symmetry

MDPI

Article

Ciphertext-Policy Attribute-Based Boolean Keyword Search with Negation for EHR Sharing

Hongjian Yin [1], Xiangbo Wang [1], Haicheng Chen [2], Guangyu Ding [3], Binrong Cheng [4,*] and Lei Zhang [1]

[1] School of Computer and Information Engineering, Henan University, Kaifeng 475004, China; zhanglei@henu.edu.cn (L.Z.)
[2] Hangzhou Muyao Technology Co., Ltd., Hangzhou 311100, China
[3] Chery New Energy Automotive Co., Ltd., Wuhu 241000, China; dingguangyu@henu.edu.cn
[4] Hangzhou Zhongcheng Consulting Supervision Co., Ltd., Hangzhou 310015, China
* Correspondence: pmp@zczxjl.com

Abstract

The growing volume of electronic health records (EHRs) has made cloud-assisted storage increasingly important, but outsourced records require both encrypted storage and controlled retrieval. Attribute-based keyword searchable encryption (ABKS) combines search with fine-grained authorization; nevertheless, many existing constructions provide only restricted Boolean queries, particularly when negation is involved. This paper presents a ciphertext-policy ABKS search layer for EHR sharing. The proposed construction operates over a fixed application dictionary and assumes a non-colluding honest-but-curious cloud server. Each record is represented by one signed literal per dictionary term, where a positive literal denotes presence and a negated literal denotes absence. Before query evaluation, Boolean formulas involving AND, OR, and NOT are converted into negation normal form using De Morgan's laws, so that NOT appears only at leaf nodes. The resulting query is then evaluated as a monotone threshold tree. An authorized user can generate a valid query token only when the user's multi-valued attribute vector matches the policy selected by the data owner. Keyed pseudorandom tags hide literal and policy values from the cloud server while enabling efficient component lookup. We define a selective chosen-keyword security model, state the resulting leakage, and give a game-based proof under the decisional bilinear Diffie–Hellman and pseudorandom-function assumptions. Theoretical and implementation-based comparisons show that the pairing-dominant costs grow linearly with the attribute and dictionary sizes. The construction therefore improves Boolean expressiveness while retaining practical search-layer overhead under the stated assumptions.

Keywords: attribute-based keyword search; Boolean keyword search; searchable encryption; electronic health records; access control

Check for updates

Received: 19 July 2026
Revised: 24 August 2026
Accepted: 27 August 2026
Published: 30 August 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

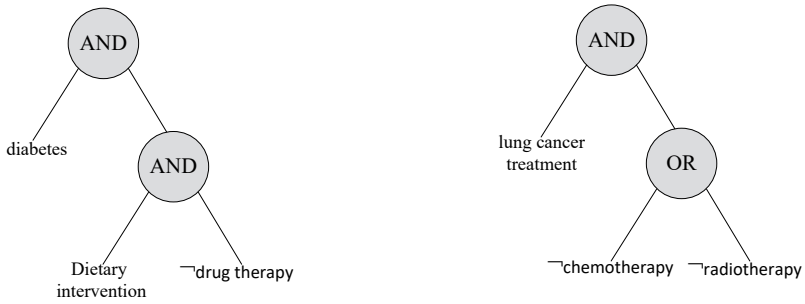
As requirements for scalable storage and high-efficiency data processing continue to rise, cloud-based platforms have gradually become a common choice for data outsourcing by organizations and individual users. In the medical field, Electronic Health Records (EHRs), encompassing highly confidential patient data, have increasingly relied on third-party cloud platforms for storage and management. To defend against potential attacks from semi-trusted cloud servers and to ensure data privacy, data owners typically encrypt

EHRs before outsourcing. However, such encryption renders traditional plaintext search algorithms ineffective, significantly limiting the flexibility of data retrieval.

Encrypted search allows users to query outsourced encrypted data while the cloud platform conducts comparison without obtaining the original plaintext content. Existing encrypted-search methods are usually classified into symmetric-key searchable encryption (SSE) [1] and public-key keyword search (PEKS) [2]. SSE usually provides high efficiency in single-owner settings, but its symmetric-key nature makes it less convenient for public cloud sharing scenarios involving multiple independent users. In PEKS, the data owner protects outsourced records with a public encryption key, whereas keyword trapdoors are generated solely by users who possess the matched secret key. These trapdoors are submitted to the cloud server for ciphertext-level keyword matching, which makes PEKS suitable for non-interactive encrypted data sharing [3].

Attribute-Based Keyword Searchable Encryption (ABKS) further combines searchable encryption with attribute-based authorization, making it useful for medical cloud systems with different access privileges. Under CP-ABKS, the data owner associates an access policy with each searchable ciphertext. As a result, the cloud-side matching process succeeds only for a requester whose attribute set conforms to the policy associated with the ciphertext [4]. Prior to outsourcing health record files, the record owner applies cryptographic protection and creates queryable index entries under the specified authorization rule. For each retrieval request, the requester derives a search trapdoor from the chosen terms and the secret key. The cloud server returns matching encrypted records only after both keyword matching and attribute-based authorization succeed.

However, existing studies still have several limitations. Many early encrypted-search schemes mainly support single-keyword or limited keyword search [5], which may force users to obtain many irrelevant ciphertexts and then filter them locally. Although some multi-keyword searchable encryption schemes support conjunction or disjunction, their query expressiveness is still insufficient for complex retrieval requirements. Boolean keyword search provides a more flexible way to describe users' search intentions by allowing logical expressions over multiple keywords. Nevertheless, many existing ABKS schemes supporting Boolean queries do not sufficiently consider the NOT operator. In practical medical retrieval, negation is also important. For example, Figure 1a shows an EHR query related to dietary intervention for diabetes while excluding drug therapy. Figure 1b describes a lung cancer treatment query that excludes chemotherapy and radiotherapy.



(**a**) AND NOT (**b**) OR NOT

Figure 1. Illustrative examples of Boolean keyword queries.

To address the above limitations, this work constructs a CP-ABKS scheme for secure and expressive EHR retrieval. The main contributions are listed below:

- We design a Boolean keyword search mechanism based on signed literals over a fixed application dictionary, where positive and negated literals represent keyword presence and absence, respectively. Boolean queries involving AND, OR, and NOT are first

converted into negation normal form, with NOT restricted to leaf nodes, and are then evaluated using a threshold tree.

- Search permission is controlled according to the attribute set owned by each requester. A query can be processed only after the requester's attributes conform to the policy defined by the data owner, thereby preventing unauthorized access to outsourced ciphertext records.
- A security framework over prime-order bilinear groups is formulated for the proposed construction. Under this framework, the proof shows that the scheme achieves selective chosen-keyword security and prevents a semi-trusted cloud server from carrying out effective keyword-guessing inference.
- We assess the scheme from the perspectives of supported functionality, theoretical overhead, and measured execution time. The comparison with representative ABKS constructions indicates that the proposed method preserves expressive Boolean retrieval while keeping computation and storage requirements at a practical level.

After the introduction, the subsequent parts are arranged below. Section 2 summarizes prior works related to searchable encryption and ABKS. Section 3 introduces the bilinear-map and access-structure preliminaries used in the construction. Section 4 defines the system architecture and the algorithmic interface. Section 5 describes the concrete algorithms and analyzes their correctness. Section 6 gives the security proof. Section 7 reports the functional comparison, theoretical overhead, and experimental results. Section 8 summarizes the main findings of this work.

2. Related Work

Encrypted search has become an important technique for querying protected data without exposing plaintext content. In early public-key encrypted-search systems, the data owner protects keyword-related information with a public key, whereas an authorized requester uses the secret key to produce a query token for matching encrypted records [2,6–8]. These works provide an important foundation for non-interactive encrypted retrieval in public cloud environments. However, their original designs mainly focus on basic keyword matching and do not directly address fine-grained user authorization. To enrich query expressiveness, subsequent studies introduced conjunctive search, multi-keyword retrieval, and predicate-based search mechanisms [9–12]. These methods improve the flexibility of encrypted search, but search authorization is generally not tightly bound to user attributes. For medical cloud applications, this limitation is significant because different doctors, departments, and healthcare organizations may have different access privileges for the same type of EHR data.

ABKS places encrypted keyword retrieval under an attribute-governed authorization mechanism. After Zheng et al. introduced this notion [13], later studies explored several directions, including finer authorization control, richer policy descriptions, and stronger protection against keyword-guessing attacks [14–17]. In such systems, search capability is not granted merely by knowing a keyword; instead, the requester must hold attribute credentials that meet the policy requirement. This property is important for cloud-based EHR sharing because knowledge of partial keyword information should not allow an unauthorized party to locate sensitive medical records. However, a considerable number of ABKS schemes still focus on single-keyword search or support only restricted multi-keyword queries. Although policy-oriented and multi-keyword ABKS constructions improve retrieval coverage [18], their query logic is commonly limited to subset matching, simple conjunction, or other narrow expression forms. Therefore, these schemes are still insufficient for medical retrieval tasks that require flexible and complex search conditions.

Boolean keyword search provides a more expressive way to formulate encrypted queries. By using logical operators, users can describe retrieval intentions more precisely

than with independent keywords. Several ABKS schemes have attempted to introduce Boolean expressions, access trees, LSSS matrices, hybrid query structures, ranked retrieval, or enhanced cloud architectures to improve search functionality [19–22]. These studies show that expressive keyword search is useful for encrypted cloud data sharing. However, important limitations remain. Some schemes do not support negated keywords, and some only support specific Boolean forms rather than arbitrary combinations. Some ranked-search schemes provide richer retrieval results but introduce additional search overhead. Moreover, access-policy hiding is not always considered, which may expose sensitive authorization information to a semi-trusted cloud server. Huang et al. [22] support AND, OR, NOT, and threshold queries using a non-monotonic access tree and further provide ranked retrieval. In contrast, our scheme realizes negation through signed literals over a fixed application dictionary, where positive and negated literals represent keyword presence and absence, respectively. Boolean queries are converted into negation normal form so that NOT appears only at leaf nodes, allowing positive and negative conditions to be handled uniformly during threshold-tree evaluation. This design provides explicit keyword-absence semantics and avoids introducing separate processing logic for negated conditions. In EHR sharing scenarios, queries involving negation are common. For example, a user may search for diabetes records involving dietary intervention but excluding drug therapy, or lung cancer treatment records excluding chemotherapy and radiotherapy. Such cases require flexible combinations of AND, OR, and NOT operators.

To address these requirements, our construction realizes negation using signed literals over a fixed application dictionary, where positive and negated literals represent keyword presence and absence, respectively. Boolean queries are converted into negation normal form so that NOT appears only at leaf nodes. Meanwhile, access authorization is enforced through multi-valued attributes, and policy-related information is protected during the search process. Therefore, the scheme aims to improve query expressiveness and privacy protection while ensuring that both computation and storage costs remain acceptable for cloud-based healthcare applications.

3. Preliminaries

3.1. Bilinear Maps

Consider two cyclic groups G and G_T with the same prime order p. Let $e : G \times G \to G_T$ be a bilinear pairing. For arbitrary elements $u, v \in G$ and exponents $\alpha, \beta \in \mathbb{Z}_p$, the pairing satisfies

$$e(u^{\alpha}, v^{\beta}) = e(u, v)^{\alpha\beta}.$$

Moreover, non-degeneracy ensures that some $u, v \in G$ can be found for which $e(u, v)$ is not equal to 1_{G_T}. These algebraic properties support the ciphertext generation, trapdoor computation, and pairing-based verification used in the proposed construction.

3.2. Access Structure

Denote the attribute universe adopted in this work by $U = \{att_1, att_2, \ldots, att_n\}$. For each attribute att_i, its possible values are taken from the domain

$$V_i = \{v_{i,1}, v_{i,2}, \ldots, v_{i,n_i}\}.$$

The access policy is denoted by $P = (p_1, p_2, \ldots, p_n)$, where $p_i \in V_i$ is the value required for attribute att_i. Thus, the policy is an AND gate over multi-valued attributes: each attribute has a multi-valued domain, while one value from that domain is selected in a particular ciphertext policy.

A user is described by an attribute vector $S = \{x_1, x_2, \ldots, x_n\}$, where $x_i \in V_i$. The policy P is fulfilled by S, written $S \models P$, exactly when $x_i = p_i$ for every $i \in [1, n]$. This equality condition is required by the cancellation used in Section 5.

It should be emphasized that the above AND-gate policy is used only for attribute-based authorization and is distinct from the Boolean keyword-search policy introduced in Section 3.3. The former determines whether a requester is eligible to perform a search: all required credential attributes, such as institutional affiliation, department, specialty, or professional role, must match the values specified by the data owner. In contrast, the latter specifies the retrieval conditions over encrypted keywords and supports AND, OR, and leaf-level NOT after the query is converted into negation normal form. We deliberately adopt the relatively simple conjunctive structure for authorization because it is consistent with the equality-based cancellation used in our construction and keeps the authorization layer separate from the expressive Boolean keyword-search mechanism, which is the main focus of this work.

Such conjunctive authorization requirements are also meaningful in practical EHR sharing. For example, access may be restricted to a user who is simultaneously affiliated with a particular hospital, holds the role of physician, and belongs to a specified medical specialty or department. Similar combinations of hospital affiliation, professional role, specialty, and department have been considered in existing EHR/PHR access-control systems [23,24]. Conjunctive policies such as "Hospital-1 AND Doctor" have also been explicitly used to model healthcare access permissions [25]. Nevertheless, this authorization structure cannot express attribute-level OR gates or threshold policies. Consequently, scenarios requiring alternatives among credential attributes or threshold-based authorization would require a more expressive access-control structure, which is beyond the scope of the present work, whose primary focus is expressive Boolean keyword search.

3.3. Access Tree

The Boolean search condition is expressed using an access tree T. For an internal node x, denote its child-node set by $children(x)$, and let k_x be the threshold assigned to this node, where $0 < k_x \leq |children(x)|$. When $k_x = 1$, the node behaves as an OR gate, since one valid child is sufficient. When $k_x = |children(x)|$, the node corresponds to an AND gate, because all child nodes must be satisfied. Each leaf node carries a signed keyword literal, and each node is assigned a unique index $index_x$. A Boolean formula is first converted to negation normal form, so that negation occurs only at leaves.

Denote by R the root node of T. For a node x, T_x refers to the subtree with x as its root, so the whole tree can be written as $T_R = T$. For a set L of signed literals representing a record, the satisfaction value of T_x is given by $T_x(L) = 1$, if L satisfies the subtree T_x; otherwise, $T_x(L) = 0$. For a leaf node x, $T_x(L) = 1$ holds exactly when its signed literal $\ell(x)$ belongs to L.

4. System Framework

4.1. System Architecture

The system involves four entities: a trusted authority, a data owner, a data user, and a cloud server. The trusted authority establishes the cryptographic environment and distributes private keys according to user attributes. Before outsourcing EHR data, the data owner applies encryption to the records and sets the search-control policy that specifies the authorized search scope. A data user then creates a search token for an intended query, whereas the cloud server stores ciphertexts and performs ciphertext-level matching without accessing plaintext records. Figure 2 presents the overall workflow, and the roles of these entities are described below.

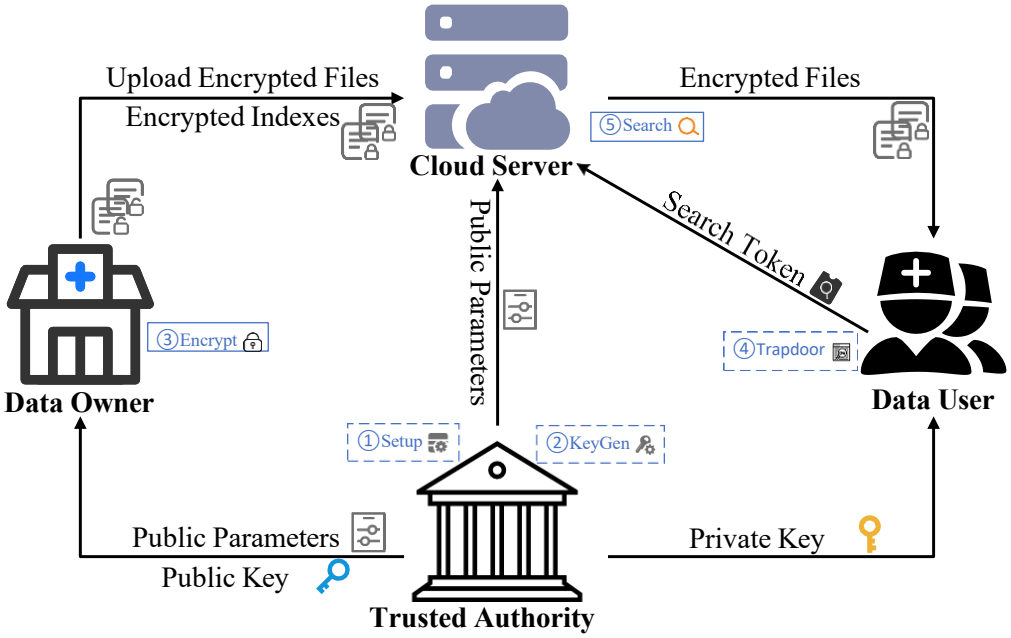


Figure 2. System model of the proposed scheme.

(1) Trusted Authority (TA): The TA produces the global cryptographic parameters while preserving the master secret. Based on the attributes of each participant, it provides the required private key components for data owners and data users. In the proposed system, the TA is only involved in the setup and key-issuing phases, and it is not required during each search process.

(2) Data Owner (DO): The DO owns the original EHR records and decides how these records are shared. Before outsourcing data to cloud storage, the DO applies encryption to medical records while constructing ciphertext components associated with keywords. It also defines the search-control policy, which specifies the attribute requirements that a requester must satisfy before performing encrypted retrieval.

(3) Data User (DU): The DU represents an authorized requester, such as a doctor, researcher, or medical institution. With an authorized secret key and a chosen Boolean keyword expression, the data user derives a query token for the target search task. This query token is delivered to the cloud server without exposing the original keywords or the user's secret credential.

(4) Cloud Server (CS): The CS offers cloud-based storage and encrypted search functionality. It stores encrypted EHR files and the corresponding searchable ciphertext components uploaded by data owners. When receiving a search token, the CS checks both the attribute-based authorization condition and the encrypted keyword-matching condition. The CS is regarded as an honest-but-curious party: it follows the required protocol operations correctly, yet may try to derive sensitive information from ciphertexts, query tokens, and returned results. As in standard searchable encryption, the CS observes ciphertext and token sizes, repeated pseudorandom tags, the query structure, the search pattern, and the access/result pattern; the construction does not claim to hide this leakage.

The cloud server is considered honest-but-curious, and the security of the proposed construction further relies on the assumption that it does not collude with credentialed data owners or authorized data users. If the CS obtains secret information such as K_W or g^c through such collusion, it may validate candidate keywords by dictionary guessing. Therefore, resistance to cloud–owner or cloud–user collusion is outside the scope of this work.

4.2. Algorithm Overview

The proposed search layer consists of five procedures: *Setup*, *KeyGen*, *Encrypt*, *Trapdoor*, and *Search*. The EHR payload itself is encrypted with an authenticated symmetric cipher under a per-record key, while key encapsulation and payload decryption are orthogonal to the searchable-index construction considered in this work. The formal interfaces of the five core algorithms are defined as follows.

(1) $\mathsf{Setup}(1^\lambda) \to (PK, MSK, EK)$: For the security parameter 1^λ, the algorithm generates the public parameters PK, the master secret key MSK, and the data-owner encryption credential EK.
(2) $\mathsf{KeyGen}(MSK, S) \to SK$: The master secret key MSK and the attribute vector S are used to derive the corresponding user secret key SK.
(3) $\mathsf{Encrypt}(PK, EK, P, \mathcal{D}, W) \to CT$: Using the public parameters PK, the encryption credential EK, the access policy P, the fixed application dictionary $\mathcal{D}$, and the record keyword set $W \subseteq \mathcal{D}$, the algorithm generates the searchable ciphertext CT.
(4) $\mathsf{Trapdoor}(PK, SK, B) \to TK$: The search token TK is generated from the public parameters PK, the user secret key SK, and the Boolean keyword expression B.
(5) $\mathsf{Search}(CT, TK) \to \{1, \perp\}$: The algorithm processes the searchable ciphertext CT and the search token TK and returns 1 when both the authorization condition and the Boolean keyword condition are satisfied; otherwise, it returns $\perp$.

The detailed construction of these algorithms is presented in Section 5.

5. Construction

5.1. Concrete Construction

The concrete algorithms of the proposed construction are specified below.
$Setup(1^\lambda) \to (PK, MSK, EK)$: Given the security parameter 1^λ, the TA first invokes $PairGen(1^\lambda)$ to generate the bilinear pairing environment. Consider a cyclic group G generated by g, together with its associated target group G_T, both of prime order p. The TA samples $a, b, c \in \mathbb{Z}_p^*$ as secret exponents and independent keys κ_A, κ_W for pseudorandom functions $F_A : \{0,1\}^* \to G$ and $F_W : \{0,1\}^* \to \{0,1\}^\lambda$. The scheme also defines $\mathbb{H}_2 : \{0,1\}^* \to \mathbb{Z}_p^*$. It obtains g^a, g^b, g^c, and $g^{\frac{1}{b}}$. Importantly, g^c and the PRF keys are omitted from the public parameters. The system values are

$$\begin{aligned} PK &= (e, G, G_T, p, \mathbb{H}_2, g, g^a, g^b, g^{\frac{1}{b}}), \\ MSK &= (a, b, c, \kappa_A, \kappa_W), \\ EK &= (g^c, \kappa_A, \kappa_W). \end{aligned}$$

The owner credential EK is delivered only to authenticated data owners. This separation is needed because publishing g^c would allow a server to validate low-entropy keyword guesses against a trapdoor.
$KeyGen(MSK, S) \to (SK)$: For a participant described by the attribute vector $S = (x_1, x_2, \ldots, x_n)$, the TA derives a private key from the master secret MSK. It chooses $t \in \mathbb{Z}_p^*$ and $u_i \in \mathbb{Z}_p^*$ independently for each $i = 1, 2, \ldots, n$. For every attribute value x_i contained in S, the corresponding key components are generated as $D_i = g^t F_A(\kappa_A, x_i)^{u_i}$, $D_i' = g^{u_i}$. The remaining component is calculated by $D = g^{(ac-t)/b}$. Finally, the private key issued to the user is written as

$$SK = \left\{ D, K_c = g^c, K_W = \kappa_W, \left\{ D_i, D_i' \right\}_{i \in [1,n]} \right\}.$$

$Encrypt(PK, EK, P, \mathcal{D}, W) \to (CT)$: The DO executes this algorithm to protect keyword-related information according to the given access policy. Given a policy $P = (p_1, p_2, \ldots, p_n)$,

a fixed application dictionary $\mathcal{D} = \{d_1, \ldots, d_m\}$, and the set $W \subseteq \mathcal{D}$ of terms occurring in the record, the DO defines one signed literal per dictionary term:

$$\ell_j = \begin{cases} d_j, & d_j \in W, \\ \neg d_j, & d_j \notin W. \end{cases}$$

Thus, a negative literal represents absence rather than a second encoding of a present keyword. The DO chooses random values $s_i \in \mathbb{Z}_p^*$ and sets $r = \sum_{i=1}^{n} s_i$, where $r \in \mathbb{Z}_p^*$. Then it computes $C' = g^{br}$. For every policy component p_i, it generates $\widehat{C}_i = F_A(\kappa_A, p_i)^{s_i}$ and $C_i = g^{s_i}$, where $i \in [1, n]$. For each signed literal ℓ_j, the DO randomly selects $k_j \in \mathbb{Z}_p^*$ and computes

$$\sigma_j = F_W(\kappa_W, \ell_j), \qquad C_{\ell_j} = (g^c)^{k_j}, \qquad C'_{\ell_j} = g^{a(r+k_j)} g^{b\mathbb{H}_2(\ell_j)k_j}.$$

The pseudorandom tag σ_j permits direct lookup without disclosing the literal to the CS. Finally, the ciphertext generated by the encryption procedure consists of the following components:

$$CT = \left\{ C', \{\widehat{C}_i, C_i\}_{i=1}^{n}, \{(\sigma_j, C_{\ell_j}, C'_{\ell_j})\}_{j=1}^{m} \right\}.$$

Trapdoor$(PK, SK, B) \rightarrow (TK)$: DU executes the algorithm. The user first transforms the Boolean query expression B (with AND, OR, and NOT) into a monotonic access tree T' by pushing each NOT operator to the leaves using De Morgan's laws. After the Boolean expression is transformed into T', each leaf contains a signed literal $\ell(x) \in \{d, \neg d : d \in \mathcal{D}\}$. For every node x in T', a polynomial q_x is constructed, and the leaves are used to represent the corresponding keyword conditions. A fresh exponent $v \in \mathbb{Z}_p^*$ is selected and assigned to the root polynomial by setting $q_R(0) = v$. The blinded component is then computed as $\hat{D} = D^v = g^{(ac-t)v/b}$. For every attribute value x_i in the set S, the corresponding key components are blinded with the randomness v as

$$\hat{D}_i = D_i^v = g^{tv} F_A(\kappa_A, x_i)^{u_i v}, \qquad \hat{D}'_i = (D'_i)^v = g^{u_i v}.$$

For a leaf node x in T', the user computes the tag $\sigma_x = F_W(K_W, \ell(x))$ and the keyword-related trapdoor components

$$\widehat{D}_x = \left(g^a g^{b\mathbb{H}_2(\ell(x))}\right)^{q_x(0)}, \qquad \widehat{D'}_x = K_c^{q_x(0)}.$$

The resulting trapdoor is given as follows:

$$TK = \left\{ T', \hat{D}, \{\hat{D}_i, \hat{D}'_i\}_{i=1}^{n}, \{(\sigma_x, \widehat{D}_x, \widehat{D'}_x)\}_{x \in \text{leaf}(T')} \right\}.$$

Search$(CT, TK) \rightarrow (1$ or $\perp)$: The CS takes CT and TK as inputs and performs ciphertext-level search. The search algorithm outputs a successful result only if two requirements are met simultaneously: the requester must comply with the access policy bound to the ciphertext, and the Boolean query must be consistent with the encrypted keyword components in CT.

To clarify the execution flow, the search procedure is performed in four steps. The CS first evaluates the attribute-based authorization information, then performs leaf-node matching for the query tree, recursively evaluates the internal nodes according to their threshold values using Lagrange interpolation, and finally verifies the root-node equation.

Step 1: Attribute-based authorization evaluation. The CS first computes the authorization factor as follows:

$$E = \frac{\prod_{i=1}^{n} e(C_i, \widehat{D}_i)}{\prod_{i=1}^{n} e(\widehat{C}_i, \widehat{D}'_i)} = \frac{\prod_{i=1}^{n} e(g^{s_i}, g^{tv} F_A(\kappa_A, x_i)^{u_i v})}{\prod_{i=1}^{n} e(F_A(\kappa_A, p_i)^{s_i}, g^{u_i v})}. \tag{1}$$

When the user's attribute vector satisfies the policy, i.e., $x_i = p_i$ for every $i \in [1, n]$, the attribute-related terms cancel and the above expression reduces to $E = e(g, g)^{rtv}$. The resulting value E is retained for the final root-node verification. If the attribute policy is not satisfied, the cancellation is incomplete, so the final verification equation cannot hold except with negligible probability.

Step 2: Leaf-node matching. The CS then examines the keyword-matching condition. Specifically, it determines whether the Boolean query carried by the trapdoor is compatible with the encrypted signed literals in the ciphertext.

For a leaf x in T', the CS uses σ_x to retrieve the ciphertext tuple with the same tag. If no such tuple exists, it sets $E_x = \perp$; otherwise, it evaluates

$$E_x = \frac{e(C'_{\ell(x)}, \widehat{D}'_x)}{e(C_{\ell(x)}, \widehat{D}_x)} = \frac{e(g^{a(r+k_j)} g^{b\mathbb{H}_2(\ell(x))k_j}, g^{cq_x(0)})}{e(g^{ck_j}, (g^a g^{b\mathbb{H}_2(\ell(x))})^{q_x(0)})} = e(g, g)^{acrq_x(0)}. \tag{2}$$

Step 3: Recursive threshold evaluation. For an internal node x, the algorithm selects k_x valid child nodes to form the set S_x. Each selected child z must have a non-null evaluation result, namely $E_z \neq \perp$. If no such subset can be found, the value of E_x is set to $\perp$; otherwise, E_x is computed using polynomial interpolation.

For a finite index set $S \subseteq \mathbb{Z}_p$ and an index $i \in S$, the Lagrange interpolation coefficient is defined as

$$\Delta_{i,S}(j) = \prod_{\substack{i' \in S \\ i' \neq i}} \frac{j - i'}{i - i'} \pmod{p}, \tag{3}$$

where i denotes the index associated with the current interpolation term, i' ranges over the remaining indices in S, and j denotes the interpolation point. In the recursive tree evaluation, $j = 0$ is used to reconstruct the polynomial value associated with the parent node.

Let $S'_x = \{index(z) : z \in S_x\}$. The value of the internal node is computed as

$$E_x = \prod_{z \in S_x} E_z^{\Delta_{index(z), S'_x}(0)} = \prod_{z \in S_x} \left(e(g, g)^{acrq_z(0)} \right)^{\Delta_{index(z), S'_x}(0)} = e(g, g)^{acrq_x(0)}. \tag{4}$$

Step 4: Root-node verification. After the recursive computation is completed, the root value is obtained as $E_R = e(g, g)^{acrv}$. The CS determines whether the search result is valid by checking

$$E_R = e(C', \widehat{D})E.$$

When the verification equation is satisfied, the server regards the search request as valid and returns 1; otherwise, the result is set to $\perp$.

5.2. Correctness Analysis

The successful case is first analyzed in three parts: authorization checking, keyword-tree evaluation, and final equality verification. We then consider two representative failure cases.

We first analyze the authorization checking stage. Assume that the user's attribute value satisfies $x_i = p_i$ for every policy component. In this case, the hash term appearing in

the ciphertext and the hash term embedded in the user's key are algebraically compatible. Thus, after applying the bilinear pairing, the attribute-related factors in $\widehat{C}_i$, C_i, $\widehat{D}_i$, and $\widehat{D}'_i$ are canceled. When all required attribute conditions are satisfied, the cloud server obtains

$$E = \frac{\prod_{i=1}^{n} e(C_i, \widehat{D}_i)}{\prod_{i=1}^{n} e(\widehat{C}_i, \widehat{D}'_i)} = e(g,g)^{rtv}.$$

The value E contains the encryption randomness r, the key randomness t, and the trapdoor randomness v. At the same time, the attribute hash components are removed through the quotient of pairing terms. If some $x_i \neq p_i$, the PRF-derived group elements differ except with negligible collision probability, the cancellation cannot be completed, and the server fails to derive the expected authorization value.

Next, we consider the Boolean keyword evaluation. After the query is converted into the tree T', each leaf node corresponds to either a normal keyword or its negated form. For a positive keyword, the server checks the ciphertext component corresponding to that keyword. For a negative keyword, the NOT operator has already been pushed to the leaf level. The leaf tag matches a ciphertext component only when the term is absent from the record, because Encrypt emits $\neg d$ exactly when $d \notin W$. Therefore, positive and negative keyword conditions are handled within the same tree-evaluation process. If a leaf label matches the encrypted keyword component, the pairing computation produces a valid leaf value. For an internal node, the server combines enough valid child nodes according to its threshold value. When the Boolean expression represented by T' is fulfilled, polynomial interpolation reconstructs the value at the root node.

After the recursive computation over the query tree, the server obtains

$$E_R = e(g,g)^{acrv}.$$

The pairing term related to C' and $\widehat{D}$ can be evaluated from $C' = g^{br}$ and $\widehat{D} = g^{(ac-t)v/b}$ as

$$e(C', \widehat{D}) = e(g^{br}, g^{(ac-t)v/b}) = e(g,g)^{r(ac-t)v}.$$

By combining this term with the authorization factor $E = e(g,g)^{rtv}$, we obtain

$$e(C', \widehat{D})E = e(g,g)^{r(ac-t)v} \cdot e(g,g)^{rtv} = e(g,g)^{acrv}.$$

Hence, the verification equation

$$E_R = e(C', \widehat{D})E$$

is satisfied when both the attribute policy and the Boolean keyword condition hold.

Accordingly, an authorized data user whose query matches the encrypted keyword structure obtains a successful search result.

We further consider the two failure cases.

Case 1: The user's attributes do not satisfy the ciphertext policy. If $x_i \neq p_i$ for at least one i, the attribute-related pairing terms in the numerator and denominator cannot be completely canceled. Therefore, the authorization factor E deviates from the expected value $e(g,g)^{rtv}$ except with negligible probability. Consequently,

$$e(C', \widehat{D})E \neq e(g,g)^{acrv},$$

and hence the final verification equation

$$E_R = e(C', \widehat{D})E$$

cannot hold even if the Boolean keyword tree itself is satisfied. Therefore, the CS returns $\perp$.

Case 2: The user's attributes satisfy the policy but the Boolean condition fails. In this case, the authorization factor is correctly obtained as $E = e(g,g)^{rtv}$. However, if a required leaf node does not match the corresponding encrypted signed literal, its evaluation result is set to $E_x = \perp$. Similarly, if an internal node x has fewer than k_x valid child nodes, its threshold condition cannot be satisfied and $E_x = \perp$. Consequently, the expected root value $E_R = e(g,g)^{acrv}$ cannot be reconstructed, and the final verification equation cannot hold. Therefore, the CS returns $\perp$.

Therefore, the construction correctly distinguishes valid search requests from invalid ones.

6. Security Analysis

Threat Model and Leakage

The TA is trusted, and credential-distribution channels are authenticated and confidential. The adversary controls the honest-but-curious CS but does not obtain EK, K_c, either PRF key, or a secret key satisfying the challenge policy. In particular, the model excludes collusion between the CS and a credentialed owner or user. The leakage function $\mathcal{L}$ contains the public dictionary size, attribute dimension, ciphertext and query-tree sizes, equality of pseudorandom tags, search pattern, access-policy outcome, and result pattern. Hiding access or search patterns requires ORAM or private-information-retrieval techniques and is outside the scope of this work.

In EHR-oriented healthcare scenarios, such leakage may still reveal sensitive behavioral information even when plaintext keywords and medical records remain encrypted. Repeated search patterns may enable the cloud server to infer that a user repeatedly queries the same disease, treatment, or diagnostic condition, while result patterns may reveal whether different queries access the same group of encrypted health records. By correlating these observations over time with auxiliary information, an honest-but-curious cloud server may infer sensitive information concerning patients' medical conditions, treatment activities, or users' clinical interests. Mitigating such leakage requires additional techniques, such as oblivious RAM (ORAM) or private information retrieval (PIR), which are beyond the scope of the present construction.

In the selective IND-sCP-CKA game, the adversary commits to two equal-size challenge record descriptions and a target policy before setup. The descriptions must induce the same leakage $\mathcal{L}$. The adversary may request keys and trapdoors adaptively, except for a key satisfying the target policy or a trapdoor that trivially distinguishes the two challenge descriptions. The challenger encrypts one description chosen by a hidden bit, and the adversary attempts to recover that bit.

Theorem 1. *Assume that F_A and F_W are secure pseudorandom functions, $\mathbb{H}_2$ is modeled as a random oracle, and the decisional bilinear Diffie–Hellman problem is hard in (G, G_T). Then the proposed construction is selectively IND-sCP-CKA secure with respect to leakage $\mathcal{L}$ against the non-colluding CS described above.*

The security reduction in Theorem 1 explicitly relies on the non-collusion assumption. Specifically, the CS is assumed not to obtain the encryption credential EK of a data owner or the secret credential of an authorized user. If the CS colludes with a credentialed data owner or user, secret information such as K_W or g^c may be disclosed, enabling dictionary-based keyword validation. In this case, the assumptions required by the reduction are violated, and the security conclusion of Theorem 1 no longer holds.

Consider a sequence of games. Game 0 is the real selective experiment. In Game 1, all F_W outputs in ciphertexts and trapdoors are consistently replaced with random tags. Distinguishing Games 0 and 1 breaks the PRF security of F_W, while preserving the tag-equality leakage admitted by $\mathcal{L}$. In Game 2, the policy encodings $F_A(\kappa_A, p_i)$ are replaced with independent random group elements, subject to the equality required for authorized keys. Distinguishing Games 1 and 2 breaks the PRF security of F_A.

Conditioned on the permitted leakage, the remaining challenge-dependent terms have the form $(g^c)^{k_j}$ and $g^{a(r+k_j)}g^{b\mathbb{H}_2(\ell_j)k_j}$ for fresh r, k_j. The omitted value g^c prevents the CS from constructing the public pairing test that would otherwise validate a guessed literal against $\widehat{D}_x$ and $\widehat{D'}_x$. A simulator embeds the bilinear Diffie–Hellman challenge in the common r-dependent term and answers legal key and trapdoor queries using the selective-policy restriction. Replacing that term with a random element gives Game 3; distinguishing Games 2 and 3 solves DBDH. In Game 3, the challenge bit is independent of the adversary's view beyond $\mathcal{L}$. Therefore,

$$\mathrm{Adv}_{\mathcal{A}}^{\text{IND-sCP-CKA}} \leq \mathrm{Adv}_{F_W}^{\mathrm{PRF}} + \mathrm{Adv}_{F_A}^{\mathrm{PRF}} + \mathrm{Adv}_{G,e}^{\mathrm{DBDH}} + \mathrm{negl}(\lambda).$$

The theorem does not hide repeated-query or result patterns. These leakage boundaries are stated explicitly so that the security claim is not interpreted as full leakage-free searchable encryption.

7. Performance Analysis

The following analysis examines the proposed scheme from three aspects: supported functions, analytical overhead, and measured execution time. Representative ABKS schemes supporting Boolean search are selected for comparison.

7.1. Functionality Analysis

Table 1 compares the main functions of different schemes, including access structure, Boolean search capability, support for NOT queries, policy hiding, and hardness assumption. Our construction supports AND, OR, and NOT queries and protects the access policy during search.

Table 1. Functional comparison of different schemes.

Scheme	Access Structure	NOT Queries	Arbitrary Boolean Logic Query	Policy Hiding	Hardness Assumption
Huang et al. [22]	LSSS	✓	✗	✗	DBDH, HDH
He et al. [20]	Access Tree	✗	✗	✗	DBDH
Zhang et al. [26]	AND-gate	✗	✗	✗	SXDH
Ours	AND-gate on multi-value attribute	✓	✓	✓	DBDH

Note: ✓ indicates that the corresponding functionality is supported, whereas ✗ indicates that it is not supported. The "Arbitrary Boolean Logic Query" capability of our scheme refers to Boolean expressions composed of AND, OR, and NOT over a fixed application dictionary. Before evaluation, the query is converted into negation normal form, with NOT appearing only at leaf nodes.

It is clear that both our scheme and the scheme [22] support NOT queries. Nevertheless, the scheme in [22] realizes negation through a different query structure and does not provide policy-hiding capability. The schemes in [20,26] are unable to process NOT-based queries, and the latter also fails to hide access policies. Different from these schemes, our construction adopts an AND-gate structure over multi-valued attributes to enforce access control. A user is regarded as authorized only if every required multi-valued attribute condition is fulfilled. Such a design ensures secure user authorization and, at the same time, avoids introducing excessive query overhead. In summary, our scheme supports Boolean queries involving AND, OR, and leaf-level NOT under the stated fixed-dictionary and negation-normal-form constraints, while also providing access policy hiding.

7.2. Computation and Storage Overhead Analysis

Tables 2 and 3 give an analytical evaluation of computation and storage requirements for our scheme and representative ABKS schemes. In Table 2, $|U|$ represents the size of the attribute universe, whereas $|S|$ indicates the number of attributes owned by the user. Here, $|W|$ denotes the number of keywords associated with a record, while $|\mathcal{D}|$ and $|W_Q|$ represent the fixed dictionary size and the number of query leaves, respectively. For the construction of Huang et al. [22], $\mathcal{N}$ represents the number of columns in the LSSS matrix, while $|\mathcal{NOT}|$ indicates the count of negated keyword conditions in the query. Besides, the notation and numerical values of the basic cryptographic operations and element sizes are summarized in Table 4. In this table, Type A parameters are used for the schemes [20,22] and our scheme, whereas Type D parameters are used for the scheme [26].

Because PRF/hash evaluations and operations in $\mathbb{Z}_p$ are relatively lightweight compared with exponentiation and pairing operations, their computation costs are excluded from the pairing-dominant comparison. Pseudorandom tags add $|\mathcal{D}|\lambda$ bits to a ciphertext and $|W_Q|\lambda$ bits to a token.

Table 2. Computation costs of different schemes.

Scheme	Setup	KeyGen	Encrypt	Trapdoor	Search
[22]	$(2\|W\|+\mathcal{N}+3)E_{\mathbb{G}}+E_{\mathbb{G}_T}$	$(3\|S\|+1)E_{\mathbb{G}}$	$(2\mathcal{N}\|U\|+3\|W\|+1)E_{\mathbb{G}}$ $+E_{\mathbb{G}_T}+t_P$	$(4\|W_Q\|+3)E_{\mathbb{G}}$	$(\|S\|+4\|W\|+\mathcal{N}+3\|\mathcal{NOT}\|+1)t_P$ $+((\mathcal{N}+1)\|S\|+\|\mathcal{NOT}\|W\|+1)E_{\mathbb{G}}$ $+(2\|\mathcal{NOT}\|+\|W\|)E_{\mathbb{G}_T}$
[20]	$3E_{\mathbb{G}}$	$(2+2\|S\|)E_{\mathbb{G}}$	$(1+2\|U\|+3\|W\|)E_{\mathbb{G}}$	$(1+2\|S\|+3\|W_Q\|)E_{\mathbb{G}}$	$(1+2\|U\|+2\|W_Q\|)t_P$ $+(\|U\|+\|W_Q\|)E_{\mathbb{G}_T}$
[26]	$4E_{\mathbb{G}_1}+E_{\mathbb{G}_2}+t_{P,d}$	$(2+3\|S\|)E_{\mathbb{G}_2}$	$(3\|U\|+\|W\|)E_{\mathbb{G}_1}+t_{P,d}$	$\|W_Q\|E_{\mathbb{G}_1}+2\|S\|E_{\mathbb{G}_2}$	$\|W_Q\|E_{\mathbb{G}_1}+\|U\|E_{\mathbb{G}_{T,d}}$ $+2\|U\|t_{P,d}$
Ours	$4E_{\mathbb{G}}$	$(1+3\|S\|)E_{\mathbb{G}}$	$(1+2\|U\|+3\|\mathcal{D}\|)E_{\mathbb{G}}$	$(1+2\|S\|+2\|W_Q\|)E_{\mathbb{G}}$	$(1+2\|U\|+2\|W_Q\|)t_P$ $+(1+\|W_Q\|)E_{\mathbb{G}_T}$

Table 3. Storage requirements of different schemes.

Scheme	Setup	KeyGen	Encrypt	Trapdoor	Search
[22]	$(3+\|W\|+\|U\|\cdot\mathcal{N})l_{\mathbb{G}}+l_{\mathbb{G}_T}+2l_{\mathbb{Z}_p^*}$	$(1+\mathcal{N}+\|S\|)l_{\mathbb{G}}$	$(1+3\|W\|+\|U\|\cdot\mathcal{N})l_{\mathbb{G}}$ $+l_{\mathbb{G}_T}+\|W\|l_{\mathbb{Z}_p^*}$	$(3\|W_Q\|+2+\mathcal{N}+\|S\|)l_{\mathbb{G}}$	$l_{\mathbb{Z}_p^*}$
[20]	$5l_{\mathbb{G}}+3l_{\mathbb{Z}_p}$	$(1+2\|S\|)l_{\mathbb{G}}$	$(1+2\|U\|+2\|W\|)l_{\mathbb{G}}$	$(1+2\|S\|+2\|W_Q\|)l_{\mathbb{G}}$	$\perp$
[26]	$(3+n)l_{\mathbb{G}_1}+l_{\mathbb{G}_{T,d}}+3l_{\mathbb{Z}_p}$	$(1+2\|S\|)l_{\mathbb{G}_2}$	$(2+2\|U\|+\|W\|)l_{\mathbb{G}_1}$ $+l_{\mathbb{G}_{T,d}}$	$(1+2\|S\|)l_{\mathbb{G}_2}$ $+\|W_Q\|l_{\mathbb{G}_1}+\|W_Q\|l_{\mathbb{Z}_p}$	$\perp$
Ours	$5l_{\mathbb{G}}+3l_{\mathbb{Z}_p}$	$(1+2\|S\|)l_{\mathbb{G}}$	$(1+2\|U\|+2\|\mathcal{D}\|)l_{\mathbb{G}}+\|\mathcal{D}\|\lambda$	$(2+2\|S\|+2\|W_Q\|)l_{\mathbb{G}}+\|W_Q\|\lambda$	$\perp$

Based on the analytical expressions in Tables 2 and 3 and the operation costs and element sizes summarized in Table 4, we evaluate and compare the computational and storage overheads of several representative ABKS schemes, including the schemes [20,22,26], together with our proposed scheme. The comparison considers the five main phases of Setup , KeyGen, Encrypt, Trapdoor, and Search.

From the perspective of computation, Table 2 shows that the Setup phase of our scheme requires only a small number of exponentiation operations. In contrast, the setup cost of Huang et al.'s scheme [22] increases with the number of record keywords and the LSSS parameter. In the KeyGen and Encrypt phases, the computational costs of the compared schemes mainly depend on the numbers of attributes and keywords involved. Similarly, the Trapdoor cost increases with the number of user attributes and query leaves. For the Search phase, Huang et al.'s construction additionally depends on the number of negated keyword conditions, whereas the search cost of our scheme mainly depends on the attribute dimension and the number of query leaves. The operation costs listed in Table 4 provide numerical values for the exponentiation and pairing operations involved in these analytical expressions.

Table 4. Values of basic cryptographic operations and element sizes.

Notation	Description	Value
$E_{\mathbb{G}}$	Exponentiation in source group $\mathbb{G}$	6.162 ms
$E_{\mathbb{G}_1}$	Exponentiation in the first source group $\mathbb{G}_1$	1.494 ms
$E_{\mathbb{G}_2}$	Exponentiation in the second source group $\mathbb{G}_2$	10.227 ms
$E_{\mathbb{G}_T}$	Exponentiation in target group $\mathbb{G}_T$	0.458 ms
$E_{\mathbb{G}_{T,d}}$	Exponentiation in Type D target group $\mathbb{G}_{T,d}$	3.435 ms
t_P	Pairing operation $e(\mathbb{G}, \mathbb{G})$	3.642 ms
$t_{P,d}$	Pairing operation $e(\mathbb{G}_1, \mathbb{G}_2)$	11.335 ms
$l_{\mathbb{G}}$	Element size in source group $\mathbb{G}$	128 bytes
$l_{\mathbb{G}_1}$	Element size in the first source group $\mathbb{G}_1$	40 bytes
$l_{\mathbb{G}_2}$	Element size in the second source group $\mathbb{G}_2$	120 bytes
$l_{\mathbb{G}_T}$	Element size in target group $\mathbb{G}_T$	128 bytes
$l_{\mathbb{G}_{T,d}}$	Element size in Type D target group $\mathbb{G}_{T,d}$	120 bytes
$l_{\mathbb{Z}_p}$	Element size in scalar field $\mathbb{Z}_p$	20 bytes
$l_{\mathbb{Z}_p^*}$	Element size in $\mathbb{Z}_p^*$	20 bytes

Regarding storage overhead, Table 3 shows that the storage requirements of the compared schemes are mainly determined by the numbers of attributes, keywords, and cryptographic elements contained in each component. For our construction, the fixed-dictionary representation additionally introduces $|\mathcal{D}|\lambda$ bits of pseudorandom tags in the ciphertext and $|W_Q|\lambda$ bits in the search token. Nevertheless, the remaining storage cost grows linearly with the attribute dimension, dictionary size, and number of query leaves. The element sizes listed in Table 4 provide the corresponding numerical basis for evaluating the storage expressions in Table 3. Overall, the computation and storage analyses show that the proposed scheme maintains moderate overhead while supporting Boolean keyword retrieval with negation and attribute-based authorization.

7.3. Experiment Analysis

All schemes were implemented in Java with the JPBC library and tested on the same Windows 11 desktop platform to ensure a fair runtime comparison.

For the experiment shown in Figure 3, we keep the dictionary size at 50 and vary the attribute size from 10 to 50 in steps of 10. As the attribute size increases, the running times of KeyGen, Encrypt, and Trapdoor generally increase, while our scheme remains less sensitive to the attribute scale, especially in the Search phase due to its aggregation-based attribute matching mechanism.

Similarly, Figure 4 presents the influence of dictionary size on execution cost when the number of attributes remains 50. The dictionary size changes from 10 to 50 in steps of 10. KeyGen is unaffected because it contains no dictionary-dependent operation, whereas the other phases grow approximately linearly.

The observed performance differences are also related to the query-processing mechanisms of the compared implementations. Our scheme can represent multiple keywords and their Boolean relations in a single query tree and process them within one search procedure, whereas an equivalent multi-keyword request in the compared implementations may require separate keyword-query executions, resulting in accumulated search costs.

The reported implementation mainly measures the pairing-dominant algebraic core. Therefore, the results support the relative algebraic-efficiency comparison rather than a complete end-to-end clinical deployment evaluation.

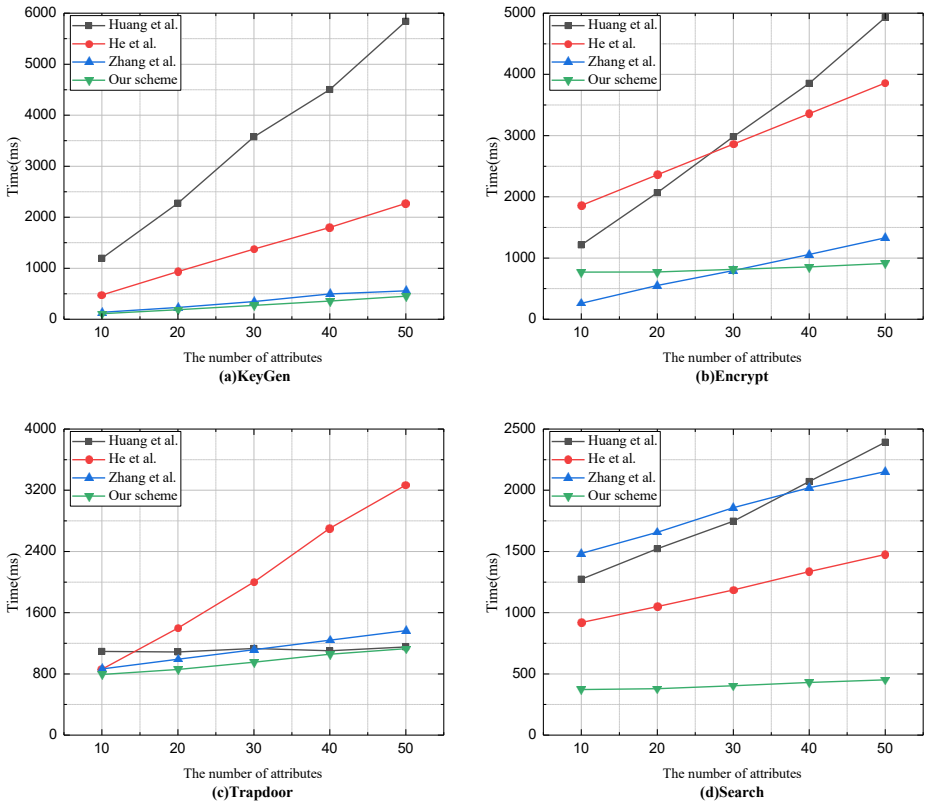


Figure 3. Runtime comparison under different attribute sizes for the proposed scheme and the schemes in [20,22,26].

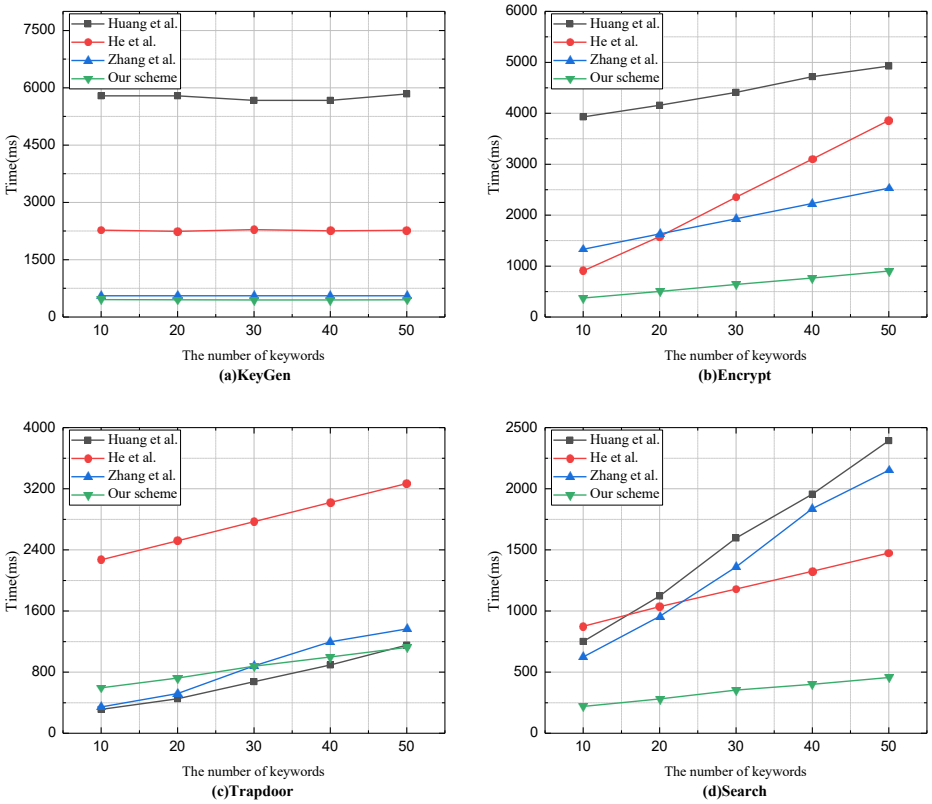


Figure 4. Runtime comparison under different keyword sizes for the proposed scheme and the schemes in [20,22,26].

The current experiments do not use real-world EHR datasets and mainly evaluate the computational overhead of the cryptographic operations in the proposed searchable-encryption layer. In particular, the evaluation does not include the costs of EHR text parsing and keyword extraction, network transmission between users and cloud servers, or encryption and transmission of the underlying medical data. In practical healthcare systems, additional engineering challenges also need to be considered, including integration with heterogeneous EHR formats and existing hospital information systems, management and update of the application dictionary, secure distribution and revocation of user credentials, and the communication and storage overhead introduced by large-scale encrypted medical records. These issues are beyond the scope of the current cryptographic evaluation and will be considered in future implementation-oriented work.

8. Conclusions

This paper presents a ciphertext-policy ABKS search layer for expressive retrieval over encrypted EHR collections. Encoding exactly one positive-or-negative literal for every term in a fixed application dictionary provides explicit keyword-absence semantics. Boolean formulas involving AND, OR, and NOT are converted into negation normal form, with NOT restricted to leaf nodes, and are then evaluated using a threshold tree. Multi-valued attribute authorization is enforced before the Boolean tree is accepted, while keyed pseudorandom encodings conceal literal and policy values from a non-colluding honest-but-curious cloud server. The security analysis establishes selective IND-sCP-CKA security with explicitly stated leakage under the DBDH and PRF assumptions. The computation and storage costs grow linearly with the attribute dimension, dictionary size, and number of query leaves. The principal limitations of the current construction are the use of a fixed dictionary, leakage of search and result patterns, reliance on a trusted credential issuer, and the exclusion of collusion between the cloud server and credentialed users or data owners. Future work will investigate incremental update mechanisms for dynamic dictionaries, so that dictionary terms and their pseudorandom tags can be updated without regenerating the entire searchable ciphertext. To mitigate access-pattern leakage, ORAM- or PIR-based techniques may be incorporated into the searchable-encryption layer. In addition, per-user or per-query blinded credentials, possibly combined with threshold-assisted token generation, will be explored to relax the current non-collusion assumption.

Author Contributions: Methodology, H.Y. and G.D.; Software, X.W., H.C. and G.D.; Formal analysis, L.Z.; Writing—original draft preparation, X.W., H.C. and G.D.; Writing—review and editing, B.C.; Supervision, H.Y. and L.Z.; Funding acquisition, B.C. and L.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Key R&D Projects in Henan Province, grant numbers 261111212500, 261111212000, and 241111212800.

Data Availability Statement: Data are contained within the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: Haicheng Chen is affiliated with Hangzhou Muyao Technology Co., Ltd.; Guangyu Ding is affiliated with Chery New Energy Automotive Co., Ltd.; and Binrong Cheng is affiliated with Hangzhou Zhongcheng Consulting Supervision Co., Ltd. These affiliations are disclosed for transparency. The authors declare that these organizations had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results. The authors declare no other conflicts of interest.

References

1. Song, D.X.; Wagner, D.A.; Perrig, A. Practical Techniques for Searches on Encrypted Data. In *Proceedings of the IEEE S&P*; IEEE: New York, NY, USA, 2000; pp. 44–55.
2. Boneh, D.; Crescenzo, G.D.; Ostrovsky, R.; Persiano, G. Public Key Encryption with Keyword Search. In *Proceedings of the EUROCRYPT*; Springer: Berlin/Heidelberg, Germany, 2004; Volume 3027, pp. 506–522.
3. Baek, J.; Safavi-Naini, R.; Susilo, W. Public Key Encryption with Keyword Search Revisited. In *Proceedings of the ICCSA*; Springer: Berlin/Heidelberg, Germany, 2008; Volume 5072, pp. 1249–1259.
4. Yu, Y.; Shi, J.; Li, H.; Li, Y.; Du, X.; Guizani, M. Key-Policy Attribute-Based Encryption with Keyword Search in Virtualized Environments. *IEEE J. Sel. Areas Commun.* **2020**, *38*, 1242–1251. [CrossRef]
5. Kamara, S.; Papamanthou, C.; Roeder, T. Dynamic searchable symmetric encryption. In *Proceedings of the ACM CCS*; ACM: New York, NY, USA, 2012; pp. 965–976.
6. Waters, B.; Balfanz, D.; Durfee, G.; Smetters, D. Building an Encrypted and Searchable Audit Log. In Proceedings of the NDSS, San Diego, CA, USA, 26 February–1 March 2004.
7. Abdalla, M.; Bellare, M.; Catalano, D.; Kiltz, E.; Kohno, T.; Lange, T.; Malone-Lee, J.; Neven, G.; Paillier, P.; Shi, H. Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions. In *Proceedings of the CRYPTO*; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3621, pp. 205–222.
8. Boneh, D.; Kushilevitz, E.; Ostrovsky, R.; William, E.S., III. Public Key Encryption That Allows PIR Queries. In *Proceedings of the CRYPTO*; Springer: Berlin/Heidelberg, Germany, 2007; Volume 4622, pp. 50–67.
9. Park, D.J.; Kim, K.; Lee, P.J. Public Key Encryption with Conjunctive Field Keyword Search. In *Proceedings of the WISA*; Springer: Berlin/Heidelberg, Germany, 2004; Volume 3325, pp. 73–86.
10. Wang, P.; Wang, H.; Pieprzyk, J. An Efficient Scheme of Common Secure Indices for Conjunctive Keyword-Based Retrieval on Encrypted Data. In *Proceedings of the WISA*; Springer: Berlin/Heidelberg, Germany, 2008; Volume 5379, pp. 145–159.
11. Boneh, D.; Waters, B. Conjunctive, Subset, and Range Queries on Encrypted Data. In *Proceedings of the TCC*; Springer: Berlin/Heidelberg, Germany, 2007; Volume 4392, pp. 535–554.
12. Lv, Z.; Hong, C.; Zhang, M.; Feng, D. Expressive and Secure Searchable Encryption in the Public Key Setting. In *Proceedings of the ISC*; Springer: Berlin/Heidelberg, Germany, 2014; Volume 8783, pp. 364–376.
13. Zheng, Q.; Xu, S.; Ateniese, G. VABKS: Verifiable attribute-based keyword search over outsourced encrypted data. In *Proceedings of the IEEE INFOCOM*; IEEE: New York, NY, USA, 2014; pp. 522–530.
14. Wang, H.; Dong, X.; Cao, Z. Multi-Value-Independent Ciphertext-Policy Attribute Based Encryption with Fast Keyword Search. *IEEE Trans. Serv. Comput.* **2020**, *13*, 1142–1151. [CrossRef]
15. Miao, Y.; Liu, X.; Choo, K.R.; Deng, R.H.; Li, J.; Li, H.; Ma, J. Privacy-Preserving Attribute-Based Keyword Search in Shared Multi-owner Setting. *IEEE Trans. Dependable Secur. Comput.* **2021**, *18*, 1080–1094. [CrossRef]
16. Xu, L.; Chen, X.; Zhang, F.; Li, W.; Wu, H.; Tang, S.; Xiang, Y. ASBKS: Towards Attribute Set Based Keyword Search Over Encrypted Personal Health Records. *IEEE Trans. Dependable Secur. Comput.* **2021**, *18*, 2941–2952. [CrossRef]
17. Sultan, N.H.; Kaaniche, N.; Laurent, M.; Barbhuiya, F.A. Authorized Keyword Search over Outsourced Encrypted Data in Cloud Environment. *IEEE Trans. Cloud Comput.* **2022**, *10*, 216–233. [CrossRef]
18. Miao, Y.; Ma, J.; Liu, X.; Li, X.; Liu, Z.; Li, H. Practical Attribute-Based Multi-Keyword Search Scheme in Mobile Crowdsourcing. *IEEE Internet Things J.* **2018**, *5*, 3008–3018. [CrossRef]
19. Yang, K.; Zhang, K.; Jia, X.; Hasan, M.A.; Shen, X.S. Privacy-preserving attribute-keyword based data publish-subscribe service on cloud platforms. *Inf. Sci.* **2017**, *387*, 116–131. [CrossRef]
20. He, K.; Guo, J.; Weng, J.; Weng, J.; Liu, J.K.; Yi, X. Attribute-Based Hybrid Boolean Keyword Search over Outsourced Encrypted Data. *IEEE Trans. Dependable Secur. Comput.* **2020**, *17*, 1207–1217. [CrossRef]
21. Chen, Y.; Li, W.; Gao, F.; Wen, Q.; Zhang, H.; Wang, H. Practical Attribute-Based Multi-Keyword Ranked Search Scheme in Cloud Computing. *IEEE Trans. Serv. Comput.* **2022**, *15*, 724–735. [CrossRef]
22. Huang, Q.; Yan, G.; Wei, Q. Attribute-Based Expressive and Ranked Keyword Search Over Encrypted Documents in Cloud Computing. *IEEE Trans. Serv. Comput.* **2023**, *16*, 957–968. [CrossRef]
23. Li, M.; Yu, S.; Zheng, Y.; Ren, K.; Lou, W. Scalable and Secure Sharing of Personal Health Records in Cloud Computing Using Attribute-Based Encryption. *IEEE Trans. Parallel Distrib. Syst.* **2013**, *24*, 131–143. [CrossRef]
24. Akinyele, J.A.; Pagano, M.W.; Green, M.D.; Lehmann, C.U.; Peterson, Z.N.J.; Rubin, A.D. Securing Electronic Medical Records Using Attribute-Based Encryption on Mobile Devices. In *Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*; ACM: New York, NY, USA, 2011; pp. 75–86. [CrossRef]

25. Khan, F.; Khan, S.; Tahir, S.; Ahmad, J.; Tahir, H.; Shah, S.A. Granular Data Access Control with a Patient-Centric Policy Update for Healthcare. *Sensors* **2021**, *21*, 3556. [CrossRef] [PubMed]
26. Zhang, K.; Wen, M.; Lu, R.; Chen, K. Multi-client sub-linear boolean keyword searching for encrypted cloud storage with owner-enforced authorization. *IEEE Trans. Dependable Secur. Comput.* **2020**, *18*, 2875–2887. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.