International Journal of Mathematics Trends and Technology
ISSN: 2231-5373/ https://doi.org/10.14445/22315373/IJMTT-V71I12P104

Volume 71 Issue 12, 19-39, December 2025

Original Article

ECSE: An Expressive Collaborative Searchable Encryption Scheme for Secure Group Data Sharing in Cloud Computing

Mishal Ismaeel[1], Chungen Xu[1,*], Lin Mei[1], GangQiang Duan[2]

[1]*School of Mathematics and Statistics, Nanjing University of Science and Technology.*
[2]*School of Cyber Science and Engineering, Nanjing University of Science and Technology.*

**Corresponding Author : xuchung@njust.edu.cn*

Received: 11 October 2025 Revised: 21 November 2025 Accepted: 13 December 2025 Published: 27 December 2025

Abstract - *With the growing usage of cloud services and multi-user collaboration, secure, privacy-preserving keyword searches are no longer optional. Conventional PEKS frameworks are susceptible to keyword-guessing attacks (KGA) and subversion attacks (SA), which become more likely when either the randomness-generating mechanism or key management fails. To address both of these risks, this work presents an Expressive Collaborative Searchable Encryption Scheme for Secure Group Data Sharing in Cloud Computing Environments (ECSE). The ECSE combines cryptographic reverse firewalls with threshold secret-sharing techniques and a collaborative approach to generating randomness, creating a robust, verified search framework. The ECSE can prevent SA attacks by employing trusted Cryptographic Randomness Generation, while it can fight KGAs by using a hybrid online/offline rate-limiting procedure with several key servers to create a ciphertext. The ECSE additionally supports Boolean search predicates, Linear Secret Sharing Schemes, and public access verification functionality to ensure that search results are reliable. Using a formal security analysis, this proposed scheme shows that the ECSE is semantically secure and resilient to semi-trusted cloud servers, key server compromise, and SA attacks. Finally, this work gives theoretical and experimental evaluations to support the conclusion that the ECSE improves security and adds functionality while maintaining acceptable efficiency. Experimental evaluation of a 2000-document dataset showed that the scheme can generate trapdoors for 15-keyword Boolean queries in less than 0.07 seconds and scales efficiently compared to TPPKS, PCSE, and TMS. Due to its Boolean expressiveness and resistance to subversion, ECSE enhances security with a slight performance overhead.*

Keywords - *Boolean Keyword Search, Cryptographic Reverse Firewall, Linear Secret Sharing, Subversion-resistant, Threshold Secret Sharing.*

1. Introduction

In the modern world, the ever-increasing amount of digital data has compelled the majority of businesses and individuals to store, manage, and analyse this data as quickly as possible while still keeping costs down. Unfortunately, there are serious hazards involved in keeping private information on cloud-based servers because all data, including private information, may be accessible to criminal hackers and/or server administrators. To limit the risk of hackers accessing private data, businesses have begun to utilise encryption to secure sensitive data stored in public clouds [1]. Encryption has emerged as a reliable method to secure critical data on untrusted servers. Searchable encryption (SE) [2] has been identified as a mechanism for searching encrypted material without first decrypting it.

When it comes to retrieving and analyzing huge amounts of encrypted data, traditional SE solutions are difficult to use [3]. Because the Ciphertext is made up of highly random pieces of data, collecting and interpreting it using traditional methods or techniques would waste a significant amount of time and resources [4]. The simplest way to execute searches on plaintext data would be to download and decrypt the ciphertext; however, this would place a significant computational and communication strain on the Accessors, limiting Ciphertext searches. One notable disadvantage of this method is that many modern systems lack the capacity to enable Keyword Searches on encrypted data. The difficulty that SE addresses is to provide searchable data on the Ciphertext by using Encrypting Trapdoors to strike a compromise between ease of use and secrecy. Existing SE research has mostly focused on enhancing security, efficiency, control over data access, and query expressivity [5].

Cloud-based group data sharing will be a more significant use case for SE as collaborative environments continue to expand. However, one barrier in providing secure and flexible access to numerous users is the distinction between access

control and user-level access control. Traditional SE implements access control for individual users (or "top-down" access control), whereas newer methods allow for fine-grained user access control via Attribute-Based Searchable Encryption (ABSE) techniques. ABSE allows individual users to acquire access to specific items in a search query by obtaining permission to view them, but it adds a significant degree of encryption/decryption overhead, making it unsuitable for lightweight and/or dynamic contexts. To ensure that sensitive data is not possibly leaked from a group interaction application, a minimum number of people must engage in a data request. Concentrating all confidence on one user would expose them to an unreasonably great danger. For example, a company's research and development team may have numerous authorised personnel who need access to sensitive information, rather than depending just on one authorised individual.

In addition, data must continue to be accessible during the absence of some members of the group (due to offline status) and in the presence of violence or threats. In order to reduce reliance on a single point of trust, threshold public key encryption (TPKES) [12, 13] and an enhanced version, TPKES with searchability [14], were suggested. TPKES allows for a fixed number of predetermined users to work together to decrypt the results of a common query. Unfortunately, many of the TPKES implementations do not offer the following benefits of correctness [15], anonymity [16], and they also provide a very limited capability to perform efficient searches using multiple keywords [17]. The intended improvement to the search options and user experiences for using a TPKES implementation can be achieved by providing a framework that includes the user using Boolean Keyword Search functionality. This new function allows the user to construct complex queries using the Boolean logical operators: AND, OR, and NOT. The advantages of this added functionality will provide end-users with improved fine-grained access to the encrypted data and more accurate retrieval of that data when compared to a TPKES that includes multi-keyword search, one or more keywords providing an improved solution for scalability and flexibility within Cloud computing applications that involve the sharing of multiple attributes simultaneously with multiple users. Combined, the proposed framework approach allows for the most practical and adaptable solution of TPKES implementations available today [18].

KGA is a major problem in frameworks such as SE and PEKS [19-21]. The majority of users select easily predictable keywords, and therefore, attackers can launch an offline brute force attack to extract the plaintext keyword based on the trapdoor. Introducing a keyword server that mediates online, rate-limited keyword generation is one method of blocking an attacker from creating checkpoints against the cipher in this fashion [22]. It limits an attacker's ability to conduct brute force keyword searches. However, if a keyword server is compromised, the SE scheme would be vulnerable again unless a threshold method was used to protect it.

Another major threat to frameworks is SA [23, 24]. SAs are fundamentally different than traditional systems that utilize CKA and KGA, as they exploit randomization weaknesses. Because of this, any system that utilizes a random number generator will be at risk from SA. Additionally, there are typically Random Number Generators that have poor randomization, which can create opportunities for attackers to launch a system attack on SE and PEKS. The Snowden documents revealed that through backdooring software/hardware, malicious suppliers could compromise the privacy of their customers [25, 26]. By embedding backdoors into Random Number Generators, backdoored encryption algorithms will intentionally choose random numbers that leak pieces of plaintext to adversaries [27]. Attackers can gain access to an SE or PEKS keyword by simply intercepting the blind hash of the keyword without proper randomization protection [18, 28]. Even though earlier research has looked into threshold decryption, attribute-based access control, and subversion-resistant PEKS, these methods are still separate and not fully developed. They either fail to provide unbiased randomization, do not allow Boolean LSSS-based predicates, or are unable to prevent off-line/online KGAs when keyword generation is compromised. Furthermore, present solutions do not integrate threshold signing, CRF sanitization, and expressive query evaluation into a single architecture. This leaves a big hole for a single system that can do keyword searches that are verifiable, expressive, and resistant to subversion, which is what you need for collaborative cloud environments.

Although earlier research has looked into threshold secret key sharing, multi-keyword searching, attribute-based access control, and swindle-safe PKSE, these various solutions tend to separate security and expressiveness concerns. As a result, present approaches offer poor protection against KGA attacks for compromised keyword creation, limited random generation, failure to support Boolean predicates, or a generic KGA assault on multi-satellite storage or hybrid cloud data centres. The same is true for subversion-safe structures, which primarily safeguard the integrity of random numbers while failing to account for collaborative keyword generation and threshold checking. The ECSE architecture, on the other hand, provides a single framework that combines threshold signings, cryptographic random functions, and Boolean LSSS-type predicates to create an expressive, auditable, and swindling-safe public keyword searchable encryption scheme that is a significant improvement over existing SE, PKSE, and SE systems.

Therefore, the goal of ECSE's design is to bridge the gap between the need for a searchable encryption scheme that enables users to conduct expressive queries on group cloud storage while offering strong security against KGA and Subversion Resistance, and practical efficiency without performance issues. The ECSE is a novel searchable encryption algorithm designed specifically for giving secure and verified access to group data held in cloud computing systems. The

ECSE design makes use of a variety of existing cryptographic primitives and mechanisms, such as threshold cryptographic schemes, KGA/SA resistance via collaborative keyword generation using reverse-firewall-assisted randomness verification, and KGA/SA-resistant verifiability and functionality, all of which contribute to an effective searchable encryption scheme. Unlike previous schemes, the ECSE does not claim to provide new or innovative methods of implementing the underlying mechanisms of "collaborative" keyword generation, KGA/SA resistant verifiability, or functionality using existing cryptographic tools, but rather combines and leverages these tools to build a secure and verifiable infrastructure to support group searching. The benefits of the ECSE-designed searchable cloud storage solutions are as follows:

1. This paper designs an ECSE scheme tailored for group data sharing that combines Cryptographic Reverse Firewall with threshold secret sharing and Distributed Key Generation (DKG), eliminating single points of failure and resisting algorithm-subversion attacks. This work introduces a server-assisted, threshold-signed keyword derivation mechanism that (i) prevents offline attacks on KGAs by binding ciphertext generation to online, rate-limited servers, and (ii) mitigates online attacks on KGAs through re-randomization and signature aggregation without exposing raw keywords.
2. This work supports Boolean LSSS expressed predicates over keywords and provides a public-key verification mechanism for server-side testing, enabling integrity verification of match results without revealing plaintext.
3. This work formally proves resistance to semi-trusted cloud servers, compromised key servers, and subversion attacks, relying on DDH and CDH hardness assumptions and functionality, security, exfiltration-resistance properties of CRFs, and the theoretical cost analysis and experimental evaluations demonstrating competitive online performance and efficient verification compared with prior schemes.

2. Related Works

In 2000, Song et al. [29] introduced searchable encryption as a significant step forward in the ongoing efforts to balance efficiency, functionality, and security for the retrieval of encrypted information. Since then, SE [21, 30] has become numerous and has received considerable attention with regard to its ability to provide confidentiality for messages and searchable encryption, which means that it allows search on encrypted data. Searchable encryptions are commonly grouped into categories according to their underlying cryptographic principles, namely, Public Key Encryption with Keyword Search [31, 32] and Symmetric Searchable Encryption [33]. SE has also progressed into new search patterns such as Boolean search [19], ranked search [34], Single keyword search [35], as well as enhancing security features such as Forward Security SE [36], Backward Security SE [36], and Access Pattern Privacy [37]. As a result of these recent advancements, the application of SE has expanded to cover a wide range of security-sensitive fields.

To enable multi-writer, multi-reader, and access control capability, ASE was created to enable the use of an asymmetric cryptographic key pair method of encrypting and decrypting during keyword search processes. This process involves encrypting keywords with the public key and using the private key for searching. The ASE model is especially useful in environments that require precise regulatory control over access rights of multiple users and multiple owners. As a result, the data was derived from a publicly issued encryption key. A similar method of using an encryption key and associating it with attributes to define a policy for searching for a keyword has also been developed using an ABSE scheme [32, 39-42]. Although these models provide users with greater flexibility, they tend to suffer from higher computational costs due to the inherent complexity of the underlying technologies. For this reason, these models retain the vulnerability to keyword guessing and guessing of terms associated with a keyword or the keyword itself when a keyword's protections are not rigorous enough and are not properly enforced regularly.

One of the most commonly used SE variants is PEKS. However, it is under constant threat from one type of attack: KGAs [43]. There are two kinds of entities that can execute KGAs: an internal attacker or an external attacker [44-47]. Although Rhee et al. [45] created the earliest countermeasures in the random oracle model for KGAs, additional research [19,51] [20,48] has shown that these early countermeasures generally depend on the use of a trusted server extensively, which creates significant overhead in terms of both communication and computation costs. Subsequent works have attempted to mitigate the threat posed by KGAs to the PEKS approach by introducing public-key authenticated encryption and improving both the methods of validating trapdoors and maintaining the original nature of the trapdoor [49,50], but have not yet fully addressed the issues associated with scalability and defence against internal attackers.

The parallel development of a second thread to the issue of server attack due to subversion resistance from the revelations made by Edward Snowden concerning the potential of backdoored cryptography, specifically the possibility of using compromised implementations of cryptographic systems by malicious actors. Bellare and his colleagues [51, 52] examined how subliminal channels created through the compromise of an implementation could result in exposing private cryptographic keys and/or sensitive information. To mitigate the risk associated with subliminal channel exposure, Zhou and colleagues [27] developed a PEKS scheme based on a CRF that was resilient to both KGAs and SAs; however, the requirement for the storage of a unique secret value for each keyword created significant storage capacity overhead. Subsequently, Jiang and colleagues [28] enhanced the designs of Zhou and colleagues by combining CRF sensitization techniques with collaborative random generation techniques to reduce reliance upon trust in a centralized authority while increasing resilience to subversion. Following these initial investigations of CRFs, further research studies have incorporated

CRF methodologies into various areas of cryptographic security, including two-factor authentication [53], firewall signatures [54], secure multiparty computation [55], and encrypted message transmission [56]. A wide range of methodologies for privacy-preserving data sharing through encryption have been proposed and improved upon in the past 15 years. For example, many methodologies rely on threshold methods, while others utilize an attribute-based framework. Still others, such as the "Subversion Resistant" PEKS methodologies developed post-2020, address the use of biased randomness when generating keywords. However, these methodologies do not address how to securely use threshold methods internally between the owner of the encrypted data and share with multiple users. Further, they also do not offer a mechanism for protecting against offline KGA or employing a secure method for creating multiple "users" within a shared decryption scheme. Finally, although many of the recent advances [59-60] have improved security and reduced reliance on centralised solutions, these solutions still rely on a single point of contact or cost-prohibitive options for those wanting flexible options for access between multiple parties. All of this casts additional light on the need for a unified methodology that combines Threshold Key Generation, CRF Randomization, and Expressive Boolean Keyword Search Techniques to enable secure and effective data sharing between groups.

2.1. Preliminaries

2.2. Bilinear Map and Hardness Problem

let G and G_T denote cyclic multiplicative groups of large prime order p. Let g and h be independently selected generators of group G. The finite field of order p can be represented by Z_p. The following qualities are said to be satisfied by bilinear mappings: $\hat{e} : G \times G \rightarrow G_T$

- Bilinearity: $\forall\, u, v \in G \text{ and } a, b \in Z_p$,

$$\hat{e}\left(u^a, v^b\right) = \hat{e}(u, v)^{ab}$$

- Non-degeneracy: There exist elements $u, v \in G$ such that $\hat{e}(u, v) \neq 1_{G_T}$.
- Computability: The bilinear map $\hat{e}$ can be computed efficiently for all $u, v \in G$.

1. Decisional Diffie–Hellman Assumption (DDH): Under this assumption, any probabilistic polynomial-time (PPT) adversary cannot computationally determine whether $g^z = g^{xy}$ or not it g^z is a random element in the group, given a generator g and elements g^x, g^y, g^z for randomly chosen $\left(x, y, z \in Z_p\right)$. This ensures the indistinguishability of the Diffie–Hellman key from random values and is fundamental to many cryptographic protocols, such as secure key exchange and encryption.
2. Decisional Bilinear Diffie–Hellman Assumption (DBDH): According to this assumption, given elements $g, g^x, g^y, g^z \in G$ and an element $Z \in G_T$, where $x, y, z \in Z_p$, no efficient adversary can distinguish whether $Z = \hat{e}(g, g)^{xyz}$ or not Z is a randomly chosen element from G_T. This assumption is very important for pairing-based cryptographic systems, like digital signature schemes and identity-based encryption.
3. Computational Diffie–Hellman Assumption (CDH): Given $g, g^x, g^y \in G$, for randomly selected $x, y \in Z_p$, Computing g^{xy}, It is computationally infeasible for any PPT adversary. Unlike the DDH assumption, which deals with distinguishing values, the CDH assumption is concerned with the impossibility of computing the shared secret itself. This assumption underpins the security of many traditional cryptographic constructions.

2.3. Cryptographic Reverse Firewall

According to [25], Stephens-Davidowitz and Mironov presented a CRF approach for randomising both received and outgoing communications. This unique method of randomising received and broadcast messages effectively protects cryptographic systems from potential security breaches due to backdoor access or compromised algorithms. Formally, assume that W and $P = (receive, next, output)$, denote the CRF and underlying party, respectively. The wrapper W acts as the CRF for the party P when provided with an initial public value σ and an incoming message m, such that the following conditions are met:

$$receive_{W \diamond P}(\sigma, m) = receive_P\left(\sigma, W(m)\right)$$

$$next_{W \diamond P}(\sigma) = W\left(next_P(\sigma)\right)$$

$$output_{W \diamond P}(\sigma) = output_P(\sigma)$$

A qualified CRF must fulfil the three properties:

- Functionality-Maintaining: Given$(k \geq 1)$, if the composed system $W^k \diamond P$ maintains the original functionality, then a wrapper W is said to retain functionality F for party P in protocolΠ. In particular, this implies that $W^1 \diamond P = W \diamond P$ and $W^{k'} \diamond P = W \diamond \left(W^{k'-1} \diamond P\right)) \forall\, k' \geq 2$. When$F$, Π, and P are comprehended from the context, W is designated as functionality-preserving.

- Weakly Security-Preserving: The system W weakly upholds the security guarantees S of the protocol Π for the party P, if the modified system $\Pi_{P\to W\diamond\overline{P}}$ fulfils S against any adversarial version of $\overline{P}$. Specifically, when P was substituted with $W \diamond \overline{P}$ the protocol continued to satisfy the security definition. When all parameters are comprehended, it offers weak security preservation.
- Weakly Exfiltration-Resistant: The CRF W mitigates data leakage by ensuring that the benefits of an adversary A in the exfiltration game (λ) are negligible in the security parameter λ.

2.4. Secret Sharing

The threshold secret sharing method employs algorithms for segmentation and reconstruction. In the segmentation process, the splitter partitions a secret into several fragments of information. Each data element is designated as a subkey, and every subkey is allocated to the participants. In the reconstruction step, the secret is retrieved by amalgamating the subkeys that satisfy or surpass the threshold. The following is the formal definition:

A (k, n) threshold secret sharing method divides the secret s into n segments of information, referred to as subkeys, which are possessed by individual group members. If the quantity of subkeys is less than or equal to k, reconstructing the original secret is unfeasible. If the subkeys are below the scheme's threshold, no information regarding the secret s can be derived when k represents the threshold. This presents a representative Shamir threshold secret sharing scheme:

The splitter selects a polynomial:

$$f(x) = a_0 + a_1 x + \cdots + a_{k-1} x^{k-1} \tag{1}$$

Such that $a_0 = s$. Each participant P_i randomly select x_i, and the subkey assigned to P_i is: $s_i = f(x_i)$, where $P_i \in P$, $P = \{P_1, \ldots, P_n\}$.

Given a subset $S \subseteq P$ with at least k elements, the original polynomial can be reconstructed using the Lagrange interpolation formula:

$$f(x) = \sum_{P_i \in S} \Delta_{x_i,S}(x) \cdot s_i \tag{2}$$

Where the Lagrange coefficient $\Delta_{x_i,S}(x)$ is defined as:

$$\Delta_{x_i,S}(x) = \prod_{\substack{P_j \in S \\ j \neq i}} \frac{x - x_j}{x_i - x_j} \tag{3}$$

Finally, the secret s is obtained by evaluating the reconstructed polynomial at $s = f(0)$.

2.5. Linear Secret Sharing Schemes

Our construction requires the use of linear secret-sharing strategies. This work first describes the access structure specification [57] that will be used in LSSS. The definition of LSSS presented here is adapted from [57].

Definition 1 (Access Structure). Let it $\{P_1, \ldots, P_m\}$ be a set of parties. An access structure $A \subseteq 2^{\{P_1,\ldots,P_m\}}$ defines which subsets are authorised to reconstruct the secret. If $B \in A, B \subseteq C$, and $C \in A$, then A is said to be monotone. In this context, the authorized subsets belong to A, while unauthorized subsets do not.

Definition 2 (Linear Secret-Sharing Schemes). Let $A = (\mathcal{A}, \rho)$ denote an LSSS, where $\mathcal{A}$ is a matrix over Z_p, and ρ maps each row of A to a participant in $\{1, \ldots, l\}$. For any subset $S \subseteq \{1, \ldots, l\}$, define the index set $I = \{i \mid \rho(i) \in S\}$. Suppose A_i represents the $i - th$ row of $\mathcal{A}$ matrix. For $S \in \mathcal{A}$, there exist coefficients. $\sigma_i \in Z_p$ such that

$$\sum_{i \in I} \sigma_i A_i = (1, 0, \ldots, 0) \tag{4}$$

The left-hand side is a linear combination of rows corresponding to parties in S, and the result reconstructs the secret (assumed to be the first element of the vector).

In this setting, the actual secret $s \in Z_p$ is embedded as the first component of the vector $v = (s, r_2, \ldots, r_m)$, where each r_i is chosen randomly. The share given to each participant is the product $\mathcal{A}_i v$, determined by the mapping $\rho(i)$. On the other hand, unauthorized sets cannot reconstruct the secret because no such set of coefficients σ_i exists that satisfies the above equation.

3. Problem Statement

3.1. System Model

The ECSE system model illustrated in Figure 1 consists of five main entities:

1. *Sender:* The data files and corresponding keywords are encrypted by the sender using the public key. The encrypted data is then uploaded to the cloud server for searchable access and secure storage.
2. *CS:* CS stores encrypted data sent from the sender. With its vast amount of capacity and processing power, CS enables recipients to search through this sort of storage by looking for keywords in the encrypted information. The information gathered from these searches will be provided to the intended recipient.
3. *Receiver*: Each receiver owns encrypted data files and performs keyword-based searches over the ciphertexts stored on the CS to retrieve the desired information.
4. *Cryptographic Reverse Firewall*: The CRF zone has two distinct components W_s and W_R, positioned between the user (either the sender or receiver) and the external environment. The CRF is responsible for sanitizing user outputs to prevent data leakage or subversion.
5. *Key Servers:* The key servers are responsible for generating and managing encrypted credentials or cryptographic materials for both the sender and receiver.

3.2. Definition of our Scheme

This proposed scheme consists of five algorithms:

1. *Setup:* Takes a security parameter λ as input and outputs the public system parameters PP, which serve as a common input for all subsequent algorithms in the scheme.
2. *SDBKGen:* Given the PP as input, this algorithm generates the user's secret key s_R and the corresponding public key V_R using a randomized procedure.
3. *CiphertextGen:* This algorithm takes as input the user's public key and the server-derived Boolean keywords W_i, and produces the ciphertext with the assistance of the CRF.
4. *TrapdoorGen:* The receiver executes this algorithm to create a trapdoor corresponding to the search predicate P.
5. *Test:* This algorithm enables CS to verify whether a trapdoor matches an encrypted keyword derived from the server. It outputs "1" if the predicate P is satisfied by the keyword set W_i, and "0" otherwise.
6. *Dec:* Given PP, the private key s_R, and ciphertextGen, this algorithm recovers the symmetric key k_m to decrypt the encrypted file $\widetilde{c_m}$.

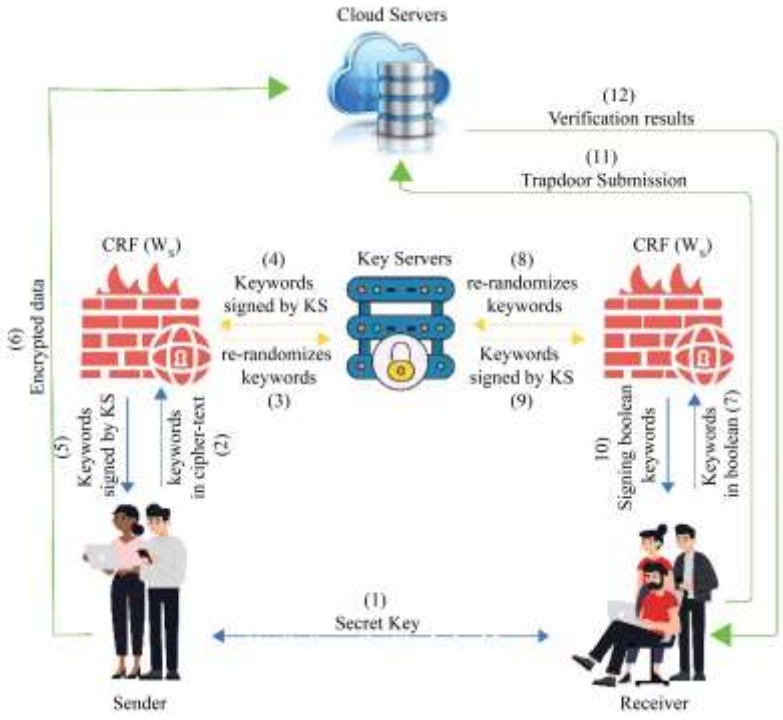

Fig. 1 System Model

3.3. Threat Model

In the ECSE scheme, this paper considers the following adversarial models and potential attack vectors that may compromise the confidentiality, integrity, or functionality of the encryption frameworks. Malicious key servers may attempt to corrupt the key generation process or manipulate the signing of server-derived keywords. However, the use of threshold secret sharing ensures that compromising a single key server does not expose the entire secret key, since key shares are distributed across multiple servers. Similarly, a compromised CRF may attempt to extract information from the generated trapdoors or alter the randomization process. The ECSE scheme employs CRF-based randomization to ensure that even if a CRF is compromised, the randomization remains secure and the adversary cannot get useful information. However, a malicious cloud server may attempt to modify search results or gain unauthorized access to stored data. However, the use of distributed keyword generation and collaborative query processing provides strong protection by restricting the server's capability to search outputs or manipulate stored ciphertext. Lastly, a more advanced adversary that can alter system implementation, like key shares or trapdoor generation procedures, is represented by subversion attacks. The combination of threshold-based secret sharing and cooperative activities across several servers helps to counter these kinds of attacks by making sure that no single compromised entity can compromise my proposed scheme's hidden security. While probabilistic cryptography is a popular technique for protecting algorithms, if an attacker uses the randomness generation maliciously, the PEKS strategy may covertly provide the adversary access to some sensitive data. This attack, which differs greatly from the traditional KGA and CKA, is known as SA. SAs are tough to identify since they are started by a stronger adversary, such as a computer manufacturer.

3.4. Design Goals

The design goals of the ECSE scheme are to achieve the following key objectives:

- *Confidentiality and privacy:* The scheme ensures that both data and queries remain confidential and private, even when stored or processed by a potentially untrusted cloud server.
- *Integrity and Authenticity:* The ECSE scheme guarantees that the search results and the keywords used during the search process are authentic and free from tampering.
- *Efficiency:* The cloud server achieves high search efficiency, while the computational and communication overhead for the sender, receiver, and CRF remains minimal.
- *Functionality:* The scheme supports secure data outsourcing and collaborative retrieval in group data-sharing environments.
- *Security:* The proposed scheme is resistant to malicious servers and compromised keys that may attempt off-line or online KGA, IND-CKA, or IND-KGA.

4. Concrete Construction

4.1. Overview of Proposed Scheme

The ECSE system presented in this article employs Distributed Key Generation (DKG), threshold secret sharing, server-side Boolean keyword calculation, and Collaborative Randomness Generation to produce a subversion-resistant ECSE that allows the KGA to be safely implemented. For example, to construct a secure, group-shared encrypted file that is hosted in the cloud (and thus vulnerable to external attacks), all keyword-serving servers must collaborate to run a DKG protocol and generate a globally available public key. Furthermore, there can be no reliance on a single reputable source. Each server receives a validated secret share that can be used for threshold signing. The employment of both CRFS and Boolean keywords prevents the option of off-line and/or internal KGAs, necessitating that both senders and recipients interact with the key-serving servers via CRFs to generate the Boolean keywords utilised in this ECSE system. Each user encrypts his or her own keyword, the CRF performs random remixes to minimise the chance of concealed leaking, and the key-serving servers provide publicly verifiable threshold signatures for keywords entered into the ECSE system. The combined signatures from the key-serving servers are run through a pseudorandom function to generate the final server-created Boolean keyword, removing any individual user's ability to create a searchable ciphertext on their own and preventing an attacker from exploiting any bias in the randomness used.

In practice, the CRF provides a fast, simple way for users to receive secure and untraceable communication from multiple key servers, while at the same time providing an easy way for users to create new blinded hash values for their keywords. During SDBKGen, the CRF Randomizes blinded hash values by applying new randomization operations to the original hash value. The CRF can only remove the randomization value when it has received sufficient responses from its key servers. Thus, even if one of the key servers is malicious, it will not be able to insert biased random numbers or provide subliminal messages into the hash. Additionally, since the CRF uses only public-key encryption for the communication process, it can be installed either as a local application or a stand-alone middleware application with minimal changes to existing systems and processes. The procedure for collaborative random generation consists of exchanging data between a Sender and a CRF. The Sender creates their random value as part of the process to create an encrypted message. After the CRF makes a contribution to the Sender's random value, the Sender opens their commitment. Ultimately, both parties have equal access to the final encryption random number, $s = a_{1i} + a_{2i}$, where a_{2i} is the random value provided by the Sender and a_{1i} is the random value provided by the CRF. The commitment verification process protects the randomness of the number from being

changed by either party prior to being used in encryption. Therefore, it is guaranteed that neither the Sender nor the CRF can alter the random number if one of them is compromised. The only requirements to implement this protocol are to have access to any cryptographic library with standard hash function support (for committing) and any pairing-based library (for performing group exponentiation).

Through the use of a sender-CRF Protocol that protects against subversion attacks, a jointly developed random number sequence is created and serves as a source of randomness for the random data to be used in constructing the ciphertexts of the encrypted file. From these randomly generated elements, the sender constructs a searchable ciphertext for the receiver and encrypts the symmetric file key using the receiver's public key. To formulate a query to perform search sessions, the ECSE system uses an LSSS to encode Boolean search terms so the search can be further expressed using logical mathematical predicates of "AND", "OR," and "NOT" as well as conjunctions and conjunctions of conjunctions of multiple logical terms. The receiver combines their private key with the Boolean search terms generated by the sending server into the trapdoor elements, using the Linear secret sharing scheme method. After the generation of trapdoor and ciphertextGen by the receiver, the cloud-based service provider executes a keyword search expression check against keyword condition elements by performing consistency checks based on pairings. The public verification equation is used to demonstrate that predicate conditions have been satisfied without revealing either the underlying plaintext data or the structure of the generated trapdoor. The receiver then retrieves the shared secret from ciphertextGen, obtains the symmetric key, and uses the symmetric key to decode the encrypted data. The ECSE uses threshold signatures, CRF-associated verification of sensitive facts, collaboratively constructed random numbers, Boolean LSSS-based predicates, and public verification of testing to offer a secure, expansive, and subversion-resistant means for searching on keywords. It mitigates the risks presented by semi-trustworthy cloud servers, rogue users, compromised key servers, and substitution attacks on associated algorithms.

4.2. Concrete Details

Setup Phase: Firstly, the system is started with a λ, along with a set of system parameters. These contain a cyclic multiplicative group G of prime order q with generator P, and a multiplicative target group G_T also of order q. A bilinear map is defined as $e: G \times G \to G_T$, $h: G \to Z_q$, $h': Z_q^* \times Z_q^* \to Z_q^*$, $H_1: \{0,1\}^* \to G$, $H_2: G_T \to \{0,1\}^{\log q}$ A Collision-resistant hash function is assumed, along with a pseudorandom function PRF: $Z_q \times \{0,1\}^* \to \{0,1\}^*$. Suppose n is the total number of key servers and the t is a threshold where $0 < t \leq n$.

Given the PP, all $\mathcal{KS}_i$ collaboratively perform the distributed key generation procedure as outlined in Algorithm 1. This involves applying a secret sharing approach for deriving the secret key s_{KS_i}, which is split among all participating key servers. Each key server $\mathcal{KS}_i$ $i \in \{1, 2, \ldots, n\}$ holds a portion of s_i, the associated V_i, and V_{KS}.

Independently, each group member generates their own key pairs (s_R, V_R). The private key s_R is selected at random from Z_q^*, and the corresponding public key is computed as $V_R = P^{s_R}$.

Algorithm 1: Distributed Key Generation

Require $1^{\lambda,n,t_1}$

Ensure s_i, V_i, V_{KS}

1. Firstly, it $\mathcal{KS}_i (i \in [1,n])$ generates its own secret by randomly selecting values $p_{i,0} \in_R Z_p^*$. It then constructs a private polynomial of degree $t_1 - 1$, defined as $p_i(x) = p_{i,0} + p_{i,1}x + \cdots + p_{i,t_1-1}x^{t_1-1}$ where the other coefficients are also chosen at random.
2. Next, the server publishes a set of public commitments to its polynomial. $\mathcal{KS}_i$ computes the group element $P^{p_{i,k}}$, for every coefficient $k \in [0, t_1 - 1]$, where P is a public generator of a cyclic group G, and these values are broadcast to all other keyword servers.
3. To distribute secret shares, $\mathcal{KS}_i$ confidentially send the evaluated values of its polynomial $p_i(j)$ to each other server $\mathcal{KS}_i$ $(j \in [1,n]$, and $j \neq i)$.
4. Upon receiving shares, $\mathcal{KS}_i$ must verify their consistency with the published commitments. For a share $p_i(j)$ received from $\mathcal{KS}_i$, it checks the verification equation:
$$P^{ik \cdot p_j(i)} \stackrel{?}{=} \prod_{k=0}^{t_1-1} (P^{p_{j,k}})^{i^k} \quad (5)$$
If this equation does not hold, the share $p_{j,k}$ is considered invalid and is rejected.
5. After successful verification, it $\mathcal{KS}_i$ calculates its final secret share $s_i = \sum_{k=1}^{n} P^{p_k(i)}$, as the share of secret s_{KS_i} and then computes its corresponding public share $V_i = P^{s_i}$, respectively.
6. Finally, it $\mathcal{KS}_i$ calculates the overall public key, which is the product of constant terms from all participants: $V_{KS} = \prod_{k=1}^{n} P^{p_{k,0}}$. The server secretly stores its secret shares s_i, and maintains the list of public values $\{V_{KS}, V_1, \cdots, V_n\}$ and deletes the intermediate values from the memory.

SDBKGen: For all keywords in the Boolean search query, S or R will generate a server-derived Boolean keyword that is securely created in collaboration with the key servers. Here is the process for generating multiple server-derived keywords for the Boolean search:

- The sender or receiver chooses a random value $c_i \in Z_q^*$ and computes $W_i = H_1(w_i)^{c_i}$ and sends it to CRF.
- CRF ($W_S or W_R$) chooses a random value $\alpha_i \in Z_q^*$ for each keyword, re-randomizes to $W_i' = (W_i)^{\alpha_i P}$, and then sends W_i' to the key servers.
- Each key server $\mathcal{KS}_i (i \in [1, n])$ signs W_i' using its secret share s_i, generating the signature $\sigma_i' = W_i'^{s_i}$, and sends it σ_i' to the CRF.
- CRF removes the randomization and sends it to the sender by calculating, $\sigma_i = (\sigma_i')^{\alpha_i P^{-1}}$ where $(\alpha_i P)(\alpha_i P^{-1}) = 1 mod q$. Note that $\sigma_i = (W_i)^{s_i}$ here represent the signatures of W_i by $\mathcal{KS}_i$.
- Upon obtaining $\sigma_i (i \in [1, n])$ The sender or receiver checks and verifies its authenticity by performing the following computation:

$$e(\sigma_i, g) \stackrel{?}{=} e(W_i, V_i) \tag{6}$$

If t_1 Signatures pass the checking, S or R compute the aggregated signature:

$$\sigma_w = \left(\prod_{k=1}^{t} \hat{\sigma}_k \right)^{c_i^{-1}} \tag{7}$$

The aggregated signature

$$e(\sigma_w, P) = e(H_1(\boldsymbol{w_i}), V_{\mathcal{KS}}) \tag{8}$$

It is verified using the valid signature produced by Boolean keyword servers; otherwise, the server-derived Boolean keyword generation process on the server side fails.

Finally, the sender or receiver derived the server-derived Boolean keyword W_i as follows:

$$W_i = \mathrm{PRF}(H_1(\sigma_w), w_i)$$

This process is repeated for all keywords $\{w_1, w_2, \ldots, w_k\}$.

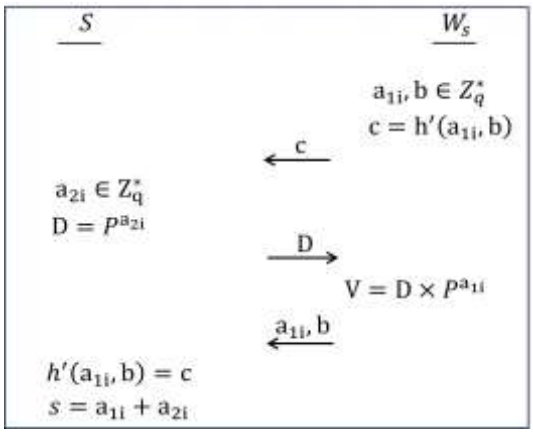

Fig. 2 Distributed Randomness Generation Protocols.

CiphertextGen: Sender S communicates with CRF W_s to produce random values s through the execution of a distributed randomness generation protocol, Figure 2:

- W_s randomly chooses $a_{1i}, b \in Z_q^*$, computes $c = h'(a_{1i}, b)$, and sends c to S to commit randomness a_{1i}.
- The sender selects $a_{2i} \in Z_q^*$ at random and sends $D = P^{a_{2i}} \in G$, and sends D to W_s for all keywords ω_i.
- W_s transmits (a_{1i}, b) to the sender S to reveal the commitment associated with a_{1i}.
- W_s calculates $V = D \times P^{a_{1i}} \in G$.
- S verifies whether $h'(a_{1i}, b) = c$; if the check passes, S accepts a_{1i} and computes $s = a_{1i} + a_{2i}$ and outputs it; otherwise, S outputs $\perp$.

Compute:

$$A = P^s, \qquad B = V_R^s, \qquad C_i = f_i^s \;\; f_i = W_i$$

The sender S computes a random symmetric key $k_m \in \{0,1\}^\lambda$ that will be used to encrypt the actual data. To securely transmit the symmetric key to the authorized receiver, the sender computes a shared secret $k_{shared} = e(V_R, P)^s$ and encrypts the symmetric key as $k_{enc} = H_1(k_{shared}) \oplus k_m$. The actual file M is then encrypted using symmetric encryption to produce $\widetilde{c_m} = Enc_{sym}(k_m, M)$. The final ciphertextGen comprises the searchable components $\{A, B, C_i, k_{enc}, \widetilde{c_m}\}$.

TrapdoorGen: The receiver takes the input parameter as pk, $sk = s_R$, and searches predicate P.

The trapdoor is generated using the LSSS for the search predicate P, represented by a $l \times m$ matrix A (e.g, $(w_1 \wedge w_2) \vee \neg w_3$), mapping rows of A to keywords. A vector $v \in Z_q^m$ is generated such that $1 \cdot v = s_R$. For each row A_i in A, a random value $r_i \in Z_q^*$ is selected, and trapdoor components are computed:

$$D_{1,i} = P^{A_i \cdot v} \cdot W_{\rho(i)}^{r_i}, \quad D_{2,i} = P^{r_i}, \quad D_{3,i} = W_{\rho(i)}^{r_i}$$

Test Algorithm: By receiving the ciphertextGen and trapdoorGen: $C_T = \{A, B, C_i\}^s$,

$T_P = \{D_{1,i}, D_{2,i}, D_{3,i}\}_{i=1}^{l}$, the cloud server computes the index set I for the predicate P based on A and ρ. For each row A_i:

- If the keyword condition is not negated, compute:

$$U_i = \frac{e\left(D_{1,i},\ A\right)}{e\left(D_{2,i},\ C_{\rho(i)}\right)} \tag{9}$$

- If the keyword is negated, compute:

$$U_i = \frac{e\left(D_{1,i},\ A\right)}{e\left(D_{3,i},\ A\right)} \tag{10}$$

These equations are used to match the keywords. After matching the keyword cloud server, verify the validations of these keywords.

Verify: Then, verify the following to match the matching keywords.

$$\prod_{i \in I} U_i^{\sigma_i} = e(P, P)^{s \cdot s_R} \tag{11}$$

that σ_i are scalars and satisfy $\sum_{i \in I} \sigma_i A_i = (1, 0, \ldots, 0)$. Return 1 if the predicate matches; otherwise, return 0.

Dec: The receiver begins by computing the shared secret $k_{shared} = e(A, P)^{s_R}$, which leverages their private key s_R and the ciphertextGen component $A = P^s$ and $e(P, P)^{s \cdot s_R}$, that was used by the sender during encryption. Using this shared secret, the receiver then recovers the symmetric key by computing $k_m = H_1(k_{shared}) \oplus k_{enc}$. Once the symmetric key is successfully recovered, the receiver decrypts the data file M by computing $M = Dec_{sym}(k_m, \widetilde{c_m})$, thus obtaining the original plaintext data.

5. Security Analysis

5.1. Correctness

Case 1: The keyword is not negated:

$$U_i = \frac{e\left(D_{1,i}, A\right)}{e\left(D_{2,i}, C_{\rho(i)}\right)}$$

$$e\left(D_{1,i}, A\right) = e\left(P^{A_i \cdot v} \cdot W_{\rho(i)}^{r_i}, P^s\right)$$

$$e\left(D_{2,i}, C_{\rho(i)}\right) = e\left(P^{r_i}, W_{\rho(i)}^{r_i}\right) = e\left(P, W_{\rho(i)}\right)^{r_i \cdot s}$$

$$U_i = e(P,P)^{s(A_i.v)} \tag{12}$$

Case 2: The Keyword is negated:

$$U_i = \frac{e\left(D_{1,i}, A\right)}{e\left(D_{3,i}, A\right)}$$

$$e\left(D_{1,i}, A\right) = e\left(P^{A_i \cdot v} \cdot W_{\rho(i)}^{r_i}, P^s\right)$$

$$e\left(D_{3,i}, A\right) = e\left(W_{\rho(i)}, P\right)^{r_i \cdot s}$$

$$U_i = e(P,P)^{s(A_i.v)} \tag{13}$$

Verify by predicate:

$$\prod_{i \in I} U_i^{\sigma_i} = e(P,P)^{s \cdot s_R}$$

as

$$\sum_{i \in I} \sigma_i A_i = (1, 0, \ldots, 0)$$

$$\prod_{i \in I} U_i^{\sigma_i} = e(P,P)^{\sigma_i \cdot (A_i.v) s}$$

simplify using $\sum_{i \in I} \sigma_i A_i = (1, 0, \ldots, 0)$

$$\prod_{i \in I} U_i^{\sigma_i} = e(P,P)^{s \cdot s_R}$$

5.2. Resistance against Semi-Trusted CS

According to the threat model, a semi-trusted cloud server could conduct an offline/online KGA to compromise the confidentiality of keywords. This paper discusses the level of security offered by the ECSE method against this form of attack. It shows how this method can protect against the threat described above by leveraging the concept of Semantic Security against Chosen-Keyword Guessing Attacks (SS-CKGA) [12]. When evaluating subversion attacks, this proposed work assumes that an adversary, such as a semi-trusted cloud server, takes advantage of the randomized components of the ciphertext (ciphertextGen) within the SS-CKGA security game. In other words, each instance of manipulation may involve altering the algorithms, resulting in the use of maliciously implemented user components. The name for the modified algorithm-based game is $Game - CKGA$. The formal definition and understanding of the concept of SS-CKGA are detailed below.

Definition 1. AN ECSE scheme will be called SS-CKGA secure if the adversary A, a PPT adversary, has no non-negligible advantage in playing $Game - CKGA$ security experiments against a challenger C is considered SS-CKGA secure if the PPT adversary has only negligible advantages in the game $Game - CKGA$ security experiment against a challenger C.

Setup. C begins by executing the setup algorithm to generate both the system parameters and the key material. The challenger will keep the secret keys and give the public key (V_R, V_{KS}) to the A. At this stage, A will provide modified versions of the algorithms for generating SDBKGen W_i^* and ciphertextGen C_T^*, and back to the challenger C.

Phase 1. A has the ability to adaptively query the C to obtain either the trapdoorGen or the ciphertextGen for any SDBKGen W_i.

Challenge. Two distinct keywords w_0 and w_1 are chosen by A, that were not used in Phase 1, and are submitted to the C. The C will indiscriminately choose a bit $b \in \{0,1\}$ and computes the secret key material W_{iw_b}, c_{iw_b}, and T_{Pw_b} using the submitted server-side algorithm.

Phase 2. A is allowed to continue the querying for the trapdoorGen or ciphertextGen of any keywords from SDBKGen, excluding w_0 and w_1.

Output. At the end of the experiment, A outputs a bit $b^* \in \{0,1\}$. If $b = b^*$, A is considered to win the game. A in this game is formally defined as:

$$Adv_{A,ciphertextGen}^{SS-CKGA} = \left| Pr(b = b^*) - \frac{1}{2} \right| \tag{14}$$

Throrem 1: ECSE provides SS-CKGA security assuming that the underlying server-aided PEKS is itself SS-CKGA secure and CRF fulfils the conditions of being functionality-maintaining, weakly security-preserving, and weakly exfiltration-resistant.

Proof: The proof for this statement follows the same line of reasoning as SS-CKGA's security proof per section VII-A of reference [28].

Definition 2. ECSE is said to achieve IND-CKG security if no PPT, A can win the $Game - CKA$ experiment with more than negligible advantages. The security game is conducted between an A and C, and proceeds as follows:

Setup: A selects a set of keyword queries W_i^*, which may include Boolean expressions over SDBKGen. Based on these, A submits the index set $I_{W_i^*}$ to C. In response, C produces and returns a modified key distribution algorithm. This includes the components $SDBKGen^*$, $ciphertextGen^*$, and $TrapdoorGen^*$.

Query 1: A queries C for the TrapdoorGen T_{Pw*} corresponding to the selected set of SDBKGen W_i^*. A also queries the index $I_{W_i^*}$ to check if the keyword set W_i^* is contained in the SDBKGen list.

Challenge: A selects a challenge set $W_i \subseteq \{1, \ldots, m^t\}$ which is a subset of Boolean server-derived keywords. A also selects a value $\sigma \in V$, where no previous trapdoorGen queries have been made, distinguishing between challenges set $Rand(W, V)$ and $Rand(W, V \setminus \{\sigma\})$. A outputs the challenge pair (W, σ) to C, then C indiscriminately chooses a bit $b \in \{0,1\}$. For the case $b = 0$, C generates the challenges set $I_{W_i^*}$ using $Rand(W, V \setminus \{\sigma\})$, applying the algorithms $SDBKGen^*$, $ciphertextGen^*$, and $TrapdoorGen^*$. If $b = 1$, C uses the full set of $Rand(W, V)$ to generate challenges.

Query 2: After receiving the challenge, A may continue querying the trapdoorGen for other keyword sets from C, but A cannot query the trapdoor; they would distinguish between W_0 and W_1, which could reveal the value of b.

Output: At the conclusion of the experiment, A returns a guess $b' \in \{0,1\}$. If $b' = b$, A is considered to win the game.

The adversaries of winning IND-CKA games are given by:

$$Adv_A^{IND-CKA} = Pr[b' = b] - \frac{1}{2} \tag{15}$$

Theorem 2: ECSE achieves IND-CKA security under the DDH assumption, provided that the DKG protocol is secure and the CRF, and satisfies functionality-maintaining, provides weak security-preserving, and weakly exfiltration-resistant, assuming the underlying scheme is IND-CKA secure.

Proof: Except for the CRF, the rest of the security argument directly follows the IND-CKA proof presented in Section 6.1 of [12]. To complete the argument, it must also be shown that the underlying ECSE constructions maintain functionality, ensuring adequate security, and exhibit resistance to data leakage. Notably, randomness generation, as suggested in [28], is used to support zero-knowledge properties and bias-resistance $s = a_{1i} + a_{2i}$ characteristics. Zero-knowledge ensures that CRF remains hidden from n during execution of the protocols, while bias-resistance guarantees ensure that no adversary can influence the selection of random values, such as n. To demonstrate that ECSE satisfies these three requirements, proceed as follows.

Functionality Maintenance: In SDBKGen, the CRF modifies the component w_i by re-randomizing it using public key operations. Specifically, it computes $W_i' = (W_i)^{\alpha_i P}$, after receiving the signature α_i' or W_i' from the key server $\mathcal{KS}_i$ for $(i \in [1,n])$, the signature α_i' is computed as

$$\alpha_i = (\alpha_i')^{\alpha_i P^{-1}} = \left(W_i'^{s_i}\right)^{\alpha_i P^{-1}} = \left(W_i^{\alpha_i P}\right)^{s_i \alpha_i P^{-1}} = W_i^{s_i}$$

The functionality of the ECSE scheme remains unchanged after integrating the CRFs, and α_i remains a valid signature on W_i from the key server $\mathcal{KS}_i$.

Furthermore, W_s and the sender works together to establish shared randomness creation, seen in Figure 2. According to the Zero-knowledge property of the randomness-generating protocol, the server's view W_s does not leak information and is indistinguishable from the uniformly random values in Z_p^*, thus preserving randomness confidentiality.

Weakly Security Preservation: To prove this, it suffices to demonstrate the IND-CKA security of the scheme ECSE remains intact when CRF is applied. This is done by showing that the adversary cannot distinguish between executions of $Game - CKA$ and the IND-CKA game under $SDBKGen^*$, and $CiphertextGen^*$, and also ensures the indistinguishability of the challenge sets and trapdoorGen queries.

Game 0: This represents the standard $Game - CKA$ security game, in which the adversary interacts with the system by submitting trapdoor queries and issuing challenge sets.

Game 1: This game is derived from *Game 0* by substituting the modified algorithm $SDBKGen^*$ in place of the original $SDBKGen$ used in the security game.

Game 2: Building upon *Game 1*, this version replaces $CiphertextGen^*$with the original $CiphertextGen$ algorithm within the security game.

Game 0 ≈ *Game 1*: To create a re-randomized element W_i', the CRF will process the potentially biased element W_i in the modified algorithm $SDBKGen^*$. The randomization property of the element ensures that W_i' is consistent with the randomness generated by $SdkGen's$. There is no observable difference between *Game 0* and *Game 1*.

Game 1 ≈ *Game 2*: In the modified algorithm $ciphertextGen^*$, the CRF (W_s) and the sender jointly generates the random number s utilized in the encryption of a server-derived boolean phrase following the protocol shown in Figure 2. The protocol's bias-resistance characteristic ensures that s is as consistently random as the one produced by $CiphertextGen$. As a result, there is no observable difference between *Game 1* and *Game 2*.

Weak Exfiltration Resistance: Because $Game - CKA$ and IND-CKA are equivalent in structure, an adversary cannot achieve a non-negligible advantage in the $Game - LEAK$ scenario, as established in [28]. This implies that the CRF provides only minimal protection against exfiltration.

5.3. Resistance against Compromised Keyword Servers

When generating the SDBKGen, there is a risk that compromised keyword servers could attempt to recover keywords through CKA. To address this, the system leverages the IND-CKA security notion, which is indistinguishable under such attacks. This security model provides the necessary protection and is formally defined in [38].

Definition 3: ECSE is said to be IND-CKA secure if no PPT A succeeds in the $Game - CKA$ experiment with a non-negligible advantage:

Initialization: A selects and submits a tampered version of the algorithm, denoted as $SDBKGen^*$, to C.

Setup: C generates the key pairs, public and private, for the keyword server, represented as $(V_{\mathcal{KS}_i}, s_{\mathcal{KS}_i})$, and shares them with A.

Challenge: A sends two distinct keywords w_0 and w_1 and submits them to C. C randomly picks a random bit $b \in \{0,1\}$, then runs the tampered algorithm $SdkbGen^*$ with the selected keyword w_b as input.

Output: A returns a guess $b^* \in \{0,1\}$. If $b^* = b$, then A is said to win the game.

The advantage of the adversary A in this game is defined as:

$$Adv_A^{IND-CKA} = \left|\Pr[b^* = b] - \frac{1}{2}\right| \tag{16}$$

Theorem 3: The proposed ECSE scheme is IND-CKA secure if no A, operating within PPT can distinguish between two encrypted keyword sets with a non-negligible advantage, as defined by the preceding security game.

Proof: This proof is based on the same approach used for the IND-CKA security proof in Section VII-A of [28].

5.4. Resistance against Subversion Attacks

Definition 4: ECSE scheme resists against SA if no $PPT\ A$ can differentiate between the following two universes:

- An honest implementation of SDBKGen is used by a user to communicate with $\mathcal{KS}_i$.
- A maliciously implemented $SDBKGen^*$ is used by a user to communicate with $\mathcal{KS}_i$.

If the ECSE scheme satisfies the aforementioned criterion of security, it will not give A any keyword information.

Theorem 4: This ECSE scheme can withstand the SA.

Proof: This proof is based on the same approach used for the IND-CKA security proof in Section VII-A of [28].

6. Performance Evaluation

In this section, this scheme provides a comprehensive analysis of the performance of the ECSE scheme. The work is distinguished by both theoretical examination and practical testing. In addition, the efficiency of ECSE will be compared to that of already available schemes such as TPPKS [15], TMS [58], PCSE [18], and ECSE.

6.1. Theoretical Evaluations

Table 1. Theoretical Computation Costs

Schemes	TPPKS [15]	TMS [58]	PCSE [18]	ECSE
KeyGen	$n(H_G + P + 2E_G + E_{GT})$	$n(n+3)E_G$	$n(H_G + 2E_G) + 2\eta(t_1+1)E_G$	$\eta t_1 + n(n-1)t_1 n + n(n-t)t_1 M_G + n(n-1)A_{Z_{q^*}} + nt_1 R_{Z_{q^*}}$
Enc	$(3+\tilde{m}) + E_G + P + E_{G_T}$	$H_g + (6+\tilde{m}+2t)E_G + E_{G_T}$	$\tilde{m}H_G + 2(t_1+1)P + [2\eta + t_1 + 4)E_G + t(P + E_{G_T}) + H_G + 16E_G$	$2(2+2\tilde{m})E_G + 2\tilde{m}H_{Z_{q^*}} + 3\tilde{m}R_{Z_{q^*}} + \tilde{m}A_{Z_{q^*}}$
TrapGen	$(l+1)t\,(H_G + 2P + 2E_G + E_{G_T})$	$(2+2t_0)E_G$	$(2+2t_0)E_G + l[H_G + 2(t_1+1)P + (2\eta + t_1 + 3)E_G]$	$3l_0E_G + l_0M_G + (l_0+m)R_{Z_{q^*}}$
Search	$2lE_G$	$(3+2t)P$	P	$2(l_1+l_2)P + \lvert I\rvert M_{G_T}$
Verify	–	$\lvert\phi\rvert(H_G + 2E_G) + 2P + E_G$	$\lvert\phi\rvert(H_G + 2E_G) + 2P + E_G$	$\lvert I\rvert E_{G_T} + (\lvert I\rvert - 1)M_{G_T} + IP$
Dec	$t(2P + 8E_{G_T}) + P$	tE_{G_T}	$3P + 2E_G$	$1P + E_{G_T} + 1H_G$

n: Number of group members, η: Number of keyword servers, $\tilde{m}$: Number of keywords in keyword set W, t_0: Threshold value of searchers, t_1: Threshold value of keyword servers, t: Number of decryptors, l: Number of boolean queried keywords, ϕ: Number of returned results, M_G: Multiplication in G, l_0: Number of rows in LSSS, m: Number of columns in LSSS, $\lvert Z_q^*\rvert$: Size of field element, $R_{Z_q^*}$: Random number generation in Z_{q^*}, $A_{Z_q^*}$: Addition in Z_q^*.

For the computation cost, this proposed scheme considers various time-consuming operations involved in the process. Regarding the storage cost, we denote the element lengths in Z_{p^*}, G, and G_T as $\lvert Z_{p^*}\rvert$, $\lvert G\rvert$, and $\lvert G_T\rvert$, respectively. Tables 2 and 3 summarize the computational and storage overhead for each phase in terms of dominant cryptographic operations.

Table 2. Theoretical Storage Costs

Schemes	TPPKS [15]	TMS [58]	PCSE [18]	ECSE
KeyGen	$n(3\lvert Z_p\rvert + 2[G] + \lvert G_T\rvert)$	$3n\lvert Z_p\rvert + n(n+2)\lvert G\rvert$	$(3n+\eta)\lvert Z_p\rvert + [2n+\eta(\eta+1)]\lvert G\rvert$	$(t_1(n-1))\lvert Z_{q^*}\rvert + t(n-1)\lvert G\rvert$
Enc	$(3+\tilde{m})\lvert G\rvert$	$(5+2\tilde{m}+t)\lvert G\rvert + \lvert G_T\rvert$	$\tilde{m}\{3\lvert Z_p\rvert + (2t_1+3)\lvert G\rvert + \lvert G_T\rvert] + 4(8\lvert Z_p\rvert + 3\lvert G\rvert) + (2\widetilde{m_1}+2t+5)\lvert Z_p\rvert + 6\lvert G\rvert + \lvert G_T\rvert$	$(2+\tilde{m})\lvert G\rvert$
TrapGen	$(l+1)\lvert Z_p\rvert + 2(l+1)t_0\lvert G_T\rvert$	$2\lvert G\rvert$	$2\lvert G\rvert + l[3\lvert Z_p\rvert + (2t_1+3)\lvert G\rvert + \lvert G_T\rvert]$	$l_0(m+1)\lvert Z_q^*\rvert + 3l\lvert G\rvert$
Search	$(2+l)\lvert G\rvert$	$(3+t)\lvert G_T\rvert$	$3\lvert G_T\rvert$	$l\lvert G_T\rvert$
Verify	–	$(\lvert\phi\rvert+1)\lvert Z_p\rvert + \lvert G\rvert + \lvert G_T\rvert$	$(\lvert\phi\rvert+1)\lvert Z_p\rvert + \lvert G\rvert + \lvert G_T\rvert$	$l\lvert G_T\rvert$
Dec	$t(5\lvert G_T\rvert + \lvert Z_p\rvert) + \lvert G_T\rvert + \lvert G\rvert$	$2\lvert G_T\rvert$	$4\lvert Z_p\rvert + 2\lvert G_T\rvert$	$2\lvert G\rvert + \lvert Z_q^*\rvert$

n: Number of group members, η: Number of keyword servers, $\tilde{m}$: Number of keywords in keyword set W, t_0: Threshold value of searchers, t_1: Threshold value of keyword servers, t: Number of decryptors, l: Number of Boolean queried keywords, ϕ: Number of returned results, M_G: Multiplication in G, l_0: Number of rows in LSSS, m: Number of columns in LSSS, $\lvert Z_q^*\rvert$: Size of field element, $R_{Z_q^*}$: Random number generation in Z_{q^*}, $A_{Z_q^*}$: Addition in Z_q^*.

It is important to note that the Setup phase, including receiver key generation, only needs to be executed once during the life-cycle of the system, and therefore, its analysis is excluded from this discussion. In addition, the number of users in a group has no effect on the size of the system's public parameters.

Furthermore, as the demand for computational and storage capacity declines, the cost of ECSE will fall. As a result, ECSE is a very efficient scheme for providing KGA resistance because it does not place new demands on the user while maintaining good security and performance at no additional expense.

As a result, ECSE outperforms pairing-based TPPKS and PCSE during both the enc and TrapGen stages, as pairing and target group exponentiation are replaced with faster cryptographic operations. Although ECSE's encoding cost includes a quadratic representation of the keyword index, this sacrifice enhances security capabilities. Conversely, TrapGen has an outstanding performance, since it is based upon a collection of well-understood core capabilities. The additional overhead produced by ECSE is small, creating a trade-off that allows the Scheme's enhanced security features to be executed without becoming overly complex.

ECSE's increased efficiency comes from its lower overhead and stronger confidentiality compared to traditional searchable encryption solutions. Specifically, the ECSE design choice to avoid performing a large number of expensive pairing computations for each verification step (as other searchable encryption solutions require) is one of the main contributors to increased efficiency.

In ECSE, the majority of checks for consistency occur at the threshold signature aggregation stage, thereby eliminating the need to perform the same grouping multiple times on the cloud server. This means ECSE can quickly process and evaluate Boolean predicates. Additionally, through the use of a commitment random function in the creation of the keyword, the randomness used to produce the keyword is consistent and pre-sanitized before being sent to the key server, therefore eliminating the risk of adversarial manipulation without requiring further checks.

In multi-attribute searches, Boolean predicates based on LSSS also offer increased efficiency over previously established PEKS systems that evaluate each keyword separately because the access structure has been encoded in an easy-to-read compact linear format in ECSE, resulting in reduced growth of trapdoor size and enabling the cloud server to conduct one structured Test instead of separate disjoint checks for each keyword, thereby allowing predicate evaluations to scale more smoothly with increasing numbers of attributes. Finally, generating collaborative randomness and performing threshold signing mitigate the negative effects of key-generating attacks and sub-attacks without incurring significant additional costs associated with cryptographic operations. Overall, these factors collectively explain why ECSE demonstrates lower computation cost, reduced communication overhead, and improved verification efficiency when compared with state-of-the-art approaches in the literature.

Table 3. Comparative Summary of Existing and ECSE Schemes

Schemes	TPPKS [15]	TMS [58]	PCSE [18]	SR-PEKS [28]	ECSE
Collaborative retrieval	✓	✓	✓	×	✓
Resistance to KGA	×	×	✓	✓	✓
Resistance to SA	×	×	×	✓	✓
Expressiveness (Boolean/LSSS)	×	×	×	×	✓
Result Verification	×	✓	✓	✓	✓
Anonymity (Search/Keyword)	×	×	✓	×	Partial

6.2. Practical Test

For our experiments, this work utilizes the Krapivin Dataset, which contains 2,000 scientific papers from the field of computer science, published by ACM, with a total size of 855 MB. Each paper includes key phases assigned by the authors and validation by reviewers. This work uses the author-provided key phrases as keywords for the documents and creates a corresponding index. The experiments are performed on an Ubuntu 24.04 server equipped with the Intel Core i7-14650HX processor and 16 GB of RAM. The implementation is done in Python, utilizing the pypbc library. For the cryptographic configuration setup, this proposed scheme adopts Type A1 pairing for the elliptic curve $E(F_q): y^2 = x^3 + x$, setting parameters p to 160 bits and q 512 bits. This configuration results in $|Z_q^*| = 160$ bits and $|G| = |G_T| = 1024$ bits.

As illustrated in Figure 3, the computational delay for key generation at keyword servers increases linearly with the number of servers, following the ratio η/t_1 where t_1 denotes the threshold value. Their project used a more comprehensive threshold to split the secret, and also used parallelized distributed key generation, Shamir secret sharing, Lagrange interpolations for polynomial constructions, and added some additional pairing-based verification on Lagrange interpolations, and also split it for multiple secrets. ECSE is designed for large-scale distributed generations, and it also uses pairing-based verification for secret share and makes it more complex and scalable; that is why the computational delay of ECSE is much higher compared to others.

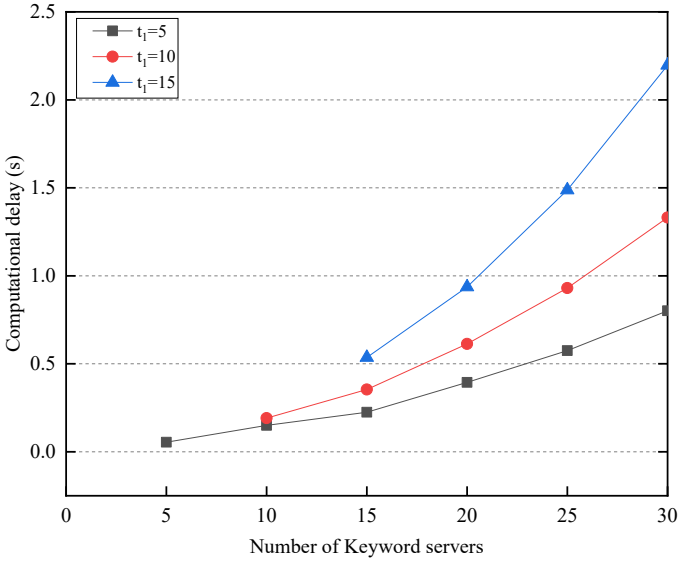

Fig. 3 Performance Evaluation of KeyGen for Keyword Servers

The graph in Figure 4 illustrates the computational delay for encryption, as the number of Boolean keywords increases, and using the various thresholds t_1, and also use CP-ABE. This is used to facilitate the encryption of messages based on an access policy tied to attributes. The results show that as the number of Boolean keywords increases, the computational delay also increases. This is because the server must process more keywords and handle the corresponding Cryptographic operations. We can see that the encryption of 30 Boolean keywords takes less than 2.5 seconds, even with $t_1 = 15$, which proves the usefulness of ECSE, suggesting that higher threshold values introduce greater complexity and processing time. This behaviour shows the trade-off between the threshold and how well encryption works when dealing with a bigger list of Boolean keywords.

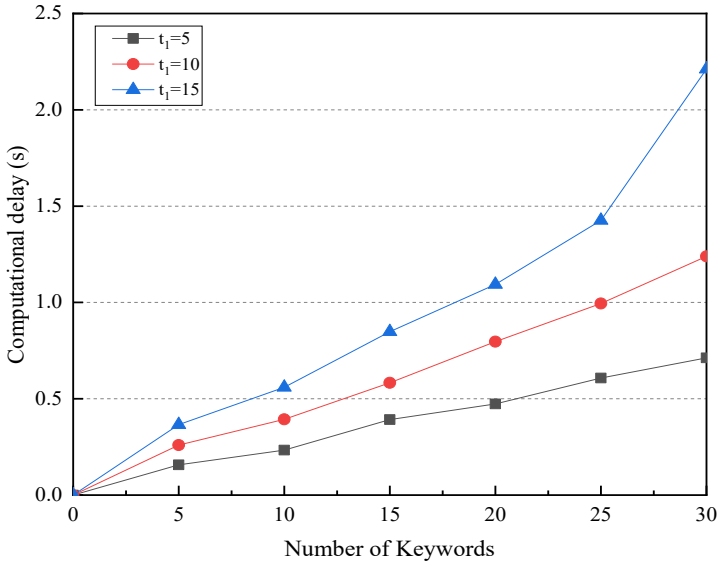

Fig. 4 Performance Evaluation Enc by using SDBKGen

The increase in the amount of computational overhead for TrapGen as the number of keyword queries rises with the value of the parameter $t = t_0 = 30$ can be seen in Figure 5a. ECSE is more efficient than TPPKS and TMS in terms of execution times. The increased execution time of ECSE is due to the use of the Boolean keyword mechanism on the export server, which provides a strong preventative measure against keyword leakage. The processing time of ECSE is less than 0.07 seconds even at 15 requested keywords, indicating that it does not adversely affect the overall performance of the system. The results in Figure 5b provide additional confirmation regarding the superior efficiency of the ECSE search algorithm compared to other search algorithms.

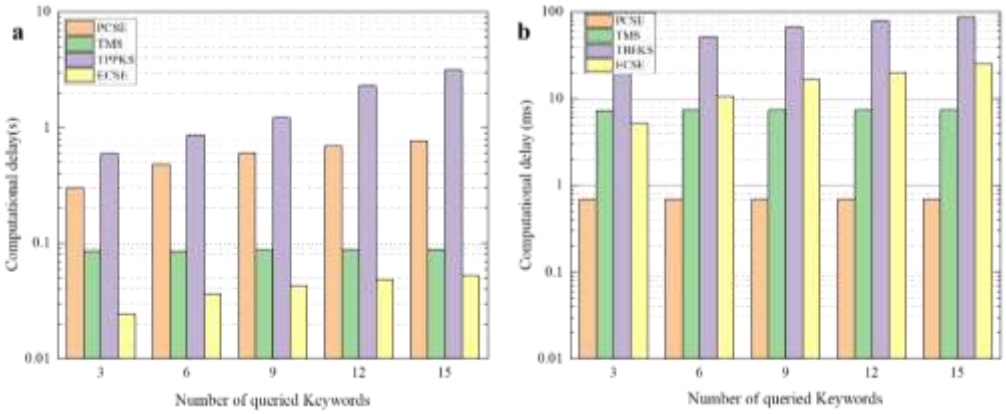

Fig. 5 Performance Evaluations (a) TrapGen, (b) Search

Compared to TMS, the verification process associated with the ECSE methodology is consistent with TMS. According to the experimental results provided in Figure 6a, the verification overhead associated with the number of processed results will directly increase as the number of searches rises. The execution time of ECSE is longer than that of PCSE due to the fact that ECSE uses Boolean keywords and requires more processing time than multi-keyword-based processing. In addition, ECSE implements the LSSS method. The execution time for decrypting information using ECSE will increase slightly for 30 and 36 intended users (receivers); the reasoning for this is that the larger number of receivers increases the access structure, thus there will be an increased number of verified steps, and a larger number of cryptographic operations will be performed.

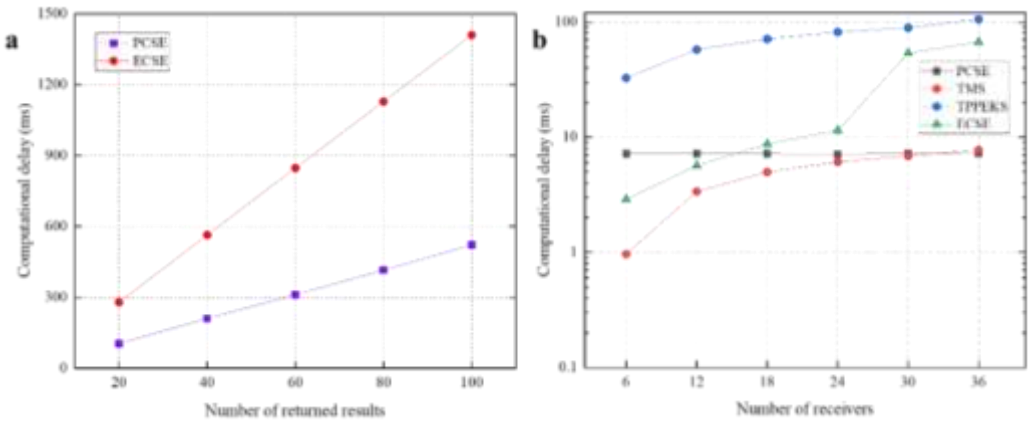

Fig. 6 Performance Evaluations (a) Verify (b) Dec

7. Conclusion

This paper describes the development of a new PKE algorithm that is intended to withstand subversive and threshold-type attacks while also allowing users to conduct secure and efficient Boolean keyword searches on encrypted data shared in a cloud-based environment. The method uses cryptographic reverse firewalls, threshold secret share generation, and collaborative random number generation. It thus offers a high level of protection against "brute-force" keyword guessing and related attacks, as well as those that exploit semi-trust cloud server providers by necessitating joint efforts from numerous entities. The proposed system also meets the requirements for ensuring the secrecy, integrity, and validity of encrypted data. Furthermore, the proposed scheme offers extraordinarily high levels of efficiency and performance. Theoretical and practical analysis of this proposed architecture show that the method strikes a practical balance between security robustness and computational efficiency, providing superior resistance to adversarial threats and better verification performance than existing methods.

Future Work

The ECSE facilitates additional interaction among users, CRFs, and essential servers, potentially leading to increased latencies in suboptimal network conditions. Boolean LSSS predicates provide greater informational complexity than basic keyword PEKS, as the trapdoors for searching multiple keywords necessitate larger trapdoors. Furthermore, the system necessitates a minimum of t honest key servers for the continuation of keyword generation.

In the future, we intend to enhance this design to include dynamic updates to the keywords being searched. The next logical steps will be to develop methods to allow blockchain auditing of the public key encryption system, as well as to provide low-latency and low-processing-power (or devices that operate in an IoT environment or a small edge-computing environment) implementations of this system for much broader real-world use cases in future project iterations.

Funding Statement

This work was supported in part by the National Natural Science Foundation of China under Grant 62072240, and by the Nanjing Science and Technology Innovation Project for Returned Overseas Scholars (Category C).

Acknowledgement

The proposed algorithm was collaboratively designed by Mishal Ismaeel and GangQiang Duan, who implemented it in the system, while Mishal Ismaeel drafted the full manuscript. Lin Mei contributed by revising and improving the manuscript. Chungen Xu supervised the research, determined how the methodology was going to be carried out, reviewed the manuscript, and provided support and resources from the institution.

References

[1] Biwen Chen et al., "Lightweight Searchable Public-key Encryption with Forward Privacy Over IIoT Outsourced Data," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 4, pp. 1753–1764, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[2] Lili Sun et al., "Efficient and Privacy-Preserving Weighted Nearby-Fit Spatial Keyword Query in Cloud," *IEEE Internet of Things Journal*, vol. 12, no. 12, pp. 20498-20511, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[3] I. Secretary, "Information Technology–security Techniques–biometric Information Protection," *International Organization for Standardization*, Standard ISO/IEC, 2011. [Google Scholar]

[4] Hongjun Li et al., "Verifiable and Forward-Secure Multi-Keyword Query in Internet of Medical Things," *IEEE Internet of Things Journal*, vol. 12, no. 13, pp. 23809-23822, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[5] Axin Wu et al., "Efficient Verifiable Searchable Encryption with Search and Access Pattern Privacy," *Security and Safety*, vol. 4, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[6] Yinbin Miao et al., "Time-controllable Keyword Search Scheme with Efficient Revocation in Mobile E-health Cloud," *IEEE Transactions on Mobile Computing*, vol. 23, no. 5, pp. 3650–3665, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[7] Chunhua Jin et al., "EBIAC: Efficient Biometric Identity-based Access Control for Wireless Body Area Networks," *Journal of Systems Architecture*, vol. 121, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[8] Liqing Chen et al., "Partially Hidden Policy Attribute-based Multi-keyword Searchable Encryption with Verification and Revocation," *IEEE Transactions on Mobile Computing*, vol. 24, no. 9, pp. 9020-9035, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[9] Dilxat Ghopur, "Attribute-based Searchable Encryption with Forward Security for Cloud-assisted IoT," *IEEE Access*, vol. 12, pp. 90840–90852, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[10] Muqadar Ali, Chungen Xu, and Abid Hussain, "Authorized Attribute-based Encryption Multi-keywords Search with Policy Updating," *Journal of New Media*, vol. 2, no. 1, pp. 31–43, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[11] Jing Wang et al., "Blockchain-enabled Lightweight Fine-grained Searchable Knowledge Sharing for Intelligent IoT," *IEEE Internet of Things Journal*, vol. 10, no. 24, pp. 21566–21579, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[12] Bo Qin et al., "Provably Secure Threshold Public-key Encryption with Adaptive Security and Short Ciphertexts," *Information Sciences*, vol. 210, pp. 67–80, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[13] Stanislaw Jarecki, and Phillip Nazarian, "Adaptively Secure Threshold Blind BLS Signatures and Threshold Oblivious PRF," *Advances in Cryptology-ASIACRYPT*, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[14] Shiwei Zhang, Yi Mu, and Guomin Yang, "Threshold Broadcast Encryption with Keyword Search," *Information Security and Cryptology*, pp. 322-337, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[15] Peishun Wang, Huaxiong Wang, and Josef Pieprzyk, "Threshold Privacy Preserving Keyword Searches," *SOFSEM 2008: Theory and Practice of Computer Science*, pp. 646–658, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[16] Hang Cheng et al., "Person Re-identification over Encrypted Outsourced Surveillance Videos," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 3, pp. 1456–1473, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[17] Shan Jiang et al., "Privacy-preserving and Efficient Multi-keyword Search Over Encrypted Data on Blockchain," *2019 IEEE International Conference on Blockchain (Blockchain)*, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[18] Yongliang Xu et al., "PCSE: Privacy-preserving Collaborative Searchable Encryption for Group Data Sharing in Cloud Computing," *IEEE Transactions on Mobile Computing*, vol. 24, no. 5, pp. 4558-4572, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[19] Axin Wu et al., "Efficient Public-key Searchable Encryption Against Inside Keyword Guessing Attacks for Cloud Storage," *Journal of Systems Architecture*, vol. 149, p. 103104, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[20] Rongmao Chen et al., "Server-aided Public Key Encryption with Keyword Search," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 12, pp. 2833–2842, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[21] Qiang Tang, and Liqun Chen, "Public-key Encryption with Registered Keyword Search," *Public Key Infrastructure, Services and Applications*, pp. 163–178, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[22] Yuan Zhang et al., "Blockchain-assisted Public-key Encryption with Keyword Search Against Keyword Guessing Attacks for Cloud Storage," *IEEE Transactions on Cloud Computing*, vol. 9, no. 4, pp. 1335–1348, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[23] Kai Zhang et al., "Subversion-resistant and Consistent Attribute-based Keyword Search for Secure Cloud Storage," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 1771–1784, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[24] Pascal Bemmann, Rongmao Chen, and Tibor Jager, "Subversion-resilient Public Key Encryption with Practical Watchdogs," *Public-Key Cryptography-PKC*, pp. 627–658, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[25] Ilya Mironov, and Noah Stephens-Davidowitz, "Cryptographic Reverse Firewalls," *Advances in Cryptology-EUROCRYPT*, pp. 657–686, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[26] Changsong Jiang et al., "Blockchain-based Immunization Against Kleptographic Attacks," *Science China Information Sciences*, vol. 67, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[27] Yuyang Zhou, Zhebin Hu, and Fagen Li, "Searchable Public-key Encryption with Cryptographic Reverse Firewalls for Cloud Storage," *IEEE Transactions on Cloud Computing*, vol. 11, no. 1, pp. 383–396, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[28] Changsong Jiang et al., "SR-PEKS: Subversion-resistant Public Key Encryption with Keyword Search," *IEEE Transactions on Cloud Computing*, vol. 11, no. 3, pp. 3168–3183, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[29] Dawn Xiaoding Song, D. Wagner, and A. Perrig, "Practical Techniques for Searches on Encrypted Data," *Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000*, 2000. [CrossRef] [Google Scholar] [Publisher Link]

[30] Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo, "Public Key Encryption with Keyword Search Revisited," *Computational Science and Its Applications-ICCSA*, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[31] Lijun Qi, and Jincheng Zhuang, "Efficient Public Key Searchable Encryption Schemes from Standard Hard Lattice Problems for Cloud Computing," *Cryptology ePrint Archive*, 2022. [Google Scholar] [Publisher Link]

[32] Liqing Chen et al., "Fair and Exculpable Attribute-Based Searchable Encryption with Revocation and Verifiable Outsourced Decryption Using Smart Contract," *IEEE Internet of Things Journal*, vol. 12, no. 4, pp. 4302-4317, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[33] Axin Wu et al., "Enabling Traceable and Verifiable Multi-user Forward Secure Searchable Encryption in Hybrid Cloud," *IEEE Transactions on Cloud Computing*, vol. 11, no. 2, pp. 1886–1898, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[34] Maryam Zarezadeh, Hamid Mala, and Maede Ashouri-Talouki, "Multi-keyword Ranked Searchable Encryption Scheme with Access Control for Cloud Storage," *Peer-to-Peer Networking and Applications*, vol. 13, pp. 207–218, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[35] Leixiao Cheng et al., "Security-enhanced Public-key Authenticated Searchable Encryption," *Information Sciences*, vol. 647, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[36] Panyu Wu et al., "MMKFB: Multi-client and Multi-keyword Searchable Symmetric Encryption with Forward and Backward Privacy," *Frontiers of Computer Science*, vol. 19, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[37] Qiyang Song et al., "SAP-SSE: Protecting Search Patterns and Access Patterns in Searchable Symmetric Encryption," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 1795–1809, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[38] Dan Boneh et al., "Public Key Encryption with Keyword Search," *Advances in Cryptology-EUROCRYPT*, pp. 506-522, 2004. [CrossRef] [Google Scholar] [Publisher Link]

[39] Yang Yang et al., "Dual Traceable Distributed Attribute-based Searchable Encryption and Ownership Transfer," *IEEE Transactions on Cloud Computing*, vol. 11, no. 1, pp. 247–262, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[40] Puning Zhang et al., "Efficient and Privacy-preserving Search Over Edge–cloud Collaborative Entity in IoT," *IEEE Internet of Things Journal*, vol. 10, no. 4, pp. 3192–3205, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[41] Chunpeng Ge et al., "Secure Keyword Search and Data Sharing Mechanism for Cloud Computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 6, pp. 2787–2800, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[42] Garima Verma, and Soumen Kanrar, "Secure Keyword Search over Encrypted Cloud Data Using Blockchain in Digital Document Sharing," *Wireless Personal Communications*, vol. 134, pp. 975–996, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[43] Jin Wook Byun et al., "Off-line Keyword Guessing Attacks on Recent Keyword Search Schemes Over Encrypted Data," *Secure Data Management*, pp. 75-83, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[44] Run Xie et al., "Lattice-based Searchable Public-key Encryption Scheme for Secure Cloud Storage," *International Journal of Web and Grid Services*, vol. 14, no. 1, pp. 3–20, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[45] Liming Fang et al., "Public Key Encryption with Keyword Search Secure Against Keyword Guessing Attacks without Random Oracle," *Information Sciences*, vol. 238, pp. 221–241, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[46] Haoyu Zhang, Baodong Qin, and Dong Zheng, "Registered Keyword Searchable Encryption Based on SM9," *Applied Sciences*, vol. 13, no. 5, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[47] Mohammad Raouf Senouci et al., "An Efficient and Secure Certificateless Searchable Encryption Scheme Against Keyword Guessing Attacks," *Journal of Systems Architecture*, vol. 119, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[48] Qiong Huang, and Hongbo Li, "An Efficient Public-key Searchable Encryption Scheme Secure Against Inside Keyword Guessing Attacks," *Information Sciences*, vol. 403-404, pp. 1–14, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[49] Xiangyu Pan, and Fagen Li, "Public-key Authenticated Encryption with Keyword Search Achieving Both Multi-Ciphertext and Multi-Trapdoor Indistinguishability," *Journal of Systems Architecture*, vol. 115, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[50] Yang Lu, and Jiguo Li, "Lightweight Public Key Authenticated Encryption with Keyword Search Against Adaptively-Chosen-Targets Adversaries for Mobile Devices," *IEEE Transactions on Mobile Computing*, vol. 21, no. 12, pp. 4397–4409, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[51] Mihir Bellare, Kenneth G. Paterson, and Phillip Rogaway, "Security of Symmetric Encryption Against Mass Surveillance," *Advances in Cryptology-CRYPTO*, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[52] Mihir Bellare, Joseph Jaeger, and Daniel Kane, "Mass-surveillance without the State: Strongly Undetectable Algorithm-substitution Attacks," *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 1431–1440, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[53] Shanshan Li et al., "A Secure Two-factor Authentication Scheme from Password-protected Hardware Tokens," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 3525–3538, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[54] Emma Dauterman et al., "True2F: Backdoor-resistant Authentication Tokens," *2019 IEEE Symposium on Security and Privacy (SP)*, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[55] Suvradip Chakraborty, Stefan Dziembowski, and Jesper Buus Nielsen, "Reverse Firewalls for Actively Secure MPCs," *Advances in Cryptology-CRYPTO*, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[56] Yevgeniy Dodis, Ilya Mironov, and Noah Stephens-Davidowitz, "Message Transmission with Reverse Firewalls—Secure Communication on Corrupted Machines," *Advances in Cryptology-CRYPTO*, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[57] A. Beimel, "*Secure Schemes for Secret Sharing and Key Distribution*," PhD Thesis, Israel Institute of Technology, Technion, 1996. [Google Scholar] [Publisher Link]

[58] Yinbin Miao et al., "Threshold Multi-keyword Search for Cloud-based Group Data Sharing," *IEEE Transactions on Cloud Computing*, vol. 10, no. 3, pp. 2146–2162, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[59] Ruizhong Du, Caixia Ma, and Mingyue Li, "Privacy-preserving Searchable Encryption Scheme Based on Public and Private Blockchains," *Tsinghua Science and Technology*, vol. 28, no. 1, pp. 13-26, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[60] Yun Wang, and Dimitrios Papadopoulos, "Multi-user Collusion-resistant Searchable Encryption for Cloud Storage," *IEEE Transactions on Cloud Computing*, vol. 11, no. 3, pp. 2993-3008, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[61] Somchart Fugkeaw et al., "EVSEB: Efficient and Verifiable Searchable Encryption with Boolean Search for Encrypted Cloud Logs," *IEEE Access*, vol. 13, pp. 101177-101195, 2025. [CrossRef] [Google Scholar] [Publisher Link]