Study and Testbed Experiments on Public Key Based Search on Encrypted Data using Cloud Storage

Obbineni Sai Nikhita , Gowlapalli Rohit
Center for Security, Theory and Algorithmic Research
International Institute of Information Technology, Hyderabad 500 032, India

I. Introduction

Searchable Encryption (SE) has emerged as a pivotal advancement in the realm of information security, adeptly balancing the imperative of data privacy with the need for swift and effective information retrieval. The evolution of SE is intricately tied to the escalating demand for secure storage solutions in cloud environments and the concurrent necessity for preserving privacy in search operations. In its early stages, cryptographic endeavors predominantly concentrated on upholding data confidentiality during data transmission and storage, inadvertently sidelining the intricate challenges posed by conducting searches on encrypted datasets.

A. *Evolution of Searchable Encryption*

The initial breakthrough in the evolution of searchable encryption materialized with the advent of Order-Preserving Encryption (OPE), a cryptographic technique permitting the comparison of encrypted data without the need for decryption. However, vulnerabilities, such as susceptibility to frequency analysis and information leakage through order preservation, prompted the exploration of more robust alternatives. Fully Homomorphic Encryption (FHE) and Partially Homomorphic Encryption (PHE) subsequently entered the scene, enabling computations on encrypted data without the necessity of decryption. Despite their inherent robustness, the computational overhead associated with these techniques limited their practicality, thereby steering the development towards more efficient schemes like Symmetric Searchable Encryption (SSE) and Attribute-Based Encryption (ABE).

Subsequent strides in the evolution of searchable encryption incorporated cutting-edge technologies such as Secure Multi-Party Computation (SMPC) and hardware-based solutions like secure enclaves. These innovations markedly enhanced the efficiency and scalability of SE, rendering it more applicable for real-world scenarios spanning diverse domains such as cloud computing, healthcare, and finance. As the digital landscape continues its dynamic evolution, the ongoing refinement of searchable encryption techniques remains paramount, ensuring the safeguarding of sensitive information while facilitating seamless and secure information retrieval processes.

Design goals of SE

- *Data privacy:* The data stored in the cloud should not be disclosed to unauthorized parties.
- *Index privacy:* Index privacy indicates that the server should not be aware of the keywords embedded in the index.
- *Keyword privacy:* The server should not be able to learn the keywords in trapdoor or authentication tags generated by the user.
- *Search pattern:* The search pattern is defined as the information that can be extracted from knowledge on whether two or more search results are from the same keyword.
- *Access pattern:* Access pattern is defined as the information that can be extracted from knowledge of a sequence of search results that contains the keyword.
- *Efficiency:* The user should be able to generate trapdoors and get search results efficiently. On the other hand, the cloud server should be able to fulfill the keyword search efficiently.
- *Verifiability:* In spite of data privacy, the results obtained from the cloud must be verified to examine data integrity.
- *Access control:* Access control prevents unauthorized access of data from the users who are not allowed to access the data. It can be achieved by performing user revocation.

B. Taxonomy of Searchable Encryption

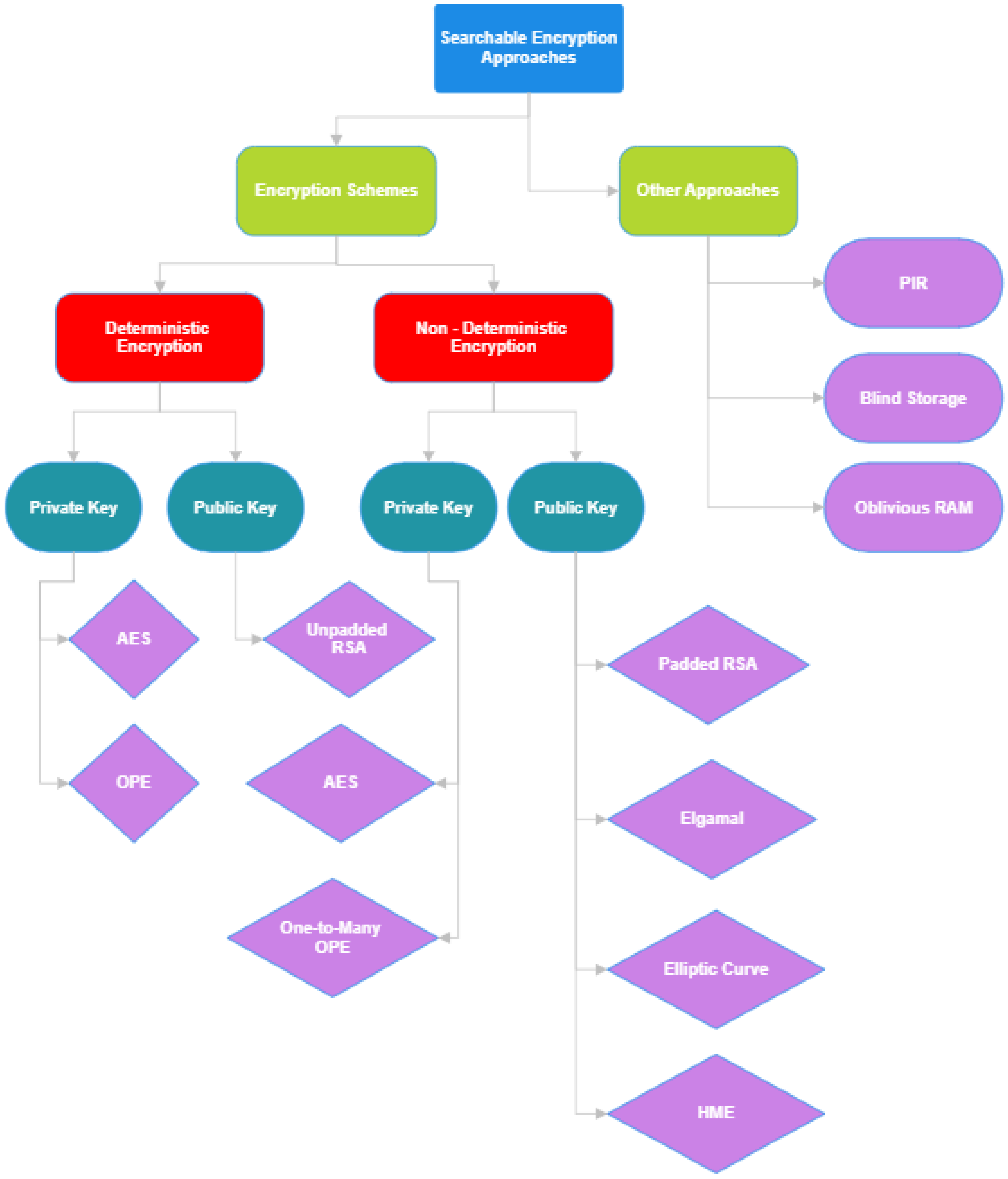

Fig. 1. **Taxonomy of Searchable Encryption**

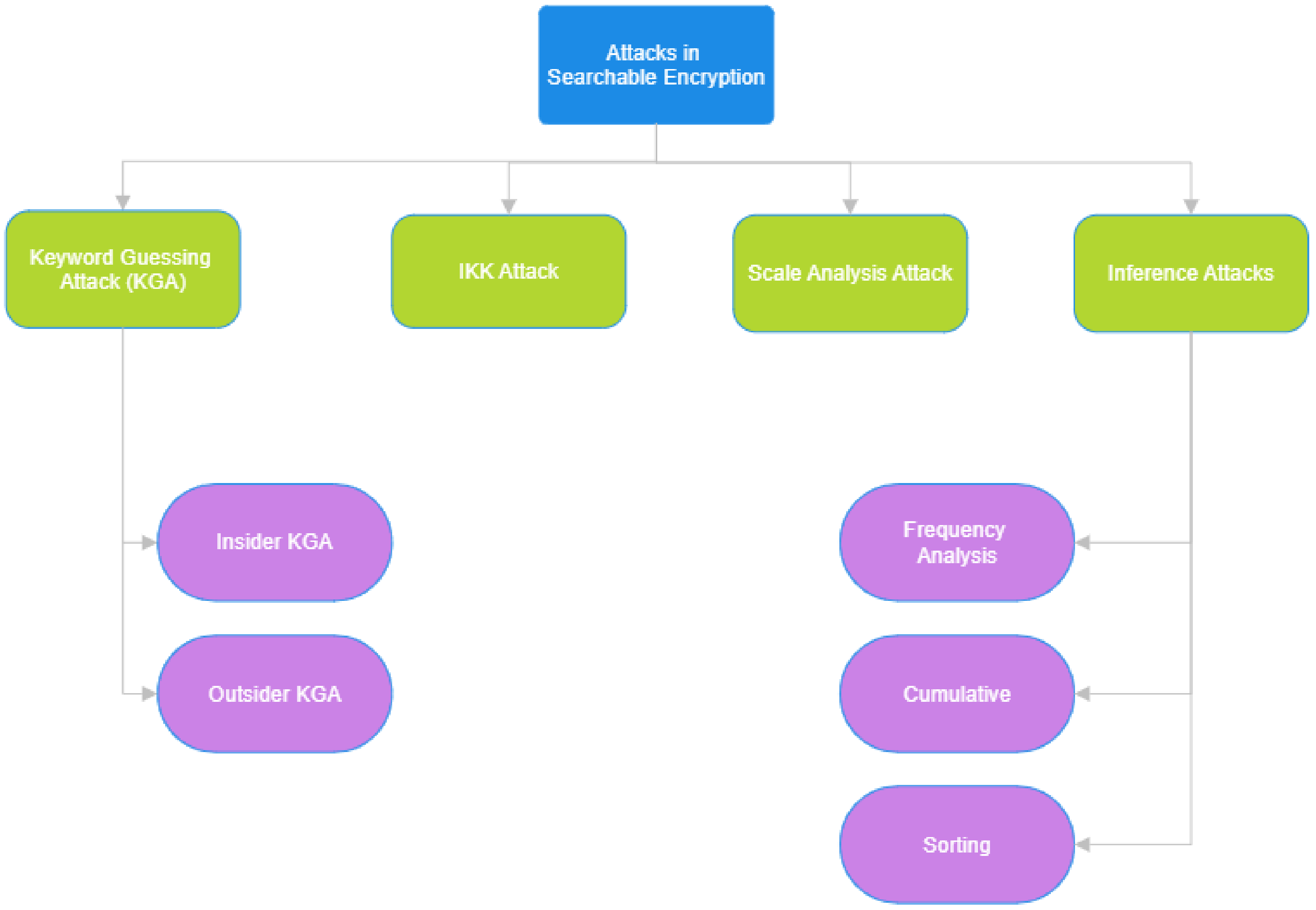

Fig. 2. **Attacks in Searchable Encryption**

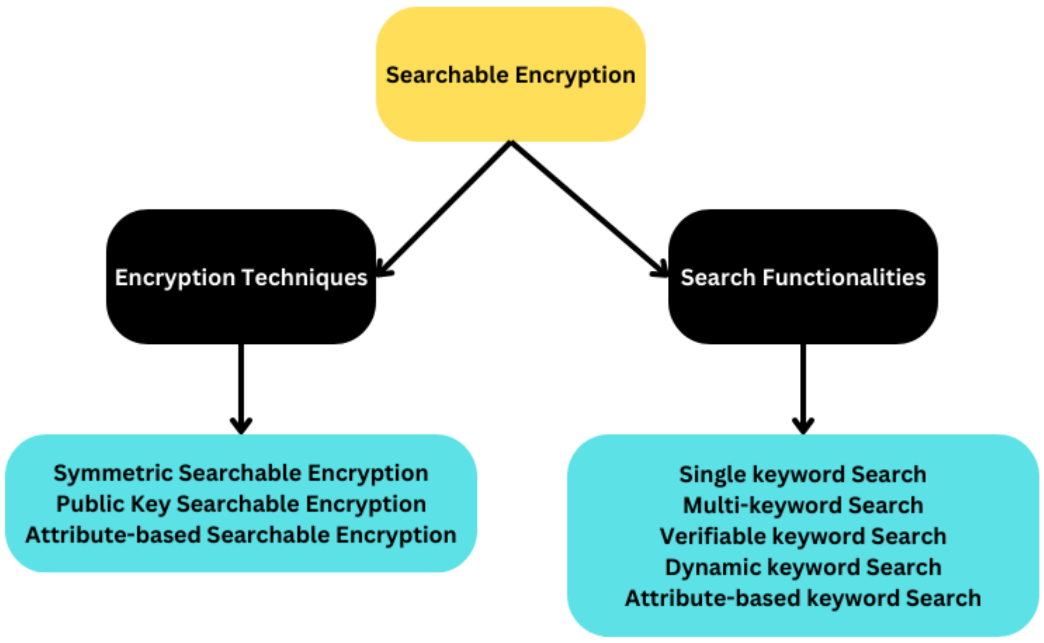
Searchable Encryption
Encryption Techniques
Search Functionalities
Symmetric Searchable Encryption
Public Key Searchable Encryption
Attribute-based Searchable Encryption
Single keyword Search
Multi-keyword Search
Verifiable keyword Search
Dynamic keyword Search
Attribute-based keyword Search

Fig. 3. **Taxonomy of Searchable Encryption**

*1) **Symmetric Searchable Encryption:*** In SSE, the documents are encrypted by using symmetric/private key encryption techniques. As shown in the below figure, the cloud server sits in between the DO and the DU. In this, DO encrypts the documents along with the index and sends them to the cloud server, and later DU can access encrypted data by generating the trapdoor to the cloud server. The cloud server performs a search over encrypted data and sends results to DU, and then, the results obtained can be decrypted by using the secret key.

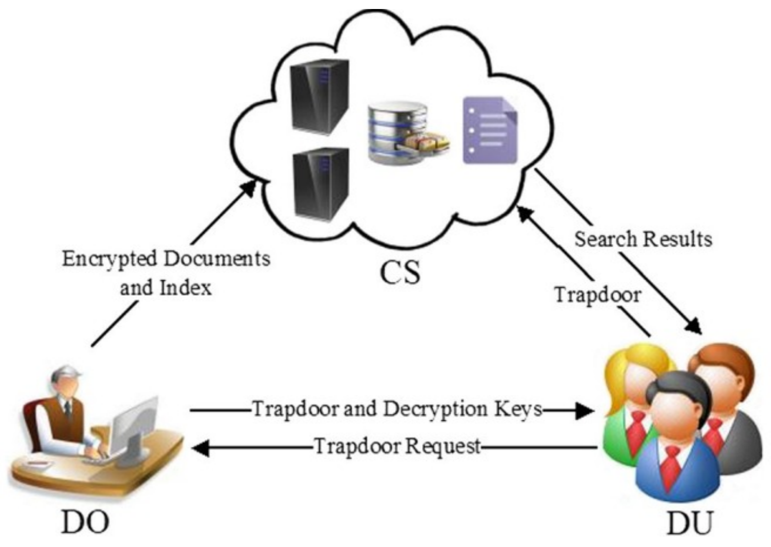

Fig. 4. **Architecture of Searchable Encryption**

SSE Algorithm

- **KeyGen**$(s) \rightarrow K$**:** The KeyGen algorithm is the very first algorithm in SE, and it is initiated by the DO. It takes security parameter s as an input and outputs the private key K.
- **BuildIndex**$(w) \rightarrow I$**:** This algorithm is run by DO. It takes a set of keywords w where $w = w_1, w_2, \ldots, w_n$ from a document set as input and produces the searchable index I as output.
- **Enc**$(D, I, K) \rightarrow E_d, E_i$**:** This algorithm is initiated by DO. It accepts document set D where $D = d_1, d_2, \ldots, d_k$, index I, and key K as input and produces the encrypted documents E_d and encrypted index E_i as output.
- **genTrapdoor**$(K, q) \rightarrow T$**:** This algorithm is run by DU. It accepts secret key K and query keyword(s) q as input. If it is a single-keyword search, then q contains only one keyword; otherwise, it contains more than one keyword. Then, it outputs trapdoor T by encrypting q with K.
- **Search**$(E_i, T) \rightarrow E_d$**:** This algorithm is run by CS. It accepts the trapdoor T and encrypted index E_i as input and performs a search operation with the trapdoor on the encrypted index before producing the encrypted search results E_d as an output.
- **Dec**$(E_d, K) \rightarrow D$**:** This algorithm is initiated by DU. It takes the search results Ed and secret key K as input and gives the actual documents D as output by decrypting E_d.

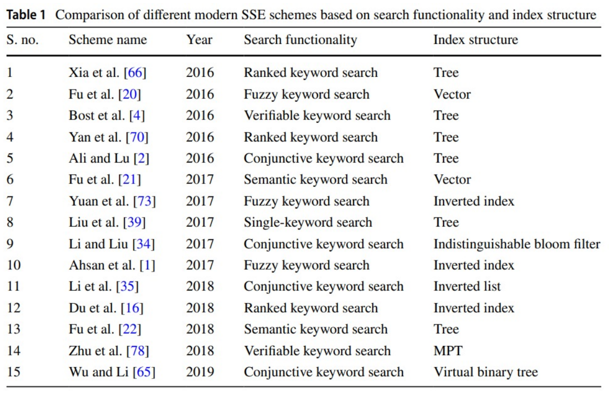

Table 1 Comparison of different modern SSE schemes based on search functionality and index structure

S. no.	Scheme name	Year	Search functionality	Index structure
1	Xia et al. [66]	2016	Ranked keyword search	Tree
2	Fu et al. [20]	2016	Fuzzy keyword search	Vector
3	Bost et al. [4]	2016	Verifiable keyword search	Tree
4	Yan et al. [70]	2016	Ranked keyword search	Tree
5	Ali and Lu [2]	2016	Conjunctive keyword search	Tree
6	Fu et al. [21]	2017	Semantic keyword search	Vector
7	Yuan et al. [73]	2017	Fuzzy keyword search	Inverted index
8	Liu et al. [39]	2017	Single-keyword search	Tree
9	Li and Liu [34]	2017	Conjunctive keyword search	Indistinguishable bloom filter
10	Ahsan et al. [1]	2017	Fuzzy keyword search	Inverted index
11	Li et al. [35]	2018	Conjunctive keyword search	Inverted list
12	Du et al. [16]	2018	Ranked keyword search	Inverted index
13	Fu et al. [22]	2018	Semantic keyword search	Tree
14	Zhu et al. [78]	2018	Verifiable keyword search	MPT
15	Wu and Li [65]	2019	Conjunctive keyword search	Virtual binary tree

Fig. 5. **Comparision of different modern SSE Schemes**

*2) **Public key Searchable Encryption:*** SSE schemes require an extra secure channel to share the secret key between DO and DU, but we cannot guarantee that the secure channel is not compromised. PSE schemes use public key encryption techniques in which two keys are used for encryption and decryption: the public key and private key, respectively. As shown in the above figure , DO encrypts the documents along with document index with the user's public key and outsources the documents along with an index to the cloud server. Now DU can generate the trapdoor to the cloud server to obtain results. Once DU gets the results, it can decrypt the results by using the user's private key

- **KeyGen(s) $\rightarrow$ (PK, SK):** The KeyGen algorithm is the very first algorithm in SE, and it is initiated by the DO. It takes security parameter s as an input and outputs the public parameter PK and secret key SK.
- **BuildIndex(w) $\rightarrow$ I:** This algorithm is run by DO. It takes a set of keywords w where $w = w_1, w_2, \ldots, w_n$ from a document set D as input and produces the searchable index I as output.
- **Enc(PK, D, I) $\rightarrow$ E_d, E_i:** This algorithm is initiated by DO. It accepts document set D where $D = d_1, d_2, \ldots, d_k$, index I, and key PK as input and produces the encrypted documents E_d and encrypted index E_i as output.
- **genTrapdoor(SK, q) $\rightarrow$ T:** This algorithm is run by DU. It accepts secret key SK and query keyword(s) $q \in I$ as input. If it is a single-keyword search, then q contains only one keyword; otherwise, it contains more than one keyword. Then, it outputs trapdoor T by encrypting q with SK.
- **Query(E_i, T) $\rightarrow$ E_d:** This algorithm is run by CS. It accepts the trapdoor T and encrypted index E_i as input and performs a search operation with the trapdoor on the encrypted index before producing the candidate set of encrypted documents E_d as an output.
- **Dec(E_d, SK) $\rightarrow$ D:** This algorithm is initiated by DU. It takes the search results E_d and secret key SK as input and gives the plaintext version of documents D as output by decrypting Ed.

C. *Organization of report*

II. Related Work

Public Key Encryption with Keyword Search in Cloud – A Survey

PEKS-PE Insights: Predicate Encryption (PEKS-PE) introduces a paradigm shift by enabling searches on ciphertext without the necessity of a private key. The capability to support disjunctive and conjunctive keyword searches adds a layer of versatility. Nevertheless, the landscape exhibits varying levels of efficiency across schemes, necessitating a nuanced balance between security and computational performance.
PEKS-PRE Unveiled: PEKS integrated with Proxy Re-Encryption (PEKS-PRE) presents a compelling solution, allowing users to delegate search functionalities without compromising sensitive information. The bidirectional features and designated testers contribute to the sophistication of the scheme. Security enhancements, including resistance against quantum attacks and the assurance of verifiable correctness, underscore the practical applications, notably in healthcare systems.
Key Takeaways: In the context of this inquiry, PEKS-PE and PEKS-PRE [1] transcend theoretical constructs. They emerge as practical solutions addressing privacy, security, and computational efficiency. Despite efficiency concerns and performance discrepancies inherent in cryptographic schemes, the tangible impact on real-world applications, particularly in healthcare and secure data sharing, underscores the practical relevance of these cryptographic innovations.

- **KeyGen(λ) $\rightarrow$ (pk, sk):**
  - **Description:** This algorithm is executed by the data receiver to generate cryptographic keys.
  - **Input:** The security parameter λ.
  - **Output:** A public key pk and a private key sk.
- **PEKS(pk, W) $\rightarrow$ S:**
  - **Description:** The encryption algorithm used by the data sender to encrypt a keyword.
  - **Input:** The public key pk and the keyword W.
  - **Output:** A keyword ciphertext S for the given keyword W.
- **Trapdoor(sk, W) $\rightarrow$ TW:**
  - **Description:** This is employed by the data receiver for generating a trapdoor corresponding to a query keyword.
  - **Input:** The private key sk and the query keyword W.
  - **Output:** The trapdoor TW for the specified query keyword W.
- **Test(pk, S, TW) $\rightarrow$ Output:**
  - **Description:** The test algorithm run by the server provider to determine if the keyword in the ciphertext matches the query keyword associated with the trapdoor.
  - **Input:** The public key pk, a ciphertext S corresponding to keyword W', and the trapdoor TW for the query keyword W.
  - **Output:** "Yes" if $W = W'$, indicating a match; otherwise, it outputs "No."

Secure and Efficient Searchable Public Key Encryption for Resource Constrained Environment Based on Pairings under Prime Order Group, Security and Communication Networks

The PECDK (Public-key Encryption with Conjunctive and Disjunctive Keyword Search) scheme allows users to securely search encrypted data without revealing the private key. In this model, a sender encrypts a message and constructs an encrypted index using the recipient's public key, along with a set of keywords. The receiver, armed with a trapdoor generated from the public key and specific keywords, sends this trapdoor to the server for a secure search. The server, without knowing the actual content, tests the trapdoor against encrypted indices and returns matched messages.

Key Algorithms in PECDK

The PECDK scheme involves four key algorithms: *KeyGen()*, *IndexBuild()*, *Trapdoor()* and *Test()*.

- *KeyGen:* Generates public and private key pairs.
- *IndexBuild:* Constructs an encrypted index based on a keyword set and public key.
- *Trapdoor:* Generates a trapdoor for secure search based on a keyword query.
- *Test:* Examines the trapdoor against the secure index and public key to determine matching messages.

1) $\mathrm{KeyGen}(1^\gamma)$: Given a security parameter γ, the algorithm generates two cyclic groups G_1, G_2 of prime order q and a bilinear pairing $\widehat{e} : G_1 \times G_1 \longrightarrow G_2$. It picks a random generator g of G_1 and two hash functions $H_1 : \{0,1\}^* \longrightarrow Z_q$ and $H_2 : G_2 \longrightarrow \{0,1\}^{\log_2 q}$. Choosing $2n+3$ random numbers $\alpha_0, \alpha_1, \ldots, \alpha_n, \beta_0, \beta_1, \ldots, \beta_n, \theta \in Z_q$, it computes $\{X_0 = g^{\alpha_0}, X_1 = g^{\alpha_1}, \ldots, X_n = g^{\alpha_n}, Y_0 = g^{\beta_0}, Y_1 = g^{\beta_1}, \ldots, Y_n = g^{\beta_n}, Z = g^\theta, \mu = \hat{e}(g,g)\}$. After this, it outputs the public key $pk = \{X_0, X_1, \ldots, X_n, Y_0, Y_1, \ldots, Y_n, Z, \mu, g, H_1, H_2, \hat{e}\}$ and the secret key $sk = \{\alpha_0, \alpha_1, \ldots, \alpha_n, \beta_0, \beta_1, \ldots, \beta_n, \theta\}$, where pk is also open to the public.
2) IndexBuild (pk, W): The algorithm generates n^2+2n random numbers $r_1, r_2, \ldots, r_n$ and $u_{ij} \in Z_q^*$, where $i \in [1,n], j \in [0,n]$. Given a keyword set $W = \{w_1, w_2, \ldots, w_n\}$ and the public key pk, for each word $w_i \in W$, it computes $A_{ij} = X_j^{r_i} \times g^{r_i H_1(\omega_i)^j + u_{ij}}$, $B_{ij} = Y_j^{u_{ij}}$, $C_i = Z^{r_i} = g^{\theta r_i}$, and $D_i = H_2(\mu^{r_i})$, where $i \in [1,n]$ and $j \in [0,n]$. The index of the keyword set W is

$$I_W = \begin{pmatrix} A_{10} & A_{11} & \ldots & A_{1n} & B_{10} & B_{11} & \ldots & B_{1n} & C_1 & D_1 \\ A_{20} & A_{21} & \ldots & A_{2n} & B_{20} & B_{21} & \ldots & B_{2n} & C_2 & D_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ A_{n0} & A_{n1} & \ldots & A_{nn} & B_{n0} & B_{n1} & \ldots & B_{nn} & C_n & D_n \end{pmatrix} \tag{6}$$

Note that the secure index can be seen as an encrypted version of matrix (3).
3) $\mathrm{Trapdoor}(sk, \{Q, \text{ sym }\})$: When the algorithm receives a keyword query $Q = \{q_1, q_2, \ldots, q_m\}$, where $m \leq n$, it will construct a vector $\vec{a} = \{a_m, a_{m-1}, \ldots, a_0\}$ by utilizing (5). Given a vector $\vec{a}$ and sk, $t_{1j} = g^{a_j / \left(\sum_{j=0}^m \alpha_j a_j + \sum_{j=0}^m a_j u \theta\right)}$ and $t_{2j} = t_{1j}^{1/\beta_j}$ can be computed, where θ is a random number in Z_q and $j \in [0,m]$. Let $t_3 = \theta$ and $\text{sym} \in \{\text{V}, \wedge\}$; the trapdoor for the keyword query Q is $T_Q = \{t_{10}, t_{11}, \ldots, t_{1m}, t_{20}, t_{21}, \ldots, t_{2m}, t_3, \text{sym}\}$.
4) $\mathrm{Test}(pk, T_Q, I_W)$: When the algorithm receives a trapdoor T_Q and a secure index I_W, it works as follows:
   a) If the symbol in T_Q is V,
      i) The algorithm chooses a counter i and sets $i = 1$.
      ii) If $i > n$, then it goes to step (c); otherwise, it checks whether $H_2(\text{test}i1/\text{test}i2) = D_i$, where $\text{test}i1 = \prod j = 0^m \hat{e}(t_{1j}, A_{ij} C_i^{t_3})$ and $\text{test}i2 = \prod j = 0^m \hat{e}(t_{2j}, B_{ij})$. If so, the algorithm outputs 1 and ends. Otherwise, it sets $i = i+1$ and goes to step (b).
      iii) The algorithm outputs 0 and ends.
   b) If the symbol in T_Q is $\wedge$,
      i) The algorithm chooses two counters i and j and sets $i = 1$ and $j = 1$.
      ii) If $i > n$, then it goes to step (c); otherwise, it tests whether $H_2(\text{test}i1/\text{test}i2) = D_i$, where $\text{test}i1 = \prod j = 0^m \hat{e}(t_{1j}, A_{ij} C_i^{t_3})$ and $\text{test}i2 = \prod j = 0^m \hat{e}(t_{2j}, B_{ij})$. If so, then the algorithm sets $j = j+1$. Otherwise, it does nothing. After that, it sets $i = i+1$ and goes to step (b).
      iii) If $j = m$, the algorithm outputs 1 and ends. Otherwise, it outputs 0 and ends.

Correctness Property

The PECDK scheme has a correctness property ensuring accurate retrieval of messages. For conjunctive keyword searches, the match holds if all query keywords are included in the index. For disjunctive searches, the match holds if there is an intersection between the query and index.

Security Definition of PECDK

The security of PECDK relies on the Decision Bilinear Diffie-Hellman Inversion (Decision -BDHI) assumption. The security game involves challenges, queries, and responses, with an attacker attempting to break the scheme. The PECDK is considered semantically secure if the attacker's advantage is negligible.

Bilinear Pairings and Complexity Assumption

PECDK uses bilinear pairings in prime order groups, relying on the Decision-BDHI assumption for security.

Keywords Conversion

To handle string keywords, a method involving matrices and vectors is introduced. These are encrypted to create the secure index and trapdoor.

PECDK Construction

Integrating keyword conversion and bilinear pairing, PECDK enables conjunctive and disjunctive keyword searches. Matrices and vectors are encrypted for efficiency, outperforming Identity-based Encryption (IBE) schemes.

Security Analysis

PECDK's security is proven under Decision-BDHI. It's secure if an attacker can't distinguish real from simulated attacks with a significant advantage.

Overall Implications

PECDK offers a secure, efficient way for conjunctive/disjunctive keyword searches on encrypted data. Using prime order groups and Decision-BDHI enhances practicality, making it promising for privacy-preserving searches in clouds.

Comparison with Previous Schemes

Compared to IBE schemes, PECDK [2] is more efficient, a viable alternative for searchable encryption. Emphasizing practical utility, PECDK is valuable where secure and efficient keyword searches are crucial.

Searchable Public Key Encryption Supporting Semantic Multi-Keywords Search

In the realm of public key-based searchable encryption (SE), [3] proposes a novel approach for "Searchable Public Key Encryption Supporting Semantic Multi-Keywords Search."

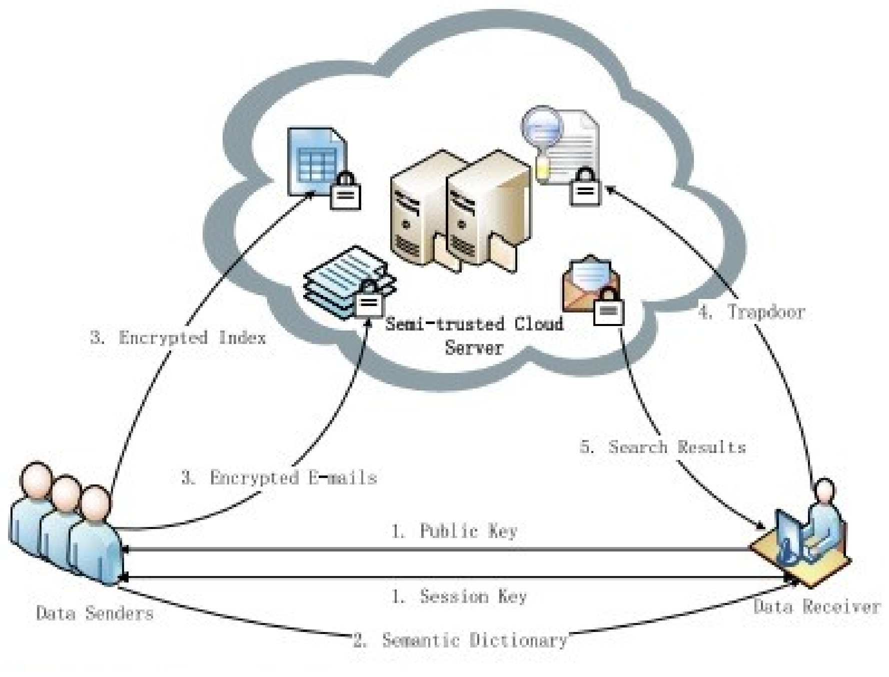

Fig. 6. **Flow of SPE-SMKS**

Semantic Index Construction The method involves constructing a semantic index and converting keyword sets into a vector space model.

Index Construction A semantic index is created by extending keyword sets through a 'word2vec'-enhanced dictionary. This index retains the semantic information of the original keywords, enabling secure multi-keyword searches.

Keywords Conversion The 'Enc' and 'KeyGen' algorithms from the PO-IPE scheme are employed for converting query and index keywords into attribute and predicate vectors.

Proposed SPE-SMKS Scheme Building upon the methods described earlier, [3] proposes a concrete SPE-SMKS scheme that leverages the PO-IPE scheme for encryption.

Concrete Scheme The SPE-SMKS scheme is presented based on the fully secure PO-IPE scheme. The security proof establishes that the security of the SPE-SMKS scheme is contingent on the underlying security of the PO-IPE scheme.

Conclusion In summary, the approach presented by [3] offers a comprehensive method for achieving secure information retrieval in the cloud through searchable public key encryption supporting semantic multi-keyword search.

Efficient Personal-Health-Records Sharing in Internet of Medical Things Using Searchable Symmetric Encryption, Blockchain, and IPFS

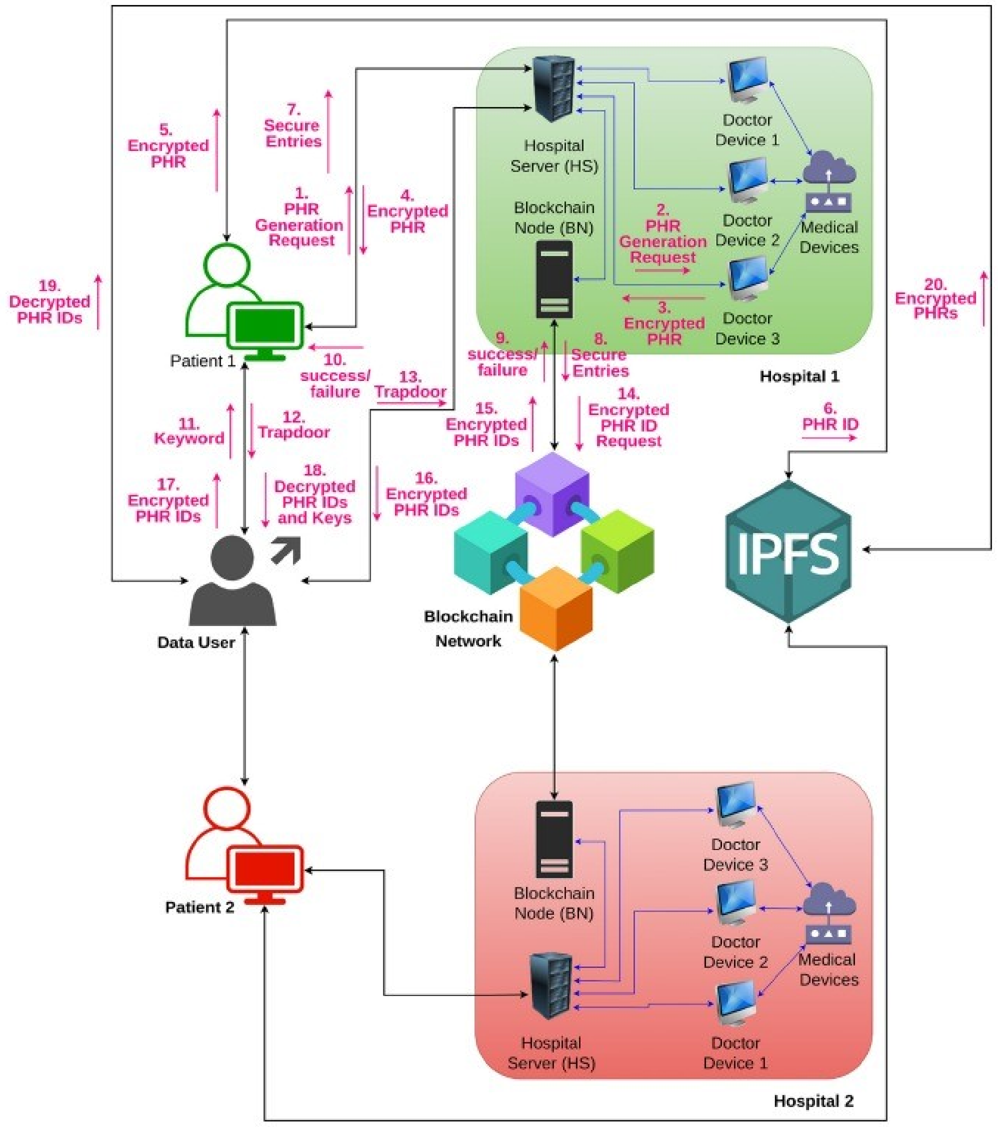

Fig. 7. **System Model**

Exploring current methods for health record sharing in IoMT, researchers adopt a secure approach using SSE, Blockchain, and IPFS [4].

Configuration and Key Generation: The system initiates by defining crucial security parameters and generating key pairs. The chosen elliptic curve $E_p(a, b)$, generator G, and hash function $h(\cdot)$ contribute to the overall system setup. The components encompass security parameters, keyword space U, and the integration of encryption algorithms.

User Configurations: Patients, hospitals (HS), doctors, and data users experience a unified setup process. This involves the selection of private keys, computation of public keys, and the establishment of dictionaries for seamless functionality.

Validation through Test-Bed: A comprehensive test-bed, utilizing desktop systems and a laptop, is implemented to simulate the real-world behavior of hospital servers, doctor devices, and patient devices. This practical validation ensures the robustness and effectiveness of the proposed system.

Code Implementation: The implementation phase involves crafting approximately 4000 lines of code in Python 3.10.6. The system leverages the cryptography Python library, ensuring a strong foundation for security measures.

Performance Analysis: A thorough analysis of time complexities is conducted for various schemes. The hybrid solution presented in this system demonstrates remarkable efficiency, especially in scenarios with a limited number of Personal-Health-Records (PHRs). This efficiency makes it a compelling solution for secure and privacy-preserving medical data sharing.

III. Details of Implemented Scheme

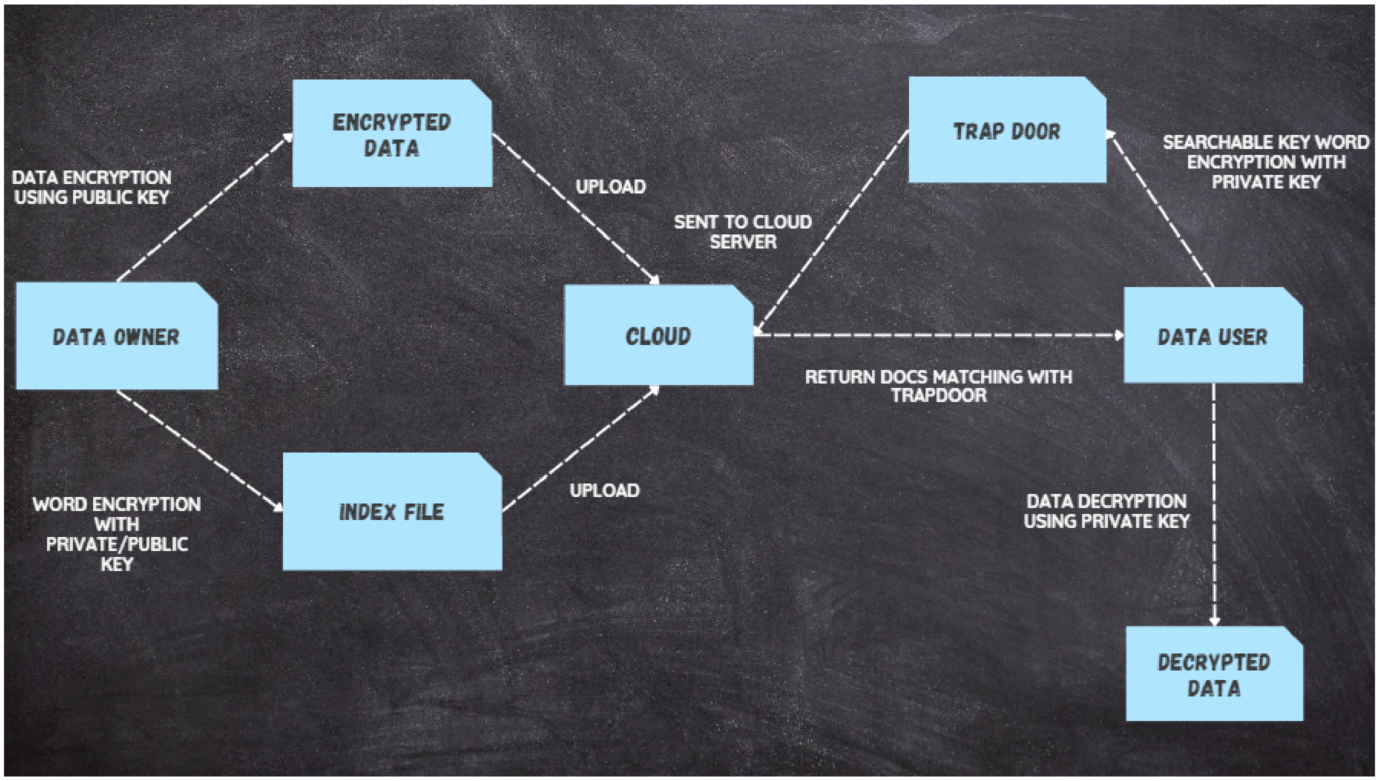
ENCRYPTED DATA
DATA ENCRYPTION USING PUBLIC KEY
UPLOAD
TRAP DOOR
SEARCHABLE KEY WORD ENCRYPTION WITH PRIVATE KEY
SENT TO CLOUD SERVER
DATA OWNER
CLOUD
DATA USER
RETURN DOCS MATCHING WITH TRAPDOOR
WORD ENCRYPTION WITH PRIVATE/PUBLIC KEY
INDEX FILE
UPLOAD
DATA DECRYPTION USING PRIVATE KEY
DECRYPTED DATA

Fig. 8. **Implemented Scheme**

IV. Literature review of paper: Certificateless authenticated encryption with keyword search

Introduction

The introduction starts by talking about how important the Industrial Internet of Things (IIoT) is in our world today, especially in industries. It mentions that by the year 2020, millions of devices would be connected to networks every second, showing how big of a deal IIoT is. It explains that cloud computing, which offers benefits like lots of storage space and fast computing, is used to handle the huge amount of data produced by IIoT.

However, there's a problem. Even though cloud computing is great for storing data, it doesn't guarantee that the data will be kept private. This is because cloud servers, where the data is stored, can't always be trusted to keep data safe. So, before sending data to the cloud, it needs to be encrypted, or coded, to keep it private.

But there's a catch with encryption. While it makes data private, it also makes it difficult to search through or share. This is where something called "searchable encryption" comes in. It's a type of encryption that keeps data private but still allows searching and sharing.

The first form of searchable encryption was called Public Key Encryption with Keyword Search (PEKS), but it had some problems. One of these problems was called the "Keyword Guessing Attack" (KGA), where someone could figure out what a searched keyword was just by looking at the encrypted data. To fix this, a new type of encryption was introduced called "searchable public key encryption with a designated tester" (dPEKS), which was better at protecting against these attacks.

Later on, even better forms of encryption were proposed, like Public Key Authenticated Encryption with Keyword Search (PAEKS). These were good at protecting against attacks from both inside and outside sources.

But there was still a problem with these encryption methods called the "certificate management problem." To solve this, new types of encryption, like Certificateless Encryption with Keyword Search (CLEKS) and Certificateless Authenticated Encryption with Keyword Search (CLAEKS), were introduced.

While these new encryption methods are promising, they still have some weaknesses. So, the goal of this paper is to improve upon these methods by proposing a new and better CLAEKS scheme that is more secure and efficient. This sets the stage for the rest of the paper, which will go into more detail about how this new encryption scheme works and why it's important.

Definition and Security Model of CLAEKS

The scheme involves several entities: the Key Generation Center (KGC), data senders, receivers, and the cloud server. Eight essential algorithms are integral to its operation, including Setup, ExtractPartialPrivateKey, SetSecretValue, SetPrivateKey, SetPublicKey, CLAEKS, TrapdoorGen, and Test.

Ciphertext Security pertains to making it computationally arduous for attackers (A1 or A2) to differentiate between ciphertexts of specific keywords. It encompasses two interactive games tailored to each attacker type, delineating this security aspect.

Trapdoor Security aims to thwart adversaries from distinguishing between trapdoors of designated keywords. Again, two games, aligned with attacker types, elucidate this security facet.

The Proposed CLAEKS Scheme

In this section, we present the detailed algorithms comprising the CLAEKS scheme, each expressed with mathematical formulas where applicable.

A. Setup Algorithm:

Performed by the Key Generation Center (KGC) with the security parameter λ as input.

- **Process:**
  1) Choose two cyclic groups G_1 and G_2 of prime order $q > 2\lambda$ and a bilinear map $e : G_1 \times G_1 \to G_2$.
  2) Choose $s \in \mathbb{Z}_q^*$ randomly as the master secret key and a generator $P \in G_1$, then compute $P_{\text{Pub}} = sP$ as the master public key.
  3) Choose three cryptographic hash functions $H : \{0,1\}^* \to G_1$, $h_1 : \{0,1\}^* \times G_1 \times \{0,1\}^* \times G_1 \times G_2 \times G_1 \times \{0,1\}^* \to \mathbb{Z}_q^*$, and $h_2 : \mathbb{Z}_q^* \times G_1 \to \mathbb{Z}_q^*$.
- **Output:** The master secret key s, secured by KGC, and the system parameters $\text{prms} = (\lambda, G_1, G_2, e, q, P, P_{\text{Pub}}, H, h_1, h_2)$ to be published.

B. ExtractPartialPrivateKey Algorithm:

Performed by KGC.

- **Input:** prms, master secret key s, and a user's identity $\text{ID}_u \in \{0,1\}^*$.
- **Process:** Compute $DID_u = sH(\text{ID}_u)$.
- **Output:** Partial private key DID_u, to be securely sent to the user with identity ID_u.

C. SetSecretValue Algorithm:

Performed by each user of the system.

- **Input:** prms and user's identity ID_u.
- **Process:** Select a random value $x_{\text{ID}_u} \in \mathbb{Z}_q^*$ as the user's secret value.
- **Output:** x_{ID_u}, secured by the user.

D. SetPrivateKey Algorithm:

Performed by each user of the system.

- **Input:** The user's secret value x_{ID_u} and their partial private key DID_u.
- **Process:** Set $\text{SK}_{\text{ID}_u} = (x_{\text{ID}_u}, DID_u)$ as the user's private key.
- **Output:** SK_{ID_u}, secured by the user.

E. SetPublicKey Algorithm:

Performed by each user of the system.

- **Input:** prms and the user's secret value x_{ID_u}.
- **Process:** Compute $PID_u = x_{\text{ID}_u}P$.
- **Output:** PID_u, to be published.

F. CLAEKS Algorithm:

Performed by the data sender.

- **Input:** prms, data sender's identity IDS, their public key P_{IDS} and private key $\text{SK}_{\text{IDS}} = (x_{\text{IDS}}, D_{\text{IDS}})$, receiver's identity IDR, their public key PID_{R}, and a keyword w.
- **Process:**
  - Choose a random number $r \in \mathbb{Z}_q^*$.
  - Compute:

$$K_1 = e(D_{\text{IDS}}, H(\text{IDR})),$$

$$K_2 = x_{\text{IDS}} PID_{\text{R}},$$

$$C_1 = rP,$$

$$C_2 = rh_2(h_1(\text{IDS}, P_{\text{IDS}}, \text{IDR}, PID_{\text{R}}, K_1, K_2, w), C_1)P.$$

- **Output:** The searchable ciphertext $CT = (C_1, C_2)$, to be uploaded to the server.

G. TrapdoorGen Algorithm:

Performed by the receiver.

- **Input:** prms, data sender's identity IDS and their public key P_{IDS}, receiver's identity IDR, their public key PID_{R}, their private key $\text{SK}_{\text{IDR}} = (x_{\text{IDR}}, d_{\text{IDR}})$, and a keyword w.
- **Process:**
  - Compute:

$$K_1 = e(H(\text{IDS}), D_{\text{IDR}}),$$

$$K_2 = x_{\text{IDR}} P_{\text{IDS}},$$

$$T_w = h_1(\text{IDS}, P_{\text{IDS}}, \text{IDR}, PID_{\text{R}}, K_1, K_2, w).$$

- **Output:** T_w

Security Analysis

The security analysis of the proposed CLAEKS scheme involves two theorems and several lemmas which are listed below:

Theorem 1: This theorem states that under GBDH (Generalized Bilinear Diffie-Hellman) and CDH (Computational Diffie-Hellman) assumptions, the CLAEKS scheme provides ciphertext indistinguishability against an adaptive chosen keyword attack. This theorem is proven through Lemmas 1 and 2.

Lemma 1: This lemma establishes that if a type 1 attacker can break the CLAEKS scheme with non-negligible advantage during Game I, then a PPT algorithm B can be constructed to solve the GBDH problem with non-negligible advantage.

Lemma 2: Similarly, this lemma states that if a type 2 attacker can break the CLAEKS scheme during Game II with non-negligible advantage, then a PPT algorithm B can be constructed to solve the CDH problem with non-negligible advantage.

Theorem 2: Under GBDH and CDH assumptions, this theorem asserts that the CLAEKS scheme provides trapdoor indistinguishability against an adaptive chosen keyword attack. This is proven through Lemmas 3 and 4.

Lemma 3: If a type 1 attacker can break the CLAEKS scheme during Game III with non-negligible advantage, then a PPT algorithm B can be constructed to solve the GBDH problem with non-negligible advantage.

Lemma 4: Similarly, if a type 2 attacker can break the CLAEKS scheme during Game IV with non-negligible advantage, then a PPT algorithm B can be constructed to solve the CDH problem with non-negligible advantage.

The security analysis demonstrates that the CLAEKS scheme is secure against various types of attacks under the assumptions of GBDH and CDH. These results provide a strong foundation for the security claims of the proposed scheme.

V. Code for CLAEKS Scheme

You can view the implementation of the CLAEKS scheme on GitHub at https://github.com/ROHIT32767/CLAEKS.

References

[1] Y. Zhou, N. Li, Y. Tian, D. An, and L. Wang, "Public Key Encryption with Keyword Search in Cloud: A Survey," *Entropy*, vol. 22, no. 4, 2020. [Online]. Available: https://www.mdpi.com/1099-4300/22/4/421

[2] Y. Zhang, Y. Li, and Y. Wang, "Secure and Efficient Searchable Public Key Encryption for Resource Constrained Environment Based on Pairings under Prime Order Group," *Security and Communication Networks*, vol. 2019, p. 5280806, 2019. [Online]. Available: https://doi.org/10.1155/2019/5280806

[3] Y. Zhang, Y. Wang, and Y. Li, "Searchable Public Key Encryption Supporting Semantic Multi-Keywords Search," *IEEE Access*, vol. 7, pp. 122 078–122 090, 2019.

[4] A. Bisht, A. K. Das, D. Niyato, and Y. Park, "Efficient Personal-Health-Records Sharing in Internet of Medical Things Using Searchable Symmetric Encryption, Blockchain, and IPFS," *IEEE Open Journal of the Communications Society*, vol. 4, pp. 2225–2244, 2023.