Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.069628

ARTICLE

Searchable Attribute-Based Encryption with Multi-Keyword Fuzzy Matching for Cloud-Based IoT

He Duan, Shi Zhang* and Dayu Li

School of Computer Science and Engineering, Northeastern University, Shenyang, 110819, China

*Corresponding Author: Shi Zhang. Email: zhangshi@mail.neu.edu.cn

Received: 27 June 2025; Accepted: 15 September 2025; Published: 09 December 2025

ABSTRACT: Internet of Things (IoT) interconnects devices via network protocols to enable intelligent sensing and control. Resource-constrained IoT devices rely on cloud servers for data storage and processing. However, this cloud-assisted architecture faces two critical challenges: the untrusted cloud services and the separation of data ownership from control. Although Attribute-based Searchable Encryption (ABSE) provides fine-grained access control and keyword search over encrypted data, existing schemes lack of error tolerance in exact multi-keyword matching. In this paper, we proposed an attribute-based multi-keyword fuzzy searchable encryption with forward ciphertext search (FCS-ABMSE) scheme that avoids computationally expensive bilinear pairing operations on the IoT device side. The scheme supports multi-keyword fuzzy search without requiring explicit keyword fields, thereby significantly enhancing error tolerance in search operations. It further incorporates forward-secure ciphertext search to mitigate trapdoor abuse, as well as offline encryption and verifiable outsourced decryption to minimize user-side computational costs. Formal security analysis proved that the FCS-ABMSE scheme meets both indistinguishability of ciphertext under the chosen keyword attacks (IND-CKA) and the indistinguishability of ciphertext under the chosen plaintext attacks (IND-CPA). In addition, we constructed an enhanced variant based on type-3 pairings. Results demonstrated that the proposed scheme outperforms existing ABSE approaches in terms of functionalities, computational cost, and communication cost.

KEYWORDS: Cloud computing; Internet of Things; ABSE; multi-keyword fuzzy matching; outsourcing decryption

1 Introduction

With the rapid development of Internet of Things (IoT), pervasive IoT devices (IoTDs) have become capable of sensing their environments and processing data, thereby bridging the physical and digital worlds. IoT has been widely applied in industrial production, agriculture, public safety monitoring, smart cities, mobile healthcare, and so on. For instance, IoT-enabled medical systems utilize embedded and wearable equipments to collect health-related data for clinical diagnosis [1]. It was predicted that the growth of wearable IoTDs will reach 265.4 USD billion in 2026 [2].

To address the computational and storage limitations of IoTDs, cloud computing has emerged as a cost-effective and scalable solution. In cloud-assisted IoT systems, IoTDs like sensors transmit data to cloud servers via wireless networks. The cloud infrastructure offers abundant storage and computational resources, enabling efficient data processing and reducing the burden on IoTDs. However, outsourcing sensitive IoT data to third-party cloud servers introduces serious privacy leakage and security concerns.

Encrypting IoT data before outsourcing is a simple solution to protecting privacy. However, once data is encrypted and stored, users cannot directly retrieve specific data. Instead, they are forced to download all ciphertexts, which significantly degrades system efficiency. Searchable encryption (SE) scheme [3] was introduced to address this issue. By leveraging keyword matching, users can search encrypted data to retrieve specific information without decrypting the entire dataset. SE schemes typically bind search capabilities to a specific user's key, limiting their effectiveness in multi-user environments. To achieve fine-grained access control of data, attribute-based encryption (ABE) mechanisms [4] have been introduced. ABE can identify users based on their attribute sets, grant each user unique identity characteristics, and provide flexible representation of access control policies at the expense of increased computational and communication costs compared to traditional encryption techniques. Attribute-based keyword-searchable encryption (ABSE) scheme [5] can address the fine-grained access control and search issue by combining ABE and SE. In ABSE schemes, data users can only retrieve and recover the plaintext if their own attribute set matches the access control policy specified by the data owner. However, most existing ABSE schemes support only exact keyword matching, which may fail to reflect a user's actual search intent and limits the expressiveness of search strategies.

1.1 Related Work

1.1.1 Searchable Encryption

In 2000, Song et al. [3] introduced searchable encryption, enabling data users to perform keyword searches on encrypted cloud data without decryption. They implemented a symmetric searchable encryption (SSE) scheme based on symmetric cryptography. However, this scheme requires a centralized key management authority to distribute keys to users, leading to complex key distribution and management challenges. After that, although many improved SSE schemes have been introduced and formally proven secure, they are unsuitable for multi-user data-sharing cases. To address this, Boneh et al. [6] in 2004 suggested a public key encryption with keyword search (PEKS) scheme based on identity-based encryption [7] for ciphertext retrieval in public key cryptosystems. PEKS allows a sender to create searchable ciphertext using a keyword and the public key of the receiver. The receiver generates a trapdoor using its private key and sends it to cloud servers. The server returns matching ciphertexts to the user if and only if the keyword in the trapdoor matches the one in the ciphertext. Chen et al. [8] improved multi-keyword PEKS by reducing computational cost and trapdoor sizes. However, their scheme strictly enforces exact-match queries. Any input errors or formatting inconsistencies in user queries will prevent the system from returning valid results, thus severely compromising practical usability. For error-tolerant scenarios, Li et al. [9] first suggested fuzzy search PEKS using wildcards and edit distance. Servers can return results similar to the queried keywords at the expense of significantly increased storage and computational overhead. Dong et al. [10] designed a homomorphic encryption based fuzzy keyword search scheme which is more efficient than [9]. Zhang et al. [11] introduced a scalable fuzzy keyword ranked search scheme over encrypted data to achieve fuzzy keyword search for any similarity threshold with a constant storage size and accurate search results. Jiang et al. [12] proposed a fuzzy keyword searchable encryption scheme based on blockchain. Their scheme leverages blockchain to ensure equitable service payment transactions between users and cloud servers.

Most existing SE schemes exhibit expressive search policies and support only single-keyword or conjunctive keyword queries. To address expressiveness challenges in SE, Lai et al. [5] proposed an expressive PEKS (EPEKS) scheme on composite-order groups with high computational costs. Cash et al. [13] designed a searchable symmetric encryption, which supports boolean queries. Xu et al. [14] developed an efficient multi-keyword search mechanism that operates without predefined keyword fields, utilizing an anti-collusion box to enable boolean search functionality. Li et al. [15] further advanced this domain by proposing

a verifiable boolean search protocol over encrypted data within an SSE framework, which guarantees forward privacy and weak backward privacy. Tong et al. [16] proposed a privacy-preserving boolean range query with temporal access control. Their scheme achieves privacy-preserving boolean range query and forward/backward derivation functions. Liu et al. [17] developed a verifiable dynamic encryption with ranked search (VDERS) scheme to secure top-K searches and verifiable results on dynamic document collections.

In terms of search privacy, Li et al. [18] introduced forward search privacy, enabling secure searches on newly added documents without leaking historical query information. Gan et al. [19] introduced a forward private SSE scheme for multiple clients. Their scheme involves no bilinear pairing computation, which has better efficiency on search. Guo et al. [20] designed a verifiable and forward-privacy SSE scheme that achieves sublinear search time and efficient file updates with guaranteed security. Cui et al. [21] proposed a dynamic and verifiable fuzzy keyword search with forward security scheme that adopts uni-gram and locality-sensitive hashing (LSH) to achieve efficient search.

1.1.2 Attributed-Based Encryption

Sahai and Waters [22] first proposed fuzzy identity-based encryption in 2005, laying the foundation for ABE as a means of achieving fine-grained access control. Building on this, Goyal et al. [4] formally defined two principal variants of ABE schemes: key-policy attribute-based encryption (KP-ABE) and ciphertext-policy attribute-based encryption (CP-ABE). In KP-ABE, secret keys are related to access control policies and ciphertexts are associated with attribute sets. CP-ABE inversely links ciphertexts to access control policies while binding secret keys to attribute sets. Bethencourt et al. [23] subsequently proposed the first concrete construction of CP-ABE. However, their implementation employed access trees as policy representation structures, resulting in limited expressive power for complex access control requirements. To improve security, efficiency, and expressiveness, Waters [24] later enhanced CP-ABE by implementing arbitrary monotonic access policies with linear secret sharing scheme (LSSS) matrices. Despite these improvements, these schemes lacked support for attribute negation. Ostrovsky et al. [25] utilized the broadcast revocation mechanism to achieve KP-ABE with negation operations, doubling the size of the ciphertext, keys, and computational costs. Lewko et al. [26] later optimized [25] by reducing public key size with longer ciphertexts.

Extensive ABE works have focused on enhancing ABE schemes. Meng et al. [27] introduced a dual-policy ABME scheme with forward security. However, their scheme requires updating the ciphertext, which is impractical in end-to-end data sharing scenarios. Luo et al. in [28] suggested a hidden policy scheme that relies on the user revocation mechanism on fog nodes, making it vulnerable if those nodes are compromised. Miao et al. [29] introduced a verifiable outsourced ABME scheme, but its lack of cryptographic agility and limited access delegation models restrict its adaptability and flexibility. Duan et al. [30] introduced a muti-authority ABME scheme. This scheme's security depends on trusted third parties (central/attribute authorities), creating a single point of failure if these entities are compromised. Ruan et al. [31] introduced an ABME scheme that supports attribute and user revocation. However, this scheme lacks public traceability, limiting its ability to identify and trace malicious users.

1.1.3 Attributed-Based Searchable Encryption

To enable searchable encryption as well as multi-user access control, Lai et al. [5] designed a key-policy ABSE scheme combining KP-ABE with PEKS. In 2014, Zheng et al. [32] presented a verifiable attribute-based keyword search (VABKS) scheme that outsources complex search operations to cloud servers. Their design uses Bloom filters and digital signatures to ensure that only authorized users can retrieve cloud data, while enabling verification of the cloud server's search integrity. Han et al. [33] introduced the concept of weak

anonymous ABE and established a generic transformation between ABE and key-policy ABSE schemes. They also proposed a multi-user ABSE scheme for flexible collaborative searches on remotely encrypted data. However, this scheme's reliance on composite-order groups resulted in low efficiency. Zhang et al. [34] suggested a blockchain-based anonymous ABSE scheme for data sharing. In their scheme, attributes of the access policy are hidden, ensuring the confidentiality of those attributes. In 2021, Ge et al. [35] proposed an ABSE scheme that supports both attribute-based keyword search and data sharing. Notably, this scheme enables keyword updates during the sharing phase without requiring interaction with the private key generator. Ali et al. [36] introduced a verifiable online/offline multi-keyword search (VMKS) scheme that implements outsourced encryption and decryption. In their scheme, the outsourcing part of the encryption and decryption computations significantly reduces the computational cost. Moreover, their scheme supports threshold-based search, where a match succeeds only when the intersection between the keywords in the trapdoor and those in the ciphertext exceeds the specified threshold. Miao et al. [37] designed an optimized verifiable fine-grained keyword search scheme in the static multi-owner setting (VFKSM) to significantly reduce both the ciphertext length and communication overhead. Their scheme achieves ciphertext length linear in the number of keywords, and employs a threshold signature mechanism to convert multiple signatures into a single threshold signature. Huang et al. [38] suggested an expressive multi-keyword ABSE scheme with ranked search results. While their solution supports AND/OR logical search predicates, it incurs substantial communication and computational overhead. To support attribute tracking and multi-keyword search, Huang et al. [39] proposed an ABSE scheme, which employs a new security model in its proof. However, this scheme suffers from a critical design flaw where portions of user private keys are exposed as trapdoors. Chen et al. [40] introduced a fair-and-exculpable-attribute-based searchable encryption with revocation and verifiable outsourced decryption (FE-ABSE-RV) scheme. Their scheme prevents malicious accusations from the client side against the cloud for returning incorrect results, thereby realizing fair exculpability for both ends. Tian et al. [41] designed an attribute-based heterogeneous data privacy sharing (AB-HDPS) scheme. Their scheme leverages the immutability of blockchain to achieve traceability and auditing of user access behaviors, while also implementing attribute revocation by subset-cover trees.

1.2 Contributions

The above ABSE schemes have limited search capabilities, supporting only single-keyword or simple conjunctive keyword searches. Issues on existing ABSE schemes are summarized as follows.

- **Only support exact match.** In most existing ABSE schemes, the data owner specifies a set of keywords for their data. The cloud server will return the search ciphertext when keywords in the ciphertext and keywords in the trapdoor match exactly (i.e., both keyword sets are identical or the keyword set in the trapdoor is a subset of the keyword set in the ciphertext). This exact match search method exhibits very poor fault tolerance, as even a single erroneous keyword input (such as a typographical error) will result in failed searches.
- **No forward security.** Existing ABSE schemes require data users to create a trapdoor for querying encrypted data stored on cloud servers. However, third-party cloud servers are often untrusted. After completing the search, the cloud server can store these trapdoors and perform unauthorized future searches. The cloud server can verify whether newly added encrypted data matches previous search queries, which may result in the frequency attack issue [42] and expose keyword privacy.
- **High computational and communication costs.** In most existing verifiable ABSE schemes, data users needs download the partially decrypted ciphertext to verify the validity of outsourced decryption. This means that even if the cloud server performs search and partial decryption operations, users still need

to download the whole ciphertext. This increases the user's bandwidth and communication overhead, which is not suitable for resource-constrained IoT applications.

To address these challenges, we introduced an attribute-based multi-keyword fuzzy searchable encryption with forward ciphertext search (FCS-ABMSE) scheme. FCS-ABMSE supports multi-keyword fuzzy search based on the number of elements of the intersection keyword sets, forward security, and outsourced decryption and verification. The main functions are listed below.

- **Multi-keyword fuzzy matching:** Our scheme allows ciphertext and query keywords to have a certain degree of lexical deviation rather than being restricted to exact matching for search fault tolerance in real-world scenarios.
- **Forward security:** Even if query a trapdoor is leaked, the scheme can ensure that attackers cannot access historical encrypted data.
- **Outsourced decryption and verification:** Outsourced encryption and verification can reduce the computational cost of data uses, and mitigate the risk of the cloud server tampering with query results or altering data.

1.3 Organization

The rest of this paper is organized as follows. Section 2 outlines cryptographic preliminaries. Section 3 presents the system model. Section 4 explains the proposed FCS-ABMSE scheme with formal security analysis, followed by performance evaluations in Section 5. Section 6 presents an enhanced scheme while Section 7 concludes this paper with future work.

2 Preliminaries

This section describes essential cryptographic preliminaries employed in our scheme.

2.1 Notations

Notations used throughout the paper are defined in Table 1.

Table 1: Notations and descriptions

Notations	Descriptions	Notations	Descriptions
p	Large prime number	U_a	System global attribute set
G, G_T	Multiplicative cyclic groups of order p	U_w	System global keyword set
g	Generator of G	att_{du}	Attribute set of data user
e	Bilinear map	ps	Global parameter set
Π	Data encapsulation scheme	Λ	Access control policy
Φ	Commitment scheme	msk	Master key
Ψ	Message authentication code scheme	$[1, n]$	$\{1, 2, \ldots, n\}$

2.2 Bilinear Map

Definition 1. Let G_1, G_2, and G_T be three cyclic groups of prime order p, with g_1 denoting a generator of G_1 and g_2 denoting a generator of G_2. A map $e : G_1 \times G_2 \to G_T$ is said to be a bilinear map if it satisfies the following properties:

- For any $v, v \in \mathbb{Z}_p^*$, $e(g_1^v, g_2^v) = e(g_1, g_2)^{vv}$;

- $e(g_1, g_2) \neq 1_{G_T}$, where 1_{G_T} denotes the identity of G_T;
- For any $v, v \in \mathbb{Z}_p^*$, $e(g_1^v, g_2^v)$ can be computed efficiently.

A bilinear map $e : G_1 \times G_2 \to G_T$ is symmetric if $G_1 = G_2 (= G)$; Otherwise, it is asymmetric.

2.3 Difficult Problems

The security of the proposed scheme relies on two difficult problems.

Definition 2. Let (G, G_T) be prime order cyclic groups. The decisional bilinear Diffie-Hellman (DBDH) problem is to determine if $T = e(g, g)^{abc}$, for $a, b, c \in \mathbb{Z}_p^*$ and $T \in G_T$.

Definition 3. Let (G, G_T) be prime order cyclic groups. The decisional q-parallel bilinear Diffie-Hellman (q-parallel-BDHE) problem is to determine whether $T = e(g, g)^{\beta q+1}$, for $\beta, s, b_1, \ldots, b_q \in \mathbb{Z}_p^*$, $(g, \tilde{y})$ and $T \in G_T$, where

$$\tilde{y} = \left(\begin{array}{llllllll} g, & g^s, & g^\beta, & \ldots, & g^{\beta^q}, & g^{(\beta^{q+2})}, & \ldots, & g^{(\beta^{2q})} \\ \forall 1 \leq j \leq q, & g^{sb_j}, & g^{a/b_j}, & \ldots, & g^{(a^q/b_j)}, & g^{(a^{q+2}/b_j)}, & \ldots, & g^{(a^{2q}/b_j)} \\ \forall 1 \leq j, k \leq q, k \neq j & g^{(asb_k/b_j)}, & \ldots, & g^{(a^q sb_k/b_j)} & & & & \end{array} \right).$$

2.4 Linear Secret Sharing Scheme

Definition 4. A linear secret sharing scheme (LSSS) for a set of participants $\mathcal{P}$ is considered linear if it can meet the following conditions:

- Each participant receives a share represented as a vector with components in $\mathbb{Z}_p^*$.
- There exists a matrix $\mathbf{M}$ with ℓ rows and n columns. Each row i of $\mathbf{M}$ (for $i \in [\ell]$) is associated with a participant labeled as $x_i \in \mathcal{P}$. To share a secret, select a column vector $\vec{v} = (s, v_2, \ldots, v_n)$, where $s \in \mathbb{Z}_p^*$ is the secret and $v_2, \ldots, v_n \in \mathbb{Z}_p^*$ are random elements. The product $\mathbf{M}\vec{v}$ produces ℓ shares of the secret s. Specifically, the share corresponding to participant x_i is given by $M_i v$, where M_i denotes the i-th row of $\mathbf{M}$.

2.5 Key Derivation Function

Definition 5. A key derivation function (KDF) accepts a secret k and a length l, and then generates a cryptographic key $key \in \{0,1\}^l$.

Definition 6. A KDF is said to be secure if the advantage $Adv_{\mathrm{KDF}}^{\mathcal{A}_d} = |\Pr[\mathcal{A}_d(KDF(k, l)) = 1] - \Pr[\mathcal{A}_d(R) = 1]|$ is negligible for any polynomial-time adversary $\mathcal{A}_d$, where $R \in \{0,1\}^l$ is a random cryptographic key chosen from the key space.

2.6 Data Encapsulation Mechanism

Definition 7. A data encapsulation mechanism (DEM) is a symmetric encryption scheme consisting of two algorithms:

- Enc: Input a symmetric key $k \in \mathcal{K}$ and a message M, output a ciphertext C_M, where $\mathcal{K}$ is the key space.
- Dec: Input $k \in \mathcal{K}$ and C_M, output the decrypted message m or a special symbol $\perp$ to indicate a decryption failure.

Definition 8. A DEM $\Pi = (Enc, Dec)$ is said to achieve indistinguishability under chosen-ciphertext attack (IND-DEM-CCA) security if the following advantage $Adv_{DEM}^{CCA}(\mathcal{A}_d)$ is negligible for any polynomial-time

adversary $\mathcal{A}_d$:

$$Adv_{DEM}^{CCA}(\mathcal{A}_d) = \Big| \Pr\Big[b = b' : (M_0, M_1) \leftarrow \mathcal{A}_d(1^\eta),\ b \leftarrow_R \{0,1\},\ K \leftarrow_R \mathcal{K}, \quad C \leftarrow \text{DEM.Enc}(K, M_b),\ b' \leftarrow \mathcal{A}_d^{O_{\text{Dec}}}(C)\Big] - \tfrac{1}{2}\Big|,$$

where η is the security parameter and O_{Dec} is the decryption oracle.

2.7 Commitment Scheme

Definition 9. A commitment scheme is a cryptographic protocol that is specified by the two following algorithms:

- $Commit$: Given an input string $s \in \{0,1\}^*$, the algorithm outputs a commitment com and a de-commitment dom;
- $Open$: Given a pair (com, dom), the algorithm outputs 1 if the pair is valid, or 0 otherwise.

Definition 10. A commitment scheme $\Phi = (Commit, Open)$ is computationally hiding if the following advantage $Adv^{Hiding}(\mathcal{A}_d)$ is negligible for any polynomial-time adversary $\mathcal{A}_d$:

$$Adv^{Hiding}(\mathcal{A}_d) = \Big| \Pr\Big[b = b' :\ (s_0, s_1) \leftarrow \mathcal{A}_d(1^\eta),\ b \leftarrow_R \{0,1\},\ com \leftarrow Commit(s_b),\ b' \leftarrow \mathcal{A}_d(com)\Big] - \tfrac{1}{2}\Big|,$$

where η is the security parameter.

2.8 Message Authentication Code

Definition 11. A message authentication code (MAC) is a cryptographic scheme that is specified by the two following algorithms:

- Mac : Given a symmetric key $k \in \mathcal{K}$ and a message m, the algorithm outputs a MAC mac, where $\mathcal{K}$ is the key space.
- $MacVrfy$: Given a symmetric key $k \in \mathcal{K}$, a message m and a MAC mac, the algorithm outputs 1 if $mac = Mac(k, m)$, or 0 otherwise.

Definition 12. A MAC scheme $\Psi = (Mac, MacVrfy)$ is said to satisfy existential unforgeability under chosen-message attack (EU-CMA) if the following advantage $Adv_{DEM}^{CCA}(\mathcal{A}_d)$ is negligible for adversary $\mathcal{A}_d$ with polynomial-time:

$Adv_{\text{MAC}}^{\mathcal{A}_d} = \Pr[\langle m', mac'\rangle \neq \langle m, mac\rangle \ \wedge\ MacVrfy(k, m', mac') = 1 | k \leftarrow \mathcal{K}, \quad m \leftarrow \mathcal{A}_d(k), \quad mac = Mac(k, m), \langle m', mac'\rangle \leftarrow \mathcal{A}_d(m, mac)]$, where $\mathcal{K}$ is the key space.

2.9 0/1—Encoding Approach

In the proposed scheme, we compare two time points t and t' using the 0/1-encoding approach [43]. The 0/1-encoding approach consists of two encoding algorithms:

- 0-encoding: Given a n-bit binary integer $t = t_n t_{n-1} \ldots t_1 \in \{0,1\}^n$, the algorithm outputs a set $S_t^0 = \{t_n t_{n-1} \cdots t_{i+1} 1 | t_i = 0, i \in [1, n]\}$ that contains at most $\log_2 n$ elements;
- 1-encoding: Given a n-bit binary integer $t = t_n t_{n-1} \ldots t_1 \in \{0,1\}^n$, the algorithm outputs a set $S_t^1 = \{t_n t_{n-1} \cdots t_i | t_i = 1, i \in [1, n]\}$ that contains at most $\log_2 n$ elements.

According to [43], for any two integer values t and t', $t' > t$ holds if and only if $S_{t'}^1 \cap S_t^0 \neq \varnothing$.

3 System Model, Synax, and Security Definition

This section gives the system model, synax, and security definitions of the FCS-ABMSE scheme in turn.

3.1 System Model

As shown in Fig. 1, the proposed FCS-ABMSE scheme involves the following six different entities:

- **Central Authority (CA):** The CA generates the system's global parameters and master key, and issues attribute keys to users.
- **Sensor's Owner (SO):** The SO encrypts keywords offline and stores offline ciphertexts of keywords onto sensors.
- **Sensors (SN):** The SN is responsible for collecting user data, creating searchable ciphertexts, and then sending them to the CS.
- **Data User (DU):** The DU retrieves matching ciphertext from the cloud server using a trapdoor, downloads them, and then decrypts the data using their decryption key.
- **Cloud Server (CS):** The CS provides ciphertexts storage and retrieval services, while assisting DUs with partial decryption of ciphertexts.
- **System Clock (SC):** The SC provides the current system timestamp in ciphertext and trapdoor generation.

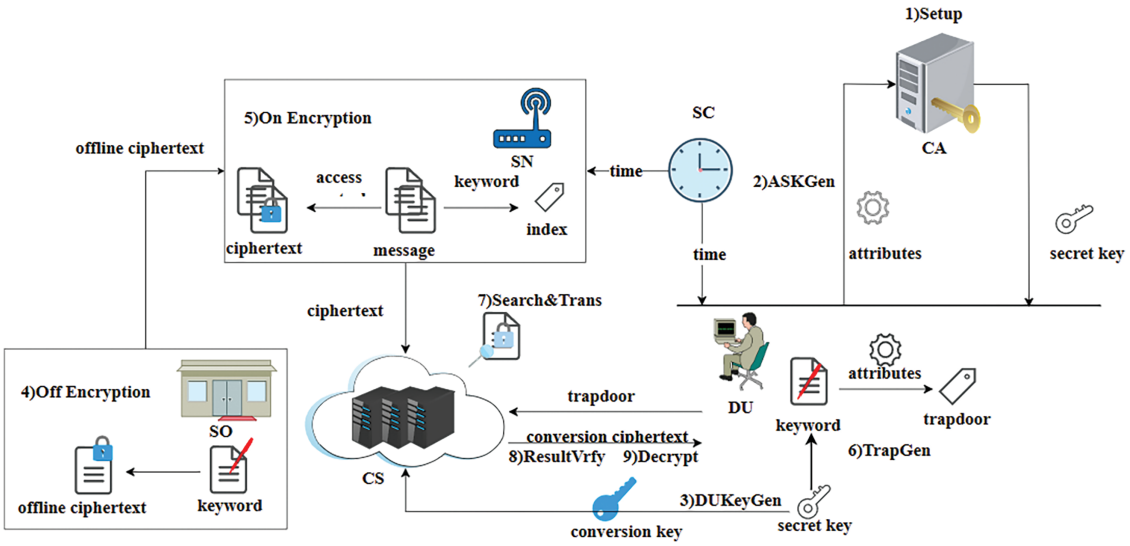

Figure 1: System model of FCS-ABMSE

The interaction among the entities proceeds as follows: First, the CA distributes the system's global parameters to other entities and transmits the attribute keys to the DU. Second, the SC transmits the current system timestamp to the SN and the DU. Third, the SO transmits the offline ciphertext to the SN. Then, the SN transmits the online ciphertext to the CS. After that, the DU transmits the trapdoor to the CS. Last, the CS stores the ciphertexts received from the SN, performs search operations upon receiving trapdoors from the DU, and returns pre-decrypted results to the DU.

3.2 Syntax of FCS-ABMSE

The FCS-ABMSE scheme is defined by the following nine algorithms:

(1) **Setup** (η, U_a, U_w): The CA executes the algorithm to generate the global parameter set ps. Given a security parameter η, the system global attribute set U_a, and the system global keyword set U_w, this algorithm outputs the global parameter set ps and a master key msk.

(2) **ASKGen** (ps, msk, att_{du}): The CA executes the algorithm to issue an attribute key for each DU. Given ps, msk, and an attribute set $att_{du} \subseteq U_a$, this algorithm outputs an attribute key ask_{du}.

(3) **DUKeyGen** (ps, ask_{du}): The DU executes the algorithm to generate a private key and a ciphertext conversion key. Given ps and an attribute key ask_{du}, this algorithm outputs a private key sk_{du} and a ciphertext conversion key tf_{du}.

(4) **offEncrypt** (ps): The SO executes the algorithm to create offline ciphertexts. Given ps, the algorithm outputs an offline ciphertext ct_{off}.

(5) **onEncrypt** $(ps, m, ct_{off}, ws, \Lambda, t)$: The SN executes the algorithm to create online ciphertexts. Given ps, a message m, the offline ciphertext ct_{off}, a keyword set $ws \subseteq U_w$, an access control policy Λ, and the current system time t, this algorithm outputs an online ciphertext ct_{on}.

(6) **TrapGen** $(ps, thd, ws', sk_{du}, t')$: The DU executes the algorithm to create a trapdoor. Given ps, a search threshold thd, a search keyword set $ws' \subseteq U_w$, the DU's private key sk_{du}, and current system time t', this algorithm outputs a trapdoor td.

(7) **Search** $(ps, ct_{on}, td, tf_{du})$: The CS executes the algorithm to make ciphertext searches and pre-decryptions. Given ps, an online ciphertext ct_{on}, a trapdoor td, and a ciphertext conversion key tf_{du}, this algorithm outputs a conversion ciphertext ct_{trans} and a pre-decryption ciphertext pct_m if ct_{on} matches td; otherwise, it returns $\perp$.

(8) **ResultVrfy** $(ps, ct_{trans}, sk_{du})$: The DU executes the algorithm to check the validness of the search results and create corresponding decryption keys. Given ps, a conversion ciphertext ct_{trans}, and the DU's private key sk_{du}, this algorithm outputs a decryption key dk if ct_{trans} is valid; otherwise, it returns an invalid flag $\perp$.

(9) **Decrypt** (ps, dk, pct_m): After verifying the validness of ct_{trans}, the DU downloads pct_m and executes the algorithm to complete final decryption. Given ps, a decryption key dk, and a pre-decryption ciphertext pct_m, this algorithm outputs a message m, or $\perp$ if decryption fails.

Definition 13. An FCS-ABMSE scheme is correct if for a message m, keyword number n, DU's attribute set $att_{du} \subseteq U_w$, search threshold thd, and two keyword sets $ws \subseteq U_w$ and $ws' \subseteq U_w$ such that $|ws \cap ws'| > thd$, $ask_{du} = \textbf{ASKGen}(ps, mtk, att_{du})$, $(sk_{du}, tf_{du}) = \textbf{DUKeyGen}(ps, ask_{du})$, $ct_{off} = \textbf{offEncrypt}(ps)$, $ct_{on} = \textbf{onEncrypt}(ps, m, ct_{off}, ws, \Lambda, t)$, $td = \textbf{TrapGen}(ps, thd, ws', sk_{du}, t')$, $(ct_{trans}, pct_m) = \textbf{Search}(ps, ct_{on}, td, tf_{du})$, and $dk = \textbf{ResultVrfy}(ps, ct_{trans}, sk_{du})$, then $m = \textbf{Decrypt}(ps, dk, pct_m)$.

3.3 Security Definitions of FCS-ABMSE

An FCS-ABMSE scheme must satisfy the two following security notions: indistinguishability of ciphertext under the chosen keyword attack (**IND-CKA**) and indistinguishability of ciphertext under the chosen plaintext attack (**IND-CPA**).

IND-CKA. The security is defined by a game between an adversary $\mathcal{A}_d$ and a challenger $\mathcal{B}$:

Initialization. $\mathcal{A}_d$ submits an access control policy Λ^* to $\mathcal{B}$.

Setup. $\mathcal{B}$ runs **Setup**(η, U) to generate ps and msk, and sends ps to $\mathcal{A}_d$.

Phase 1. $\mathcal{A}_d$ adaptively queries the oracles with the restriction that att_{du} does not satisfy Λ^*. $\mathcal{B}$ answers as follows:

- $\mathcal{O}_{ask}$: Given an attribute set att_{du}, $\mathcal{B}$ runs **ASKGen** (ps, msk, att_{du}) to generate ask_{du}. Then, it returns ask_{du} to $\mathcal{A}_d$.
- $\mathcal{O}_{tf}$: Given an attribute set att_{du}, $\mathcal{B}$ runs **ASKGen** (ps, msk, att_{du}) to generate ask_{du}, and **DUKeyGen** (ps, ask_{du}) to generate tf_{du}. Then, it returns tf_{du} to $\mathcal{A}_d$.
- $\mathcal{O}_{sk}$: Given an attribute set att_{du}, $\mathcal{B}$ runs **ASKGen** (ps, msk, att_{du}) to generate ask_{du}, and **DUKeyGen** (ps, ask_{du}) to generate sk_{du}. Then, it returns sk_{du} to $\mathcal{A}_d$.
- $\mathcal{O}_{td}$: Given a search threshold thd, a keyword set ws', an attribute set att_{du}, and a time point t', $\mathcal{B}$ first obtains sk_{du} via $\mathcal{O}_{sk}$, then runs **TrapGen** $(ps, thd, ws', sk_{du}, t')$ to generate a trapdoor td. Finally, it returns td to $\mathcal{A}_d$.

Challenge. $\mathcal{A}_d$ outputs two keyword sets ws_0 and ws_1 of equal length, a message m^*, and a timestamp t^*. $\mathcal{B}$ runs **offEncrypt** (ps) to generate ct^*_{off}. Then, $\mathcal{B}$ randomly selects $v \in \{0,1\}$ and runs **onEncrypt** $(ps, m^*, ct^*_{off}, ws_v, \Lambda^*, t^*)$ to generate a challenge ciphertext ct^*_{on}. Finally, $\mathcal{B}$ sends ct^*_{on} to $\mathcal{A}_d$.

Phase 2. This phase is identical to Phase 1.

Guess. $\mathcal{A}_d$ outputs a guess v'. If $v = v'$, it wins the game with advantage $Adv^{IND-CKA}_{\mathcal{A}_d} = |\Pr[v = v'] - 1/2|$.

Definition 14. An FCS-ABMSE scheme is **IND-CKA** secure if $Adv^{\text{IND-CKA}}_{\mathcal{A}_d}$ is negligible for all polynomial-time adversaries $\mathcal{A}_d$.

IND-CPA. The security is defined by a game between an adversary $\mathcal{A}_d$ and a challenger $\mathcal{B}$:

Initialization. Identical to the initialization process in the IND-CKA security game.

Setup. Identical to the setup process in the IND-CKA security game.

Phase 1. Identical to Phase 1 in IND-CKA security game, except that $\mathcal{A}_d$ can not make queries to the oracle $\mathcal{O}_{td}$.

Challenge. $\mathcal{A}_d$ outputs two messages m_0 and m_1 of equal length, a keyword set ws^* and a time point t^*. $\mathcal{B}$ first runs **offEncrypt** (ps) to generate ct^*_{off}. $\mathcal{B}$ randomly selects $v \in \{0,1\}$ and runs **onEncrypt** $(ps, m_v, ct^*_{off}, ws^*, \Lambda^*, t^*)$ to generate a challenge ciphertext ct^*_{on}. $\mathcal{B}$ sends ct^*_{on} to $\mathcal{A}_d$.

Phase 2. Same as the IND-CKA security game, except that $\mathcal{A}_d$ cannot make queries to the oracle $\mathcal{O}_{td}$.

Guess. $\mathcal{A}_d$ outputs a guess v'. If $v = v'$, it wins the game with advantage $Adv^{IND-CPA}_{\mathcal{A}_d} = |\Pr[v = v'] - 1/2|$.

Definition 15. An FCS-ABMSE scheme is **IND-CPA** secure if $Adv^{\text{IND-CPA}}_{\mathcal{A}_d}$ is negligible for all polynomial-time adversaries $\mathcal{A}_d$.

4 The Proposed FCS-ABMSE Scheme

This section presents the concrete construction of the FCS-ABMSE scheme, along with its correctness analysis and security proof.

The proposed FCS-ABMSE scheme is described as follows. (1) **Setup** (η, U_a, U_w): Given η, U_a and U_w, it is executed as follows:

- Generate two prime p-order groups (G, G_T) and $e : G \times G \to G_T$;
- Randomly choose $g, g_0, g_1 \in G$ and $\alpha, \beta, \gamma \in \mathbb{Z}^*_p$, compute $X = g^\beta$, $Y = g^\gamma$, $E_1 = e(g^\alpha, g)$, $E_2 = e(X, g_1)$, and choose a hash function $H : \{0,1\}^* \to \mathbb{Z}^*_p$;
- For each attribute in U_a, select a random element $h_i \in G$ where $i \in [1, |U_a|]$;
- Choose an IND-DEM-CCA secure data encapsulation scheme $\Pi = (Enc, Dec)$ with symmetric key space $\{0,1\}^l$, a computationally hiding commitment scheme $\Phi = (Commit, Open)$, an EU-CMA secure message authentication code scheme $\Psi = (Mac, MacVrfy)$ and a secure key derivation function $KDF : G_T \to \{0,1\}^l$;

- Output $ps = (p, G, G_T, e, g, g_0, g_1, X, Y, E_1, E_2, h_1, \ldots, h_{|U_a|}, H, \Pi, \Phi, \Psi, KDF)$ and $msk = (\alpha, \beta, \gamma)$.

(2) **ASKGen**(ps, msk, att_{du}): Given ps, $msk = (\alpha, \beta, \gamma)$ and att_{du}, it is executed as follows:

- Randomly select $z \in \mathbb{Z}_p^*$, compute $k_1 = g^{\alpha} g^{\beta z}$ and $k_2 = g^z$;
- For each $x \in att_{du}$, compute $k_0 = g_1^{\beta} g_0^{\gamma \sum H(h_x)}$ and $k_x = h_x^z$;
- Output $ask_{du} = (k_0, k_1, k_2, \{k_x\}_{x \in att_{du}})$.

(3) **DUKeyGen**(ps, ask_{du}): Given $ask_{du} = (k_0, k_1, k_2, \{k_x\}_{x \in att_{du}})$, it is executed as follows:

- Randomly select $\lambda \in \mathbb{Z}_p^*$, compute $K_1 = k_1^{\lambda} = g^{\alpha\lambda} g^{\lambda\beta z}$ and $K_2 = k_2^{\lambda} = g^{\lambda z}$;
- For each $x \in att_{du}$, compute $K_x = k_x^{\lambda} = h_x^{\lambda z}$;
- Output $sk_{du} = (ask_{du}, \lambda)$ and $tf_{du} = (K_1, K_2, \{K_x\}_{x \in att_{du}})$.

(4) **OffEncryption**(ps): Given ps, it is executed as follows:

- For each $j \in [1, n]$ where $n = |U_w|$, randomly select $u_j \in \mathbb{Z}_p^*$, and compute $W_j' = E_2^{-u_j}$;
- Output $ct_{off} = (\{u_j, W_j'\}_{j \in [1,n]})$.

(5) **OnEncryption**$(ps, m, ct_{off}, ws, \Lambda, t)$: Given ps, m, $ct_{off} = (\{u_j, W_j'\}_{j \in [1,n]})$, $ws = \{w_1, w_2, \ldots, w_{n_1}\}$, $\Lambda = (M, \rho)$ and t where $M \in \mathbb{Z}_p^{row \times col}$ is a $row \times col$ matrix and ρ is a function mapping each row of M to an attribute in U_a, it is executed as follows:

- Randomly select $k \in G_T$ and run $KDF(k, l)$ to extract a symmetric key $key \in \{0,1\}^l$, compute $C_M = \Pi.Enc(key, m)$ and $mac = \Psi.Mac(key, C_M)$;
- Run $\Phi.Commit(m \| k)$ to generate (com, dom) and compute $c_{com} = \Pi.Enc(key, dom)$;
- Randomly select $s, v_2, v_3, \ldots, v_{row} \in \mathbb{Z}_p^*$, and set $\tilde{v} = (s, v_2, v_3, \ldots, v_{row})^{\perp}$;
- Compute $C_1 = g^s$ and $C_2 = Y^s$;
- Randomly select $d \in \mathbb{Z}_p^*$, and compute $C_3 = k \cdot E_1^s$ and $C_4 = g^d$;
- For each $i \in [1, row]$, compute $\mu_i = M_i \tilde{v}$ and $C_i = g^{\beta \mu_i} \cdot h_{\rho(i)}^{-d}$, where M_i is the i-th row of M;
- Set $C_V = (C_M, mac, com, C_{com})$ and $C = (C_1, C_2, C_3, C_4, \{C_i\}_{i \in [1,row]})$.
- Run the 0-encoding algorithm to transform t into a set T;
- For each $\tau \in T$, compute $I_\tau = g_1^{-sH(\tau)}$;
- For each $j \in [1, n_1]$, compute $W_j = H(w_j) + u_j - s$;
- Set $I = (T, \{I_\tau\}_{\tau \in T}, \{W_j, W_j'\}_{j \in [1,n_1]})$;
- Set $ct_{on} = (C_V, C, I)$.

(6) **TrapGen**$(ps, thd, ws', sk_{du}, t')$: Given ps, thd, $ws' = \{w_1', w_2', \ldots, w_{n_2}'\}$, $sk_{du} = (k_0, k_1, k_2, \{k_x\}_{x \in att_{du}}, \lambda)$ and t', it is executed as follows:

- Run 1-encoding algorithm to transform t' into a set T';
- Randomly select $\kappa \in \mathbb{Z}_p^*$ and compute $t_{\tau'} = k_0 \cdot g_1^{H(\tau')\kappa}$ for each $\tau \in T'$;
- Compute $t_1 = g^{\lambda}$, $t_2 = g^{\kappa}$, $t_3 = g_0^{\lambda}$, and $t_4 = \lambda\kappa$;
- For each $j \in [1, n_2]$, compute $\tilde{W}_j = H(E_2^{H(w_j')+\lambda})$;
- Set $td = (thd, T', t_1, t_2, t_3, t_4, \{t_\tau\}_{\tau \in T'}, \{\tilde{W}_j\}_{j \in [1,n_2]})$.

(7) **Search&Trans**$(ps, ct_{on}, td, tf_{du})$: Given ps, ct_{on}, td and tf_{du}, it is executed as follows:

- If att_{du} associated with tf_{du} satisfies the access structure in ct_{on} and $T \cap T' \neq \varnothing$, randomly select $y \in T \cap T'$ and compute: $IV = \frac{e(C_1 \cdot t_1, t_y) \cdot e(t_2, I_y) \cdot e(g^{-t_4}, g_1^{H(y)})}{e\left(g^{r \Sigma H(h_x)}, t_3\right) \cdot e\left((C_2)^{\Sigma H(h_x)}, g_0\right)}$, where $x \in att_{du}$;
- Check whether $\left| \left\{ H\left(IV \cdot E_2^{W_j} \cdot W_j \right) \right\}_{j=1}^{n_1} \cap \left\{ \widetilde{W}_j \right\}_{j=1}^{n_2} \right| > thd$;

- If not, output $\perp$; else, for all $i \in N$ where $N = \{i : \rho(i) \in att_{du}\}$, find $\{\omega_i\} \in \mathbb{Z}_p^*$ that satisfy $\sum_{i\in N} \omega_i M_i = (1, 0, 0, \ldots, 0)$, compute $TF = \frac{e(K_1, C_1)}{e\left(\prod_{i\in N} C_i^{\omega_i}, K_2\right) \cdot e\left(\prod_{i\in N} K_{\rho(i)}^{\omega_i}, C_4\right)}$;
- Output $ct_{trans} = (C_{com}, com, C_3, TF)$ and $pct_m = (mac, C_M)$.

(8) **ResultVrfy**(ps, ct_{trans}): Given ps and ct_{trans}, it is executed as follows:

- Before downloading the decryption ciphertext $pct_m = (mac, C_M)$, compute $k = \frac{C_3}{(TF)^{1/\lambda}}, key = KDF(k, l), dom' = \Pi.Dec(key, C_{com})$ and run $\Phi.Open(com, dom')$ to verify whether dom' and com matches.
- If $\Phi.Open$ outputs 1, it indicates successful validation. Then download decryption ciphertext pct_m and output the decryption key $dk = key$. If $\Phi.Open$ outputs 0, output $\perp$.

(9) **Decrypt**(ps, dk, pct_m): Given ps, dk, and $pct_m = (mac, C_M)$, it is executed as follows:

- Compute $m = \Pi.Dec(dk, C_M)$ and $mac' = \Psi.MacVrfy(dk, m, C_M)$;
- If mac is equal to mac', output m; otherwise, output $\perp$.

4.1 Correctness Analysis

According to the scheme description, we can deduce that

$$\begin{aligned} IV &= \frac{e(C_1 \cdot t_1, t_y) \cdot e(t_2, I_y) \cdot e(g^{-t_4}, g_1^{H(y)})}{e\left(g^{\gamma\Sigma H(h_x)}, t_3\right) \cdot e\left((C_2)^{\Sigma H(h_x)}, g_0\right)} \\ &= \frac{e(g^s \cdot g^\lambda, g_1^\beta g_0^{\gamma\Sigma H(h_x)} \cdot g_1^{H(y)\cdot\kappa}) \cdot e\left(g^\kappa, g_1^{-s\cdot H(y)}\right) \cdot e\left(g^{-\lambda\kappa}, g_1^{H(y)}\right)}{e(g^{\gamma\cdot\Sigma H(h_x)}, g_0^\lambda) \cdot e(g^{\gamma s\cdot\Sigma H(h_x)}, g_0)} \\ &= \frac{e(g^{s+\lambda}, g_1^\beta \cdot g_0^{\gamma\Sigma h_x} \cdot g_1^{H(y)\kappa}) \cdot e(g^{\lambda+s}, g_1^{-\kappa\cdot H(y)})}{e(g^{\gamma\cdot\Sigma h_x}, g_0^{\lambda+s})} \\ &= e(g^\beta, g_1)^{s+\lambda} \\ &= E_2^{s+\lambda}. \end{aligned} \tag{1}$$

Clearly, if $|ws \cap ws'| > thd$, then we have

$$\begin{aligned} \left|\left\{H\left(IV \cdot E_2^{W_j} \cdot W_j'\right)\right\}_{j=1}^{n_1} \cap \left\{\widetilde{W}_j\right\}_{j=1}^{n_2}\right| &= \left|\left\{H\left(E_2^{s+\lambda} \cdot E_2^{H(w_j)-s+u_j} \cdot E_2^{-u_j}\right)\right\}_{j=1}^{n_1} \cap \left\{H(E_2^{H(w_j)+\lambda})\right\}_{j=1}^{n_2}\right| \\ &= \left|\left\{H(E_2^{H(w_j)+\lambda})_{j=1}^{n_1} \cap H\left(E_2^{H(\widetilde{w}_j)+\lambda}\right)\right\}_{j=1}^{n_2}\right| \\ &> thd. \end{aligned} \tag{2}$$

Therefore, keyword search works correctly. In addition, we can deduce that

$$\begin{aligned} \frac{C_3}{(TF)^{1/\lambda}} &= \frac{k \cdot E_1^s \cdot e\left(\prod_{i\in N} C_i^{\omega_i}, K_2\right)^{1/\lambda} \cdot e\left(\prod_{i\in N} K_{\rho(i)}^{\omega_i}, C_4\right)^{1/\lambda}}{e(K_1, C_1)^{1/\lambda}} \\ &= \frac{k \cdot E_1^s \cdot e\left(\prod_{i\in N} g^{\beta\mu_i\cdot\omega_i} \cdot h_{\rho(i)}^{-d\cdot\omega_i}, g^z\right) \cdot e\left(\prod_{i\in N} h_{\rho(i)}^{z\cdot\omega_i}, g^d\right)}{e(g^\alpha \cdot g^{\beta z}, g^s)} \\ &= \frac{k \cdot E_1^s \cdot e(g^{\beta s}, g^z) \cdot e\left(\prod_{i\in N} h_{\rho(i)}^{-d\cdot\omega_i}, g^z\right) \cdot e\left(\prod_{i\in N} h_{\rho(i)}^{z\cdot\omega_i}, g^d\right)}{e(g^\alpha \cdot g^{\beta z}, g^s)} \end{aligned}$$

$$= \frac{k \cdot e(g,g)^{\alpha s}}{e(g,g)^{\alpha s}}$$
$$= k. \tag{3}$$

4.2 Security Proofs

We demonstrate that our FCS-ABMSE scheme meets the IND-CKA and the IND-CPA security.

Theorem 1. If a polynomial-time adversary $\mathcal{A}_d$ breaks the IND-CKA security of our scheme with advantage ϵ, there exists an algorithm $\mathcal{B}$ solving the DBDH problem with advantage ϵ.

Proof. Given a random instance of the DBDH problem $(p, G, G_T, e, g, g^a, g^b, g^c, Z)$, $\mathcal{B}$ interacts with $\mathcal{A}$ to determine whether $Z = e(g,g)^{abc}$ as follows:

Initialization. $\mathcal{A}_d$ submits a policy $\Lambda^* = (M^*, \rho^*)$, where M^* is a row^* columns matrix.

Setup. $\mathcal{B}$ first sets $X = g^b$, $g_1 = g^c$, and $E_2 = e(g^b, g^c)$. Then, it sets other parameters $(g_0, Y, E_1, h_1, \ldots, h_{|U|},$

$H, \Pi, \Phi, \Psi, KDF)$ as in the **Setup** algorithm. Finally, it returns $ps = (p, G, G_T, e, g, g_0, g_1,$

$X, Y, E_1, E_2, h_1, \ldots, h_{|U|}, H, \Pi, \Phi, \Psi, KDF)$ to $\mathcal{A}_d$.

Phase 1. $\mathcal{B}$ maintains a key list $\mathcal{L}_{key} = \{\langle att_{du}, ask_{du}, tf_{du}, sk_{du}\rangle\}$ and a trapdoor list $\mathcal{L}_{td} = \{\langle att_{du}, ws', td\rangle\}$, both of which are initially empty. $\mathcal{B}$ answers $\mathcal{A}_d$'s queries as follows:

- $\mathcal{O}_{ask}$: Given an attribute set att_{du}, $\mathcal{B}$ returns ask_{du} if att_{du} already exists in a record $\langle att_{du}, ask_{du}, tf_{du}, sk_{du}\rangle$ on the list $\mathcal{L}_{key}$. Otherwise, $\mathcal{B}$ randomly selects $z \in \mathbb{Z}_p^*$, calculates $k_0 = g_1^b g_0^{y \sum h_x}$, $k_1 = g^\alpha g^{bz}$, $k_2 = g^z$, $k_x = h_x^z$, for all $x \in att_{du}$. Finally, $\mathcal{B}$ returns the attribute key: $ask_{du} = (k_0, k_1, k_2, \{k_x\}_{x \in att_{du}})$ to $\mathcal{A}_d$ and adds $\langle att_{du}, ask_{du}, -, -\rangle$ to the list $\mathcal{L}_{key}$.
- $\mathcal{O}_{tf}$: Given an attribute set att_{du}, $\mathcal{B}$ returns tf_{du} if tf_{du} already exists in a record $\langle att_{du}, ask_{du}, tf_{du}, sk_{du}\rangle$ on the list $\mathcal{L}_{key}$. Otherwise, $\mathcal{B}$ checks the list to see if ask_{du} corresponding to att_{du} exists. If so, $\mathcal{B}$ selects $\lambda \in \mathbb{Z}_p^*$ and calculates: $K_1 = k_1^\lambda$, $K_2 = k_2^\lambda$, $K_x = k_x^\lambda$, $\forall x \in att_{du}$. It generates the conversion key: $tf = (K_1, K_2, \{K_x\}_{x \in att_{du}})$, and the private key: $sk = (ask, \lambda)$. If ask does not exist, $\mathcal{B}$ first performs $\mathcal{O}_{ask_{du}}$ to obtain ask_{du}, and then generates the corresponding tf_{du} and sk according to the above steps. Finally, $\mathcal{B}$ returns the conversion key tf_{du} to $\mathcal{A}_d$ and adds the tuple $\langle att_{du}, ask_{du}, tf_{du}, sk_{du}\rangle$ to the list $\mathcal{L}_{key}$.
- $\mathcal{O}_{sk}$: Given an attribute set att_{du}, $\mathcal{B}$ returns tf_{du} if tf_{du} already exists in a record $\langle att_{du}, ask_{du}, tf_{du}, sk_{du}\rangle$ on the list $\mathcal{L}_{key}$. Otherwise, $\mathcal{B}$ performs $\mathcal{O}_{tf}$ to generate the conversion key: $tf_{du} = (K_1, K_2, \{K_x\}_{x \in att_{du}})$, and the private key:$sk = (ask, \lambda)$. Finally, $\mathcal{B}$ returns sk_{du} to $\mathcal{A}_d$, and adds the tuple $\langle att_{du}, ask_{du}, tf_{du}, sk_{du}\rangle$ to the list $\mathcal{L}_{key}$.
- $\mathcal{O}_{trapdoor}$: Given a threshold value thd, a keyword set $ws' = \{w_1', w_2', \ldots, w_{n_2}'\}$, an attribute set att_{du}, and time t', $\mathcal{B}$ returns td if td already exists in a record $\langle att_{du}, ws', td\rangle$ on the list $\mathcal{L}_{td}$. Otherwise, $\mathcal{B}$ performs $\mathcal{O}_{ask}$ and $\mathcal{O}_{sk}$ to obtain attribute keys ask_{du} and private keys sk. $\mathcal{B}$ runs the 1-encoding algorithm to transform the time t' into a set T'. For each $\tau' \in T'$, $\mathcal{B}$ randomly chooses $\kappa \in \mathbb{Z}_p^*$, and computes $t_{\tau'} = k_0 \cdot g_1^{H(\tau')\kappa}$. $\mathcal{B}$ computes $t_1 = g^\lambda$, $t_2 = g^\kappa$, $t_3 = g_0^\lambda$, $t_4 = \lambda\kappa$. For each $j = 1, \ldots, n_2$, it compute:$W_j' = H(E_2^{H(w_j')+\lambda})$. Finally, it outputs the trapdoor $td = (T', t_1, t_2, t_3, t_4, \{t_{\tau'}\}_{\tau' \in T'}, thd, \{W_j'\}_1^{n_2})$ to $\mathcal{A}_d$ and adds td to the list $\mathcal{L}_{td}$.

Challenge. $\mathcal{A}_d$ outputs two keyword sets with the same number of keywords $ws_0 = \{w_{0,1}, \ldots, w_{0,n_1}\}$ and $ws_1 = \{w_{1,1}, \ldots, w_{1,n_1}\}$, a message m^*, an access control policy $\Lambda^* = (M^*, \rho^*)$, and a time t^*, where M^*

is a $col^* \times row^*$ matrix, and ρ^* is a function that maps the rows of M^* to attributes. $\mathcal{B}$ randomly picks a coin $v \in \{0,1\}$ and encrypts the keyword set ws_v. The following steps are executed:

- For each $j = 1, \ldots, n_1$, $\mathcal{B}$ randomly selects $W_j \in \mathbb{Z}_p^*$. Assume that for an unknown u_j, $W_j = H(w_j) + a + u_j$, and calculates $W_j' = E_2^{W_j} E_2^{-H(w_j)} Z$.
- $\mathcal{B}$ runs the 0-encoding algorithm to transform the time t^* into a set T^*. For each $\tau \in T^*$, it computes $I_\tau^* = e(g^\lambda, g^\kappa \cdot H(\tau))$.
- $\mathcal{B}$ sets the keyword ciphertext: $I^* = (T^*, \{I_\tau^*\}_{\tau \in T^*}, \{I_j\}_{j=1}^{n_1})$.
- $\mathcal{B}$ sets the other ciphertext according to the **OnEncryption** algorithm.

Phase 2. This phase is identical to Phase 1. att_{du} queried by $\mathcal{A}_d$ cannot meet Λ^*.

Guess. $\mathcal{A}_d$ outputs a guess v' for v. If $v = v'$, $\mathcal{B}$ outputs 1, which means that $Z = e(g,g)^{abc}$. Otherwise, $\mathcal{B}$ outputs 0, which means that $Z \neq e(g,g)^{abc}$.

If $Z = e(g,g)^{abc}$, this simulation is effective for the adversary, and $Adv_{\mathcal{A}_d}^{IND-CKA} = 1/2 + \epsilon$. If Z is a random element, $Adv_{\mathcal{A}_d}^{IND-CKA} = 1/2$. Thus, the advantage of $\mathcal{B}$ in breaking the DBDH problem is $Adv_{\mathcal{B}} = |\Pr[\mathcal{A}_d(p, G, G_T, e, g, g^a, g^b, g^c, e(g,g)^{abc}) = 1] - \Pr[\mathcal{A}_d(p, G, G_T, e, g, g^a, g^b, g^c, Z) = 1]| = |1/2 + \varepsilon - 1/2| = \varepsilon$.□

Theorem 2. Assuming that the q-parallel-BDHE problem is difficult, the FCS-ABMSE meets the IND-CPA security. If a polynomial-time adversary $\mathcal{A}_d$ breaks the IND-CPA security of our scheme with advantage ε, there exists an algorithm $\mathcal{B}$ solving the q-parallel-BDHE problem with advantage ε.

Proof. Given a random instance of the decisional q-parallel-BDHE problem $(p, G, G_T, e, g, \tilde{y}, T)$, $\mathcal{B}$ interacts with $\mathcal{A}$ to determine whether $T = e(g,g)^{\alpha^{q+1}s}$ as follows:

Initialization. $\mathcal{A}_d$ outputs a policy $\Lambda^* = (M^*, \rho^*)$, where M^* is a matrix with row^* columns.

Setup. $\mathcal{B}$ randomly selects $g, g_0, g_1 \in G$ and $\alpha', \gamma \in \mathbb{Z}_p^*$, sets $X = g^\beta$, $Y = g^\gamma$, $E_2 = e(X, g_1)$, and calculates $E_1 = e(g,g)^{\alpha'} = e(g,g)^\alpha$, where $\alpha = \alpha' + \beta^{q+1}$ is implicitly given. For $1 \le x \le U_a$, it randomly selects a number $z_x \in \mathbb{Z}_p^*$. Let set_x represent the set of i that satisfies $\rho^*(i) = x$. $\mathcal{B}$ calculates h_x corresponding to attribute x as: $h_x = g^{z_x} \cdot \prod_{i \in set_x} g^{\beta M_{i,1}^*/b_i} g^{\beta^2 M_{i,2}^*/b_i} \cdots g^{\beta^{n^*} M_{i,n}^*/b_i}$. $\mathcal{B}$ selects a hash function $H: \{0,1\}^* \to \mathbb{Z}_p^*$. $\mathcal{B}$ chooses a symmetric encryption scheme $\Pi = (Enc, Dec)$, a commitment scheme $\Phi = (Commit, Open)$, a message authentication code scheme $\Psi = (Mac, MacVrfy)$, and a key export function $KDF: G_T \to \kappa$, where κ is the symmetric key space. Finally, it outputs the global parameter set: $ps = (p, G, G_T, e, g, g_0, g_1, h_1, \ldots, h_U, X, Y, E_1, E_2, H, \Pi, \Phi, \Psi, KDF)$ to $\mathcal{A}_d$.

Phase 1. In order to answer the queries in this phase, $\mathcal{B}$ maintains a key list: $\mathcal{L}_{key} = \{\langle att_{du}, ask_{du}, tf_{du}, sk_{du}\rangle\}$. $\mathcal{A}_d$ adaptively makes the following queries with the restriction that att_{du} cannot meet Λ^*.

- $\mathcal{O}_{ask}$: Given an attribute set att_{du}, $\mathcal{B}$ returns ask_{du} if att_{du} already exists in a record $\langle att_{du}, ask_{du}, tf_{du}, sk_{du}\rangle$ on the list $\mathcal{L}_{key}$. Otherwise, $\mathcal{B}$ randomly selects $r \in \mathbb{Z}_p^*$. Next, $\mathcal{B}$ chooses $\vec{w} = (w_1, w_2, \ldots, w_{n^*})$, where $w_1 = 1$. For all i where $\mu^*(i) = x$ satisfies $\vec{w} M_i^* = 0$, $\mathcal{B}$ sets: $z = r + w_1 a^q + w_2 a^{q-1} + \cdots + w_{n^*} a^{q-n^*+1}$, and calculates $k_0 = g_1^\beta g_0^{\gamma \sum h_x}$, $k_1 = g^{\alpha'} \cdot g^{\beta r} \prod_{i \in [n^*]} (g^{\beta^{q+2-i}})^{w_i} = g^\alpha g^{\beta z}$, $k_2 = g^r \cdot \prod_{i \in [n^*]} \left(g^{\beta^{q+1-i}}\right)^{w_i} = g^z$. For any attribute x in set att_{du}, if there is no i that satisfies $\rho^*(i) = x$, $\mathcal{B}$ sets $h_x = g^{zx}$. And computes $k_x = h_x^z = k_2^{zx}$ Otherwise, let X indicate the set of i that satisfies $\rho^*(i) = x$, and $\mathcal{B}$ calculates $k_x = L^{z_x} \cdot \prod_{i \in X} \prod_{j \in [n^*]} (g^{(\beta^j/b_i)^r} \cdot \prod_{k \in [n^*], k \neq j} (g^{\beta^{q+1+j-k}/b_i})^{w_k}) = h_x^t$ as follows: $k_x = g^z \cdot \prod_{i \in X} g^{\beta^i}$. Finally, $\mathcal{B}$ returns the attribute key $ask = (k_0, k_1, k_2, k_x)$ to $\mathcal{A}_d$ and adds $\langle att_{du}, ask_{du}, -, -\rangle$ to the list $\mathcal{L}$.
- $\mathcal{O}_{tf}$: Identical to that in IND-CKA.

- $\mathcal{O}_{sk}$: Identical to that in IND-CKA.

Challenge. $\mathcal{A}_d$ outputs two messages m_0, m_1, a keyword set ws^*, and a time t^*. $\mathcal{B}$ randomly picks a coin $v \in \{0,1\}$. $\mathcal{B}$ computes $C_3^* = k \cdot T \cdot e(g^s, g^{\alpha'})$, selects $d \in \mathbb{Z}_p^*$ and for $i \in [1, row^*]$ computes $g^{r_i} = g^{-d} g^{sb_i}$, where $d = -r_i + sb_i$ is implicitly given and $C_4^* = g^{-r_i + sb_i} = g^d$. $\mathcal{B}$ defines the set R to represent the set of k that satisfies $\rho^*(i) = \rho^*(k)$ and $k \neq i$. $\mathcal{B}$ chooses $(y'_{2*}, \ldots, y'_{n*}) \in \mathbb{Z}_p^*$ and calculates $C_i^* = h_{\rho^*(i)}^{r_i} \cdot (\prod_{j=2,\ldots,n} g^{\beta M_{i,j}^* \cdot y_j})(g^{b_i \cdot s})^{-z_{\rho^*(i)}} \cdot \prod_{k \in R} \prod_{j \in [n]} (g^{s \cdot \beta^j \cdot M_{k,j}^* (b_i/b_k)})^{-1}$, where $\vec{v} = (s,\ s\beta + y'_2,\ s\beta^2 + y'_3,\ \ldots,\ s\beta^{n^*-1} + y'_{n^*}) \in \mathbb{Z}_p^{n^*}$ is implicitly given. $\mathcal{B}$ sets the other ciphertext according to the **OnEncryption** algorithm.

Phase 2. Identical to Phase 1. The queried att_{du} by $\mathcal{A}_d$ cannot meet Λ^*.

Guess. $\mathcal{A}_d$ outputs a guess v' for v. If $v = v'$, $\mathcal{B}$ outputs 1, which means $T = e(g,g)^{\alpha^{q+1_s}}$. Otherwise, $\mathcal{B}$ outputs 0, which means $T \neq e(g,g)^{\alpha^{q+1_s}}$.

If $T = e(g,g)^{\alpha^{q+1_s}}$, this simulation is effective for the adversary, and $Adv_{\mathcal{A}_d}^{IND-CPA} = 1/2 + \varepsilon$. If T is a random element, $Adv_{\mathcal{A}_d}^{IND-CPA} = 1/2$. Thus, the advantage of $\mathcal{B}$ in breaking the DBDH problem is $Adv_{\mathcal{B}} = |\Pr[\mathcal{A}_d(p, G, G_T, e, g, g^a, g^b, g^c, e(g,g)^{\alpha^{q+1_s}}) = 1] - \Pr[\mathcal{A}_d(p, G, G_T, e, g, g^a, g^b, g^c, T) = 1]| = |1/2 + \varepsilon - 1/2| = \varepsilon$. □

5 Evaluation and Discussions

This section compares the proposed scheme with four benchmark approaches [36,37,40], and [41] for IoT environments in terms of functionalities, computational cost, and communication cost.

The Java Pairing-Based Cryptography Library (JPBC) was adopted [44] for simulations. The experiment was conducted using the JPBC-2.0.0 library on a system equipped with a Core i5-13500HX processor at 2.50 GHz and 16 GB of RAM. The bilinear map follows a type-1 pairings structure, with the curve equation $y^2 = x^3 + x$ and a 160-bit prime order p. The lengths of an element on Groups G and G_T are 512 and 1024 bits, respectively. The symbols are listed in Table 2:

Table 2: Symbols and definitions

Symbols	Definitions	Symbols	Definitions
T_p	Time cost of a bilinear pair	l	Number of attributes in LSSS
T_e	Time cost of an exponent on group G	$\|S\|$	Number of user attributes
T_z	Time cost of an exponent on group G_T	n_1	Number of keywords in encryption
T_{mp}	Time cost of a Map-to-Point hash function	n_2	Number of keywords in trapdoor
T_h	Operation time of a general hash function	$\|G\|$	Element length of group G
T_m	Time cost of a multiplication on group G	$\|G_T\|$	Element length of group G_T
T_c	Time cost of a multiplication on group G_T	$\|Z_p^*\|$	Element length pf group $\|Z_p^*\|$

5.1 Functional Comparison

Table 3 compares this scheme with VMKS [36], VFKSM [37], FE-ABSE-RV [40], AB-HDPS [41]. VMKS, VFKSM, FE-ABSE-RV can only support multi-keyword search while AB-HDPS can support multi-keyword search and forward ciphertext search. Only the proposed scheme can support two additional functions including multi-keyword fuzzy matching, enabling tolerance to minor input errors, and expressive access control policies, which allow for fine-grained and flexible authorization beyond the capabilities of prior schemes.

Table 3: Comparison of Functions in different schemes

Scheme	Multi-keyword search	Multi-keyword fuzzy matching	Forward ciphertext search	Expressive access control policy
VMKS [36]	✓	×	×	×
VFKSM [37]	✓	×	×	×
FE-ABSE-RV [40]	✓	×	×	×
AB-HDPS [41]	✓	×	✓	×
Ours	✓	✓	✓	✓

Current fuzzy keyword search schemes mainly adopt two architectural approaches: (1) edit distance; (2) LSH. The former approach primarily utilizes edit distance to construct a collection of fuzzy keywords approximating original keywords. The latter employs the LSH algorithm to generate a bloom filter for keyword sets, subsequently encrypting the filter into index structures. Both approaches operate by building indexes and resembling original keywords to enable fuzzy search. In contrast, our scheme's core innovation lies in threshold-based intersection search, which achieves multi-keyword fuzzy search by comparing intersections between distinct keyword sets.

Traditional fuzzy keyword search is designed to tolerate typos and spelling mistakes in user queries. It typically performs single-keyword search, determining whether a potentially incorrect search term belongs to a predefined set of fuzzy variants of the specific keyword. In contrast, our scheme is designed to evaluate whether a ciphertext contains a partial or complete match to a set of keywords being searched for. It supports multi-keyword fuzzy matching by assessing the similarity between two distinct keyword sets—one encrypted within the ciphertext and the other encoded in the trapdoor based on the size of their intersection. As we know, the search efficiency of single-keyword search is significantly lower than that of multi-keyword search. To conduct a fair comparison, we only include the ABSE schemes that support multi-keyword search in the performance comparison.

5.2 Computational Cost

The experiment obtains the system time generated timestamp with a fixed length of 40 bits. For 0-1 encoding, its computational efficiency is related to the numbers of 0 and 1 characters in the bit string. We randomly generated eight bit strings with a total length of 40 bits. Different numbers of 0/1 bits are used to test the computational efficiency of 0 encoding and 1 encoding. As shown in Table 4, it can be seen that 0-1 encoding is highly efficient. When the number of "0" or "1" is 40, the time for all 0 bit strings and all 1 bit strings is 11.08 and 9.603 μs, respectively. Hence, the time consumption for the operation of generating a time set using 0-1 encoding in this scheme is negligible for IoT applications with constrained resources.

Table 4: Time for varying numbers of "0" or "1" in a string

Type	5 bits	10 bits	15 bits	20 bits	25 bits	30 bits	35 bits	40 bits
0	3.986 μs	5.546 μs	6.173 μs	7.476 μs	8.761 μs	9.465 μs	10.79 μs	11.08 μs
1	4.478 μs	4.625 μs	5.558 μs	6.616 μs	7.694 μs	8.824 μs	9.202 μs	9.603 μs

The impact of time sets in ciphertext generation and trapdoor generation on computation efficiency was investigated. Due to the use of 0-1 encoding in this scheme to generate a time set, elements in the set are

then subjected to exponential operation on group G. Therefore, as the number of elements in the time set grows, the corresponding ciphertext and trapdoor time costs also increase. Table 5 shows the time required to generate ciphertext using 0-encoding and the time required to generate trapdoors using 1-encoding technology. Similarly, in order to demonstrate the authenticity of the experiment, the total length of the string is 40. From Table 5, it can be observed that the time consumed by all 0 and 1 bit strings is 115.2 and 482.8 μs, respectively, which are relatively small and acceptable for cloud-based IoT applications.

Table 5: Time for different phases of 0-1 encoding

Phase	5 bits	10 bits	15 bits	20 bits	25 bits	30 bits	35 bits	40 bits
Encryption	24.7 μs	39.1 μs	51.9 μs	63.5 μs	74.8 μs	88.4 μs	101.1 μs	115.2 μs
Trapdoor	39.3 μs	77.4 μs	118.2 μs	245.9 μs	305.4 μs	369.3 μs	427.1 μs	482.8 μs

Table 6 summarizes computational costs associated with key generation, encryption, trapdoor generation, search, and decryption operations. The computational cost of an algorithm is calculated as the sum of the time costs of all related operations. As the symmetric encryption, commitment methods, mac functions, and other mature methods in the encryption process require very small time, they are omitted from the analysis. In addition, these schemes only require an exponential operation of Group G and decryption calculation in the symmetric encryption algorithm during the user decryption process. Therefore, we will not elaborate on them here. Finally, although the owner performs offline encryption, this operation only needs to be calculated once in this scheme, [36], and [40]. Hence, the computational cost of this operation is not discussed.

Table 6: Comparison of key generation, encryption, trapdoor, search, and decryption operations

Scheme	Key generation	Encryption	Trapdoor	Search	Decryption
VMKS [36]	$(2+2\|S\|+\|U\|)T_e+T_z+(1+3l)T_m+T_c$	$(1+2l)T_e+T_z+(1+3l)T_m+T_c$	$(4+\|S\|)T_e+n_2T_h+n_2T_z+\|S\|T_m$	$8T_p+(1+l)T_c+(2+2l)T_m$	$2T_z+T_c+T_h$
VFKSM [37]	$(7+\|S\|)T_e+2T_{mp}+T_m$	$(2n_1+2+2l)T_e+T_z+T_p+n_1(T_h+T_m)$	$3T_e+T_m+n_2T_h$	$(4+2l)T_e+3T_p+(1+2l)T_m$	$2T_c+T_p$
FE-ABSE-RV [40]	$(5+2\|S\|)T_e$	$(3+n_1+l)T_e+lT_p+T_c+lT_m$	$(n_2+2)T_e+\|S\|T_p$	$(2l+5)T_p+lT_e$	$T_e+T_z+T_p$
AB-HDPS [41]	$(5\|S\|+2)T_e+(4\|S\|-2)T_m$	$(5l+6)T_e+T_z+lT_m+(n_1+1)T_h$	$4T_e+3T_m+(n_2+1)T_h$	$3T_p+T_c$	$T_e+T_c+T_p$
Ours	$(5+\|S\|)T_e$	$(3+2l+\|T\|)T_e+T_z+T_c$	$(3+T')T_e+n_2(T_z+T_h)+T'T_m$	$5T_p+3T_e+n_1T_z+3T_m+T_h$	T_z+T_c

Fig. 2 illustrates the computational cost in key generation for each scheme. The computation time of our scheme in key generation is the lowest. Taking the computational cost of a set of 6 attributes as an example, the key generation algorithm in our scheme takes about 13.86 ms, while that in schemes [36,37,40,41] is about 18.9, 15.918, 21.42, and 40.45 ms, respectively. As shown in Fig. 3, the computational cost of our scheme in trapdoor is higher than that of [37] and [41] due to its support for multi-keyword fuzzy search functions. Compared to [36] and [40], our computational cost is lower. For a search with a set of 6 keywords, the trapdoor algorithm in our scheme takes about 19.81 ms, while that in schemes [36,37,40,41] is about 23.58, 12.24, 25.52, and 14.91 ms, respectively.

The computational cost of the encryption algorithm is affected by both the number of attributes and the number of keywords. Fig. 4 illustrates the relationship between computational cost and the number of attributes on encryption time. The number of keywords is set to two. The encryption time increases with the number of attributes. For a set of six attributes, the encryption algorithm in our scheme takes about 18.82 ms, while that in schemes [36,37,40,41] is about 19.54, 39.02, 60.36, and 62.19 ms, respectively. When the number of attributes is large enough, the encryption time in [40] becomes the longest. Fig. 5 illustrates the relationship between computational cost and the number of keywords on encryption time. The number of attributes is two. As our scheme and [36] can achieve outsourced assisted keyword encryption, the computational cost is independent of the number of keywords. Due to the implementation of multi-keyword fuzzy search, the cost of our scheme is higher than that of [36]. For six attributes, the encryption time in our scheme is about 10.08 ms, while that in [36,37,40,41] is about 7.56, 31.66, 29.3, and 31.35 ms, respectively. As our scheme and [36] implement outsourced keyword encryption, both schemes can achieve computational cost invariance regardless of the number of keywords.

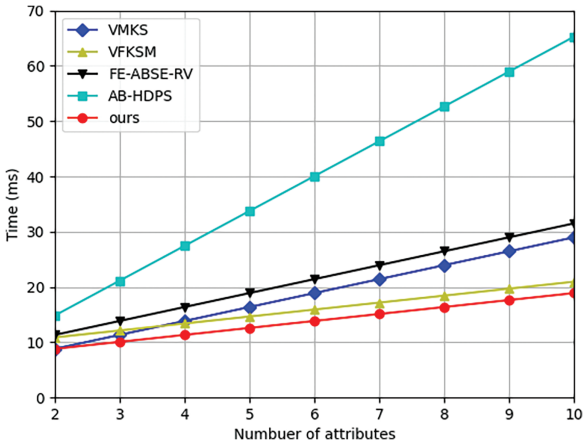

Figure 2: Computational cost of key generation

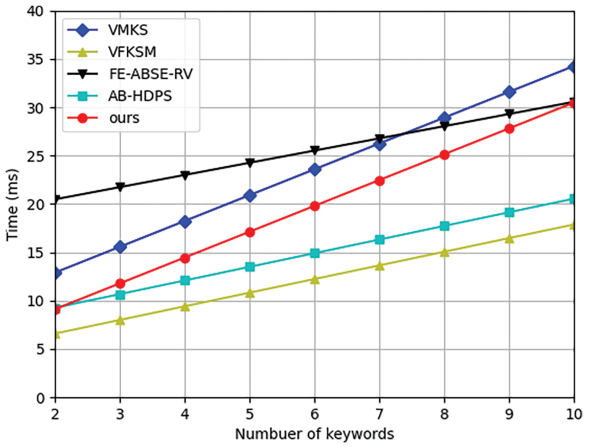

Figure 3: Computational cost of trapdoor

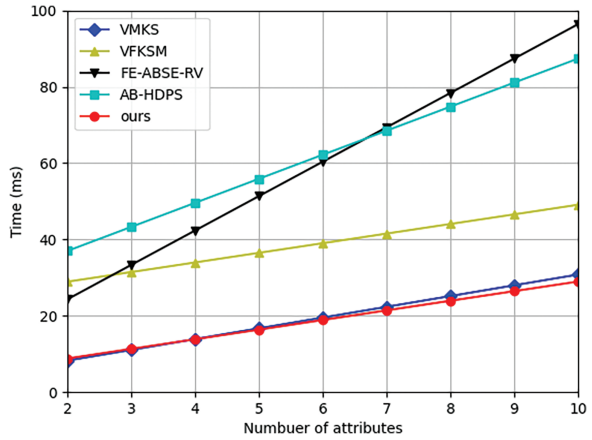

Figure 4: Computational cost of encryption vs. number of attributes

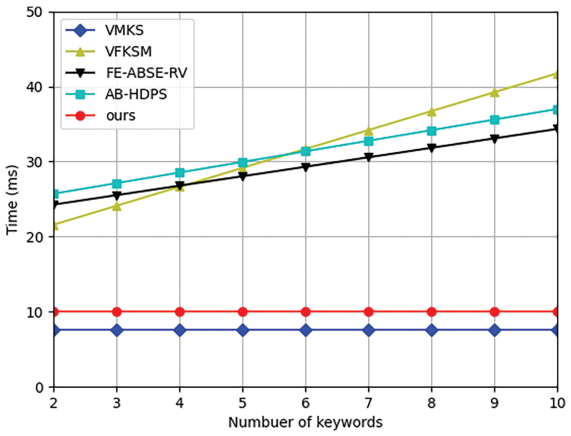

Figure 5: Computational cost of encryption vs. number of keywords

Fig. 6 presents the computational cost of the search operation. Besides our scheme, the computational cost of other schemes is independent of the number of keywords. Due to the support of multi-keyword fuzzy search, the cost of our scheme is higher than that in [37] and [41]. However, this workload is computed by cloud servers, and this additional computational cost is acceptable in practical deployment scenarios. For six keywords, the search algorithm in our scheme takes about 51.37 ms, while that in schemes [36,37,40,41] is about 61.76, 133.24, 72, and 23.16 ms.

Fig. 7 compares the computational cost of decryption of each scheme. All schemes implement outsourced decryption and our scheme performs best. The decryption algorithm in our scheme takes about 1.26 ms, while that in schemes [36,37,40,41] is about 3.93, 7.72, 10.24, and 8.98 ms, respectively.

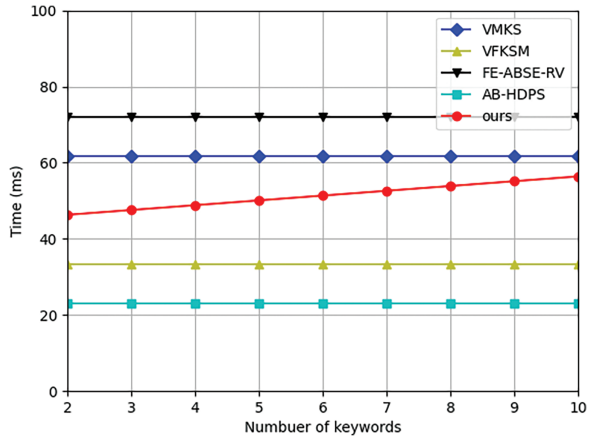

Figure 6: Computational cost of search

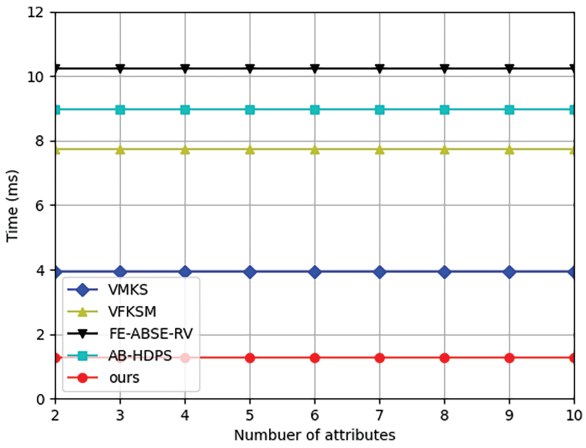

Figure 7: Computational cost of decryption

5.3 Communication Cost

Due to the incorporation of a forward security function in our scheme, the communication cost of ciphertexts and trapdoors is affected by the number of elements in the time set. The validity period of the trapdoor is related to the value of parameter T. The larger the value of T, the longer the validity period.

In the experiment, T is set to be 5. The proposed scheme demonstrates superior overall communication efficiency compared to existing alternatives. As we can see from Table 7, our scheme achieves lower key management communication cost compared to the others, while incurring higher ciphertext cost. The trapdoor communication cost of our scheme is slightly higher than that of [41].

Table 7: Comparison of communication cost

Scheme	Key	Ciphertext	Trapdoor
VMKS [36]	$(2+\|U\|+\|S\|)\|G\|+\|G_T\|$	$(1+2l)\|G\|+(2+n_1)\|G_T\|+n_1\|\mathbb{Z}_p^*\|$	$(4+\|S\|)\|G\|+(n_2+1)\|\mathbb{Z}_p^*\|$
VFKSM [37]	$(6+\|S\|)\|G\|+3\|\mathbb{Z}_p^*\|$	$(4+\|S\|+n_1)\|G\|+2\|G_T\|$	$(2\|G\|+\|\mathbb{Z}_p^*\|)n_2$
FE-ABSE-RV [40]	$(4+2\|S\|)\|G\|+\|\mathbb{Z}_p^*\|$	$(5+2l+n_1)\|G\|+\|S\|\|G_T\|$	$4\|G\|+\|S\|\|G_T\|$
AB-HDPS [41]	$(\|S\|+4)\|G\|+2\|\mathbb{Z}_p^*\|$	$(3l+5)\|G\|$	$2\|G\|$
Ours	$(3+\|S\|)\|G\|$	$(5+l+\|T\|)\|G\|+(1+n_1)\|G_T\|+n_1\|\mathbb{Z}_p^*\|$	$(3+\|T'\|)\|G\|+(1+n_2)\|\mathbb{Z}_p^*\|$

To better compare the communication cost among different schemes, Fig. 8 compares the communication costs between each scheme in the key, ciphertext, and trapdoor. We set the number of attributes and keywords to be ten and remove the communication costs generated by the forward security function of this scheme. Our scheme achieves the lowest communication cost during key generation, with trapdoor transmission costs only higher than [41]. Although the ciphertext storage cost is relatively higher, this overhead is deemed acceptable given that ciphertexts are stored on cloud servers with substantial storage capacity.

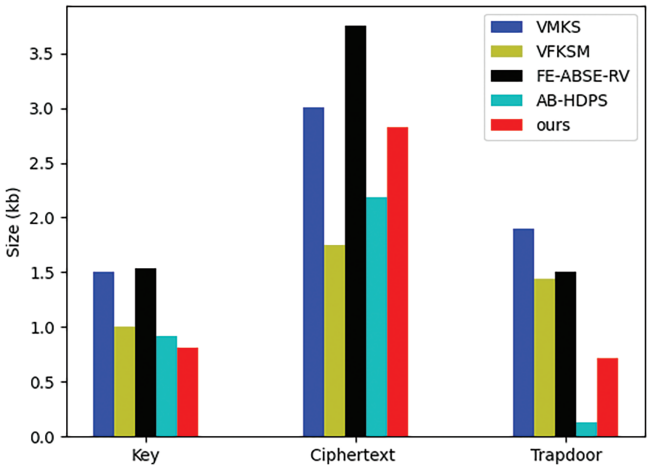

Figure 8: Comparison of communication cost

6 Enhancement of the FCS-ABMSE Scheme with Type-3 Pairings

This section illustrates the enhanced scheme based on type-3 pairings. Type-3 elliptic curves offer superior computational performance and security. It can be used to enhance the FCS-ABMSE scheme by adjusting certain parameters. It is worth noting that the core contribution of multi-keyword fuzzy search and forward ciphertext search remains unaffected by this algorithmic-level adjustment.

(1) **Setup**(η, U_a, U_w): Given η, U_a and U_w, it is executed as follows:

- Generate three prime p-order groups (G_1, G_2, G_T) and $e : G_1 \times G_2 \to G_T$;

- Randomly choose $g_1 \in G_1, g_0, g_2 \in G_2$, and $\alpha, \beta, \gamma \in \mathbb{Z}_p^*$, compute $X = g_1^{\beta}$, $Y = g_1^{\gamma}$, $E_1 = e(g_1^{\alpha}, g_2)$, $E_2 = e(X, g_2)$, and choose a hash function $H : \{0,1\}^* \to \mathbb{Z}_p^*$;
- For each attribute in U_a, select a random element $h_i \in G_2$ where $i \in [1, |U_a|]$;
- Choose an IND-DEM-CCA secure data encapsulation scheme $\Pi = (Enc, Dec)$ with symmetric key space $\{0,1\}^l$, a computationally hiding commitment scheme $\Phi = (Commit, Open)$, an EU-CMA secure message authentication code scheme $\Psi = (Mac, MacVrfy)$, and a secure key derivation function $KDF : G_T \to \{0,1\}^l$;
- Output $ps = (p, G_1, G_2, G_T, e, g_0, g_1, g_2, X, Y, E_1, E_2, h_1, \ldots, h_{|U_a|}, H, \Pi, \Phi, \Psi, KDF)$ and $msk = (\alpha, \beta, \gamma)$.

(2) **ASKGen**(ps, msk, att_{du}): Given ps, $msk = (\alpha, \beta, \gamma)$, and att_{du}, it is executed as follows:

- Randomly select $z \in \mathbb{Z}_p^*$, compute $k_1 = g_2^{\alpha} g_2^{\beta z}$ and $k_2 = g_1^z$;
- For each $x \in att_{du}$, compute $k_0 = g_2^{\beta} g_0^{\gamma \sum H(h_x)}$ and $k_x = h_x^z$;
- Output $ask_{du} = (k_0, k_1, k_2, \{k_x\}_{x \in att_{du}})$.

(3) **DUKeyGen**(ps, ask_{du}): Given $ask_{du} = (k_0, k_1, k_2, \{k_x\}_{x \in att_{du}})$, it is executed as follows:

- Randomly select $\lambda \in \mathbb{Z}_p^*$, compute $K_1 = k_1^{\lambda} = g_2^{\alpha\lambda} g_2^{\lambda\beta z}$ and $K_2 = k_2^{\lambda} = g_1^{\lambda z}$;
- For each $x \in att_{du}$, compute $K_x = k_x^{\lambda} = h_x^{\lambda z}$;
- Output $sk_{du} = (ask_{du}, \lambda)$ and $tf_{du} = (K_1, K_2, \{K_x\}_{x \in att_{du}})$.

(4) **OffEncryption**(ps): Same as that in the original scheme.

(5) **OnEncryption**$(ps, m, ct_{off}, ws, \Lambda, t)$: Given ps, m, $ct_{off} = (\{u_j, W_j'\}_{j \in [1,n]})$, $ws = \{w_1, w_2, \ldots, w_{n_1}\}$, $\Lambda = (M, \rho)$, and t where $M \in \mathbb{Z}_p^{row \times col}$ is a $row \times col$ matrix and ρ is a function mapping each row of M to an attribute in U_a, it is executed as follows:

- Randomly select $k \in G_T$ and run $KDF(k, l)$ to extract a symmetric key $key \in \{0,1\}^l$, compute $C_M = \Pi.Enc(key, m)$, and $mac = \Psi.Mac(key, C_M)$;
- Run $\Phi.Commit(m \| k)$ to generate (com, dom) and compute $c_{com} = \Pi.Enc(key, dom)$;
- Randomly select $s, v_2, v_3, \ldots, v_{row} \in \mathbb{Z}_p^*$ and set $\tilde{v} = (s, v_2, v_3, \ldots, v_{row})^{\perp}$;
- Compute $C_1 = g_1^s$ and $C_2 = Y^s$;
- Randomly select $d \in \mathbb{Z}_p^*$ and compute $C_3 = k \cdot E_1^s$ and $C_4 = g_1^d$;
- For each $i \in [1, row]$, compute $\mu_i = M_i \tilde{v}$ and $C_i = g_2^{\beta\mu_i} \cdot h_{\rho(i)}^{-d}$, where M_i is the i-th row of M;
- Set $C_V = (C_M, mac, com, C_{com})$ and $C = (C_1, C_2, C_3, C_4, \{C_i\}_{i \in [1,row]})$.
- Run the 0-encoding algorithm to transform t into a set T;
- For each $\tau \in T$, compute $I_\tau = g_2^{-sH(\tau)}$;
- For each $j \in [1, n_1]$, compute $W_j = H(w_j) + u_j - s$;
- Set $I = (T, \{I_\tau\}_{\tau \in T}, \{W_j, W_j'\}_{j \in [1,n_1]})$;
- Set $ct_{on} = (C_V, C, I)$.

(6) **TrapGen**$(ps, thd, ws', sk_{du}, t')$: Given ps, thd, $ws' = \{w_1', w_2', \ldots, w_{n_2}'\}$, $sk_{du} = (k_0, k_1, k_2, \{k_x\}_{x \in att_{du}}, \lambda)$ and t', it is executed as follows:

- Run 1-encoding algorithm to transform t' into a set T';
- Randomly select $\kappa \in \mathbb{Z}_p^*$ and compute $t_{\tau'} = k_0 \cdot g_2^{H(\tau')\kappa}$ for each $\tau \in T'$;
- Compute $t_1 = g_1^{\lambda}$, $t_2 = g_1^{\kappa}$, $t_3 = g_0^{\lambda}$, and $t_4 = \lambda\kappa$;
- For each $j \in [1, n_2]$, compute $\tilde{W}_j = H(E_2^{H(w_j')+\lambda})$;
- Set $td = (thd, T', t_1, t_2, t_3, t_4, \{t_\tau\}_{\tau \in T'}, \{\tilde{W}_j\}_{j \in [1,n_2]})$.

(7) **Search&Trans**$(ps, ct_{on}, td, tf_{du})$: Given ps, ct_{on}, td and tf_{du}, it is executed as follows:

- If att_{du} associated with tf_{du} satisfies the access structure in ct_{on} and $T \cap T' \neq \varnothing$, randomly select $y \in T \cap T'$ and compute: $IV = \frac{e(C_1 \cdot t_1, t_y) \cdot e(t_2, I_y) \cdot e(g_1^{-t_4}, g_2^{H(y)})}{e\left(g_1^{r\Sigma H(h_x)}, t_3\right) \cdot e\left((C_2)^{\Sigma H(h_x)}, g_0\right)}$, where $x \in att_{du}$;
- Check whether $\left| \left\{ H\left(IV \cdot E_2^{W_j} \cdot W_j\right) \right\}_{j=1}^{n_1} \cap \left\{ \widetilde{W}_j \right\}_{j=1}^{n_2} \right| > thd$;
- If not, output $\perp$; else, for all $i \in N$ where $N = \{i : \rho(i) \in att_{du}\}$, find $\{\omega_i\} \in \mathbb{Z}_p^*$ that satisfy $\sum_{i \in N} \omega_i M_i = (1, 0, 0, \ldots, 0)$, compute $TF = \frac{e(C_1, K_1)}{e\left(K_2, \prod_{i \in N} C_i^{\omega_i}\right) \cdot e\left(C_4, \prod_{i \in N} K_{\rho(i)}^{\omega_i}\right)}$;
- Output $ct_{trans} = (C_{com}, com, C_3, TF)$ and $pct_m = (mac, C_M)$.

(8) **ResultVrfy**(ps, ct_{trans}): Same as that in the original scheme.

(9) **Decrypt**(ps, dk, pct_m): Same as that in the original scheme.

7 Conclusion

In this paper, we proposed an FCS-ABMSE scheme and an improved construction based on type-3 elliptic curves. The scheme supports multi-keyword fuzzy search without relying on predefined keyword fields, thereby overcoming the lack of error tolerance in exact matching. It also incorporates forward ciphertext search to mitigate trapdoor reuse vulnerabilities in IoT environments. Furthermore, the scheme enables verifiable outsourced decryption, which eliminates bilinear pairing operations on the client side and reduces computational overhead. This deign achieves millisecond-level decryption of search results on user devices while allowing verification of outsourced decryption correctness. However, the incorporation of multi-keyword fuzzy search and forward ciphertext search introduces a slight efficiency loss in encryption and trapdoor generation, and search verification incurs substantial bilinear pairing costs. Rigorous security analysis proved that the FCS-ABMSE scheme meets both IND-CKA and IND-CPA. Additionally, the workload of the cloud server remains high and access policies lack flexibility. Future works will focus on designing more efficient and flexible ABMSE schemes with expressive access policy and reduced server-side burden.

Acknowledgement: Not applicable.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: The authors confirm contribution to the paper as follows: conceptualization, He Duan and Shi Zhang; methodology, He Duan; validation, He Duan and Shi Zhang; formal analysis, He Duan; simulations, He Duan; resources, He Duan and Dayu Li; writing—original draft preparation, He Duan, Shi Zhang, and Dayu Li; writing—review and editing, He Duan; visualization, He Duan; supervision, Shi Zhang and Dayu Li; project administration, He Duan. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Not applicable.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Shwetha D, Swetha. IoT's impact on personal devices and health: wearable technology. In: 2024 International Conference on Advances in Modern Age Technologies for Health and Engineering Science (AMATHE); 2024 May 16–17; Shivamogga, India. p. 1–8.

2. Bisht RS, Jain S, Tewari N. Study of wearable IoT devices in 2021: analysis & future prospects. In: 2021 2nd International Conference on Intelligent Engineering and Management (ICIEM); 2021 Apr 28–30; London, UK. p. 577–81.
3. Song DX, Wagner D, Perrig A. Practical techniques for searches on encrypted data. In: Proceeding 2000 IEEE Symposium on Security and Privacy (S&P '00); 2000 May 15–18; Oakland, CA, USA. p. 44–55.
4. Goyal V, Pandey O, Sahai A, Waters B. Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS '06). New York, NY, USA: Association for Computing Machinery; 2006. p. 89–98.
5. Lai J, Zhou X, Deng RH, Li Y, Chen K. Expressive search on encrypted data. In: Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security (ASIA CCS '13). New York, NY, USA: Association for Computing Machinery; 2013. p. 243–52.
6. Boneh D, Di Crescenzo G, Ostrovsky R, Persiano G. Public key encryption with keyword search. In: Cachin C, Camenisch JL, editors. Advances in cryptology—EUROCRYPT 2004. Berlin/Heidelberg, Germany: Springer; 2004. p. 506–22. doi:10.1007/978-3-540-24676-3_30.
7. Boneh D, Franklin M. Identity-based encryption from the weil pairing. In: Kilian J, editor. Advances in cryptology—CRYPTO 2001. Berlin/Heidelberg, Germany: Springer; 2001. p. 213–29. doi:10.1007/3-540-44647-8_13.
8. Chen Z, Wu C, Wang D, Li S. Conjunctive keywords searchable encryption with efficient pairing, constant ciphertext and short trapdoor. In: Chau M, Wang GA, Yue WT, Chen H, editors. Intelligence and security informatics. Berlin/Heidelberg, Germany: Springer; 2012. p. 176–89. doi:10.1007/978-3-642-30428-6_15.
9. Li J, Wang Q, Wang C, Cao N, Ren K, Lou W. Fuzzy keyword search over encrypted data in cloud computing. In: 2010 Proceedings IEEE INFOCOM; 2010 Mar 14–19; San Diego, CA, USA. p. 1–5.
10. Dong Q, Guan Z, Wu L, Chen Z. Fuzzy keyword search over encrypted data in the public key setting. In: Wang J, Xiong H, Ishikawa Y, Xu J, Zhou J, editors. Web-age information management. Berlin/Heidelberg, Germany: Springer; 2013. p. 729–40. doi:10.1007/978-3-642-38562-9_74.
11. Zhang H, Zhao S, Guo Z, Wen Q, Li W, Gao F. Scalable fuzzy keyword ranked search over encrypted data on hybrid clouds. IEEE Trans Cloud Comput. 2023;11(1):308–23. doi:10.1109/tcc.2021.3092358.
12. Jiang Y, Lu J, Feng T. Fuzzy keyword searchable encryption scheme based on blockchain. Information. 2022;13(11):517. doi:10.3390/info13110517.
13. Cash D, Jarecki S, Jutla C, Krawczyk H, Roşu MC, Steiner M. Highly-scalable searchable symmetric encryption with support for boolean queries. In: Canetti R, Garay JA, editors. Advances in cryptology—CRYPTO 2013. Berlin/Heidelberg, Germany: Springer; 2013. p. 353–73. doi:10.1007/978-3-642-40041-4_20.
14. Xu P, Tang S, Xu P, Wu Q, Hu H, Susilo W. Practical multi-keyword and boolean search over encrypted e-mail in cloud server. IEEE Trans Serv Comput. 2021;14(6):1877–89. doi:10.1109/tsc.2019.2903502.
15. Li F, Ma J, Miao Y, Liu Z, Choo KKR, Liu X, et al. Towards efficient verifiable boolean search over encrypted cloud data. IEEE Trans Cloud Comput. 2023;11(1):839–53. doi:10.1109/tcc.2021.3118692.
16. Tong Q, Li X, Miao Y, Liu X, Weng J, Deng RH. Privacy-preserving boolean range query with temporal access control in mobile computing. IEEE Trans Knowl Data Eng. 2023;35(5):5159–72. doi:10.1109/tkde.2022.3152168.
17. Liu Q, Tian Y, Wu J, Peng T, Wang G. Enabling verifiable and dynamic ranked search over outsourced data. IEEE Trans Serv Comput. 2022;15(1):69–82. doi:10.1109/tsc.2019.2922177.
18. Li J, Huang Y, Wei Y, Lv S, Liu Z, Dong C, et al. Searchable symmetric encryption with forward search privacy. IEEE Trans Dependable Secure Comput. 2021;18(1):460–74. doi:10.1109/tdsc.2019.2894411.
19. Gan Q, Wang X, Huang D, Li J, Zhou D, Wang C. Towards multi-client forward private searchable symmetric encryption in cloud computing. IEEE Trans Serv Comput. 2022;15(6):3566–76. doi:10.1109/tsc.2021.3087155.
20. Guo Y, Zhang C, Wang C, Jia X. Towards public verifiable and forward-privacy encrypted search by using blockchain. IEEE Trans Dependable Secure Comput. 2023;20(3):2111–26. doi:10.1109/tdsc.2022.3173291.
21. Cui N, Wang Z, Pei L, Wang M, Wang D, Cui J, et al. Dynamic and verifiable fuzzy keyword search with forward security in cloud environments. IEEE Internet Things J. 2025;12(9):11482–94. doi:10.1109/jiot.2024.3517165.

22. Sahai A, Waters B. Fuzzy identity-based encryption. In: Cramer R, editor. Advances in cryptology—EUROCRYPT 2005. Berlin/Heidelberg, Germany: Springer; 2005. p. 457–73. doi:10.1007/11426639_27.
23. Bethencourt J, Sahai A, Waters B. Ciphertext-policy attribute-based encryption. In: 2007 IEEE Symposium on Security and Privacy (SP '07); 2007 May 20–23; Berkeley, CA, USA. p. 321–34.
24. Waters B. Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. In: Catalano D, Fazio N, Gennaro R, Nicolosi A, editors. Public key cryptography—PKC 2011. Berlin/Heidelberg, Germany: Springer; 2011. p. 53–70. doi:10.1007/978-3-642-19379-8_4.
25. Ostrovsky R, Sahai A, Waters B. Attribute-based encryption with non-monotonic access structures. In: CCS '07. New York, NY, USA: Association for Computing Machinery; 2007. p. 195–203. doi:10.1145/1315245.1315270.
26. Lewko A, Sahai A, Waters B. Revocation systems with very small private keys. In: 2010 IEEE Symposium on Security and Privacy; 2010 May 16–19; Oakland, CA, USA. p. 273–85.
27. Meng L, Xu H, Tang R, Zhou X, Han Z. Dual Hybrid CP-ABE: how to provide forward security without a trusted authority in vehicular opportunistic computing. IEEE Internet Things J. 2024;11(5):8800–14. doi:10.1109/jiot.2023.3321563.
28. Luo W, Lv Z, Yang L, Han G, Zhang X. FOC-PH-CP-ABE: an efficient CP-ABE scheme with fully outsourced computation and policy hidden in the industrial internet of things. IEEE Sensors J. 2024;24(18):28971–81. doi:10.1109/jsen.2024.3432276.
29. Miao Y, Li F, Li X, Ning J, Li H, Choo KKR, et al. Verifiable outsourced attribute-based encryption scheme for cloud-assisted mobile e-health system. IEEE Trans Dependable Secure Comput. 2024;21(4):1845–62. doi:10.1109/tdsc.2023.3292129.
30. Duan P, Ma Z, Gao H, Tian T, Zhang Y. Multi-authority attribute-based encryption scheme with access delegation for cross blockchain data sharing. IEEE Trans Information Forensics Secur. 2025;20(2327):323–37. doi:10.1109/tifs.2024.3515812.
31. Ruan C, Hu C, Li X, Deng S, Liu Z, Yu J. A revocable and fair outsourcing attribute-based access control scheme in metaverse. IEEE Trans Consumer Electron. 2024;70(1):3781–91. doi:10.1109/tce.2024.3377107.
32. Zheng Q, Xu S, Ateniese G. VABKS: Verifiable attribute-based keyword search over outsourced encrypted data. In: IEEE INFOCOM 2014—IEEE Conference on Computer Communications; 2014 Apr 27–May 2; Toronto, ON, Canada. p. 522–30.
33. Han F, Qin J, Zhao H, Hu J. A general transformation from KP-ABE to searchable encryption. Fut Gener Comput Syst. 2014;30(2–3):107–15. doi:10.1016/j.future.2013.09.013.
34. Zhang K, Zhang Y, Li Y, Liu X, Lu L. A blockchain-based anonymous attribute-based searchable encryption scheme for data sharing. IEEE Internet Things J. 2024;11(1):1685–97. doi:10.1109/jiot.2023.3290975.
35. Ge C, Susilo W, Liu Z, Xia J, Szalachowski P, Fang L. Secure keyword search and data sharing mechanism for cloud computing. IEEE Trans Dependable Secure Comput. 2021;18(6):2787–800. doi:10.1109/tdsc.2020.2963978.
36. Ali M, Sadeghi MR, Liu X, Miao Y, Vasilakos AV. Verifiable online/offline multi-keyword search for cloud-assisted Industrial Internet of Things. J Inf Secur Appl. 2022;65(3):1–12. doi:10.1016/j.jisa.2021.103101.
37. Miao Y, Deng RH, Choo KKR, Liu X, Ning J, Li H. Optimized verifiable fine-grained keyword search in dynamic multi-owner settings. IEEE Trans Dependable Secure Comput. 2021;18(4):1804–20. doi:10.1109/tdsc.2019.2940573.
38. Huang Q, Yan G, Wei Q. Attribute-based expressive and ranked keyword search over encrypted documents in cloud computing. IEEE Trans Services Comput. 2023;16(2):957–68. doi:10.1109/tsc.2022.3149761.
39. Huang Q, Yan G, Yang Y. Privacy-preserving traceable attribute-based keyword search in multi-authority medical cloud. IEEE Trans Cloud Comput. 2023;11(1):678–91. doi:10.1109/tcc.2021.3109282.
40. Chen L, Xu S, Zhang H, Weng J. Fair-and-exculpable-attribute-based searchable encryption with revocation and verifiable outsourced decryption using smart contract. IEEE Internet Things J. 2025;12(4):4302–17. doi:10.1109/jiot.2024.3484227.
41. Tian T, Shen Y, Gao H, Ma Z, Guo Z, Duan P. Attribute-based heterogeneous data privacy sharing in blockchain-assisted Industrial IoT. IEEE Internet Things J. 2025;12(8):10404–19. doi:10.1109/jiot.2024.3510872.

42. Li J, Qin C, Lee PPC, Zhang X. Information leakage in encrypted deduplication via frequency analysis. In: 2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN); 2017 Jun 26–29; Denver, CO, USA. p. 1–12.
43. Lin HY, Tzeng WG. An efficient solution to the millionaires' problem based on homomorphic encryption. In: Ioannidis J, Keromytis A, Yung M, editors. Applied cryptography and network security. Berlin/Heidelberg, Germnay: Springer; 2005. p. 456–66. doi:10.1007/11496137_31.
44. De Caro A, Iovino V. JPBC: java pairing based cryptography. In: Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011; 2011 Jun 28–Jul 1; Corfu, Greece. p. 850–5.