboration, forward security, fine-grained bilateral access control, puncture outsourcing, Internet of Things

Citation Hong K Y, Ma J F, Tian C, et al. Forward secure bilateral access control for end-cloud collaborative Internet of Things. Sci China Inf Sci, 2026, 69(2): 122305, https://doi.org/10.1007/s11432-025-4571-x

1 Introduction

The Internet of Things (IoT) [1] integrates data sensing, aggregation, computation, and storage. It fundamentally changes communication between the terminals and the cloud, thereby realizing the connection of all things. As an important carrier for secure data sharing, cloud-based cryptographic computation plays a crucial role in processing vast amounts of data. After years of wireless communication applications, the end-cloud collaborative model [2, 3] further enhances the agility and real-time capabilities of cloud services [4]. Specifically, IoT terminals perform privacy-preserving computation tasks to respond locally, while the cloud offers communication resources and storage capacity [5]. End-cloud collaborative IoT improves resource usage and transmission efficiency through coordination at the physical, network, and application layers, as well as cooperation between the devices and the cloud.

As the IoT network grows, the explosive growth of data and its diverse formats present significant challenges for storage and management [6, 7]. Furthermore, during the computation, storage, and distribution of encrypted data, privacy incidents caused by illegal access and forgery occur frequently. These incidents range from personal privacy leaks to infrastructure attacks, thereby posing a serious threat to the stable operation of IoT [8]. Data collected by lightweight terminals are constantly exposed to the security threats of forgery and tampering [9]. As one of the general techniques for tamper resistance, traditional authentication methods rely on identity validation, but their source identification efficiency remains debatable for massive devices. In addition, it is not trivial to efficiently handle complex access requests in current access control mechanisms for the one-to-one communication model [10], due to the increasing user scale and high heterogeneity of IoT. How to simultaneously achieve fine-grained access control for privacy resources [11] and fine-grained authentication for source identification [10] is a crucial issue in IoT security. The former allows access control of confidential data at a more precise level according to the predefined

* Corresponding author (email: chuangtian@proton.me)

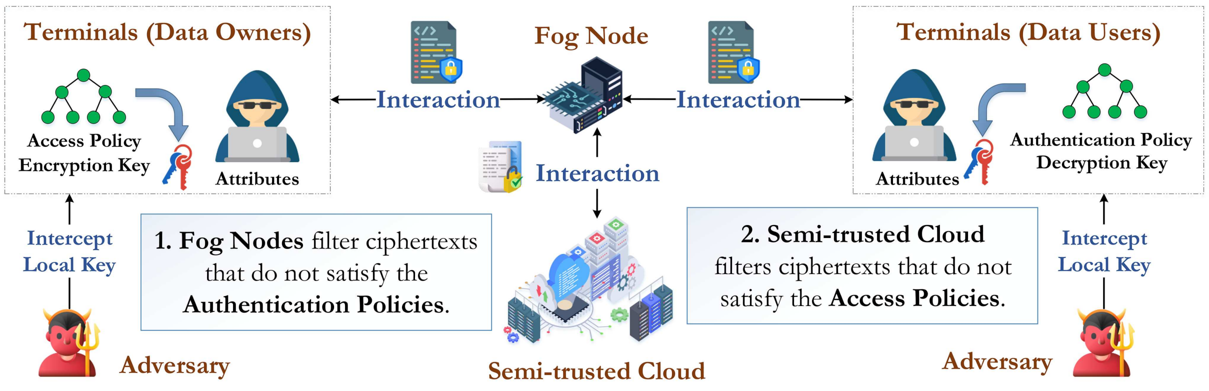

Figure 1 (Color online) Fine-grained bilateral access control in end-cloud computing.

access policies, whereas the latter identifies the sender in a fine-grained manner through the authentication policies, ensuring source legitimacy.

To achieve both authenticity and privacy, the bilateral access control methods [12] arise at an opportune time. It requires bi-directional privilege verification before data transmission [13, 14], which allows the recipients to identify the source without decrypting the data. Besides preventing unauthorized entities from accessing sensitive data, the dual restriction ensures that both parties meet each other's policies. Matchmaking encryption (ME) [14–16] for bilateral access control is suitable for the cloud computing paradigm and provides on-demand services in edge networks. As shown in Figure 1, end-cloud collaborative architecture extends the functionality of cloud computing. It enables source identification and outsourced data confidentiality. The fog nodes filter out ciphertexts that do not comply with the authentication policies, while the cloud filters out ciphertexts that do not meet the access policies. However, as a common issue, the current ME methods perform inadequately in preventing local key compromise and historical data leakage. If the decryption key suffers from exposure attacks, adversaries may attempt to decrypt past communication content, thereby threatening the entire transmission [17]. Under the key rotation, it is impractical for low-power terminals to frequently perform global updates. To address these issues in ME methods, forward security [18] is implemented to ensure that long-term transmissions do not expose historical data due to key leakage. In other words, data privacy has no relation to the historical key state, so as to realize the secure isolation of historical communication content.

As a significant technology for achieving forward security, puncturable encryption (PE) obtained popularity among researchers. It enables targeted revocation of decryption capabilities by embedding specific labels within the local key [19]. As a result, the fine-grained bilateral access control combines attribute-based matching encryption (ABME) [20] and PE emerged [21, 22]. Nevertheless, they face the bottlenecks of high storage overhead, large communication loads, and complex key management in many-to-many communication models. Resource-limited terminals perform local key updates and experience longer processing times. Besides, the local storage cost for keys multiplies with the increasing number of updates [23]. Consequently, there is an enormous scope for current ABME methods to develop in supporting both lightweight bilateral access control and forward security. Designing a forward secure bilateral access control method with secure lightweight updates becomes a pressing issue. Considering the resource constraints and the extensive network distribution, this paper follows the pattern of ME and presents a novel ABME protocol with secure outsourced updating that supports fine-grained bilateral access control. It achieves a reasonable balance between secure access control, source identification, and lightweight performance.

1.1 Contribution

To meet the practical demand for secure sharing, alleviate the issues posed by large-scale terminals with limited resources, and ensure data source credibility, we propose a novel fine-grained bilateral access control scheme for end-cloud collaborative IoT. The contributions are as follows.

(1) We propose a novel forward secure bilateral access control scheme with fine-grained access control and authentication, called FSBiAC. By employing linear matching between the outsourced decryption key, policy token, and encrypted file, FSBiAC satisfies the bi-directional verification between data owners and data users. Different from traditional PE, the puncture operation is divided into outsourced updating and local updating, thus a significant portion of updates is offloaded to the cloud. This process not only ensures forward security and key exposure attack defense, but also reduces the users' local computation and storage costs.

(2) We define a semantic security model under selective chosen plaintext attacks (IND-sCPA) and an existential unforgeability model under chosen message attacks (EUF-CMA). Rigorous security analysis proves that FSBiAC can realize privacy and authenticity under two q-type parallel bilinear Diffie-Hellman exponent assumptions, respectively.

(3) The experimental results show that FSBiAC has significant computational advantages in the local update, encryption, and decryption operations with the Java pairing-based cryptography (JPBC) library [24]. The storage complexity of local update will not increase with the number of punctures, which is only 788, 908, and 404 Bytes for Type-E, Type-A1, and Type-A curves, respectively. When the numbers of shares, attributes, and labels are all 50, the computational efficiency of encryption and decryption on real-world datasets [25] is improved by about 1.5 times and 40 times, respectively, compared to [21].

1.2 Related work

To realize both privacy and authenticity, Ateniese et al. [26] first introduced the concept of ME and constructed an identity-based encryption (IBE) scheme for policy-matched communication between the senders and receivers. They proved its security under the random oracle model based on the bilinear Diffie-Hellman (BDH) assumption. Since then, numerous identity-based cryptographic techniques [12, 27–29] have been designed for ME, but none have ensured the applicability of fine-grained access control in large-scale endpoint environments. Besides, predicate functions in identity-based policies require exponential-sized descriptions and therefore have limited flexibility. Several lightweight schemes [7, 30–33] offer solid solutions to relieve insufficient granularity for end-cloud collaborative scenarios. Tanveer et al. [7] proposed a lightweight authentication protocol for smart healthcare, which is based on a random-or-real method to mitigate the inherent risks in the public channel. The flexible attribute-based encryption (ABE) schemes were proposed by Li et al. [30, 31], which support user revocation and addition at the group level without disclosing the files and the keys. Nevertheless, Refs. [7, 30, 31] unilaterally pursued lightweight access control or authentication, resulting in the inability to balance privacy and authenticity. Wu et al. [32, 33] put forward the formal definitions for the fuzzy identity-based ME primitive. If the overlap between the tags of senders and receivers exceeds a certain threshold, the encrypted files can be correctly decrypted. Since static matching of tags, the granularity of both two schemes is still not suitable for a wireless network with large amounts of data. Xu et al. [13, 14, 34] designed several ABME schemes suitable for cloud-fog computing, which can optimize performance by offloading some computational tasks to fog nodes and providing bilateral access control. Nevertheless, they did not consider the impact of key leakage on communication content. Ma et al. [15] introduced the security properties of blindness and unlinkability, and designed an adaptively secure ME scheme by incorporating anonymous credentials. Huang et al. [20] designed a certificateless and revocable attribute-based bilateral access control scheme suitable for edge-cloud environments. To support the functionalities of attribute and identity revocation, however, the system needs to maintain both the revocation list and attribute information when the users' roles change. Zhang et al. [35] introduced a cloud-enabled publish/subscribe system that features attribute-based bilateral access control. It significantly improves matching efficiency and ensures the anonymity of publishers. Unfortunately, the data access privilege relies solely on the AND-Gate structure. Hu et al. [36] introduced a bilateral data-sharing scheme based on extended ME. It effectively addresses the privacy challenges in vehicular networks but has not escaped the one-to-one communication model in its revocation process.

On the other hand, PE primitive was first proposed by Green et al. [37]. They draw on the idea of puncturable pseudorandom function (PPRF) proposed by Goldreich [38]. Specifically, a constrained PPRF key can delete the pseudorandom output parts for specific inputs, such that PPRF no longer computes the values of those inputs. Hence, PPRF embodies a forward secure mechanism with a selective non-decryptable feature. Technically, PE and PPRF are not inherently related, but PE similarly restricts user keys from decrypting specific ciphertexts through label embedment. Phuong et al. [39] introduced attribute-based puncturable encryption (ABPE), which achieves targeted revocation for specific labels and ensures fine-grained access control. Several schemes for ABPE based on revocation trees were presented [23, 40]. Unfortunately, the parallel employment of puncture policies and access policies within the tree structure significantly affects the overall computational performance. In [23, 40], the initial puncture key, as a crucial component for local decryption, is exposed to untrusted clouds, greatly reducing the scheme's security. Additionally, the number of public parameters is fixed relative to the attribute size, which results in lacking scalable key management and flexible data distribution. Recently, Liu et al. [41] proposed a puncturable and traceable broadcast encryption scheme that facilitates the hash uniform distribution to prevent malicious encryptors. Nie et al. [21] constructed an attribute-based puncturable and matchmaking encryption scheme, which realizes bi-directional fine-grained access control, data authenticity, and forward security. A fog-based online ride-hailing system was designed in [22], capable of both fine-grained and bilateral matching. It ensures the authenticity of passenger orders and meets the time constraints of passenger requests. Besides, this scheme allows both parties

Table 1 Functional comparison in various schemes. – : meaningless; ✓ : support; ✗ : non-support; Tree: access tree structure; LSSS: linear secret sharing scheme; $\mathcal{F}_1$: IND-sCPA (indistinguishability under selective chosen plaintext attacks); $\mathcal{F}_2$: IND-CCA (indistinguishability under chosen ciphertext attacks); $\mathcal{F}_3$: IND-aCPA (indistinguishability under adaptive chosen plaintext attacks); $\mathcal{F}_4$: EUF-CMA (existential unforgeability under chosen message attacks).

Characterization	Ref. [7]	Ref. [12]	Ref. [14]	Ref. [20]	Ref. [21]	Ref. [22]	Ref. [23]	Ref. [32]	Ref. [35]	Ref. [36]	Ref. [40]	FSBiAC
Bilateral access control	✗	✓	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓
Forward security	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗	✓	✓
Lightweight key rotation	✓	✗	✗	✓	✗	✗	✓	✗	✗	✓	✓	✓
Outsourced decryption	✓	✗	✓	✓	✗	✗	✓	✗	✓	✓	✓	✓
Exposure attacks defense	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✓
Large universe application	✗	✗	✓	✗	✓	✓	✗	✗	✓	✗	✗	✓
Access structure	–	–	LSSS	–	LSSS	LSSS	Tree	Tree	–	LSSS	Tree	LSSS
Security model	–	$\mathcal{F}_1\mathcal{F}_3$	$\mathcal{F}_1\mathcal{F}_4$	$\mathcal{F}_2$	$\mathcal{F}_1\mathcal{F}_4$	$\mathcal{F}_1\mathcal{F}_4$	$\mathcal{F}_1$	$\mathcal{F}_1\mathcal{F}_4$	$\mathcal{F}_4$	–	$\mathcal{F}_1$	$\mathcal{F}_1\mathcal{F}_4$

to specify policies, with encrypted orders only revealed when both parties' policies are matched. Nevertheless, in [21,22], the puncture process and key maintenance are performed locally by the receivers, which results in heavy communication and storage overhead. Thus, these schemes are unsuitable for resource-limited and large-scale terminal devices.

The functional comparison of FSBiAC with the current studies is provided in Table 1. In summary, PE achieves forward security by embedding labels into keys or ciphertexts. It provides solid content protection and defends against malicious key attacks. Simply combining PE with ME obviously fails to meet the dual requirements of efficient key updates and ciphertext privacy. Current bilateral access control methods cannot efficiently address the issue of historical data exposure after key leakage, due to neglect of the balance between the computational cost of key updates and forward security features. Besides, some schemes lack policy flexibility and remain limited to the one-to-one communication model.

Roadmap. We provide the preliminaries in Section 2. Section 3 introduces the specific system entities, threat model, algorithm description, and security model, followed by concrete construction and correctness analysis. Section 4 describes IND-sCPA and EUF-CMA in detail. Section 5 analyzes the theoretical and experimental performance of our scheme, respectively. Section 6 concludes this paper.

2 Preliminaries

This section provides some preliminaries, including linear secret sharing scheme, share generation matrix, Lagrange interpolation function, $(q-1)$-type assumption, and $(q-2)$-type assumption.

Linear secret sharing scheme and share generation matrix. Assume $\mathbb{P}$ is a set of participants, and $\mathbb{M}$ is an $l \times n$ sharing generation matrix. Let $\rho(\cdot)$ be a mapping function that maps each row of the matrix $\mathbb{M}$ to a specific participant in $\mathbb{P}$. For the access structure $\Theta \subseteq 2^{\mathbb{P}} \backslash \emptyset$, the linear secret sharing scheme [42] consists of the following two steps.

(1) Secret division: This algorithm takes as input the secret value s to be shared. It randomly selects $\mu_2, \ldots, \mu_n$ from $\mathbb{Z}_p$ to form secret vector $\boldsymbol{\mu} = (s, \mu_2, \ldots, \mu_n)^\top$. Define $\boldsymbol{\lambda} = \mathbb{M} \cdot \boldsymbol{\mu}$, which means computing $\lambda_{\rho(k)} = \mathbb{M}_k \cdot \boldsymbol{\mu}$. It outputs $\left\{\lambda_{\rho(k)}\right\}_{k \in [1-l]}$ as the shares of secret value. Here, $\mathbb{M}_k$ denotes the k-th row of matrix $\mathbb{M}$, and the share $\lambda_{\rho(k)}$ belongs to participant $\rho(k)$.

(2) Secret reconstruction: This algorithm takes as input an authorized set $S \in \Theta$, and computes a set of constants $\{u_k\}_{k \in [1-l]} \subset \mathbb{Z}_p$ such that $\sum_{k \in I} \mathbb{M}_k \cdot u_k = (1, 0, \ldots, 0)$. Here, I represents the rows that satisfy $\rho(k) \in S$. It reconstructs secret value by computing $\sum_{k \in I} \lambda_{\rho(k)} \cdot u_k = s$.

Lagrange interpolation function. Given $(n+1)$ known points $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$, the Lagrange interpolation function can construct an interpolating polynomial through the above data points. The n-th degree polynomial $P(x)$ and the basis functions $L_i(x)$ can be expressed as $P(x) = \sum_{i=0}^{n} L_i(x) \cdot y_i$, in which $L_i(x) = \prod_{j \in [0,n]} \frac{x - x_j}{x_i - x_j}$ $(j \neq i)$. $L_i(x)$ takes the value 1 at $x = x_i$ and 0 at all other $x = x_j$. In consequence, $P(x)$ passes through all known points.

$(q-1)$-type assumption. Let g be the generator of the cyclic group G_0 with prime order p, and $\mathbb{Z}$ be a random element in G_T. Choose $a, \vartheta, b_1, \ldots, b_q$ from $\mathbb{Z}_p^*$. If any probabilistic polynomial time (PPT) adversary $\mathcal{A}$ cannot distinguish between $\left(\boldsymbol{y}, e(g,g)^{a^{q+1}\vartheta}\right)$ and $(\boldsymbol{y}, \mathbb{Z} \in G_T)$ with a non-negligible advantage, then (decisional) $(q-1)$-type assumption is said to hold [43]. The advantage of $\mathcal{A}$ is defined as $Adv_{\mathcal{A}}^{(q-1)} = |\Pr\left[\mathcal{A}(\mathbb{Z} = e(g,g)^{a^{q+1}\vartheta}, \boldsymbol{y}) = 1\right]$

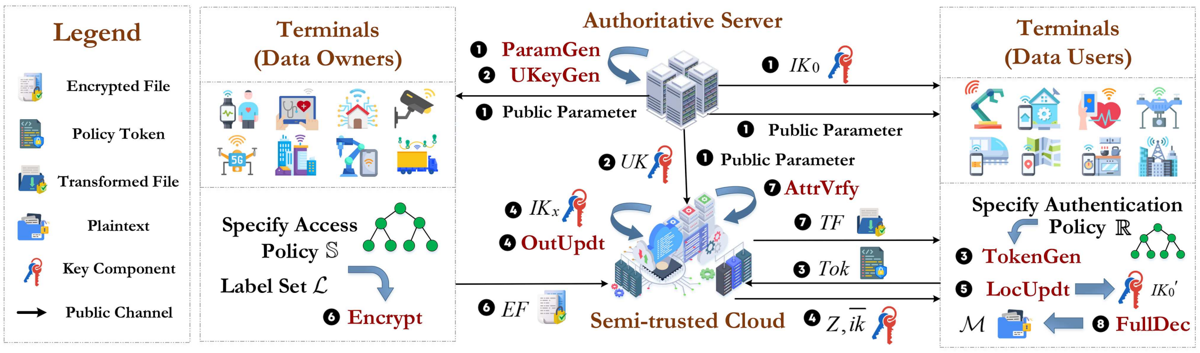

Figure 2 (Color online) Our bilateral access control architecture for end-cloud collaborative IoT.

$-\Pr[\mathcal{A}(\mathbb{Z} \in G_T, \boldsymbol{y}) = 1]|$. And $\boldsymbol{y}$ is given by

$$\Big[g, g^{\vartheta}, g^{1/a}, \{g^{a^i}, g^{b_j}, g^{\vartheta b_j}, g^{a^i b_j}, g^{a^i/b_j^2}, \forall(i,j) \in [q,q]\}, \{g^{a^i b_j/b_{j'}^2}, \forall(i,j,j') \in [2q,q,q], j \neq j'\},$$

$$\{g^{a^i/b_j}, \forall(i,j) \in [2q,q], j \neq q+1\}, \{g^{\vartheta a^i b_j/b_{j'}}, g^{\vartheta a^i b_j/b_{j'}^2}, \forall(i,j,j') \in [q,q,q], j \neq j'\}\Big].$$

$(q-2)$-type assumption. Let g be the generator of the cyclic group G_0 with prime order p, and $\mathbb{Z}$ be a random element in G_T. Choose $\vartheta, \varrho, \varsigma, b_1, \ldots, b_q$ from $\mathbb{Z}_p^*$. If any PPT adversary $\mathcal{F}$, given $\boldsymbol{y}$, can only compute $e(g,g)^{\vartheta\varrho\varsigma}$ with negligible probability $\Pr\left[\mathcal{F}^{(q-2)}(\boldsymbol{y}) = e(g,g)^{\vartheta\varrho\varsigma}\right]$, then (computational) $(q-2)$-type assumption holds [42]. And the vector $\boldsymbol{y}$ is given by

$$\Big[g, g^{\vartheta}, g^{\varrho}, g^{\varsigma}, g^{(\vartheta\varsigma)^2}, \{g^{b_i}, g^{\vartheta\varsigma b_i}, g^{\vartheta\varsigma/b_i}, g^{\vartheta^2\varsigma b_i}, g^{\varrho/b_i^2}, g^{\varrho^2/b_i^2}, \quad \forall i \in [q]\},$$

$$\{g^{\vartheta\varsigma b_i/b_j}, g^{\varrho b_i/b_j^2}, g^{\vartheta\varrho\varsigma b_i/b_j^2}, g^{(\vartheta\varsigma)^2 b_i/b_j}, \qquad \forall i,j \in [q,q], i \neq j\}\Big].$$

3 System overview and construction

This section provides the system architecture, security model, formal algorithms, and construction.

3.1 System architecture

As shown in Figure 2, FSBiAC involves four entities: authoritative server (AS), semi-trusted cloud (SC), data owner (DO), and data user (DU).

(1) AS: As a trusted entity, AS is responsible for generating system public parameters and master secret key, as well as generating decryption keys for DU based on attribute sets. **(2) SC:** Acting as an honest-but-curious cloud server, SC is responsible for performing several verification operations on encrypted files, outsourced keys, and policy tokens. Additionally, SC assists DU in securely computing the puncture of outsourced update key. **(3) DO:** As the data sender, DO is responsible for executing data encryption and sending encrypted files to SC. **(4) DU:** As the data receiver, DU is responsible for performing local key updating and decryption operation. Since a large portion of decryption tasks has been outsourced to SC, DU only needs to carry out lightweight decryption to recover the plaintext.

In our architecture, SC is a semi-honest and inquisitive server that honestly performs verification at the access policy layer and the authentication layer, but is interested in outsourced keys and encrypted files. It may generate illegal communication with the terminals.

3.2 Formal algorithm description

FSBiAC consists of the following eight algorithms, including **ParamGen**, **UKeyGen**, **TokenGen**, **OutUpdt**, **LocUpdt**, **Encrypt**, **AttrVrfy**, and **FullDec** algorithms.

$(pp, msk) \leftarrow$ **ParamGen**$(\mathbb{U}, 1^\kappa, d)$: This algorithm takes as input large attribute universe $\mathbb{U}$, system security parameter κ, and the maximum number of labels d, and outputs system public parameters pp and master secret key msk. It is executed by AS.

$(IK_0, UK) \leftarrow$ **UKeyGen**(pp, msk, Θ_R): This algorithm takes as input public parameters pp, master secret key msk, and DU's attribute set Θ_R, and outputs initial update key IK_0 and outsourced decryption key UK. It is executed by AS.

$Tok \leftarrow$ **TokenGen**$(pp, \mathbb{R})$: This algorithm takes as input public parameters pp and DU's authentication policy $\mathbb{R}$, and outputs policy token Tok. It is executed by DU.

$(Z, \overline{ik}, IK_x) \leftarrow$ **OutUpdt**(pp, msk, IK_{x-1}, l_x): This algorithm takes as input public parameters pp, master secret key msk, the last updated key IK_{x-1} and any length specific label $l_x \neq l_0$, and outputs accumulated value Z, commitment $\overline{ik}$ of δ, and outsourced update key IK_x. It is executed by SC.

$IK_0' \leftarrow$ **LocUpdt**(pp, IK_0'): This algorithm takes as input public parameters pp and current local update key IK_0', and outputs updated key. It is executed by DU.

$EF \leftarrow$ **Encrypt**$(\mathcal{M}, pp, \Theta_S, \mathcal{L}, \mathbb{S})$: This algorithm takes as input public parameters pp, plaintext $\mathcal{M}$, DO's attribute set Θ_S, label set $\mathcal{L}$, and DO's access policy $\mathbb{S}$, and outputs encrypted file EF. It is executed by DO.

$TF/\perp \leftarrow$ **AttrVrfy**(pp, EF, Tok, UK, IK_x): This algorithm takes as input public parameters pp, encrypted file EF, policy token Tok, outsourced update key IK_x, and outsourced decryption key UK, and finally outputs transformed file TF. It is executed by SC.

$\mathcal{M} \leftarrow$ **FullDec**(pp, IK_0', TF): This algorithm takes as input public parameters pp, the current local update key IK_0', and transformed file TF, and finally outputs plaintext $\mathcal{M}$. It is executed by DU.

3.3 Security model definition

To verify the security of FSBiAC, this section defines the security models for IND-sCPA and EUF-CMA.

IND-sCPA model. The security model for our FSBiAC is formally described. The game below captures the interactions between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. $\mathcal{A}$ first declares a target access policy $\mathbb{S}^* = (\mathbb{M}^*, \rho^*)$ and a target label set $\mathcal{L}^* = (l_1^*, \ldots, l_d^*)$.

(1) *Initialization.* $\mathcal{C}$ obtains public parameters pp and master secret key msk by **ParamGen**, then transfers pp to $\mathcal{A}$. Besides, $\mathcal{C}$ remains msk and sets up two empty list $\mathcal{E}_x$ and $\mathcal{E}_y$.

(2) *The first query phase.* After initializing a counter $V = 0$, $\mathcal{A}$ adaptively queries outsourced decryption key UK and the most recent update key IK_x. $\mathcal{C}$ responds as follows.

(a) *UKeyGen Query.* After the query being sent on an attribute set $\Theta_R = (\Theta_{R,1}, \ldots, \Theta_{R,k})$, $\mathcal{C}$ performs **UKeyGen** and returns IK_0 and UK, where Θ_R is not satisfied with $\mathbb{S}^*$. Lastly, $\mathcal{C}$ sends them to $\mathcal{A}$.

(b) *OutUpdt Query.* After the key query is sent on a label l_x and $(x-1)$-th outsourced update key, $\mathcal{C}$ performs **OutUpdt** and obtains $(IK_x, Z, \bar{ik}, l_x)$, where $l_x \notin \mathcal{L}^* = (l_1^*, \ldots, l_d^*)$. Then $\mathcal{C}$ sends IK_x to $\mathcal{A}$. Lastly, $\mathcal{C}$ increments the counter V and stores $(IK_x, l_x, Z, \bar{ik})$ in execution record $\mathcal{E}_x$.

(c) *LocUpdt Query.* $\mathcal{A}$ sends on a label l_x and $(x-1)$-th local update key to $\mathcal{C}$. $\mathcal{C}$ first checks if there is a record l_x in $\mathcal{E}_x$, where $l_x \notin \mathcal{L}^* = (l_1^*, \ldots, l_d^*)$. If the tuple $(l_x, Z, \bar{ik})$ exists, $\mathcal{C}$ executes **LocUpdt** and returns IK_0' to $\mathcal{A}$. If it does not exist, $\mathcal{C}$ simultaneously executes *OutUpdt Query* and *LocUpdt Query*, followed with incrementing V and storing $(IK_0', IK_x, l_x, Z, \bar{ik})$ in $\mathcal{E}_x$.

(d) *Corrupt Query.* This query is executed only once, i.e., when $\mathcal{A}$ issues for the first time, $\mathcal{C}$ assigns $\mathcal{E}_x$ to $\mathcal{E}_y$ and provides $\mathcal{A}$ with the last x-th update key (IK_0', IK_x). Subsequently, all *Corrupt Query* do not return any result. If $\{l_1, \cdots, l_d\} \cup \mathcal{E}_x = \emptyset$, it returns $\perp$.

(3) *Challenge.* Two same-size message $\mathcal{M}_0$ and $\mathcal{M}_1$ are selected by $\mathcal{A}$. It sends $(\mathcal{M}_0, \mathcal{M}_1, \mathbb{S}^*, \mathcal{L}^*)$ to $\mathcal{C}$. Notice that $\{l_1^*, \ldots, l_d^*\} \in \mathcal{L}^*$ has never been issued by *Corrupt Query.* Next, $\mathcal{C}$ takes pp, $\mathcal{M}_\psi$, $\mathbb{S}^* = (\mathbb{M}^*, \rho^*)$ and $\{l_1^*, \ldots, l_d^*\} \in \mathcal{L}^*$ as input, then runs **Encrypt**. It sends the challenge ciphertext EF^* to $\mathcal{A}$, where $\psi \in \{0, 1\}$.

(4) *The Second Query Phase.* $\mathcal{A}$ requests $\mathcal{C}$ with the same operations in the first query phase.

(5) *Guess.* $\mathcal{A}$ generates a guess bit $\psi' \in \{0, 1\}$, and wins in this game if $\psi = \psi'$. The advantage of adversary $\mathcal{A}$ is defined as $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{sCPA}} = |\Pr[\psi' = \psi] - 1/2|$.

Definition 1 (IND-sCPA)**.** Our scheme is secure under IND-sCPA if any PPT adversary has at most a negligible advantage in distinguishing $\psi \in \{0, 1\}$.

EUF-CMA model. The security model in EUF-CMA is formally described by the following EUF-CMA game. It captures the interaction between a polynomial time forger $\mathcal{F}$ and a challenger $\mathcal{C}$. $\mathcal{F}$ is allowed to access various oracles. It attempts to forge signatures or break encryption through interactions. FSBiAC ensures that even if $\mathcal{F}$ has query privileges, it cannot effectively forge valid ciphertext.

(1) *Initialization.* The forger $\mathcal{F}$ specifies a target attribute set $\Theta_S^* = (\Theta_{S,1}^*, \ldots, \Theta_{S,k}^*)$ and sends it to $\mathcal{C}$. $\mathcal{C}$ runs **ParamGen** to generate relevant public parameters pp and master secret key msk, and finally sends pp to $\mathcal{F}$.

(2) *Query Phase.* The challenger $\mathcal{C}$ provides two relevant oracles *TokenGen Query* and *Signature Query*, allowing the forger $\mathcal{F}$ to make queries. $\mathcal{C}$ responds as follows.

(a) *TokenGen Query.* $\mathcal{F}$ selects an authentication policy $\mathbb{R} = (\mathbb{N}, \sigma)$, then $\mathcal{C}$ runs **TokenGen** to generate the corresponding policy token Tok, and finally sends it to $\mathcal{F}$.

(b) *Signature Query.* $\mathcal{F}$ selects a plaintext $\mathcal{M}$, an authentication policy $\mathbb{R} = (\mathbb{N}, \sigma)$, and an attribute set $\Theta_S = (\Theta_{S,1}, \ldots, \Theta_{S,k})$. It is worth noting that $\mathbb{R} = (\mathbb{N}, \sigma)$ in *Signature Query* must not match Θ_S^*. $\mathcal{C}$ runs **Encrypt** to generate the signature components $\{\{C_{\tau,6}, C_{\tau,7}\}_{\tau\in[1-k]}, C_8, C_9\}$ corresponding to encrypted file. Finally, $\mathcal{C}$ returns the tuple to $\mathcal{F}$.

(3) *Forgery.* Once $\mathcal{F}$ completes the above *Query Phase*, it generates a forged signature $\{C_{\tau,6}^*, C_{\tau,7}^*, C_8^*, C_9^*\}_{\tau\in[1-k]}$ under the target attribute set $\Theta_S^* = (\Theta_{S,1}^*, \ldots, \Theta_{S,k}^*)$ that satisfies the target authentication policy $\mathbb{R}^* = (\mathbb{N}^*, \sigma^*)$. Note that $\mathcal{F}$ must not have queried the signature on Θ_S^*. Besides, the advantage of forger $\mathcal{F}$ is defined as $\mathrm{Adv}_{\mathcal{F}}^{\mathrm{CMA}}(1^\kappa) = \Pr[\mathrm{Exp}_{\mathcal{F}}^{\mathrm{CMA}}(1^\kappa) = 1]$.

Definition 2 (EUF-CMA). Our scheme is secure under EUF-CMA if any PPT forger $\mathcal{F}$ has at most a negligible advantage $\mathrm{Adv}_{\mathcal{F}}^{\mathrm{CMA}}(1^\kappa)$ in forging signatures.

3.4 Forward secure bilateral access control construction

According to the formal definitions in Section 3.2, we describe the concrete construction.

ParamGen: This algorithm takes as input a large attribute universe $\mathbb{U}$, system security parameter κ, and the maximum number of labels d. It randomly selects a multiplicative cyclic group G_0 and G_T with prime order p and generator g, and designates a mapping $e : G_0 \times G_0 \to G_T$. Then, it selects w, v, u, h from G_T and α, β from $\mathbb{Z}_p$. It initializes label l_0 and randomly selects coefficients for a d-degree polynomial from $\mathbb{Z}_p$. Subsequently, it defines $Q(\cdot) = g^{q(\cdot)}$ and $q(0) = \beta$. Two collision-resistant hash functions $H_1 : \{0,1\}^* \to \mathbb{Z}_p$ and $H_2 : \{0,1\}^* \to G_0$ are defined. Finally, this algorithm outputs system public parameters $pp = \{w, v, u, h, g, Q(1), \ldots, Q(d), e(g,g)^\alpha, g^\beta, l_0, H_1, H_2\}$ and master secret key $msk = \alpha$.

UKeyGen: This algorithm takes as input pp, msk, and DU's attribute set $\Theta_R = (\Theta_{R,1}, \ldots, \Theta_{R,k})$. It selects random ω, t from $\mathbb{Z}_p$, and calculates the initial update key $IK_0 = ik_{[0,1-4]}$:

$$ik_{0,1} = g^\omega, \quad ik_{0,2} = l_0, \quad ik_{0,3} = (g^\beta)^{\omega+t}, \quad ik_{0,4} = Q(H_1(l_0))^\omega.$$

It selects random elements $r, r_1, \ldots, r_k$ from $\mathbb{Z}_p$ and outputs the outsourced decryption key $UK = uk_{[1-4]}$, where $\tau \in [1-k]$:

$$uk_1 = (g^\beta)^t g^\alpha w^r, \quad uk_2 = g^r, \quad uk_{\tau,3} = g^{r_\tau}, \quad uk_{\tau,4} = \left(u^{\Theta_{R,\tau}} h\right)^{r_\tau} v^{-r}.$$

TokenGen: This algorithm takes as input pp and DU's authentication policy $\mathbb{R} = (\mathbb{N}, \sigma)$, where $\mathbb{R}$ contains an $l \times n$ matrix $\mathbb{N}$ (the matrix elements all belonging to $\mathbb{Z}_p$) and a mapping $\sigma : [1-l] \to \mathbb{Z}_p$. It selects random $v_2, \ldots, v_n$ from $\mathbb{Z}_p$ to form secret vector $\boldsymbol{v} = (1, v_2, \ldots, v_n)^\top$, and computes $\boldsymbol{\pi} = \mathbb{N} \cdot \boldsymbol{v} = (\pi_1, \pi_2, \ldots, \pi_l)^\top$. It calculates the row set I in vector $\boldsymbol{\pi}$ corresponding to DU's attributes Θ_R, represented as $I = \{i : \sigma(i) \in \Theta_R \wedge \sigma : [1-l] \to \mathbb{Z}_p\}$. It selects l random numbers γ_i from $\mathbb{Z}_p$, calculates and outputs policy token $Tok = \{T_{i,1}, T_{i,2}, T_{i,3}\}$:

$$T_{i,1} = \left(u^{\sigma(i)} h\right)^{-\gamma_i}, \qquad T_{i,2} = g^{\gamma_i}, \qquad T_{i,3} = g^{\pi_i} w^{\gamma_i}.$$

OutUpdt: This algorithm takes as input pp and an arbitrary length-specific tag $l_x \neq l_0$, then selects random $\delta, x_{x,0}, z_x$ from $\mathbb{Z}_p$. It calculates and outputs the accumulated value $Z = \sum_{\xi=1}^{x} z_\xi$, a commitment $\bar{ik} = (g^\beta)^\delta$ of δ, and an outsourced update key $IK_x = ik_{[x,1-4]}$:

$$ik_{x,1} = g^{z_x \cdot x_{x,0}/x}, \quad ik_{x,2} = l_x, \quad ik_{x,3} = (g^\beta)^{z_x \cdot (\delta + x_{x,0})/x}, \quad ik_{x,4} = Q(H_1(l_x))^{z_x \cdot x_{x,0}/x}.$$

LocUpdt: This algorithm takes as input pp, $\bar{ik}$ and the current local update key IK_0'. It selects a random $x_{x,1}$ from $\mathbb{Z}_p$, and then outputs new local update key $IK_0' = ik'_{[0,1-4]}$:

$$\begin{aligned} &ik'_{0,1} = g^{x_{x,1}} \cdot ik_{0,1} = g^{\omega + x_{x,1}}, \qquad ik'_{0,2} = ik_{0,2} = l_0, \\ &ik'_{0,3} = (g^\beta)^{x_{x,1}} / \bar{ik} \cdot ik_{0,3} = (g^\beta)^{\omega + t + x_{x,1} - \delta}, \\ &ik'_{0,4} = Q(H_1(l_0))^{x_{x,1}} \cdot ik_{0,4} = Q(H_1(l_0))^{\omega + x_{x,1}}. \end{aligned}$$

Encrypt: This algorithm takes as input *pp*, plaintext $\mathcal{M}$, DO's attribute set $\Theta_S = (\Theta_{S,1}, \ldots, \Theta_{S,k})$, label set $\mathcal{L} = (l_1, \ldots, l_d)$, and DO's access policy $\mathbb{S} = (\mathbb{M}, \rho)$. Here, $\mathbb{S}$ contains an $l \times n$ matrix $\mathbb{M}$ (matrix elements all belonging to $\mathbb{Z}_p$) and a mapping $\rho : [1-l] \to \mathbb{Z}_p$. It selects a secret value ϑ and random $\mu_2, \ldots, \mu_n$ from $\mathbb{Z}_p$ to form secret vector $\boldsymbol{\mu} = (\vartheta, \mu_2, \ldots, \mu_n)^\top$ and computes $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_l)^\top = \mathbb{M} \cdot \boldsymbol{\mu}$. It selects l random numbers η_j from $\mathbb{Z}_p$ and computes ciphertext components $\{C, \{C_{j,1}, C_{j,2}, C_{j,3}\}_{j \in [1-l]}\}$:

$$C = \mathcal{M} \cdot e(g,g)^{\alpha\vartheta}, \quad C_{j,1} = (u^{\rho(j)} h)^{-\eta_j}, \quad C_{j,2} = g^{\eta_j}, \quad C_{j,3} = w^{\lambda_j} v^{\eta_j}.$$

For $x \in [1-d]$, it computes $C_{x,4} = Q(H_1(l_x))^{\vartheta}$. It computes the commitment of ϑ as $C_5 = g^{\vartheta}$. Randomly selects $\varpi, s_1, \ldots, s_k$ from $\mathbb{Z}_p$ and calculates ciphertext components $\{C_{\tau,6} = g^{s_\tau}, C_{\tau,7} = (u^{\Theta_{s,\tau}} h)^{s_\tau} w^{-\varpi}, C_8 = g^{\varpi}, C_9 = g^{\varpi} \cdot H_2(C_{6-8})\}$1), where $\tau \in [1-k]$. The encrypted file EF is $\{\{C_{j,1}, C_{j,2}, C_{j,3}\}_{j \in [1-l]}, \{C_{x,4}\}_{x \in [1-d]}, C_5, \{C_{\tau,6}, C_{\tau,7}\}_{\tau \in [1-k]}, C_8, C_9, C, \mathbb{S}, \Theta_S\}$.

AttrVrfy: This algorithm takes as input *pp*, EF, Tok, IK_x, and UK.

• *Fine-Grained Authentication.* It selects a random number γ from $\mathbb{Z}_p$, and re-randomizes policy token $Tok' = \{T'_{i,1}, T'_{i,2}, T'_{i,3}, T_4\}$ as follows:

$$T'_{i,1} = (T_{i,1})^{\gamma} = (u^{\sigma(i)} h)^{-\gamma \cdot \gamma_i}, \quad T'_{i,2} = (T_{i,2})^{\gamma} = g^{\gamma \cdot \gamma_i}, \quad T'_{i,3} = (T_{i,3})^{\gamma} = g^{\gamma \cdot \pi_i} w^{\gamma \cdot \gamma_i}, \quad T_4 = g^{\gamma}.$$

If Θ_S is an authorized set, this algorithm can obtain a set of constants $\{c_i \in \mathbb{Z}_p\}_{i \in I}$ such that $\sum_{i \in I} c_i \pi_i = 1$, and thus the equation $\prod_{i \in I} \left(e(T'_{i,1}, C_{\tau,6}) \cdot e(T'_{i,2}, C_{\tau,7}) \cdot e(T'_{i,3}, C_8)\right)^{c_i} \cdot e(T_4, H_2(C_{6-8})) = e(T_4, C_9)$ holds. If Θ_S is an unauthorized set, this algorithm returns a termination symbol $\perp$.

• *Outsourced Puncture Decryption.* According to the interpolation function in Section 3.2, define $(d+1)$ function values as $\{H_1(ik_{x,2}), H_1(l_\ell)_{\ell \in [1-d]}\}$. Then for $\xi \in [x]^*$, it computes $(d+1)$ Lagrange coefficients $\{\chi^*, \chi_1, \ldots, \chi_d\}$ such that the equation $\chi^* \cdot q(H_1(ik_{x,2})) + \sum_{\ell=1}^{d} (q(H_1(l_\ell)) \cdot \chi_\ell) = q(0) = \beta$ holds. Afterwards, it performs the following calculation:

$$\mathcal{Q} = \prod_{\xi=1}^{x} \left(e\left(ik_{\xi,3}, C_5\right) / e\left(ik_{\xi,1}, \prod_{\ell=1}^{d} \left(C_{\ell,4}\right)^{\chi_\xi}\right) \cdot e\left(ik_{\xi,4}, C_5\right)^{\chi^*} \right).$$

If $\{\chi^*, \chi_1, \ldots, \chi_d\}$ cannot be computed, this algorithm returns $\perp$.

• *Fine-Grained Access Control.* The encrypted file EF contains DO's access policy $\mathbb{S} = (\mathbb{M}, \rho)$. Compute the row set J in $\boldsymbol{\lambda}$ corresponding to DU's attributes Θ_R, i.e., $J = \{j : \rho(j) \in \Theta_R \wedge \rho : [1-l] \to \mathbb{Z}_p\}$. If Θ_R is an authorized set, this algorithm can obtain a set of constants $\{f_j \in \mathbb{Z}_p\}_{j \in J}$ and compute the secret $\vartheta = \sum_{j \in J} f_j \lambda_j$, and thus successfully compute the following equation:

$$\mathcal{P} = e\left(C_5, uk_1\right) / \prod_{j \in J} \left(e\left(C_{j,3}, uk_2\right) \cdot e\left(C_{j,1}, uk_{\tau,3}\right) \cdot e\left(C_{j,2}, uk_{\tau,4}\right)\right)^{\lambda_j}.$$

If Θ_R is unauthorized, this algorithm returns $\perp$. It outputs transformed files $TF = \{\mathcal{Q}, \mathcal{P}, Z, C, C_{x,4}, C_5\}$.

FullDec: This algorithm takes as input *pp*, the current local update key IK'_0, and transformed file TF. It performs local puncture decryption:

$$\mathcal{B} = e(ik'_{0,3}, C_5) / e\left(ik'_{0,1}, \prod_{\ell=1}^{d} (C_{\ell,4})^{\chi_\xi}\right) \cdot e(ik'_{0,4}, C_5)^{\chi^*}.$$

Finally, this algorithm recovers the plaintext $C \cdot \mathcal{Q}^{1/Z} \cdot \mathcal{B} / \mathcal{P} = \mathcal{M}$.

3.5 Correctness analysis

(1) Correctness of **AttrVrfy** algorithm.

• *Fine-Grained Authentication.* Assume that Θ_S is an authorized set, then $\sum_{i \in I} c_i \pi_i = 1$:

$$\begin{aligned} e(T_4, C_9) &= e(g^{\gamma}, g^{\varpi} \cdot H_2(C_{6-8})) = e\left(g, g^{\varpi}\right)^{\gamma \cdot \sum_{i \in I} \pi_i c_i} \cdot e(g^{\gamma}, H_2(C_{6-8})) \\ &= \prod_{i \in I} \left(e\left((u^{\sigma(i)} h)^{-\gamma \cdot \gamma_i}, g^{s_\tau} \right) \cdot e\left(g^{\gamma \cdot \gamma_i}, (u^{\Theta_{S,\tau}} h)^{s_\tau} \right) \cdot e\left(g^{\gamma \cdot \gamma_i}, w^{-\varpi} \right) \right. \end{aligned}$$

1) $H_2(C_{6-8})$ denotes concatenating the ciphertexts $\{\{C_{\tau,6}, C_{\tau,7}\}_{\tau \in [1-k]}, C_8\}$ together to perform compression on a string with arbitrary length.

$$\begin{aligned}
&\cdot e\left(g^{\gamma\cdot\pi_i}, g^{\varpi}\right)\cdot e\left(w^{\gamma\cdot\gamma_i}, g^{\varpi}\right)\right)^{c_i}\cdot e(g^{\gamma}, H_2(C_{6-8}))\\
=&\prod_{i\in I}\left(e\left((u^{\sigma(i)}h)^{-\gamma\cdot\gamma_i}, g^{s_\tau}\right)\cdot e\left(g^{\gamma\cdot\gamma_i}, (u^{\Theta_{S,\tau}}h)^{s_\tau}w^{-\varpi}\right)\cdot e\left(g^{\gamma\cdot\pi_i}w^{\gamma\cdot\gamma_i}, g^{\varpi}\right)\right)^{c_i}\cdot e(g^{\gamma}, H_2(C_{6-8}))\\
=&\prod_{i\in I}\left(e(T'_{i,1}, C_{\tau,6})\cdot e(T'_{i,2}, C_{\tau,7})\cdot e(T'_{i,3}, C_8)\right)^{c_i}\cdot e(T_4, H_2(C_{6-8})).
\end{aligned}$$

- *Outsourced Puncture Decryption.* Assume that $ik_{x,2}\notin\{l_1,\ldots,l_d\}$, then,

$$\begin{aligned}
\mathcal{Q}=&\prod_{\xi=1}^{x}\frac{e\left(ik_{\xi,3}, C_5\right)}{e\left(ik_{\xi,1}, \prod_{\ell=1}^{d}(C_{\ell,4})^{\chi_\xi}\right)\cdot e\left(ik_{\xi,4}, C_5\right)^{\chi^*}}=\prod_{\xi=1}^{x}\frac{e\left((g^{\beta})^{z_\xi\cdot(\delta+x_{\xi,0})/x}, g^{\vartheta}\right)}{e\left(g^{z_\xi\cdot x_{\xi,0}/x}, \prod_{\ell=1}^{d}(Q(H_1(l_\ell))^{\vartheta})^{\chi_\xi}\right)}\\
&\cdot\prod_{\xi=1}^{x}\frac{1}{e\left(Q(H_1(l_\xi))^{z_\xi\cdot x_{\xi,0}/x}, g^{\vartheta}\right)^{\chi^*}}=\prod_{\xi=1}^{x}\frac{e\left(g^{z_\xi\cdot\delta\cdot\beta/x}, g^{\vartheta}\right)\cdot e\left(g^{z_\xi\cdot x_{\xi,0}\cdot\beta/x}, g^{\vartheta}\right)}{e\left(g^{x_{\xi,0}/x}, g^{\vartheta}\right)^{z_\xi(\chi^*\cdot q(H_1(l_x))+\chi_\ell\cdot\sum_{\ell=1}^{d}q(H_1(l_\ell)))}}\\
=&\prod_{\xi=1}^{x}\frac{e\left(g^{z_\xi\cdot\delta\cdot\beta/x}, g^{\vartheta}\right)\cdot e\left(g^{z_\xi\cdot x_{\xi,0}\cdot\beta/x}, g^{\vartheta}\right)}{e\left(g^{x_{\xi,0}/x}, g^{\vartheta}\right)^{z_\xi\cdot q(0)}}=\prod_{\xi=1}^{x}e\left(g^{z_\xi\cdot\delta\cdot\beta/x}, g^{\vartheta}\right)=e\left(g^{\sum_{\xi=1}^{x}z_\xi\cdot\delta\cdot\beta}, g^{\vartheta}\right)=e\left(g,g\right)^{Z\delta\beta\vartheta}.
\end{aligned}$$

- *Fine-Grained Access Control.* Assume that Θ_R is an authorized set, then $\vartheta=\sum_{j\in J}f_j\lambda_j$:

$$\begin{aligned}
&\prod_{j\in J}\left(e\left(C_{j,3}, uk_2\right)\cdot e\left(C_{j,1}, uk_{\tau,3}\right)\cdot e\left(C_{j,2}, uk_{\tau,4}\right)\right)^{\lambda_j}=\prod_{j\in J}\left(e\left(w^{\lambda_j}v^{\eta_j}, g^{r}\right)\cdot e\left((u^{\rho(j)}h)^{-\eta_j}, g^{r_\tau}\right)\right.\\
&\left.\cdot e\left(g^{\eta_j}, (u^{\Theta_{R,\tau}}h)^{r_\tau}v^{-r}\right)\right)^{\lambda_j}=\prod_{j\in J}\left(e\left(w^{\lambda_j}, g^{r}\right)\cdot e\left(v^{\eta_j}, g^{r}\right)\cdot e\left((u^{\rho(j)}h)^{-\eta_j}, g^{r_\tau}\right)\cdot e\left(g^{\eta_j}, (u^{\Theta_{R,\tau}}h)^{r_\tau}\right)\right.\\
&\left.\cdot e\left(g^{\eta_j}, v^{-r}\right)\right)^{\lambda_j}=\prod_{j\in J}e\left(w^{\lambda_j}, g^{r}\right)^{f_j}=e\left(w, g^{r}\right)^{\sum_{j\in J}\lambda_j f_j}=e(g,w)^{\vartheta r}.
\end{aligned}$$

$$\mathcal{P}=e(C_5, uk_1)/e(g,w)^{\vartheta r}=e(g^{\vartheta}, (g^{\beta})^{t}g^{\alpha}w^{r})/e(g,w)^{\vartheta r}=e(g^{\vartheta}, g^{\alpha})\cdot e(g^{\vartheta}, g^{\beta t}).$$

Remark: According to linear secret sharing scheme and Lagrange interpolation function in Section 2, the correctness of **AttrVrfy** is divided into the following three parts. Specifically, if the authentication policy $\mathbb{R}=(\mathbb{N},\sigma)$ embedded in Tok' is matched by Θ_S, *Fine-Grained Authentication* is achieved. Furthermore, if the coefficients $\{\chi^*,\chi_1,\ldots,\chi_d\}$ can reconstruct β in the exponent, *Outsourced Puncture Decryption* is realizable. Similarly, if the access policy $\mathbb{S}=(\mathbb{M},\rho)$ in EF is matched by Θ_R, FSBiAC enables *Fine-Grained Access Control*.

(2) Correctness of **FullDec** algorithm.

$$\begin{aligned}
\mathcal{B}=&\frac{e\left(ik_{0,3}{}', C_5\right)}{e\left(ik_{0,1}{}', \prod_{\ell=1}^{d}(C_{\ell,4})^{\chi_\xi}\right)\cdot e\left(ik_{0,4}{}', C_5\right)^{\chi^*}}=\frac{e\left(\left(g^{\beta}\right)^{\omega+t+x_{x,1}-\delta}, g^{\vartheta}\right)}{e\left(g^{\omega+x_{x,1}}, \prod_{\ell=1}^{d}(Q(H_1(l_\ell))^{\vartheta})^{\chi_\xi}\right)}\\
&\cdot\frac{1}{e(Q(H_1(l_0))^{\omega+x_{x,1}}, g^{\vartheta})^{\chi^*}}=\frac{e\left(\left(g^{\beta}\right)^{\omega+t+x_{x,1}-\delta}, g^{\vartheta}\right)}{e\left(g^{\omega+x_{x,1}}, g^{\vartheta}\right)^{q(0)}}=e(g,g)^{\vartheta\beta(t-\delta)}.
\end{aligned}$$

$$\mathcal{M}=\mathcal{C}\cdot\mathcal{Q}^{1/Z}\cdot\mathcal{B}/\mathcal{P}=\mathcal{M}\cdot\frac{e(g,g)^{\alpha\vartheta}\cdot e(g,g)^{(Z\delta\beta\vartheta)\cdot 1/Z}\cdot e(g,g)^{\vartheta\beta(t-\delta)}}{e(g^{\vartheta}, g^{\alpha})\cdot e(g^{\vartheta}, g^{\beta t})}=\mathcal{M}\cdot\frac{e(g,g)^{\alpha\vartheta}\cdot e(g,g)^{\vartheta\beta t}}{e(g^{\vartheta}, g^{\alpha})\cdot e(g^{\vartheta}, g^{\beta t})}.$$

4 Security proof and analysis

This section rigorously proves the privacy and authenticity based on the security model in Section 3.3, including IND-sCPA and EUF-CMA. Besides, we conduct an in-depth analysis of forward security and exposure attack defense, then examine how the Lagrange interpolation function in Section 2 facilitates secure outsourced updating.

4.1 IND-sCPA security proof

Theorem 1 (IND-sCPA security). If $(q-1)$-type assumption in Section 2.3 holds, the proposed scheme can achieve IND-sCPA security.

Relying on the construction in Section 3.4, the selective semantic security game can be established. If there exists a PPT adversary $\mathcal{A}$ that can break the game with a non-negligible advantage ε, a simulator $\mathcal{B}$ can be constructed to solve $(q-1)$-type problem.

(1) *Initialization.* $\mathcal{A}$ first declares a target policy $\mathbb{S}^* = (\mathbb{M}^*, \rho^*)$ and a set of target label $\mathcal{L}^* = (l_1^*, \ldots, l_d^*)$. $\mathcal{B}$ accepts the given instances from decisional $(q-1)$-type assumption, and obtains $\mathbb{S}^*$ and $\mathcal{L}^*$ from $\mathcal{A}$. Then, $\mathcal{B}$ sets up two empty list $\mathcal{E}_x$ and $\mathcal{E}_y$. $\mathcal{B}$ selects $\tilde{\alpha}$ from $\mathbb{Z}_p$, and computes $e(g,g)^{\alpha} = e(g,g)^{\tilde{\alpha}} \cdot e(g^a, g^{a^q})$, implicitly setting $msk = \alpha = a^{q+1} + \tilde{\alpha}$. Subsequently, $\mathcal{B}$ sets $w = g^a$, and randomly selects $\tilde{v}, \tilde{u}, \tilde{h}$ from $\mathbb{Z}_p$, finally sets public parameters pp as follows: $v = g^{\tilde{v}} \cdot \prod_{(j,k)\in[l,n]} (g^{a^k/b_j})^{\mathbb{M}^*_{j,k}}$, $u = g^{\tilde{u}} \cdot \prod_{(j,k)\in[l,n]} (g^{a^k/b_j^2})^{\mathbb{M}^*_{j,k}}$, $h = g^{\tilde{h}} \cdot \prod_{(j,k)\in[l,n]} (g^{a^k/b_j^2})^{-\rho^*(j)\cdot\mathbb{M}^*_{j,k}}$. Besides, $\mathcal{B}$ randomly selects $(d+1)$ points $\gamma_0, \ldots, \gamma_d$ from $\mathbb{Z}_p$, where γ_0 is only used as a distinguishing term and is not used in the simulation. By computing $g^{\beta} = g^{1/a}$, $\mathcal{B}$ implicitly sets $q(0) = \beta = 1/a$. It sets $q(\sigma) = \gamma_\sigma$, so that $Q(\sigma) = g^{q(\sigma)} = g^{\gamma_\sigma}$. Finally, $\mathcal{B}$ transfers pp to $\mathcal{A}$ and remains msk.

(2) *The First Query Phase.* After initializing a counter $V = 0$, $\mathcal{A}$ adaptively queries UK and IK_x.

(a) *UKeyGen Query.* After the query is sent on an attribute set $\Theta_R = (\Theta_{R,1}, \ldots, \Theta_{R,k})$ (Θ_R must not be satisfied with $\mathbb{S}^*$), $\mathcal{B}$ randomly selects $\tilde{\omega}$ and $\tilde{t}$ from $\mathbb{Z}_p$. Then, it sets $IK_0 = ik_{[0,1-4]}$, where $ik_{0,1} = g^{\tilde{\omega}}$, $ik_{0,2} = l_0$, $ik_{0,3} = (g^{\beta})^{\tilde{\omega}+\tilde{t}}$, $ik_{0,4} = Q(H_1(l_0))^{\tilde{\omega}}$. Next, $\mathcal{B}$ calculates UK as follows:

$$uk_1 = (g^{1/a})^{\tilde{t}} \cdot g^{a^{q+1}} \cdot g^{\tilde{\alpha}} \cdot g^{a\tilde{r}} \prod_{i\in[1-n]} g^{\lambda_i a^{q+2-i}} = g^{\tilde{t}/a} \cdot g^{a\tilde{r}} \cdot g^{\tilde{\alpha}} \prod_{i=2}^{n} (g^{a^{q+2-i}})^{\lambda_i},$$

$$uk_2 = g^{\tilde{r}} \prod_{i\in[1-n]} (g^{a^{q+1-i}})^{\lambda_i}, \quad uk_{\tau,3} = g^{r_\tau} = g^{\tilde{r}_\tau} \cdot g^{\tilde{r}\cdot\sum_{\substack{i'\in[1-l] \\ \rho^*(i')\notin\Theta_R}} \frac{b_{i'}}{\Theta_{R,\tau}-\rho^*(i')}} \cdot g^{\sum_{\substack{(i,i')\in[n,l] \\ \rho^*(i')\notin\Theta_R}} \frac{\lambda_i b_{i'} a^{q+1-i}}{\Theta_{R,\tau}-\rho^*(i')}},$$

$$uk_{\tau,4} = (u^{\Theta_{R,\tau}} h)^{r_\tau} \cdot v^{-r} = \Psi \cdot \Phi, \qquad (u^{\Theta_{R,\tau}} h)^{r_\tau} = \Psi \cdot \prod_{\substack{j\in[1-l] \\ \rho^*(j)\notin\Theta_R}} g^{\langle \boldsymbol{\lambda}, \mathbb{M}^*_j \rangle a^{q+1}/b_j},$$

$$v^{-r} = v^{-\tilde{r}} \left(g^{\tilde{v}} \prod_{(j,k)\in[l,n]} g^{a^k \cdot \mathbb{M}^*_{j,k}/b_j} \right)^{-\sum_{i\in[1-n]} \lambda_i \cdot a^{q+1-i}} = \Phi \cdot \prod_{\substack{j\in[1-l] \\ \rho^*(j)\notin\Theta_R}} g^{-\langle \boldsymbol{\lambda}, \mathbb{M}^*_j \rangle a^{q+1}/b_j}.$$

Since Θ_R does not satisfy the target policy $\mathbb{S}^*$, there must exist a vector $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_n)^{\top}$ such that $\lambda_1 = -1$ and for the row $i \in I$, it holds that $\langle \mathbb{M}^*_i, \boldsymbol{\lambda} \rangle = 0$. Through the calculation on IK, the implicit setting r can be achieved by $r = \tilde{r} + \sum_{i\in[1-n]} \lambda_i \cdot a^{q+1-i}$, where r is randomly selected from $\mathbb{Z}_p$. Beside, r_τ is randomly selected from $\mathbb{Z}_p$, and both Φ and Ψ can be calculated by specific $(q-1)$-type instance [43]. Lastly, $\mathcal{B}$ sends the tuple $(IK_0 = ik_{[0,1-4]}, UK = uk_{[1-4]})$ to $\mathcal{A}$.

(b) *OutUpdt Query.* After the key query is sent on a label $l_x \notin \mathcal{L}^* = (l_1^*, \ldots, l_d^*)$, a unauthorized attribute set $\Theta_R = (\Theta_{R,1}, \ldots, \Theta_{R,k})$ and $(x-1)$-th outsourced update key, $\mathcal{B}$ check the counter V about x, and add l_x to the execution record $\mathcal{E}_x$. When $\mathcal{A}$ first performs *Corrupt Query*, $\mathcal{B}$ returns the latest update key IK_{x-1} to $\mathcal{A}$ and assigns all labels in $\mathcal{E}_x$ to $\mathcal{E}_y$. When no *Corrupt Query* is performed, or $\mathcal{L}^*$ has a non-empty intersection with $\mathcal{E}_x$, $\mathcal{B}$ randomly selects $\tilde{\delta}, \tilde{x_0}, \tilde{z_x}$ from $\mathbb{Z}_p$, and sets the last outsourced update key IK_x as $\bar{ik} = (g^{1/a})^{\tilde{\delta}}$, $ik_{x,1} = g^{\tilde{z_x}\cdot\tilde{x_0}/x}$, $ik_{x,2} = l_x$, $ik_{x,3} = (g^{1/a})^{\tilde{z_x}\cdot(\tilde{\delta}+\tilde{x_0})/x}$, $ik_{x,4} = Q(H_1(l_x))^{\tilde{z_x}\cdot\tilde{x_0}/x}$. Finally, $\mathcal{B}$ calculates $\tilde{Z} = \sum_{\xi=1}^{x} \tilde{z}_\xi$, increments the counter V, and sends the tuple $(IK_x, l_x, \tilde{Z}, \bar{ik})$ to $\mathcal{A}$.

(c) *LocUpdt Query.* $\mathcal{A}$ sends on a label $l_x \notin \mathcal{L}^* = (l_1^*, \ldots, l_d^*)$, an unauthorized attribute set $\Theta_R = (\Theta_{R,1}, \ldots, \Theta_{R,k})$ and $(x-1)$-th local update key to $\mathcal{B}$. If the tuple $(l_x, IK_x, \tilde{Z}, \bar{ik})$ has been recorded, $\mathcal{B}$ randomly selects $\tilde{x_1}$ from $\mathbb{Z}_p$, sets the last local update key IK'_0 as follows:

$$ik'_{0,1} = g^{\tilde{\omega}+\tilde{x_1}}, \qquad ik'_{0,2} = l_0, \qquad ik'_{0,3} = (g^{1/a})^{\tilde{\omega}+\tilde{t}+\tilde{x_1}-\tilde{\delta}}, \qquad ik'_{0,4} = Q(H_1(l_0))^{\tilde{\omega}+\tilde{x_1}}.$$

Otherwise, if the tuple $(l_x, Z, \bar{ik})$ exists, $\mathcal{B}$ simultaneously executes *OutUpdt Query* and *LocUpdt Query*, followed with incrementing V and storing $(IK'_0, IK_x, l_x, Z, \bar{ik})$ in $\mathcal{E}_x$. Lastly, $\mathcal{B}$ sends the tuple to $\mathcal{A}$.

(d) *Corrupt Query.* When $\mathcal{A}$ sends the first query, $\mathcal{B}$ assigns $\mathcal{E}_x$ to $\mathcal{E}_y$, and returns the x-th update key (IK_0', IK_x). For all subsequent queries, $\mathcal{B}$ returns $\perp$.

(3) *Challenge.* Two same-size message $\mathcal{M}_0^*$ and $\mathcal{M}_1^*$ are selected by $\mathcal{A}$. It sends $(\mathcal{M}_0^*, \mathcal{M}_1^*, \mathbb{S}^*, \mathcal{L}^*)$ to $\mathcal{B}$. Notice that $\{l_1^*, \ldots, l_d^*\} \in \mathcal{L}^*$ has never been issued by *Corrupt Query.* $\mathcal{B}$ randomly selects $\psi \in \{0,1\}$ and constructs the challenge encrypted components as follows:

$$C = \mathcal{M}_\varphi^* \cdot \mathcal{T} \cdot e\left(g, g^\vartheta\right)^{\tilde{\alpha}}, \quad C_{i,1} = \left(g^{\vartheta b_i}\right)^{-\tilde{u}\rho^*(i) - \tilde{h}} \cdot \prod_{\substack{(j,k)\in[l,n] \\ j \neq i}} \left(g^{\vartheta a^k b_i / b_j^2}\right)^{-(\rho^*(i) - \rho^*(j)) \cdot \mathbb{M}_{j,k}^*},$$

$$C_{i,2} = \left(g^{\vartheta b_i}\right)^{-1} = g^{-\vartheta \cdot b_i}, \quad C_{i,3} = w^{\tilde{\lambda}_i} \cdot \left(g^{\vartheta b_i}\right)^{-\tilde{v}} \cdot \prod_{\substack{(j,k)\in[l,n] \\ j \neq i}} \left(g^{\vartheta a^k b_i / b_j}\right)^{-\mathbb{M}_{j,k}^*}, \quad C_{x,4} = \left(g^\vartheta\right)^{\gamma_x}.$$

Among the above components, we have $\vartheta_i = -\vartheta \cdot b_i$, $i \in [l]$, $x \in [d]$. The secret vector is set as $\boldsymbol{v} = \left(\vartheta, \vartheta a + \tilde{v}_2, \vartheta a^2 + \tilde{v}_3, \ldots, \vartheta a^{n-1} + \tilde{v}_n\right)^\top$ ($\tilde{v}_2, \ldots, \tilde{v}_n$ is randomly selected from $\mathbb{Z}_p$), and the sharing vector of $\mathbb{M}^*$ is $\lambda_i = \sum_{\tau\in[1-n]} \mathbb{M}_{i,\tau}^* \vartheta a^{\tau-1} + \sum_{\tau=2}^n \mathbb{M}_{i,\tau}^* \tilde{v}_\tau = \sum_{\tau\in[1-n]} \mathbb{M}_{i,\tau}^* \vartheta a^{\tau-1} + \tilde{\lambda}_i$, in which $\tilde{\lambda}_i = \sum_{\tau=2}^n \mathbb{M}_{i,\tau}^* \tilde{v}_\tau$. Then, $\mathcal{B}$ defines the assumption term g^ϑ as C_5. If $\psi = 0$, let the distinguishing term as $\mathcal{T} = e(g,g)^{\vartheta \cdot a^{q+1}}$, then $C = \mathcal{M}_\varphi^* \cdot e(g,g)^{\vartheta \cdot a^{q+1}} \cdot e(g, g^\vartheta)^{\tilde{\alpha}} = \mathcal{M}_\varphi^* \cdot e(g,g)^{\alpha \cdot \vartheta}$. Otherwise, $\mathcal{B}$ randomly selects $\mathcal{T}$ from G_T, then defines $C = \mathcal{M}_\varphi^* \cdot \mathcal{T} \cdot e(g, g^\vartheta)^{\tilde{\alpha}}$. Finally, $\mathcal{B}$ returns the challenge tuple $(C, C_{i,1}, C_{i,2}, C_{i,3}, C_{x,4}, C_5)$ to $\mathcal{A}$.

(4) *The Second Query Phase.* $\mathcal{A}$ requests $\mathcal{B}$ with the same operations in the first query phase, but the queried plaintext $\mathcal{M}$ must be different from $\mathcal{M}_0^*$ and $\mathcal{M}_1^*$. If previous queries have led to the corruption of $\mathcal{L}^* = \{l_1^*, \ldots, l_d^*\}$, then *UKeyGen*, *OutUpdt* and *LocUpdt* queries return $\perp$.

(5) *Guess.* $\mathcal{A}$ generates a guess bit $\varphi' \in \{0,1\}$, and wins in this game if $\varphi = \varphi'$. $\mathcal{B}$ constructs an environment that is equivalent to the interaction between $\mathcal{A}$ and $\mathcal{C}$. When $\psi = 0$, we have $\mathcal{T} = e(g,g)^{s \cdot a^{q+1}}$; $\mathcal{A}$ knows the effective ciphertext of $\mathcal{M}_\varphi^*$, thus $\Pr[\varphi' = \varphi | \psi = 0] = \varepsilon + 1/2$. When $\psi = 1$, $\mathcal{T}$ is randomly chosen from G_T, hence no information about φ can be obtained from $\mathcal{A}$'s view, thus $\Pr[\varphi' = \varphi | \psi = 1] = 1/2$.

The advantage of $\mathcal{B}$ in solving $(q-1)$-type Problem is $Adv_{\mathcal{B}}^{(q-1)} = 1/2 \cdot \Pr[\varphi' = \varphi | \psi = 0] - 1/2 \cdot \Pr[\varphi' = \varphi | \psi = 1] = 1/2 \cdot (1/2 + \varepsilon) - 1/2 \times 1/2 = \varepsilon/2$. If there exists a PPT adversary $\mathcal{A}$ that can break the proposed scheme with a non-negligible advantage, $\mathcal{A}$'s ability can be leveraged to solve $(q-1)$-type problem. Since this assumption is proven to be unsolvable in polynomial time, FSBiAC achieves IND-sCPA security.

4.2 EUF-CMA security proof

Theorem 2 (EUF-CMA security)**.** If $(q-2)$-type assumption in Section 2.3 holds, the proposed scheme can realize EUF-CMA security.

Relying on the construction in Section 3.4, the following existential unforgeability game can be established. Specifically, if there exists a PPT forger $\mathcal{F}$ that can forge a legal signature with a non-negligible advantage ε, a simulator $\mathcal{B}$ can be constructed to solve $(q-2)$-type problem.

(1) *Initialization.* The forger $\mathcal{F}$ specifies the target attribute set $\Theta_S^* = (\Theta_{S,1}^*, \ldots, \Theta_{S,k}^*)$. $\mathcal{B}$ accepts the given instances from $(q-2)$-type assumption, and obtains Θ_S^* from $\mathcal{F}$. $\mathcal{B}$ randomly selects $\tilde{\alpha}, \tilde{h}$ from $\mathbb{Z}_p$, then computes $e(g,g)^\alpha = e(g^\vartheta, g^\varrho)$, where it implicitly set $msk = \alpha = \vartheta \cdot \varrho$, and ϑ and ϱ are already offered in $(q-2)$ instance. $\mathcal{B}$ randomly selects $\tilde{u}, \tilde{h}$ from $\mathbb{Z}_p$, and calculates the relevant parameters pp as $w = g^\vartheta$, $u = g^{\tilde{u}} \cdot \prod_{i\in[1-k]} g^{\varrho/b_i^2}$, $h = g^{\tilde{h}} \cdot \prod_{i\in[1-k]} g^{\vartheta\varsigma/b_i} \cdot \prod_{i\in[1-k]} \left(g^{\varrho/b_i^2}\right)^{-\Theta_{S,i}^*}$. Finally, $\mathcal{B}$ selects a collision-resistant hash function $H_2 : \{0,1\}^* \to G_0$, and transfers pp to $\mathcal{F}$. For brevity, $\mathcal{B}$ only offers parameter settings associated with authentication in this process. The above parameters are properly distributed in $\mathcal{F}$'s view and can be calculated by the given term in $(q-2)$ instance and the target attribute set Θ_S^*.

(2) *Query Phase.* The forger $\mathcal{F}$ issues *TokenGen Query* and *Signature Query.* $\mathcal{C}$ responds as follows.

(a) *TokenGen Query.* $\mathcal{F}$ selects an authentication policy $\mathbb{R} = (\mathbb{N}, \sigma)$ and transfers it to $\mathcal{B}$. Here, the target attribute set $\Theta_S^* = (\Theta_{S,1}^*, \ldots, \Theta_{S,k}^*)$ must be not satisfied with $\mathbb{R} = (\mathbb{N}, \sigma)$. Since Θ_S^* is an unauthorized set for $\mathbb{R} = (\mathbb{N}, \sigma)$, there exists a vector $\boldsymbol{\nu} = (1/\vartheta\varrho, v_2, \ldots, v_n)^\top$, and for all $\tau \in [1-l]$ where $\sigma(\tau) \in \Theta_S^*$, it holds that $\langle \mathbb{N}_\tau, \boldsymbol{\nu} \rangle = 0$. In consequence, by randomly selecting $\tilde{v}_2, \ldots, \tilde{v}_n$ from $\mathbb{Z}_p$, the vector $\boldsymbol{v}$ which contains $1/\vartheta\varrho$, can be implicitly set as $\boldsymbol{v} = \vartheta\varrho\boldsymbol{\nu} + (0, \tilde{v}_2, \ldots, \tilde{v}_n)^\top$. For each row in $[1-l]$, the share vector is set as $\boldsymbol{\pi}_\tau =$

$\vartheta\varrho\langle\mathbb{N}_\tau, \boldsymbol{\nu}\rangle + \langle\mathbb{N}_\tau, (0, \tilde{v}_2, \ldots, \tilde{v}_n)^\top\rangle = \vartheta\varrho\langle\mathbb{N}_\tau, \boldsymbol{\nu}\rangle + \tilde{\pi}_\tau$. If $\sigma(\tau) \in \Theta_S^*$, then it must hold that $\langle\mathbb{N}_\tau, \boldsymbol{\nu}\rangle = 0$, thus there exists $\pi_\tau = \left\langle\mathbb{N}_\tau, (0, \tilde{v}_2, \ldots, \tilde{v}_n)^\top\right\rangle$. If $\sigma(\tau) \notin \Theta_S^*$, then $\mathcal{B}$ randomly selects $\tilde{\gamma}_\tau$ from $\mathbb{Z}_p$ and sets Tok as follows:

$$T_{\tau,1} = \left(u^{\sigma(\tau)}h\right)^{-\tilde{\gamma}_\tau} \cdot \left(\prod_{i\in[1-k]} g^{\frac{\vartheta\varsigma}{b_i}} \cdot \prod_{i\in[1-k]} g^{\frac{\varrho(\sigma(\tau)-\Theta_S^*)}{b_i^2}} \cdot g^{\sigma(\tau)\cdot\tilde{u}+\tilde{h}}\right)^{\varrho\langle\mathbb{N}_\tau,\boldsymbol{\nu}\rangle - \sum_{i\in[1-k]}\frac{\vartheta\varsigma b_i\langle\mathbb{N}_\tau,\boldsymbol{\nu}\rangle}{\sigma(\tau)-\Theta_S^*}},$$

$$T_{\tau,2} = (g^\varrho)^{-\langle\mathbb{N}_\tau,\boldsymbol{\nu}\rangle} \cdot g^{\tilde{t}_\tau} \cdot \prod_{i\in[1-k]} \left(g^{\vartheta\varsigma b_i}\right)^{\frac{\langle\mathbb{N}_\tau,\boldsymbol{\nu}\rangle}{\sigma(\tau)-\Theta_S^*}}, \quad T_{\tau,3} = g^{\tilde{\pi}_\tau} \cdot w^{\tilde{\gamma}_\tau} \cdot \prod_{i\in[1-n]} \left(g^{\vartheta^2\varsigma b_i}\right)^{\frac{\langle\mathbb{N}_\tau,\boldsymbol{\nu}\rangle}{\sigma(\tau)-\Theta_S^*}}.$$

Here, by calculating the above equation, $\mathcal{B}$ implicitly sets $\gamma_\tau = -\varrho\langle\mathbb{N}_\tau, \boldsymbol{\nu}\rangle + \sum_{i\in[1-k]} \vartheta\varsigma b_i\langle\mathbb{N}_\tau, \boldsymbol{\nu}\rangle/(\sigma(\tau) - \Theta_S^*) + \tilde{\gamma}_\tau$, and finally sends $Tok = \{T_{\tau,1}, T_{\tau,2}, T_{\tau,3})\}_{\tau\in[1-k]}$ to $\mathcal{F}$.

(b) *Signature Query.* $\mathcal{F}$ selects a plaintext $\mathcal{M}$ and an attribute set $\Theta_S = (\Theta_{S,1}, \ldots, \Theta_{S,k}) \neq \Theta_S^*$, and transfers it to $\mathcal{B}$. For all $\tau \in [1-k]$, $\mathcal{B}$ implicitly sets $s_\tau = b_\tau$ by computing $C_{\tau,6} = g^{s_\tau} = g^{b_\tau}$. Then, $\mathcal{B}$ calculates $C_{\tau,7}$ and C_8 as follows:

$$C_{\tau,7} = g^{b_\tau\left(\tilde{u}\Theta_\tau+\tilde{h}\right)} \cdot g^{-\vartheta\varsigma} \cdot \prod_{i\in[1-k]} g^{\frac{\vartheta\varsigma b_\tau}{b_i}} \cdot \prod_{i\in[1-k]} g^{\frac{\varrho b_\tau\left(\Theta_{S,k}-\Theta_{S,i}\right)}{b_i^2}}, \quad C_8 = g^{-\varsigma}.$$

$\mathcal{B}$ performs a concatenation of $\{C_{\tau,6}, C_{\tau,7}\}_{\tau\in[1-k]}$ and C_8, then calculates $C_9 = g^{-\varsigma} \cdot H_2(C_{6-8})$. Finally, it sends the signature tuple $\{\{C_{\tau,6}, C_{\tau,7}\}_{\tau\in[1-k]}, C_8, C_9\}$ to $\mathcal{F}$.

(3) *Forgery.* Once $\mathcal{F}$ completes *Query Phase*, it generates a forged signature $\{C_{\tau,6}^*, C_{\tau,7}^*, C_8^*, C_9^*\}$ under the target attribute set $\Theta_S^* = (\Theta_{S,1}^*, \ldots, \Theta_{S,k}^*)$. Note that $\mathcal{F}$ must not have queried the signature on Θ_S^*. $\mathcal{F}$ submits the forged signature to $\mathcal{B}$, claiming that it is valid. Based on the signature definition, if the simulation is successful, $\mathcal{B}$ randomly selects $\tilde{v}_2, \ldots, \tilde{v}_n$ and sets $\boldsymbol{v} = \vartheta\varrho\boldsymbol{\nu} + (0, \tilde{v}_2, \ldots, \tilde{v}_n)^\top$. Next, let $I = \{i : \sigma(i) \in \Theta_R \wedge \sigma : [1-l] \to \mathbb{Z}_p\}$, and for $\{c_i \in \mathbb{Z}_p\}_{i\in I}$, ensure that $\sum_{i\in I} c_i\mathbb{N}_i = (1, 0, \ldots, 0)$. $\mathcal{B}$ computes $\prod = \prod_{i\in I}\left(e((u^{\sigma(i)}h)^{-1}, C_{\tau,6}^*) \cdot e(g, C_{\tau,7}^*) \cdot e(g^{\pi_i+\vartheta}, C_8^*)\right)^{c_i}$ $\cdot e(g^\varsigma, H_2(C_{6-8}^*))$. Since $\mathcal{B}$ holds g^ϱ and g^ς, it can compute $\prod/(e(g, g^\varsigma \cdot C_9^*) \cdot e(g^\varrho, w^{-\varpi}) = e(g, g)^{\vartheta\varrho\varsigma}$ as the solution to $(q-2)$-type problem. If there exists a PPT forger $\mathcal{F}$ that can forge the valid signature with a non-negligible advantage, $\mathcal{F}$'s ability can be leveraged to solve $(q-2)$ problem. Since $e(g, g)^{\vartheta\varrho\varsigma}$ is proven to be uncomputable, it can be deduced that FSBiAC achieves EUF-CMA security.

4.3 Forward security and exposure attack defense

In the traditional definition, forward security [44] refers to the ability which maintains the previously encrypted information confidentiality, even if the long-term secret key is maliciously compromised or stolen by an adversary. It effectively prevents the adversary from using historical keys to breach system security. In our FSBiAC, the semi-trusted cloud continuously stores past encrypted information. The encrypted file EF is generated by **Encrypt**, which embeds a set of predefined labels $\mathcal{L} = \{l_1, \ldots, l_d\}$ established by DO. Moreover, by inputting a specific label $IK_{x,2} = l_x$, the update key IK_{x-1} runs $(Z, \overline{ik}, IK_x) \leftarrow$ **OutUpdt**(pp, msk, IK_{x-1}, l_x) and $IK_0' \leftarrow$ **LocUpdt**(pp, IK_0') algorithms to ensure that the update key IK_x can only decrypt EF that not containing l_x (see **AttrVrfy** and **FullDec** algorithms in Subsection 3.4). The key is updated after each puncture, which disables the decryption capability for the specific labels. After each update, the current key can only decrypt future data, whereas previously disabled ciphertexts can no longer be decrypted. This characteristic is achieved through Lagrange interpolation. If $ik_{x,2}$ belongs to $\{l_\ell\}_{\ell\in[1-d]}$, then $q(H_1(ik_{x,2})) \cup \{q(H_1(l_\ell))\}_{\ell\in[1-d]}$ degenerates into d points, thus no set of coefficients $\{\chi^*, \chi_1, \ldots, \chi_d\}$ can satisfy the conditions for reconstructing β. Even if the update key IK_x is subjected to malicious attacks, EF can still guarantee confidentiality. Therefore, FSBiAC achieves forward security, thereby effectively resisting key exposure attacks.

5 Experimental evaluation

This section analyzes schemes [14, 21, 23], and FSBiAC from both theoretical and experimental perspectives. We demonstrate the applicability using Stanford Network SNAP datasets [25].

Table 2 Comparison of theoretical computational cost. E_0 and E_T represent exponentiations in G_0 and G_T, respectively. M_0 and M_T denote multiplications in G_0 and G_T, respectively. P denotes pairing operations, and H denotes hash function operations.

Algorithm	Xu et al. [14]	Ghopur et al. [23]	Nie et al. [21]	FSBiAC
UKeyGen	$3\vert\mathbb{N}\vert E_0 + \vert\mathbb{N}\vert M_0 + \vert\mathbb{N}\vert H$	$2(\vert\Theta_R\vert + \vert\mathbb{D}\vert)E_0 + 2M_0$	$3\vert\mathbb{N}\vert + 5E_0+$ $(\vert\mathbb{N}\vert + 1)(M_0 + H)$	$3(\vert\Theta_R\vert + 7)E_0+$ $(\vert\Theta_R\vert + 2)M_0 + H$
TokenGen	$\vert\Theta_S\vert(2E_0 + H) + M_0$	–	$\vert\Theta_S\vert(2E_0 + M_0 + H)$	$\vert\mathbb{N}\vert(3E_0 + 2M_0)$
LocUpdt	–	–	$\vert\mathbb{I}\vert(6E_0 + 5M_0)$	$\vert\mathbb{I}\vert(3E_0 + 4M_0)$
Encrypt	$(2\vert\Theta_R\vert + 2\vert\Theta_S\vert + 3)E_0$ $+(\vert\Theta_S\vert + \vert\Theta_R\vert)M_0 + E_T$ $+M_T + (\vert\Theta_R\vert + \vert\Theta_S\vert + 1)H$	$(6\vert\mathbb{I}\vert + 2\vert\mathbb{M}\vert + 2)E_0$ $+(5\vert\mathbb{I}\vert + \vert\mathbb{M}\vert)M_0 + 2M_T$	$(2\vert\Theta_R\vert + \vert\mathbb{D}\vert + 2\vert\Theta_S\vert + 3)E_0$ $+(\vert\Theta_S\vert + \vert\Theta_R\vert)M_0 + E_T + M_T$ $+(\vert\Theta_R\vert + \vert\mathbb{D}\vert + \vert\Theta_S\vert + 1)H$	$(3\vert\mathbb{M}\vert + \vert\mathbb{D}\vert + 2\vert\Theta_S\vert + 3)E_0$ $+\vert\mathbb{D}\vert H + E_T + M_T$ $2(\vert\mathbb{M}\vert + \vert\Theta_S\vert)M_0$
FullDec	$\vert\mathbb{N}\vert E_T + 2\vert\mathbb{N}\vert(M_T + P)$	$2E_0 + 2M_0 + P$	$\vert\mathbb{I}\vert M_0 + (\vert\mathbb{N}\vert + \vert\mathbb{I}\vert)E_T$ $+(2\vert\mathbb{N}\vert + \vert\mathbb{I}\vert)M_T$ $+(2\vert\mathbb{N}\vert + 3\vert\mathbb{I}\vert)P$	$\vert\mathbb{I}\vert(E_0 + M_0)$ $+2E_T + 5M_T + 3P$

Table 3 Comparison of theoretical storage cost. $|G_0|$, $|G_T|$, and $|\mathbb{Z}_p|$ represent the element sizes in G_0, G_T, and $\mathbb{Z}_p$, respectively.

Components	Xu et al. [14]	Ghopur et al. [23]	Nie et al. [21]	FSBiAC
Initial update key	–	$3\vert G_0\vert + \vert\mathbb{Z}_p\vert$	$3\vert G_0\vert + \vert\mathbb{Z}_p\vert$	$3\vert G_0\vert + \vert\mathbb{Z}_p\vert$
Outsourced key	$2\vert\Theta_R\vert\vert G_0\vert$	$(\vert\Theta_R\vert + \vert\mathbb{D}\vert + 2)\vert G_0\vert$	$2\vert\Theta_R\vert + \vert G_0\vert$	$(\vert\Theta_R\vert + 2)\vert G_0\vert$
Policy token	$(\vert\Theta_S\vert + 1)\vert G_0\vert$	–	$2\vert\Theta_S\vert + \vert G_0\vert$	$3\vert\mathbb{N}\vert + \vert G_0\vert$
Update key	–	$(\vert\mathbb{I}\vert + 1)(3\vert G_0\vert + \vert\mathbb{Z}_p\vert)$	$(\vert\mathbb{I}\vert + 1)(3\vert G_0\vert + \vert\mathbb{Z}_p\vert)$	Out: $\vert\mathbb{I}\vert(3\vert G_0\vert + \vert\mathbb{Z}_p\vert)$ Local: $3\vert G_0\vert + \vert\mathbb{Z}_p\vert$
Encrypted file	$3(\vert\Theta_S\vert + 1)$ $\vert G_0\vert + \vert G_T\vert$	$(3\vert\mathbb{I}\vert + \vert\mathbb{M}\vert + 6)\vert G_0\vert$ $+(\vert\mathbb{I}\vert + 1)\vert\mathbb{Z}_p\vert + \vert G_T\vert$	$(2\vert\Theta_S\vert + 2\vert\Theta_R\vert + \vert\mathbb{D}\vert$ $+2)\vert G_0\vert + \vert G_T\vert$	$(2\vert\Theta_S\vert + \vert\mathbb{M}\vert + \vert\mathbb{D}\vert$ $+2)\vert G_0\vert + \vert G_T\vert$
Transformed file	$(\vert\Theta_S\vert + 1)$ $\vert G_0\vert + \vert G_T\vert$	$\vert G_0\vert + 2\vert G_T\vert$	$(\vert\Theta_S\vert + \vert\mathbb{I}\vert$ $+1)\vert G_0\vert + \vert G_T\vert$	$(\vert\mathbb{I}\vert + 1)\vert G_0\vert + 3\vert G_T\vert + \vert\mathbb{Z}_p\vert$

5.1 Theoretical analysis

This section analyzes the computational and storage costs from a theoretical perspective, as shown in Tables 2 and 3. It is worth noting that $|\mathbb{R}|$ represents the number of attributes in DU's attribute set, $|\mathbb{S}|$ represents the number of attributes in DO's attribute set, $|\mathbb{N}|$ represents the number of shares in authentication policy $\mathbb{R}$, $|\mathbb{M}|$ represents the number of shares in access policy $\mathbb{S}$, $|\mathbb{I}|$ represents the number of puncture operations on update key, and $|\mathbb{D}|$ represents the number of elements in label set.

Table 2 provides a detailed computational comparison for five algorithms, including **UKeyGen**, **TokenGen**, **LocUpdt**, **Encrypt**, and **FullDec**. Different from [14,21], in terms of user key and policy token generation, FSBiAC embeds DU's attributes into keys and embeds DU's policies into tokens with a large attribute universe, which shows comparable computational overhead with [14,21,23]. During the puncture process, our FSBiAC outsources update operation to the cloud, and requires only $|\mathbb{I}|(3E_0 + 4M_0)$ local computation. While SC is responsible for executing $|\mathbb{I}|(3E_0+M_0)$ secure outsourced computation. In terms of **Encrypt**, our scheme has essentially the same complexity as [14,21]. Due to the embedding of puncture components in encrypted files, Ghopur et al. [23] requires a higher order of resources during the encryption process. In **FullDec** part, compared to [21], FSBiAC effectively reduces local burden through outsourced verification, and requires only $|\mathbb{D}|(E_0 + M_0)$ computation to achieve lightweight puncture verification. Table 3 provides a detailed storage comparison for six components, including initial update key IK_0, outsourced decryption key UK, policy token Tok, update keys IK_x and IK_0', encrypted files EF, and transformed file TF. It is worth noting that during update key phase, since FSBiAC outsources puncture process, SC is responsible for its secure maintenance. As the number of punctures $|\mathbb{I}|$ increases, SC grows its storage cost by $3|\mathbb{I}||\mathbb{G}_0| + |\mathbb{D}|$ each time. And DU only needs to consume initial update key IK_0 required $3|\mathbb{I}||\mathbb{G}_0|$ overhead and accumulated value $Z = \sum_{\xi=1}^{x} z_\xi$ required $|\mathbb{Z}_p|$ overhead, which avoids keeping history puncture information locally. In contrast, this process is performed by DU in both [21,23]. As the update frequency increases, they would face the issues in terminal overheads that are inconvenient to deploy.

5.2 Experimental results

This section conducts a simulation based on the JPBC Library [24] to compare the computational costs of key algorithms in FSBiAC with those in [21]. The simulation experiment was run on a Lenovo desktop equipped with an Intel(R) Core(TM) i7-9700 processor and 16.0 GB onboard memory. For lightweight computation and high security, the simulation utilizes "e.properties", "a1.properties", and "a.properties" (referred to as Type-E, Type-A1, and Type-A) from the JPBC library[2]. Therein, Type-E based on the Diophantine equation $DV^2 = 4q - t^3$ is

2) The main algorithm code can be found in an open-source repository at https://github.com/Reichenbachxd1202/FSBiAC.

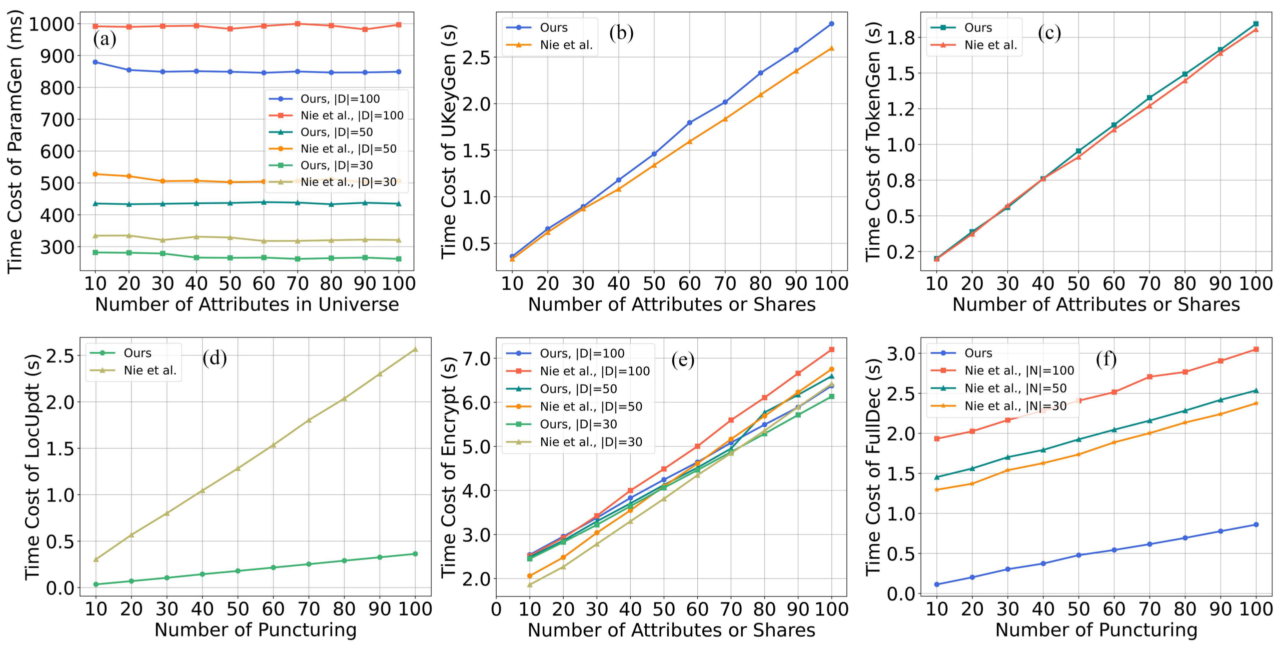

Figure 3 (Color online) Computation cost for main algorithms. (a) **ParamGen**; (b) **UKeyGen**; (c) **TokenGen**; (d) **LocUpdt**; (e) **Encrypt**; (f) **FullDec**.

Table 4 Computational overhead (ms) of various algorithms under Type-E and Type-A1 curves.

Algorithm (attribute/share/label)	Type-E [21]		Type-A1 [21]		Type-E FSBiAC		Type-A1 FSBiAC	
	50	100	50	100	50	100	50	100
ParamGen	2757.34	2754.62	15893.26	15952.24	2314.14	2327.99	15127.74	14672.84
UKeyGen	3697.37	7239.41	22561.91	43874.35	3943.89	7775.78	24707.18	48303.33
TokenGen	2522.87	5006.29	15389.92	30403.49	3308.69	6668.81	20675.94	41199.71
LocUpdt	3621.14	6949.98	22366.81	44043.79	483.19	977.31	2937.46	5850.84
Encrypt	12364.13	19642.41	77161.12	124296.15	11794.87	17322.65	72053.52	102951.4
FullDec	7623.68	9885.08	43953.95	55857.49	1206.83	2369.32	7546.18	14765.54

evaluated within the finite field $\mathbb{F}_q$, and no field expansion is required. Both Type-A1 and Type-A are built on the super-singular curve $y^2 = x^3 + x$.

Figure 3 shows the time overhead comparison of several algorithms between our scheme and [21] under the Type-A elliptic curve. In **ParamGen**, we calculated the time cost when the number $|\mathbb{D}|$ in the label set is 30, 50, and 100, respectively. Due to collision-resistant hash functions and the public parameter computation in groups G_0 and G_T, Nie et al. [21] incurs slightly higher overhead. Since **UKeyGen** and **TokenGen** require access policies specification and shares generation, their computational and storage costs are slightly higher than those of [21], but still fall within an acceptable computational range. In **LocUpdt**, the outsourcing of key updates is securely computed by the semi-trusted cloud, thus greatly reducing local computational costs. When the number of punctures $|\mathbb{I}|$ reaches 50 and 100, the cost of FSBiAC is 179.28 and 363.30 ms, respectively, significantly better than scheme [21]'s 1283.55 and 2566.89 ms. In **Encrypt**, Nie et al. [21] requires re-encryption operations, thus incurring slightly higher costs than FSBiAC. It is noteworthy that in **FullDec**, due to the outsourcing maintenance of updated keys and the verification of outsourced punctures, the actual performance of FSBiAC is better than [21]. Specifically, when decrypting with 100 shares and the number of punctures $|\mathbb{I}|$ reaches 50 and 100, the cost of **FullDec** is 478.54 and 860.45 ms, respectively, which is better than [21]'s 2408.09 and 3050.49 ms.

For brevity, the data under the Type-E and Type-A1 elliptic curves can be intuitively obtained from Table 4. Similar to the results of Type-A curve in Figure 3, FSBiAC has a slightly higher cost in **UKeyGen** and **TokenGen** due to policy embedding and the application of Boneh-Boyen hash function. **LocUpdt**, **Encrypt**, and **FullDec** algorithms executed by terminals (DOs and DUs) show a clear efficiency advantage.

Figure 4 illustrates the storage comparison between FSBiAC and [21] under Type-E, Type-A1, and Type-A elliptic curves. Notably in **LocUpdt** algorithm, the outsourced key update significantly reduces the actual storage overhead of FSBiAC compared to [21]. Specifically in [21], key update storage increases linearly with the number

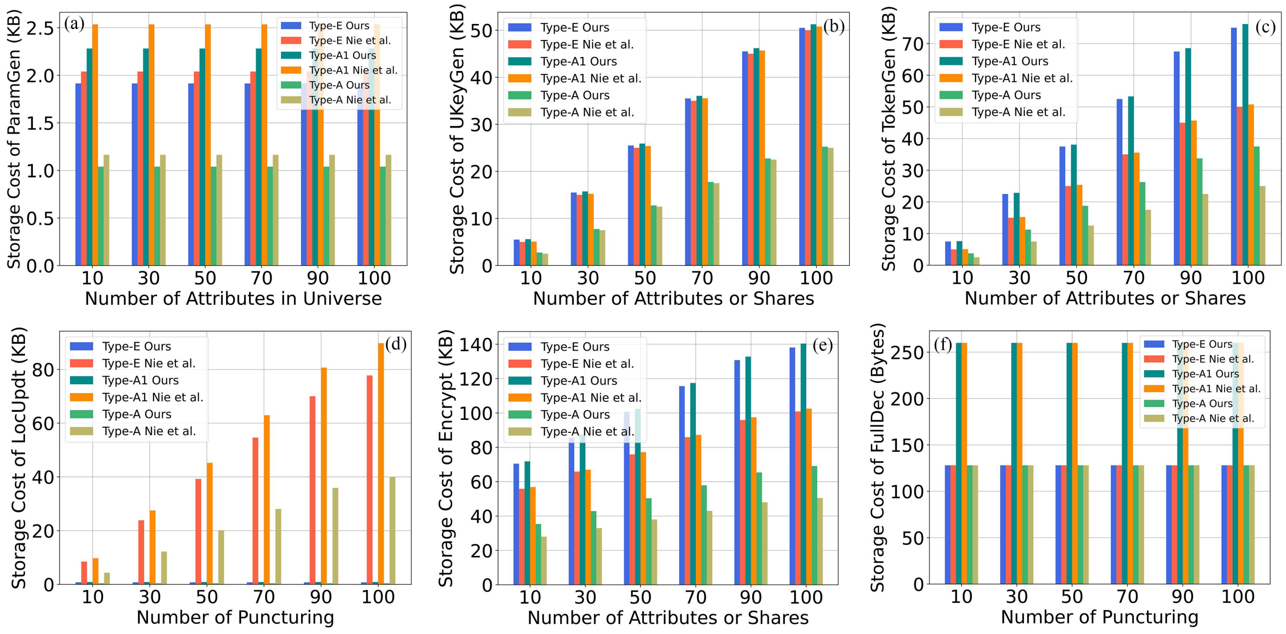

Figure 4 (Color online) Storage cost for main algorithms. (a) **ParamGen**; (b) **UKeyGen**; (c) **TokenGen**; (d) **LocUpdt**; (e) **Encrypt**; (f) **FullDec**.

Table 5 Storage overhead (Bytes) of various algorithms under Type-E and Type-A1 curves. ①: outsourced decryption key $UK = uk_{[1-4]}$; ②: initial update key $IK_0 = ik_{[0,1-4]}$.

Algorithm (attribute/share/label)	Type-E [21]		Type-A1 [21]		Type-E FSBiAC		Type-A1 FSBiAC	
	50	100	50	100	50	100	50	100
ParamGen	2088	2088	2596	2596	1960	1960	2336	2336
UKeyGen	① 25600 ② 788	① 51200 ② 788	① 26000 ② 908	① 52000 ② 908	① 26112 ② 788	① 51712 ② 788	① 26520 ② 908	① 52520 ② 908
TokenGen	25600	51200	26000	52000	38400	76800	39000	78000
LocUpdt	40188	79588	46308	91708	788	788	908	908
Encrypt	77696	103296	79040	105040	103040	141400	104780	143780
FullDec	128	128	260	260	128	128	260	260

Table 6 Algorithm execution results (s, KB) on Gnutella, wiki-Talk, and CA-AstroPh datasets.

Algorithm (dataset)	Scheme [21]			FSBiAC		
	Gnutella	wiki-Talk	CA-AstroPh	Gnutella	wiki-Talk	CA-AstroPh
Encrypt	Comp: 39.62 Stor: 5031.38	Comp: 3038.99 Stor: 627708.37	Comp: 394.37 Stor: 49552.13	Comp: 25.04 Stor: 4999.75	Comp: 2041.81 Stor: 627676.75	Comp: 220.15 Stor: 49520.50
FullDec	Comp: 144.63 Stor: 421.04	Comp: 14398.33 Stor: 64910.30	Comp: 1657.44 Stor: 5163.55	Comp: 3.58 Stor: 421.04	Comp: 312.57 Stor: 64910.31	Comp: 43.93 Stor: 5163.55

of punctures $\mathbb{I}$. When $|\mathbb{I}|$ reaches 50 and 100, the costs are 40188 and 79588 Bytes under Type-E curve, 46308 and 91708 Bytes under Type-A1 curve, 20604 and 40804 Bytes under Type-A curve, respectively. In contrast, FSBiAC maintains a stable overhead for updated key storage across all three types of curves, with costs of 788, 908, and 404 Bytes, which do not increase with the number of punctures $|\mathbb{I}|$. For clarity, some storage overhead under Type-E and Type-A1 curves can be obtained from Table 5. Additionally, based on p2p-Gnutella, wiki-Talk, and CA-AstroPh datasets collected from Stanford Network SNAP [25][3], we test the computational and storage costs of **Encrypt** and **FullDec** for both schemes, with the number of shares, attributes, and elements in the label set all set to 50. We manage the thread pool through *ExecutorService* and break down the computing task into several blocks. As shown in Table 6, the computational efficiency of **Encrypt** and **FullDec** on three datasets is improved by about 1.5 times and 40 times, respectively, compared to [21]. Hence, our FSBiAC demonstrates better scalability and is suitable for resource-limited IoT terminals.

3) Link to the datasets is available at http://snap.stanford.edu/data/other.html.

6 Conclusion

To address the issue of illegal access and forgery of sensitive data in end-cloud collaborative IoT, this paper proposes a forward secure bilateral access control scheme, called FSBiAC, to ensure the full lifecycle security of encrypted data. It integrates ABME and PE to achieve both fine-grained access control and fine-grained authentication. FSBiAC adopts the ME paradigm to enable bilateral verification between policies and attributes. Furthermore, FSBiAC revokes directional keys' access to specific labels, and prevents adversaries from using the current keys to expose historical data, thereby ensuring forward security. Different from traditional ABME, our scheme offloads a significant portion of puncture tasks to the cloud, thus effectively reducing the high local computation and storage costs in traditional forward secure methods. Security analysis and simulation results prove that it offers superior security and practicality in end-cloud collaborative IoT compared to previous studies. In the complexly deployed cloud-network-end collaborative architecture [2], the revocation and addition of terminals and edge devices frequently change. With the separation of data ownership, data management, and usage permissions, it is a worthwhile consideration that compromised keys cannot access historical and future data. With the collaborative nature of terminals, edge networks, and the cloud, future research will focus on constructing a lightweight bilateral access control scheme that supports both forward and backward security.

Acknowledgements This work was supported by Foundation for Innovative Research Groups of the National Natural Science Foundation of China (Grant No. 62121001), Key Program of the National Natural Science Foundation of China (Grant No. 62232013), National Natural Science Foundation of China (Grant Nos. 62125205, 62202364, U23A20303, 62372345), Fellowship of China National Postdoctoral Program for Innovation Talents (Grant No. BX20230279), Natural Science Basic Research Program of Shaanxi Province (Grant No. 2023-JC-JQ-49), China Postdoctoral Science Foundation (Grant No. 2024M752534), and Fundamental Research Funds for the Central Universities (Grant No. ZYTS25070).

References

1 Wu W W, Hu S, Lin D, et al. Reliable resource allocation with RF fingerprinting authentication in secure IoT networks. Sci China Inf Sci, 2022, 65: 170304

2 Wang X, Ma J. Cloud-network-end collaborative security for wireless networks: architecture, mechanisms, and applications. Tsinghua Sci Technol, 2025, 30: 18–33

3 Yang Z M, Ji W, Wang Z. Adaptive joint configuration optimization for collaborative inference in edge-cloud systems. Sci China Inf Sci, 2024, 67: 149103

4 Zhang J, Li T, Ying Z, et al. Trust-based secure multi-cloud collaboration framework in cloud-fog-assisted IoT. IEEE Trans Cloud Comput, 2023, 11: 1546–1561

5 Tian C, Ma J, Li T, et al. Provably and physically secure UAV-assisted authentication protocol for IoT devices in unattended settings. IEEE Trans Inform Forensic Secur, 2024, 19: 4448–4463

6 Kong L, Tan J, Huang J, et al. Edge-computing-driven Internet of Things: a survey. ACM Comput Surv, 2022, 55: 1–41

7 Tanveer M, Chelloug S A, Alabdulhafith M, et al. Lightweight authentication protocol for connected medical IoT through privacy-preserving access. Egyptian Inf J, 2024, 26: 100474

8 Cisco. 2023 Global Networking Trends Report: New Rules for a Multicloud World. 2023. https://www.cisco.com/c/dam/global/en_ca/solutions/enterprise-networks/xa-09-2023-networking-report.pdf

9 Zhang W, Deng D, Wu X, et al. An adaptive asynchronous federated learning framework for heterogeneous Internet of Things. Inf Sci, 2025, 689: 121458

10 Song M, Wang D. AB-PAKE: achieving fine-grained access control and flexible authentication. IEEE Trans Inform Forensic Secur, 2024, 19: 6197–6212

11 Liu C, Hou C, Jiang T, et al. FACOS: enabling privacy protection through fine-grained access control with on-chain and off-chain system. IEEE Trans Inform Forensic Secur, 2024, 19: 7060–7074

12 Zhang Y, Chen J, Ning J, et al. Flexible privacy-preserving data computing with bilateral access control for cloud-assisted IoT. IEEE Trans Dependable Secure Comput, 2025, 22: 3166–3178

13 Xu S, Ning J, Huang X, et al. Server-aided bilateral access control for secure data sharing with dynamic user groups. IEEE Trans Inform Forensic Secur, 2021, 16: 4746–4761

14 Xu S M, Ning J T, Li Y J, et al. Match in my way: fine-grained bilateral access control for secure cloud-fog computing. IEEE Trans Depend Secure Comput, 2022, 19: 1064–1077

15 Ma J, Xu S, Ning J, et al. Catch me if you can: a secure bilateral access control system with anonymous credentials. IEEE Trans Serv Comput, 2023, 16: 4444–4455

16 Yang N, Tang C, Xiong Z, et al. RCME: a reputation incentive committee consensus-based for matchmaking encryption in IoT healthcare. IEEE Trans Serv Comput, 2024, 17: 2790–2806

17 Deng H, Yin H, Qin Z, et al. Toward fine-grained and forward-secure access control in cloud-assisted IoT. IEEE Internet Things J, 2024, 11: 36569–36580

18 Wei J, Chen X, Wang J, et al. Enabling (end-to-end) encrypted cloud emails with practical forward secrecy. IEEE Trans Dependable Secure Comput, 2022, 19: 2318–2332

19 Wei J, Chen X, Wang J, et al. Towards secure asynchronous messaging with forward secrecy and mutual authentication. Inf Sci, 2023, 626: 114–132

20 Huang Q A, Si Y W. Certificateless and revocable bilateral access control for privacy-preserving edge-cloud computing. IEEE Internet Things J, 2025, 12: 10333–10348

21 Nie X Y, Yuan Y, Sun J F. Puncturable attribute-based matchmaking encryption scheme. J Cryptol Res, 2022, 9: 883–898

22 Sun J, Xu G, Zhang T, et al. A practical fog-based privacy-preserving online car-hailing service system. IEEE Trans Inform Forensic Secur, 2022, 17: 2862–2877

23 Ghopur D, Ma J F, Ma X D, et al. Puncturable ciphertext-policy attribute-based encryption scheme for efficient and flexible user revocation. Sci China Inf Sci, 2023, 66: 172104

24 de Caro A, Iovino V. jPBC: Java pairing based cryptography. In: Proceedings of the 16th IEEE Symposium on Computers and Communications (ISCC 2011), Kerkyra, 2011. 850–855

25 Leskovec J, Sosič R. SNAP: a general-purpose network analysis and graph-mining library. ACM Trans Intell Syst Technol, 2016, 8: 1–20

26 Ateniese G, Francati D, Nunez D, et al. Match me if you can: matchmaking encryption and its applications. In: Proceedings of the 39th Annual International Cryptology Conference (CRYPTO 2019), Santa Barbara, 2019. 701–731

27 Wu A X, Weng J, Luo W Q, et al. Cross-domain identity-based matchmaking encryption. Cryptology ePrint Archive, 2022. https://eprint.iacr.org/2022/085

28 Yan Z, Lin X, Zhang X, et al. Identity-based matchmaking encryption with equality test. Entropy, 2024, 26: 74

29 Lin S, Li Y, Chen J. CCA-secure identity-based matchmaking encryption from standard assumptions. In: Proceedings of the 19th International Conference on Information Security and Cryptology (Inscrypt 2023). Singapore: Springer, 2024. 253–273

30 Li J, Zhang E, Han J, et al. PH-MG-ABE: a flexible policy-hidden multigroup attribute-based encryption scheme for secure cloud storage. IEEE Internet Things J, 2025, 12: 2146–2157

31 Li J, Yao W, Zhang Y, et al. Flexible and fine-grained attribute-based data storage in cloud computing. IEEE Trans Serv Comput, 2017, 10: 785–796

32 Wu A, Luo W, Weng J, et al. Fuzzy identity-based matchmaking encryption and its application. IEEE Trans Inform Forensic Secur, 2023, 18: 5592–5607

33 Huang W, Wu A, Xu S, et al. EASNs: efficient anonymous social networks with enhanced security and high scalability. IEEE Trans Inform Forensic Secur, 2025, 20: 796–806

34 Xu S M, Ning J T, Ma J H, et al. Expressive bilateral access control for Internet-of-Things in cloud-fog computing. In: Proceedings of the 26th Symposium Access Control Models and Technologies (SACMAT 2021), 2021: 143–154

35 Zhang K, Wang X, Ning J, et al. Secure cloud-assisted data pub/sub service with fine-grained bilateral access control. IEEE Trans Inform Forensic Secur, 2023, 18: 5286–5301

36 Hu X, Wang L, Gu L, et al. A bilateral access control data sharing scheme for Internet of vehicles. IEEE Internet Things J, 2024, 11: 36748–36762

37 Green M D, Miers I. Forward secure asynchronous messaging from puncturable encryption. In: Proceedings of the 36th IEEE Symposium on Security and Privacy (S&P 2015), San Jose, 2015. 305–320

38 Goldreich O, Goldwasser S, Micali S. How to construct random functions. J ACM, 1986, 33: 792–807

39 Phuong T V X, Ning R, Xin C S, et al. Puncturable attribute-based encryption for secure data delivery in Internet of Things. In: Proceedings of the 37th IEEE Conference on Computer Communications (INFOCOM 2018), Honolulu, 2018. 1511–1519

40 Ghopur D, Ma J, Ma X, et al. Puncturable key-policy attribute-based encryption scheme for efficient user revocation. IEEE Trans Serv Comput, 2023, 16: 3999–4011

41 Liu S G, Hu Y Z, Wang X A, et al. Puncturable-based broadcast encryption with tracking for preventing malicious encryptors in cloud file sharing. J Inf Secur Appl, 2024, 84: 103803

42 Rouselakis Y, Waters B. Practical constructions and new proof methods for large universe attribute-based encryption. In: Proceedings of the 20th ACM SIGSAC Conference on Computer and Communications Security (CCS 2013), Berlin, 2013. 463–474

43 Qin B, Zhao Q, Zheng D, et al. (Dual) server-aided revocable attribute-based encryption with decryption key exposure resistance. Inf Sci, 2019, 490: 74–92

44 Wei J H, Tian G H, Wang D, et al. Pixel+ and Pixel++: compact and efficient forward-secure multi-signatures for PoS blockchain consensus. In: Proceedings of the 33rd USENIX Security Symposium (USENIX Security 2024), Seattle, 2024. 6327–6254