Open Journal of Physical Science (OJPS)
ISSN: 2734-2123
Article Details:
DOI: 10.52417/ojps.v4i1.471
Article Ref. No.: OJPS0401003-418
Volume: 4; Issue: 1, Pages: 34 - 53 (2023)
Accepted Date: 31st May, 2023

Open Journals Nigeria (OJN)
Open Access | Bi-annual | Peer-reviewed
www.openjournalsnigeria.org.ng
editorial@openjournalsnigeria.org.ng

RESEARCH ARTICLE **OJPS0401003-418**

PUBLIC KEY ENCRYPTION SCHEME BASED ON KEYWORD SEARCH

Soroush, A. M.

Computer Science Department, Shanxi University, China

***Corresponding Author Email:** mahdisoroush2016@gmail.com; 008619935474100*

ABSTRACT

This study explores the concept of a Public Key Encryption Scheme (PKES) with an emphasis on enabling keyword search functionality. The aim of this research is to develop a cryptographic framework that allows users to securely search over encrypted data without revealing sensitive information. The study investigates various methods and techniques employed in PKES to achieve efficient and privacy-preserving keyword search capabilities. The research highlights the increasing importance of data privacy and the need for secure information retrieval over encrypted data. Traditional encryption methods hinder search functionality, making it challenging to perform keyword searches on encrypted data. To address this limitation, the study focuses on PKES, which facilitates search operations while preserving the confidentiality of the underlying data. Experimental research design was used for this research. To accomplish the objectives, the study employs a combination of cryptographic algorithms and data transformation techniques. Public key encryption algorithms are utilized to secure the data, ensuring that only authorized users can access the information. Additionally, searchable encryption methods, such as Searchable Symmetric Encryption (SSE) and Index-Based Encryption (IBE), are explored to enable efficient and secure keyword searches on encrypted data. The results of the research demonstrate the feasibility and effectiveness of the proposed PKES with keyword search. The developed framework allows users to encrypt their data and store it securely, while still being able to search for specific keywords without decrypting the data. The experiments conducted on different datasets showcase the efficiency of the implemented techniques, providing a practical solution for secure data retrieval. In conclusion, the study presents a robust and privacy-preserving Public Key Encryption Scheme that incorporates keyword search capabilities. The research contributes to the field of cryptography by addressing the limitations of traditional encryption methods and offering an effective solution for secure information retrieval. The findings highlight the significance of PKES in safeguarding sensitive data while allowing users to perform keyword searches conveniently.

Keywords: *Cryptanalysis, Generic construction, Keyword search, Post-quantum, Trapdoor privacy, Public-key authenticated encryption.*

INTRODUCTION

Public-key asymmetric cryptography also referred to as encryption, involves the study of cryptographic systems that use a pair of similar keys. The key pair consists of public and private keys that are generated using cryptographic techniques based on one-way functions (Shirey, 2007; Bernstein & Lange, 2017). To ensure security, the private key in a public-key cryptography system must be safeguarded, while the public key can be distributed widely without posing any security risks (Stallings, 1990). With the use of public-key encryption, any individual who has access to a public key can make use of it to encrypt a message and create a ciphertext. But, decryption of the ciphertext and revealing the original message is possible only for those holding the relevant private key (Menezes *et al.*, 1997). In a digital signature mechanism, a sender can sign a message with the help of a private key. Other individuals cannot forge any pairing of messages or signatures that could pass public key validation using the private key. Nevertheless, individuals in possession of the corresponding public key can confirm the match of the message and signature (Menezes *et al.*, 1997; Bernstein, 2008).

Most cloud servers are believed to be dependable but daring. To maintain the security of these data and the ability to search for data, cloud storage must give functional and trustworthy solutions. Data transmission is critical to combat illegal institution penetration (Aloqaily, *et al.*, 2019; Otoum, *et al.*, 2017), especially for critical infrastructure. Encrypting sensitive data before outsourcing is a simple way to protect the information, but it makes it difficult for users to search for a specific phrase in the data. At the time, two methodologies were used, both of which are no longer acceptable.

Key management can be difficult in multiuser systems. To remedy this issue, it was suggested that PEKS plans be used. As a result, Public Key Encryption with Keyword Search (PEKS) methods are more commonly used than Searchable Symmetric Encryption (SSE) techniques. Furthermore, the bulk of these solutions needs the establishment of a secure communication channel, which is extremely difficult to execute in real-world circumstances. PEKS designs were proposed as a solution to this problem. In 2004, Boneh and colleagues devised the original PEKS (Public-key Encryption with Keyword Search) method by combining searchable encryption and public-key encryption (Boneh *et al.*, 2004). This method employs the public keys of recipients to generate encrypted keywords or searchable ciphertext. Afterward, the recipient can use their private key to create a trapdoor and send it to a cloud server to search for matching ciphertexts. The introduction of PEKS not only provided a new encryption method but also highlighted the importance of ensuring the security of the ciphertext. Consequently, Boneh *et al* (2004). suggested the concept of ciphertext indistinguishability (CI) or ciphertext indistinguishability against selected keyword attacks (CKA) to prevent malicious entities from extracting keyword data from the ciphertext. By incorporating CI and CKA in PEKS, the security of the method is enhanced, ensuring that private and confidential data is not vulnerable to attacks (Boneh *et al.*, 2004.

Hence, the consideration of CKA alone, as noted by Byun *et al.* (2006), is insufficient. The attacker can employ ciphertexts generated adaptively to test and guess keywords, thereby retrieving keyword information from the trapdoor. They also proposed the idea of trapdoor privacy (TP), which assesses indistinguishability from keyword guessing attacks (KGA), to account for this attack scenario (Emura *et al.*, 2015). Two types of KGA can be distinguished, namely, inner

KGA (IKGA) and outer KGA, launched by internal adversaries such as malicious cloud servers (Byun *et al.*, 2006). According to Byun *et al.* (2006), the keyword space in PEKS systems is constrained and small. The probability that the adversary will successfully use a brute force attack to obtain the keyword information concealed by the trapdoor is therefore quite high.

Although there have been numerous KGA-secure PEKS schemes developed (Emura*et al.*, 2015; Jeong, *et al.*, 2009; Fang, *et al.*, 2009); Jiang *et al.*, 2016), the issue of Inner Key Generation Algorithm (IKGA) in the single-server context was finally resolved without sender-to-receiver communication by Huang and Li's (2017) PAEKS concept, which allows for public-key authenticated encryption with keyword search. Their theory states that only verified senders may use the trapdoor created by the receiver; thus, the adaptively-produced ciphertexts for any keyword cannot be tested by adversaries using KGA. PAEKS eliminates privacy concerns and has resulted in multiple variations of PAEKS schemes being presented to suit various scenarios (Yang and Jiguo, 2021; Yang *et al.*, 2019).

The focus of this research revolves around creating a generic Public Key Encryption with Keyword Search which includes the use of its algorithms, models, attack models, applications, and classifications. Additionally, the research addresses the issues regarding lattice-based public-key encryption by employing a method that can defend against inside keyword guess attacks (IKGAs). These IKGAs have been a point of concern in previous research, where Zhang *et al.* (2021) investigated the use of the defense mechanism. Overall, the paper provides a comprehensive understanding of Public Key Encryption with Keyword Search and proposed a generic Public Key Encryption with Keyword Search.

GENERAL KNOWLEDGE OF PUBLIC KEY ENCRYPTION WITH KEYWORD SEARCH

This section provides a thorough overview of Public Key Encryption with Keyword Search, including its algorithms, models, attack models, and applications.

The PEKS broad structure, which consists of the user, the data owner (DO), and the virtual server, was suggested by Boneh *et al.* (2004). Before moving the files to the virtual server, the DO first encrypts the files and securely indexes them using the user's public key. Before launching a search request to the server, the user must have the private key to the matching public key. Using their private key, the user creates a real trapdoor and forwards it to the virtual server. Thereafter, the virtual server determines if the search query has any matches in the index after receiving it. If so, the user receives the classified content in their search results from the server. The confidential documents can then be decrypted by the user. PEKS is appropriate for open networks (including insecure networks).

PUBLIC KEY ENCRYPTION WITH KEYWORD SEARCH ALGORITHMS

Every public key system, in theory, might be subject to a "brute-force key search assault" (Yang, 2015). However, if all potential attackers lack the computational power required to succeed, which Claude (2015) referred to as the "work factor," such an attack becomes impractical. The effort factor may usually be improved by just picking a longer key. However, some strategies may already be significantly less computationally costly than others, rendering protection against a brute-force assault (for example, employing longer keys) irrelevant. To assist in defeating various public key

encryption schemes, some unique algorithms have been created. The four algorithms listed below are typically included in each PEKS scheme (Boneh *et al.*, 2004):

KeyGen(): Using a security parameter, this technique creates a public key and a private key (PK, SK). This key-generating process is used by the person who receives the search results.

PEKS (PK, W): The S code is produced using an encryption method that requires the public key of the PK user and a W keyword as input. Significantly, the data owner always executes the PEKS algorithm.

Trapdoor (SK, W): The method needs to input the private key of an SK user and the query term W to create a trapdoor. It then creates the output TW and sends it to the server. The individual who receives the search results is in charge of running the trapdoor algorithm.

Test (PK, S, TW): The public PK key, the ciphertext S, and the plaintext TW are all inputs into the search algorithm Test (PK, S, TW). The procedure returns the answer "Yes," and the user is given the pertinent documents if W is discovered during the search process. In that case, the test algorithm will reply, "No.".

PUBLIC KEY ENCRYPTION WITH KEYWORD SEARCH SECURITY MODELS

The notion of Indistinguishability against a chosen keyword assault (IND-CKA) was introduced by Boneh *et al*. (2004) by developing a game that pitted attacker A against challenger C. The game involves C initially running KeyGen(), which generates a key pair (PK, SK). C keeps the SK secret while giving PK alone to A. This game setup serves as the basis for IND-CKA.

Subsequently, in the first step of the game, adversary A has the flexibility to request information from challenger C specifically about the keywords that are of their choosing.

During the game, Challenger C is given two new keywords, W_0 and W_1, by Attacker A. C chooses a random bit b and generates a $C = PEKS(PK, Wb)$ ciphertext which is then sent to A.

In the second step of the game, Adversary A can request any additional term other than W_0 *and* W_1 *in Phase* 1. If A correctly guesses the bit b_0 and it is the same as the bit chosen by C, then he wins the game.

The probability of A guessing correctly is denoted as $A(s) = Pr\,[b = b0]\ 1/2$ (where the absolute value of the probability is considered). Therefore, the PEKS IND-CKA strategy is considered secure, even if Adversary A has a slight advantage in the game.

PUBLIC KEY ENCRYPTION WITH KEYWORD SEARCH ATTACK MODELS

The attack models that have been introduced for PEKS schemes specify that file injection and keyword guessing attacks are a risk for these schemes. In the subsequent paragraphs, we will review these attack models for PEKS techniques.

1. FILE INJECTION ATTACK

The first file injection attack on searchable encryption was developed by Zhang and Zhang (2011). The attack becomes particularly perilous when an internal attacker, in the form of a rogue cloud server, injects specific data to gain access to critical information regarding a particular keyword query.

The file injection attack reveals the access pattern, which violates privacy by giving the attacker access to a huge amount of sensitive information. As a result, it is critical to develop PEKS methods that are both successful and resistant to this attack.

2. KEYWORD GUESSING ATTACK

Boneh *et al.* (2004) were the first to use a keyword guess attack against various PEKS techniques. Due to the limited keyword space, this approach is considered to be a practical one. With this strategy, the attacker can successfully get the encrypted keyword. External attacks occur when an external attacker is present, while internal assaults occur when an internal attacker is present (Lu, 2019).

AREAS OF APPLICATION OF PUBLIC KEY ENCRYPTION WITH KEYWORD SEARCH

The potential applications of PEKS are scenarios where confidential information is stored in the cloud by a third-party server provider and the usability of the stored data is maintained have garnered significant attention in the field of searchable encryption research. This section delves into various PEKS applications, including email routing, healthcare, smart grid, and more.

1. E-MAIL ROUTING

For email routing circumstances, Boneh *et al.* presented the first PEKS algorithm in 2004. Assume that Emeka sends Ada an email in this situation that utilizes an unreliable mail server and contains specific keywords. The server is in charge of sending the email to Ada's suitable device based on the keyword data even if it is unable to view the email body and associated keywords. As an illustration, if the keyword is "urgent," the server will send the message to Ada's cell phone, and if it is "lunch," it will go to her laptop. Ada is free to use any device to access the message. The recipient's public key must be used to encrypt the email for privacy. PEKS might be used to search encrypted email messages.

2. HEALTHCARE

Healthcare providers are progressively turning to third-party cloud services to save on costs when storing electronic medical records and application services, all while prioritizing the protection of sensitive data. As healthcare professionals use a patient's information for diagnosis within electronic healthcare systems and share it with other medical staff, they also endeavor to encrypt that data before sending the data to the remote cloud server.

CLASSIFICATION OF EXISTING PUBLIC KEY ENCRYPTION WITH KEYWORD SEARCH SCHEMES

The PEKS technique has been presented to examine key management for SSE. This section's goal is to present a rundown of the numerous PEKS schemes that have been published to date as well as an overview of the research and documentation that has been generated. We categorize PEKS systems into six groups to achieve this purpose;

PUBLIC KEY ENCRYPTION WITH KEYWORD SEARCH BASED ON PUBLIC KEY INFRASTRUCTURE (PEKS-PKI)

Many PEKS techniques involve managing digital certificates through a public key infrastructure (PKI). With these approaches, sending parties, like Ada in this example, use the recipient's digital certificate to verify their public key before encrypting the data and keywords with that key. The encrypted data is then transmitted to a cloud server for storage. On receipt, the intended recipient, like Emeka, can use their secret key to locate matching keywords in the encrypted data.

Typically, a trustworthy third party creates these digital public key certificates within a PKI system. Boneh *et al.* (2004) first introduced a PKI-based method that uses a bilinear pairing map, but transmitting the trapdoor requires a secure communication channel which can be difficult and costly.

Baek *et al.* (2008) proposed a different PEKS system that eliminates the need for encrypted communication, making it a more efficient PKI solution. However, this scheme still relies on a server to have a pair of public and private keys, and only the chosen server can execute the test algorithm. Security for this scheme falls within the Oracle random model under the BDH assumption.

PUBLIC KEY ENCRYPTION WITH KEYWORD SEARCH BASED ON IDENTITY-BASED ENCRYPTION (PEKS-IBE)

Shamir (1984) initially proposed Identity-Based Encryption (IBE) encryption in 1984 as a solution to streamline certificate administration based on PKI by identifying the public key holder by different user variables like their email or phone number. The private key is produced using this manner after user authentication. The sender can encrypt data using the PEKS-IBE technique and send it to a cloud server while the recipient authenticates themself with the Public Key Generator (PKG). Their private key is used to create a trapdoor, which is sent to the cloud server, which then does a keyword search on it. Boneh *et al.* (2004) presented the first PEKS-IBE method in which the term serves as an identifier. The first PEKS-IBE strategy based on Jacobi symbols and the quadratic residual problem without bilinear mappings was introduced by Di Crescenzo and Saraswat in 2017. The implementation of this plan would need a secure communication connection and a lot of storage, though (Boneh *et al.*, 2004).

Public Key Encryption with Keyword Search Based on Attribute-Based Encryption (PEKS-ABE)

As a method of identifying based on collections of qualities, Sahay and Waters (2005) introduced Attribute-Based Encryption (ABE). These attributes serve as settings for user information. Using conventional techniques, a sender must create numerous ciphertexts from a single document to encrypt it using the public keys of different recipients.

The recipient then uses their private key to decrypt their text. With ABE, however, the sender may generate a single ciphertext by employing a single access policy, which many consumers can use to decrypt the encrypted file. If the user's characteristics match the relevant access criteria, decryption is permitted. The access policy is linked to the private key holder. Ciphertext-policy attribute-based encryption (CP-ABE) and key-policy attribute-based encryption (KP-ABE) are the two main varieties of attribute-based encryption.

Public Key Encryption with Keyword Search Based on Predicate Encryption (PEKS-PE)

Predicate encryption, a novel type of public key encryption that enables users to do searches without needing to know the private key, was first proposed by Katz *et al.* (2008). Depending on the access control strategy, precise access control over the encrypted data enables predicate encryption to hide concealed information and messages. The receiver receives the results without providing any more information to the server after the server uses the matching algorithm to link a trapdoor with encrypted indices. The resultant encrypted messages that relate to the characteristics can only be received and decrypted using this encryption method by the private key holder. Zhang and Lu created a PEKS-PE technique in 2014 that can not only look for disjunctive keywords but also conjunctive keywords.

PUBLIC KEY ENCRYPTION WITH KEYWORD SEARCH BASED ON CERTIFICATELESS ENCRYPTION (PEKS-CLE)

Certificateless encryption, a fresh take on public key encryption based on IBE (Identity-Based Encryption), was first developed by AI-Riyami and Paterson (2003). The distinctive aspect of this method is that to produce the private key, the user and PKG must work together to generate the secret key. A PEKS-CLE technique is one type of certificateless encryption that encrypts both the keyword and the data using the recipient identity and public key before transmitting the encrypted data to the server. PKG initially sends the receiver a partial private key; the recipient then combines that partial key with a secret value of their choosing to create the full private key. The trapdoor is then created by the receiver and sent to the server. Since PKG does not know the private key, certificateless encryption avoids storing keys in identity-based encryption. Yanguo *et al.* (2014)presented a PEKS-CLE scheme that was immune to keyword guessing and chosen keyword attacks and did not require a secure channel of communication.

PUBLIC KEY ENCRYPTION WITH KEYWORD SEARCH SUPPORTING PROXY RE-ENCRYPTION (PEKS-PRE)

At EUROCRYPT'98 (The 17th Annual International Conference on the Theory and Application of Cryptographic Techniques, which took place in Espoo, Finland, from May 31 to June 4, 1998), Blaze *et al.* (1998) initially introduced this encryption approach. The semi-trusted third party is the proxy in this approach, responsible for transforming ciphertext. Notably, the sender of the material, after re-encrypting the key, may assign its searching right to a delegate without disclosing their private key. The method is designed so that the proxy does not have access to the required unencrypted data. A proxy re-encryption method called PEKS-PRE enables searching encrypted material without decrypting it. As a proxy, the server converts encrypted information into encrypted text that the delegate may examine. A trapdoor is sent to the proxy server for the verification procedure, and the data stays with the transmitter so the recipient may sift through it.

RESEARCH METHOD

The experimental method was used in the implementation of the Public Key Encryption Scheme Based On Keyword Search Construction. This approach involves the design and implementation of the proposed encryption scheme, followed by the performance of experiments to evaluate its performance. This method provides valuable insights into the practical feasibility of the proposed scheme, in terms of its efficiency and security properties.

In employing the experimental method, several steps can be taken. The first step is the design and general construction of PAEKS in adherence to PEKS and SPHF schemes. This involves defining the mathematical framework of the construction, selecting appropriate algorithms and parameters, and implementing the encryption and decryption processes.

The next step is to perform experiments to evaluate the correctness and security analysis of the encryption scheme. This involves creating a series of games that measures if the PEAKS construction satisfies the requirements of CL and TP. This involves performing attacks on the encryption scheme, such as brute-force attacks, dictionary attacks, and chosen-plaintext attacks. The results of these experiments were used to evaluate the security of the encryption scheme and to identify weaknesses that need to be addressed.

One advantage of employing the experimental method is that it provides a practical evaluation of the proposed encryption scheme. This can be useful in assessing its real-world performance and identifying potential issues that may not be captured by mathematical analysis or simulation. However, it is important to carefully design and conduct the experiments to ensure that the results are accurate and reliable.

PAEKS CONSTRUCTION

This section outlines a general construction of PAEKS in adherence to the following two schemes: PEKS and SPHF.

- The proposed PEKS scheme, denoted as $PEKS = (KeyGen, PEKS, Trapdoor, Test)$, has a keyword space KS_{PEKS} that fulfills CI.
- Additionally, an $\epsilon - approximate, word - independent, and\ pseudo - random\ SPHF\ scheme$, $SPHF = (HashKG, ProjKG, Hash, ProjHash)$, is required, which has an output length of $\{0,1\}c$ $for\ the\ language\ of\ the\ ciphertext\ of\ a\ CCA2 - secure\ labeled\ PKE$ scheme, $PKE = (KeyGen, Encrypt, Decrypt).\ PKS_{PKE}\ is\ the\ public\ key\ space, and\ PS_{PKE},$ is the $plaintext\ space$, where ϵ has negligible value.

LANGUAGE OF CIPHERTEXT

Consider (Ipar, Itrap) $= (ek_{PKE}, dk_{PKE})$, where ek $k_{PKE} \in$ PKS_{PKE} and dk_{PKE} is its corresponding decryption key. . $The\ language\ of\ ciphertexts\ is\ defined\ as$: $\tilde{\mathcal{L}} :=$ {(label, ct PKE, mPKE) | $\exists \rho$, ct $_{PKE} \leftarrow$ Encrypt(ekPKE, label, mPKE; ρ)} and $\mathcal{L} :=$ {(label, ct PKE, MPKE) Decrypt (dk_{PKE}, label, ct PKE) $= m_{PKE}$}, $where\ the\ witness\ relation\ \widetilde{K}$ is implicit defined as: $\widetilde{K}$((label, ctPKE, MPKE), ρ) $= 1$ if and only if $ct_{PKE} \leftarrow$ Encrypt(ekPKE, label, mPKE; ρ).

The complete structure is detailed as follows:

During the setup phase, denoted as Setup(1^{λ}), this algorithm performs the subsequent operations given a security parameter λ.

- PKE.KeyGen(1^{λ}) generates the key pair (ek,eKE) for public key encryption and stores it as an output.
- A plaintext message m_{PKE} is randomly chosen from the space of valid plaintexts $\longleftarrow^{\$} PS_{PKE}$, along with a label label from $\{0,1\}^{*}$.
- This algorithm also chooses two functions $H_1: PKS_{PKE} \times PS_{PKE} \times \{0,1\}^{*} \rightarrow PKS_{PKE}$ and $H_2: KS_{PEKS} \times \{0,1\}^{*} \rightarrow KS_{PEKS}$, modeled as random oracles, responsible for assigning key shares to the public and private key components.
- The output public parameter, denoted pp, is a tuple consisting of $\lambda, mpk, and\ ek$.

- $KeyGen_S(pp)$: This algorithm undertakes the following operations after receiving the public parameter "pp":

- First, it verifies whether the equality mpk $=^{?} H_1$ (ekPKE, mPKE, label) holds. If the equation is not satisfied, the process is aborted.
- Next, it computes hp_Susing the hash key generation function $SPHF.ProjKG$ on mpk.
- Then, it derives hp_S, by applying SPHF.ProjKG to hks and mpk.
- Subsequently, the algorithm generates the ciphertext ct $_{PKE,S}$ by invoking PKE.Encrypt with inputs mpk, label, m_{PKE}, and the randomly chosen witness ρ_S satisfying $\tilde{K}((label, ct_{PKE}), \rho S) = A$.
- Finally, it produces the public key pk_S by concatenating $(hp_S, and\ ct_{PKE,S})$ and assigns it to the sender. The private key sk $_S$: $contains\ the\ pair\ (hk_S, \rho_S)$.

- $KeyGen_R(pp)$: This algorithm starts with the public parameter pp as input and proceeds as follows:

- It first verifies whether $mpk\ =^{?} H_1(ek_{PKE}, m_{PKE}, label)$. holds. If this condition is not satisfied, the algorithm terminates.
- The next step involves computing hk_R using the hash key generation function $SPHF.HashKG$ on mpk.
- Afterwards, it derives hk_R *by applying* $SPHP.ProjKG$ *to* Hk_R *and* mpk.
- *Then, the algorithm generates the ciphertext* $ct_{PKE,R}$ by calling $PKE.Encrypt$ with inputs $mpk, label, mPKE$, and the randomly chosen witness ρ_R that satisfies K̃((label, ct PKE,R, m_PKE),ρ_R)=1.
- Subsequently, it generates the pair (pk_{PEKS}, sk $_{PEKS}$) using PEKS.KeyGen with parameter (1^{λ}).
- Consequently, the public key pk_R is constructed by concatenating (hp_R,, ct $_{PKE,R}$ and pk_{PEKS}). On the other hand, the private key sk_R is computed as the tuple (hk_R, ρ_R, sk $_{PEKS}$).

- PAEKS(pp, pk_S, sk_S, pk_R, kw) : *Starting with the public parameter pp*, the *public key* pk_S and *the private key* sk_S of *the sender, the public key* pk_R *of* the receiver, *and a keyword* $kw' \in KS_{PEKS}$, *the algorithm carries out the following steps:*

    1. *Computes* $H_S \leftarrow SPH.Hash(hks, mpk, (ct_{PKE,R}, m_{PKE}))$ and $pH_S \leftarrow$ SPHF.ProjHash($hp_R, mpk, (ct_{PKE,S}, m_{PKE}), \rho_S)$
    2. Computes der-kws $\leftarrow H_2(kw, H_S \oplus pH_S)$.
    3. *Generates* ct PEKS,der-kw $_S$
    4. $PEKS.PEKS(pk_{PEKS}, der - kws)$.
    5. Outputs a searchable ciphertext $ct_{kw} := ct^S_{PEKS,der-kw}$.

- Trapdoor(pp, pk_S, pk_R, sk_R, kw') : Given the public parameter pp, *the public key* pk_S *of the sender, the public key* pk_R *and private key* sk_R *of the receiver, and a keyword* $kw' \in KS_{PEKS,}$, *this algorithm runs the following steps*:
- *Computes* $H_R \leftarrow SPHF.Hash(hk_R, mpk, (ct_{PKE,S}, m_{PKE})$ and $pH_R \leftarrow$ SPHF.ProjHash(hp_S, mpk, (ct $ct_{PKE,R}, m_{PKE}), \rho_R)$
- *Computes* der-kw $k'_R \leftarrow H_2(kw', H_R \oplus pH_R)$
- *Generates* $td_{PEAKS,der-kw'_R} \leftarrow PEKS.Trapdoor(sK_{PEKS,} der - kw'_R)$.

    - *Outputs a trapdoor* $td_{kw'} = td_{PEKS,der-kw'_R}$.

- $Test(pp, ct_{kw}, td_{kw'})$: *his algorithm takes as input the public parameter pp, the searchable ciphertext* ct_{kw}, *and* the trapdoor $td_{kw'}$, and it outputs the result of applying $PEKS.Test$ to $(ct_{kw}, td_{kw'})$.

Correctness of PEKS scheme

Assuming that the public parameter, denoted as p , and the public/private key pairs, denoted as $(pk_S, sk_S), (pk_R, sk_R)$, have been generated truthfully. Let ct_{kw} represent the searchable ciphertext that relates to the generated keyword kw sent by the transmitter, while $td_{kw'}$ represents the trapdoor relating to the keyword kw' generated by the receiver.

As the underlying SPHF is ϵ-correct for some $\epsilon = negl(\lambda)$, it follows that

$$H_S = SPHF \cdot Hash(hk_S, mpk, (ct_{PKE,R}, m_{PKE})) = SPHF.ProjHash(hp_S, mpk, (ct_{PKE,R}, m_{PKE}), \rho_R) = pH_R; \quad H_R = SPHF.ProjHash(hp_R, mpk(ct_{PKE,S}, m_{PKE}), \rho_S) = pH_S.$$

Therefore, $H_S \oplus pH_S = H_R \oplus pH_R$ *holds. Clearly,* if $kw = kw'$, then der-kw $w_S = H_2(kw, H_S \oplus pH_S) = H_2(kw', H_R \oplus pH_R)$, *and therefore* $ct_{PEKS,der-kw_S}$ *and* $td_{PEKS,der-kw'_R}$ *are related to the same extended keyword.*

As the underlying PEKS scheme is correct, $PAEKS.Test$ $(pp, c_{kw}, td_{kw'}) = 1$ holds

$with\ overwhelming\ probability.\ \ In\ contrast, since\ H_2\ is\ modeled$ as a random oracle, if $kw \neq kw'$, $then\ der - kw\ w_S = H_2(kw, H_S \oplus pH_S) \neq H_2(kw', H_R \oplus pHH_R) = der - kww'_R,$ $and\ therefore,$ $ct_{PEKS,der-kw_S}\ and\ td_{PEKS,der-kw'_R} are\ related\ \ to\ different\ \ extended\ keywords. Consequently,$ PAEKS.Test $(pp, ct_{kw}, td_{kw'}) = 0$ holds with overwhelming probability.

Security Analysis of PEKS Scheme

The suggested PEKS construction has been demonstrated to meet the requirements of CI and TP through Theorems 1 and 2, respectively, using the sequence-of-games method. To prove the security of structure, a series of games is created, starting with a game that closely resembles the actual attack scenario, wherein the attacker can only distinguish between the games with slight benefit. The series of games are designed such that the last game in the sequence is equivalent to the ideal game in which the attacker has no advantage in distinguishing between the games. Through this method, it is shown that the suggested PEKS construction satisfies the requirements of CI and TP. The success of this method in proving the security of the construction highlights its effectiveness in evaluating and analyzing the security of various PEKS schemes by progressively increasing the difficulty of the games in the sequence. Overall, Theorems 1 and 2 provide strong evidence for the security of the suggested PEKS construction and its potential for practical use in various applications that require secure storage and access to encrypted data.

For simplicity, let $Adv_A^{Game_i}(\lambda)$ indicate the benefit of A in game $Game\ _i, where\ i \in \{0, \cdots, 3\}$. In addition, Theorem 3 demonstrates that the suggested construction also meets the requirement of MCI.

The first theorem affirms that the PAEKS construction proposed in this study is secure, given that the underlying SPHF scheme is pseudo-random and modeled as a random oracle. The proof for this theorem is composed of four games, which are illustrated below;

Game $_0$: In this game, the random oracle is exchanged with a fixed, pre-determined function, which is known to the hacker . the challenges for the hacker is to still guess the is to still correctly guess the targeted keyword, despite the fixed function.

The actual $IND - CKA$ game described in Section 3.2 and this game are both equivalent. Imagine that the definition of A's benefit in this game is $Adv_A^{Game_0}(\lambda) := \epsilon$ Additionally, the $challenger\ C$ responds as follows when A asks for a specific keyword in order to mimic a real view for A.

- O_C: $For\ keyword\ kw, C\ computes\ ct_{kw} \leftarrow PAEKS(pp, pk_S, sk_S, pk_R, kw)$ and $returns\ ct_{kw}$ to A.
- O_T : For keyword kw, C computes $td_{kw} \leftarrow$ Trapdoor $(pp, pk_S, pk_R, sk_R, kw)$ and returns td_{kw} to A.

$Game\ _1$: Game 1 : here the random oracle is exchanged with a fixed, pre-determined function, which is known to the hacker. The challenge for the hacker is to still correctly guess the targeted keyword, despite the fixed function. $This\ game\ is\ identical\ to$ Game $_0$, $except\ for\ the\ generation\ of\ the$

$challenge\ ciphertext\ ct^*\ in\ the\ Challenge\ phase.\ More\ concretely, instead\ of\ generating\ H_S \leftarrow SPHF.Hash\left(hk_S, mpk, \left(ct_{PKE,R}, m_{PKE}\right)\right), C\ randomly\ chooses\ H_S$ $from\ the\ output\ space\ of\ the\ SPHF.Hash\ algorithm.\ Since\ the\ underlying\ SPHF$ $scheme\ satisfies\ pseudo-randomness, A\ cannot\ distinguish\ the\ view\ between$ $Game_0\ and\ Game_1. Therefore, we\ obtain$

$$\left|Adv_A^{Game_1}(\lambda) - Adv_A^{Game_0}(\lambda)\right| \leq negl(\lambda).$$

$Game_2$: $This\ game\ further\ changes\ the\ generation\ of\ the\ challenge$ ciphertext $ct*$ in the Challenge phase. $In\ this\ game, der-kw\ w_S\ is\ randomly\ from\ KS_{PEKS},$ indtead of $coxmputifng\ der-kws \leftarrow H_2(kw_b^*, H_S \oplus pH_S)\ for\ some\ b \in \{0,1\}$.

As H_S is randomly $chosen\ and\ H_2\ is\ modeled\ as\ random\ oracle, the\ output\ of\ H_2(kw_b^*, H_S \oplus pH_S)$ is $random. Therefore, A\ cannot\ distinguish\ the\ view\ between\ Game_1\ and\ Game_2$. consequently, we obtain PE KS

$$\left|Adv_A^{Game_2}(\lambda) - Adv_A^{Game_1}(\lambda)\right| \leq negl(\lambda).$$

$Game_3$: Game 3: The game is designed to be equivalent to an ideal game, in which the adversary cannot differentiate between the targeted keyword and a random string. In this game, the PEKS oracle is accepted with random oracle, and the challenge for the adversary is to guess the targeted keyword. This game is the last game. Because the challenge ciphertex $ct^* = ct_{PEKS,der-kws}$ is generated from $PEKS.PEKS(pk_{peks}, der-kws)$ and $der-kws\ is\ now\ randomly$ chosen from KS_{PEKS}, $the\ challenge\ ciphertext\ does\ not\ contain\ any\ information\ about\ the\ challenge\ keywords\ (kw_0^*, kw_1^*)$ given by $A.\ The\ only\ way\ for\ A\ is\ to\ guess$. Therefore, we have

$$Advv_A^{Game_3}(\lambda) = 0.$$

Incorporating the aforementioned games above, we have $\epsilon \leq negl(\lambda)$. The proof is completed. Through the sequence of games, it is demonstrated that the suggested PAEKS structure fulfills the requirements of CI, thereby indicating its potential for practical use in various applications that require secure storage and access to encrypted data.

THEOREM 2. $The\ proposed\ generic\ PAEKS\ construction\ satisfies\ TP$ if the $underlying\ SPHF\ scheme\ satisfies\ pseudo-randomness\ and\ H_2\ is\ modeled$ as a random oracle.

Proof: This proof used a similar structure as the evidence for Theorem 1 and involved four games.

Game 0: The first game, denoted as "Game 0," is similar as the real game $IND-IKGA$ game described in section 3.2. let's say the benefit of player A in this game is represented by " $Adv_A^{Game_0}(\lambda) := \epsilon$ " Also, note that the view that the challenger C simulates in this game is identical to the view in $Game\ 0$ of the proof for Theorem 5.1.

Proof. To demonstrate this, we will use a proof that is comparable to the one used for Theorem 1. However, this time we will use four games.

Game $_1$: In $Game_1$, the only difference from $Game_0$ is how the challenge trapdoor td^* is generated during the Challenge phase. Specifically, instead of using $H_R \leftarrow SPH.Hash\left(hk_R, mpk, \left(ct_{PKE,S}, m_{PKE}\right)\right)$, $C\ randomly$ chosen chooses H_R from the output of the $SPHF.Hash\ algorithm$.

Proof. The evidence for this statement is quite similar to that of Theorem 1 and also involves the examination of four separate games. Upon closer inspection, it can be observed that the understanding and analysis of these games play a crucial role in demonstrating the veracity of the theorem. By exploring and understanding the mechanisms at play in each game, it is possible to provide a robust and defensible argument for the correctness of the proposed result.

Game $_1$: $This\ game\ is\ identical\ to\ Game\ _0, except\ for\ the\ generation\ of\ the\ challenge\ trapdoor\ td^*$ in the Challenge phase. More concretely, instead of generating $H_R \leftarrow SPH.Hash\left(hk_R, mpk, \left(ct_{PKE,S}, m_{PKE}\right)\right)$, $C\ randomly$ chosen chooses H_R from the output space of the SPHF.Hash algorithm. A is unable to discriminate between the views of Game $_0$ and Game _1 since the underlying SPHF system fulfills pseudo-randomness. Consequently, we do have;

$$\left|Adv_A^{Game_1}(\lambda) - Adv_A^{Game\ 0}(\lambda)\right| \leq negl(\lambda).$$

Game $_2$:During the challenge phase, there is a modification to how the challenge trapdoor td^* is generated in this game. $In\ this\ gamne, der - kwR\ is\ randomly\ chosen\ from\ KS_{PEKS}$,$instead\ of\ by\ computing\ der - kw_R \leftarrow H_2(kw_b^*, H_R \oplus pH_R)$ for some $b \in \{0,1\}$. As $H_R\ is\ randomly\ chosen$, the output of $H_2(kw_b^*, H_R \oplus pH_R)$ is random.

$Therefore, A\ cannot\ distinguish\ the\ view\ between\ Game_1\ and\ Game_2$. Consequently, we obtain

$$\left|Adv_A^{Game_2}(\lambda) - Adv_A^{Game\ 1}(\lambda)\right| \leq negl(\lambda).$$

Game $_3$: $This\ game\ is\ the\ last\ game.\ Because\ the\ challenge\ trapdoor\ td^* = td_{PEKS,der-kw_R}$is genesrated from $PEKS.PEKS(pk_{PEKS}, der - kw_R)$ and $der - kw_R is\ now\ randomly$ chosen from from KS_{PEKS}, $the\ challenge\ trapdoor\ does\ not$ contain any information about the challenge keywords (kw_0^*, kw_1^*) given by A. The only way for A is to guess. Therefore, we have

$$Adv_A^{Game_3}(\lambda) = 0.$$

$Finally, combining\ the\ above\ games,$ we have $\epsilon \leq negl(\lambda)$. The proof is now complete.

THEOREM 3. The generic PAEKS design has been demonstrated to fulfill MCI when Theorem 1 and Theorem 2 are valid, and the underlying scheme is probabilistic, as is the case with many commonly utilized PEKS schemes."

When we bring together the findings of Theorem 1 and Theorem 2, we can establish that the suggested design satisfies the MCI (Message-Carried Information) requirement. This means that the information conveyed in the encrypted message remains consistent with the original message after decryption, and the decryption process does not reveal any additional information that was not present in the original message. In other words, the suggested construction can maintain message privacy and integrity throughout the encryption and decryption process.

LATTICE-BASED INSTANTIATION

We present a lattice-based scheme for PAEKS that is resilient to quantum attacks. This is achieved by using three lattice-based primitives, which build the foundation of our scheme and inherit their securities against quantum attacks. Our approach makes use of the labelled $IND-CCA1$ PKE scheme published by Micciancio and Peikert (2012), the word-independent SPHF scheme created by Li and Wang (2019), and the PEKS system proposed by Behnia *et al.* (2020). For the convenience of explanation, it is vital to remember that we are only presenting the $IND-CCA1$ variant of the PKE scheme (Micciancio and Peikert, 2012). Additionally, we want to emphasize that the labelled $IND-CCA1\ PKE$ can be converted to $IND-CCA2\ PKE$ using a one-time signature scheme.

DEFINE SOME IMPORTANT NOTATIONS

Before presenting our instantiation, we need to establish key notations. We define $\mathcal{R}$ as a ring, while U is a subset of $\mathcal{R}^{\times}$ comprising the $invertible\ elements$. Additionally, we set $G:=I_n\otimes g^{\top}$ as the matrix gadget defined in Micciancio and Peikert (2012). Here, $g^{\top}:=[1,2,\cdots,2^k]$ and $k:=\lceil log\ q\rceil-1$.. To complete the important notations, we introduce $he\ encoding\ function\ \ Encode(\mu\in\{0,1\}):=\mu.\ \ (0,\cdots,0,\lceil q/2\rceil)^{\top}$ and the $deterministic\ rounding\ function\ R(x):=\lfloor 2x/q\rceil mod 2$.. Finally, we refer to $[A\mid B]$ and $[A;B]=[A^{\top}\mid B^{\top}]^{\top}$ $as\ horizontal\ concatenation\ and\ vertical\ concatenation$, respectively, of $matrices\ A\ and\ B$.

The instantiation follows the Setup (1^{λ}) the procedure, which obtains its input from several parameters including $q,n,m,\sigma_1,\sigma_2,\alpha$ and α (set as per instructions in subsequent parameter selection section). This algorithm performs the following operations:

- First, it sets κ,ρ, and l to a polynomial (n) and randomly selects $=m_1m_2\cdots m_{\kappa}\longleftarrow^{\$}\{0,1\}^{\kappa}$.
- Next, it computes $(A_0,T)\leftarrow TrapGen(1^n,1^m,q)$.
- It then sets $kPKE:=A_0, dk_{PKE}:=T$, and m $_{PKE}:=m$.
- Random element $u\longleftarrow^{\$}U$ is chosen, and the label is set as "u".
- The algorithm proceeds to choose secure hash functions $H_1:\mathbb{Z}_q^{n\times m}\times\{0,1\}^{\kappa}\times U\rightarrow\mathbb{Z}_q^{n\times m}, H_2:\{1,-1\}^l\times\{0,1\}^{\kappa}\rightarrow\{1,-1\}^l$, $and\ an\ injective\ ring\ homomorphism\ h:\mathcal{R}\rightarrow\mathbb{Z}_q^{n\times n}$.
- Finally, it computes $A\leftarrow H_1(A_0,m,u)\in\mathbb{Z}_q^{n\times m}$ and $sets\ the\ mpk$ as A.
- $The\ output\ of\ the\ algorithm$ is $herefore\ pp:=(\lambda,n,m,q,\sigma_1,\sigma_2,\kappa,\rho,l,$ ekPKE $:=A_0,$ mpk :=A, $m_{PKE}:=m$, label := $u,H_1,H_2,h)$.

$KeyGen_S(pp)$: This algorithm executes the following steps based on the public parameter (pp).

- $Checks\ whether\ A =^{?} H_1(A_0, m, u)$.
- Computes $A_u = A + [0; Gh(u)]$, randomly chooses a matrix $h_S := k_S \leftarrow^{\$} D_{\mathbb{Z},s}^m$, and computes $hp_S := p_S = A_u^\top \cdot k_S \in \mathbb{Z}_q^n$, where $s \geq \eta_\epsilon\left(\Lambda^\perp(A_u)\right)$ for some $\epsilon = negl(n)$.
- For $i = 1, \cdots, \kappa$, randomly chooses vectors $s_{S,i} \longleftarrow^{\$} \mathbb{Z}_q^n$ as well as $e_{S,i} \leftarrow^{\$} D_{\mathbb{Z},t}^{m\ \$}$ (re-select $e_{S,i}$ if $\| e_{S,i} \| > 2t\sqrt{m}$), and computes $c_{S,i} = A_u^\top \cdot s_{S,i} + e_{S,i} + Encode(m_i) mod q$, where $t = \sigma_1\sqrt{m} \cdot \omega(\sqrt{log\ n}$.
- Outputs the public key $pk_S := (hp_S := p_S$, ct $_{PKE,S} := \{c_{S,i}\}_{i=1}^{\kappa})$ and the private key $sk_S := (hkp_S := k_S, \rho_S := \{s_{S,i}\}_{i=1}^{K})\ of\ the\ sender$.

$KeyGen_R(pp)$: This algorithm executes the following steps based on the public parameter (pp):

- $Checks\ whether\ A =^{?} H_1(A_0, m, u)$.
- Computes $A_u = A + [0; Gh(u)]$, randomly chooses a matrix $hk_R := k_R \leftarrow D_{\mathbb{Z},s}^m$, and computes $hp_R := p_R = A_u^\top \cdot k_R \in \mathbb{Z}_q^n$, where $s \geq \eta_\epsilon\left(\Lambda^\perp(A_u)\right)$ for some $\epsilon = negl(n)$.
- For $i = 1, \cdots, \kappa$, randomly chooses vectors $s_{R,i} \leftarrow^{\$} \mathbb{Z}_q^n$ as well as $e_{R,i} \leftarrow D_{\mathbb{Z},t}^m$ (re-select $e_{R,i}$ if $\| e_{R,i} \| > 2t\sqrt{m}$), and computes $c_{R,i} = A_u^\top \cdot s_{R,i} + e_{R,i} + Encode(m_i) mod q$, where $t = \sigma_1\sqrt{m} \cdot \omega(\sqrt{log\ n}$.
- Generates $(B_R,\ S_R) \leftarrow TrapGen(1^n, 1^m, q)$.
- Selects $l + 1$ random matrices $B_{R,1}, \cdots, B_{R,l}, C_R \longleftarrow^{\$} \mathbb{Z}_q^{n \times m}\ and\ a\ random\ vector\ r_R \leftarrow^{\$} \mathbb{Z}_q^n$.
- $Outputs\ the\ public\ key\ pk_R := (hp_S := p_R, ct_{PKE,R} := \{c_R, i\}\}_{i=1}^{\kappa}, pK_{PEKS} :=$ $\{B_{R,}, \{B_{R,1}\}_{i=1}^{L}, C_R, r_R\})$and the private key vate key $sk_R := (hk_R := k_R, \rho_R := \{s_{R,i}\}_{i=1}^{\kappa}$, sk $_{PEKS} := S_R)$ of the receiver.

- PAEKS (pp, pk_S, s_S, pk_R, kw) : This algorithm operates in the presence of $the\ public\ parameter\ pp$, the public key pk_S and private key sk_S of the sender, the public key pk_R of $the\ receiver, and\ a\ keyword\ kw$ that belongs to the set of $\in \{1, -1\}^l$ is given as;

  - $For\ i = 1, \cdots, \kappa, computes\ h_{S,i} \leftarrow R\left(c_{R,i}^\top \cdot k_S (mod q)\right), p_{S,i} \leftarrow R\left(s_{S,i}^\top \cdot p_R (mod q)\right)$ and $y_{S,i} = h_{S,i} \cdot p_{S,i}$.
  - Sets $y_S = y_{S,1} y_{S,2} \cdots y_{S,\kappa} \in \{0,1\}^\kappa$.
  - Computes $der - kws := dk_S = dk_{S,1} dk_{S,2} \ldots dk_{S,l} \leftarrow H_2(kw, y_S) \in \{1, -1\}^l$.

- Computes $B_{dk} = C_R + \sum_{i=1}^{l} dk_{S,i} B_{R,i}$ and $F_{dk} = [B_R \mid B_{dk}] \in \mathbb{Z}_q^{n \times 2m}$
- $For\ j = 1, \cdots, \rho, performs\ the\ following\ steps$:

  - $Chooses\ b_j \$^{\leftarrow}, 1\},\ random\ s_j \leftarrow^{\$} \mathbb{Z}_q^n, and\ matrices\ R_{i_j} \leftarrow^{\$} \{1, -1\}^{m \times m}$ for $i = 1, \cdots, l$.
  - Sets $\underline{R}_j = \sum_{i=1}^{l} dk_{S,i} R_{i_j} \in \{-l, \cdots, l\}^{m \times m}$.

- $Chooses\ noise\ vectors\ x_j \leftarrow^{\Psi_\alpha} \mathbb{Z}_q$ and $y_j \leftarrow^{\Psi_\alpha^m} \mathbb{Z}_q^m$.
- Sets $z_j \leftarrow \underline{R_j}^\top y_j \in \mathbb{Z}_q^m, c_{0_j} = r_R^\top s_j + x_j + b_j \lfloor q/2 \rfloor \in \mathbb{Z}_q$, and $c_{1_j} = F_{dk}^\top s_j + \left[y_j; z_j\right] \in \mathbb{Z}_q^{2m}$.

○ Outputs a searchable ciphertext $ct_{kw} \coloneqq (ct_{PEAKS,der-kw_S} \coloneqq \left\{c_{0_j}, c_{1_j}, b_j\right\}_{j=1}^{\rho})$.

- Trapdoor $(pp, pk_S, pk_R, sk_R, kw')$: This algorithm is executed using the $public\ parameter\ (pp)$, and the $public\ key\ pk_S\ of\ the\ sender$, as well as the public key pk_R and $private\ keys k_R$ of the receiver. It also requires a $keyword\ kw'$ that belongs to the set of $\in \{1, -1\}^l$ as shown;

  - For $i = 1, \cdots, \kappa$, computes $h_{R,i} \leftarrow R\left(c_{S,i}^\top \cdot k_R (modq)\right)$ and $p_{R,i} \leftarrow R\left(s_{R,i}^\top \cdot p_S (modq)\right)$, and $y_{R,i} = h_{R,i} \cdot p_{R,i}$.
  - Sets $y_R = y_{R,1} y_{R,2} \cdots y_{R,\kappa} \in \{0,1\}^\kappa$.
  - Computes der-kw $'_R \coloneqq dk_R = dk_{R,1} dk_{R,2} \cdots dk_{R,l} \leftarrow H_2(kw', y_R)$.
  - Computes $B_{dk} = C_R + \sum_{i=1}^{l} dk_{R,i} B_{R,i}$ and samples $td_{PEKS,} der - kw'_R \coloneqq t_{dk} \leftarrow sample\ left\ (B_{R,}, B_{dk}, S_{R,}, r_R, \sigma_2)$.

○ Outputs $td_{tw'} \coloneqq td_{PEKS,} der - kw'_R \coloneqq t_{dk})$.

- Test $(pp, ct_{kw}, td_{kw'})$: $Given\ the\ public\ parameter\ pp,\ the\ searchable\ ciphertext\ ct_{kw}$, and the $trapdoor\ td_{kw'}, this\ algorithm\ runs\ as\ follows.$

  - For $j = 1, \cdots, \rho$, computes $v_j = c_{0_j} - t_{dk} c_{1_j} \in \mathbb{Z}_q$.
  - Checks whether $\left|v_j - \lfloor q/2 \rfloor\right| < \lfloor q/4 \rfloor$; sets $v_j = 1\ if\ the\ equation\ holds;\ otherwise$, sets $v_j = 0$.
  - If $v_j = b_j$ for all $j = 1, \cdots, \rho$, outputs 1 ; otherwise, outputs 0.

CONCLUSION

PEKS, which stands for Public Key Encryption with Keyword Search, is an advanced cryptographic system that offers a powerful solution to the Single Keyword Search problem. Essentially, this refers to the ability to search for particular keywords or phrases within a large collection of encrypted data. This can be useful in both personal and commercial settings where privacy and security are of utmost importance. Importantly, PEKS differs from the standard Public Key Infrastructure (PKI) in many ways. For example, PEKS is based on Identity-Based Encryption (IBE) rather than PKI. Another notable difference is that PEKS does not require a trusted third party, like a Certificate Authority (CA), to authorize the public key.

While PEKS (Public Key Encryption with Keyword Search) offers several benefits, it is essential to consider its limitations. Initially, secure channels are required between the receiver and server to send Trapdoor inquiries, which can be difficult and resource-intensive to establish. In some scenarios, it may be unfeasible to set up these connections.

Additionally, most PEKS approaches can only tackle the problem of Single Keyword Search, rendering them unsuitable for Multiple Keyword Search tasks. Consequently, it is not appropriate to use these PEKS systems in vast public networks.

Our research introduces a new generic PAEKS structure that aims to tackle the limitations of existing systems. We suggest using advanced security models such as multi-ciphertext indistinguishability and trapdoor privacy based on lattices to construct a quantum-resistant PAEKS implementation. Our experiment results indicate that the proposed approach delivers enhanced security features without incurring significant additional costs, making it more suitable for various contexts in practice.

CONTRIBUTIONS TO KNOWLEDGE

The construction of a generic Public Key Encryption (PKE) scheme based on keyword search has several significant contributions to knowledge. Some of these contributions include:

1. **Secure Searching:** This generic PKE scheme allows for the secure searching of encrypted data based on specific keywords. This enables users to search for relevant information without compromising the privacy and confidentiality of the data. The scheme ensures that only authorized parties can access the decrypted data while maintaining the confidentiality of the encrypted data.
2. **Efficient Keyword-Based Retrieval:** The PKE scheme focuses on efficient keyword-based retrieval of encrypted data. It provides a mechanism for indexing and organizing the encrypted data in a way that enables efficient and fast retrieval of relevant data items based on keyword queries. This contributes to improving the efficiency and usability of encrypted data storage and retrieval systems.
3. **Privacy-Preserving**: The PKE scheme based on keyword search contributes to preserving the privacy of both the data owner and the user performing the search. By encrypting the data and allowing searching on the encrypted data, the scheme ensures that the sensitive information remains hidden from unauthorized parties, including cloud service providers or potential attackers. This enhances the privacy and confidentiality of the data in storage and retrieval scenarios.
4. **Generic Design**: This generic PKE scheme implies that it can be applied across various domains and use cases. The scheme provides a flexible and adaptable framework that can be used in different applications and environments, such as cloud computing, data-sharing platforms, or secure search systems. This generality contributes to the broader applicability and adoption of secure keyword-based searching techniques.
5. **Security Guarantees**: The new PKE scheme ensures strong security guarantees, protecting the encrypted data against potential attacks and unauthorized access. The scheme leverages cryptographic primitives and techniques to provide robust security features, such as resistance to chosen keyword attacks, keyword privacy, and data confidentiality. These security guarantees contribute to building trust and confidence in the usage of the PKE scheme in real-world scenarios.

REFERENCES

Aloqaily, M., Otoum, S., AlRidhawi, I. & Jararweh, Y. (2019). An intrusion detection system for connected vehicles in smart cities, *Ad Hoc Networks*, 90, Article ID 101842.

Al-Riyami, S. S. & Paterson, K. G. (2003). Certificateless public key cryptography, in *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security*, Taipei, Taiwan,

Baek, J., Safavi-Naini, R. & Susilo, W. (2008). Public key encryption with keyword search revisited, in *Proceedings of the International conference on Computational Science and Its Applications*, Perugia, Italy, July 2008.

Behnia, R., Ozmen, M. O., & Yavuz, A. A. (2020). Lattice-Based Public Key Searchable Encryption from Experimental Perspectives. *IEEE Transactions on Dependable and Secure Computing*, **17**(6): 1269–1282. https://doi.org/10.1109/tdsc.2018.2867462

Bernstein, D. & Lange, J. T. (2017). Post-quantum cryptography. *Nature.* **549** (7671): 188 194.

Bernstein, D. J. (2008). *Protecting communications against forgery*, *Algorithmic Number Theory* (PDF). 44. MSRI Publications.

Blaze, M., Bleumer, G. & Strauss, M. (1998). Divertible protocols and atomic proxy cryptography, in *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, Espoo, Finland,

Boneh, D., Di Crescenzo, G., Ostrovsky, R., & Persiano, G. (2004). Public Key Encryption with Keyword Search. *Advances in Cryptology - EUROCRYPT 2004*, 506–522. https://doi.org/10.1007/978-3-540-24676-3_30

Byun, J. W., Rhee, H. S., Park, H.-A., & Lee, D. H. (2006). Off-Line Keyword Guessing Attacks on Recent Keyword Search Schemes over Encrypted Data. *Lecture Notes in Computer Science*, 75–83. https://doi.org/10.1007/11844662_6

Emura, K., Miyaji, A., Rahman, M. S. & Kazumasa, O. (2015). Generic Constructions of Secure-channel Free Searchable Encryption with Adaptive Security. *Secur. Commun. Networks* **8**(8): 1547–1560.

Fang, L., Susilo, W., Ge, C., & Wang, J. (2009). A Secure Channel Free Public Key Encryption with Keyword Search Scheme without Random Oracle. *Cryptology and Network Security*, 248–258. https://doi.org/10.1007/978-3-642-10433-6_16

Goh, E. J. (2003). Secure indexes, IACR Cryptol. *ePrint Arch.*, 2003, 216.

Goldreich, O. & Ostrovsky, R. (1996). Software protection and simulation on oblivious RAMs, *Journal of the ACM*, 43, (3) 431–473, 1996.

Guo, L., & Yau, W.-C. (2015). Efficient Secure-Channel Free Public Key Encryption with Keyword Search for EMRs in Cloud Storage. *Journal of Medical Systems*, **39**(2). https://doi.org/10.1007/s10916-014-0178-y.

Hu, C., & Liu, P. (2012). An Enhanced Searchable Public Key Encryption Scheme with a Designated Tester and Its Extensions. *Journal of Computer*, **7**(716-723). https://www.semanticscholar.org/paper/An-Enhanced-Searchable-Public-Key-Encryption-Scheme-Hu-Liu/b40d43bfc04d821f7f6127a644ad49656c9c8333

Huang, Q., & Li, H. (2017). An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks. *Information Sciences*, *403-404*, 1–14. https://doi.org/10.1016/j.ins.2017.03.038.

Jeong, I. R., Kwon, J. O., Hong, D., & Lee, D. H. (2009). Constructing PEKS schemes secure against keyword guessing attacks is possible? *Computer Communications*, **32**(2): 394–396. https://doi.org/10.1016/j.comcom.2008.11.018

Jiang, P., Mu, Y., Guo, F., Wang, X., & Wen, Q. (2015). Online/Offline Ciphertext Retrieval on Resource Constrained Devices. *The Computer Journal*, **59**(7): 955–969. https://doi.org/10.1093/comjnl/bxv099

Kamara, S. & Papamanthou, C. (2013). Parallel and dynamic searchable symmetric encryption, in Proceedings of the International conference on financial cryptography and data security, Okinawa, Japan, April 2013.

Katz, J., Sahai, A., & Waters, B. (2012). Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products. *Journal of Cryptology*, **26**(2): 191–224. https://doi.org/10.1007/s00145-012-9119-4

Konstantopoulos, M., Diamantopoulos, P., Chondros, N. & Roussopoulos, M. (2019). Distributed personal cloud storage without third parties, *IEEE Transactions on Parallel and Distributed Systems*, **30**(11): 2434–2448

Li, M., Yu, S., Zheng, Y., Ren, K., & Lou, W. (2013). Scalable and Secure Sharing of Personal Health Records in Cloud Computing Using Attribute-Based Encryption. *IEEE Transactions on Parallel and Distributed Systems*, **24**(1): 131–143. https://doi.org/10.1109/tpds.2012.97.

Liu, Z.-Y., Tseng, Y.-F., Tso, R., Mambo, M., & Chen, Y.-C. (2021b). *Public-key Authenticated Encryption with Keyword Search: Cryptanalysis, Enhanced Security, and Quantum-resistant Instantiation * CCS CONCEPTS*. https://eprint.iacr.org/2021/1008.pdf.

Liu, Z.-Y., Tseng, Y.-F., Tso, R., Mambo, M., & Chen, Y.-C. (n.d.). *Public-key Authenticated Encryption with Keyword Search: Cryptanalysis, Enhanced Security, and Quantum-resistant Instantiation * CCS CONCEPTS*. Retrieved March 15, 2023, from https://eprint.iacr.org/2021/1008.pdf.

Lu, J., Li, H., Huang, J., Ma, S., Au, M. H. A., & Huang, Q. (2023). Certificateless Public Key Authenticated Encryption with Keyword Search Achieving Stronger Security. *Information*, **14**(3): 142. https://doi.org/10.3390/info14030142

Lu, Y., Li, J., & Zhang, Y. (2020). Secure Channel Free Certificate-Based Searchable Encryption Withstanding Outside and Inside Keyword Guessing Attacks. *IEEE Transactions on Services Computing*, **14**(6): 2041. https://www.academia.edu/40194346/Secure_Channel_Free_Certificate-Based_Searchable_Encryption_Withstanding_Outside_and_Inside_Keyword_Guessing_Attacks

Lu, Y., Wang, G., & Li, J. (2019). Keyword guessing attacks on a public key encryption with keyword search scheme without random oracle and its improvement. *Inf. Sci.* https://www.semanticscholar.org/paper/Keyword-guessing-attacks-on-a-public-key-encryption-Lu-Wang/f3288d4e8bc7b30bd6e38f2b86bd2f8c2db22071.

Menezes, A. J., van Oorschot, P. C. & Vanstone, S. A. (1996). *8: Public-key encryption. Handbook of Applied Cryptography (PDF)*. CRC Press. pp. 283–319. ISBN 0-8493-8523-7.

Micciancio, D., & Peikert, C. (2011). Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller. *Cryptology EPrint Archive*, **10**(21-26). https://eprint.iacr.org/2011/501

Noroozi, M., & Eslami, Z. (2019). Public key authenticated encryption with keyword search: revisited. *IET Information Security*, **13**(4): 336–342. https://doi.org/10.1049/iet-ifs.2018.5315.

Otoum, S., Kantarci, B. & Mouftah, H. T. (2017). Detection of known and unknown intrusive sensor behavior in critical applications, *IEEE Sensors Letters*, **1**(5): 1–4, 2017.

Rhee, H. S., Park, J. H., Susilo, W., & Lee, D. H. (2010). Trapdoor security in a searchable public-key encryption scheme with a designated tester. *Journal of Systems and Software*, **83**(5): 763–771. https://doi.org/10.1016/j.jss.2009.11.726

Shamir, A. (1984). Identity-Based Cryptosystems and Signature Schemes. In Advances in Cryptology – CRYPTO '84 (pp. 47-53). Springer.

Shirey, R. (2007). *Internet Security Glossary, Version 2*. Network Working Group. doi:10.17487/RFC4949.

Stallings, W. (1990). *Cryptography and Network Security: Principles and Practice.* Prentice Hall, United States of America, p. 165. ISBN 9780138690175.

Suzuki, T., Emura, K., & Ohigashi, T. (2019). A Generic Construction of Integrated Secure-Channel Free PEKS and PKE and its Application to EMRs in Cloud Storage. *Journal of Medical Systems*, **43**(5). https://doi.org/10.1007/s10916-019-1244-2.

Tang, Q., & Chen, L. (2010). Public-Key Encryption with Registered Keyword Search. *Public Key Infrastructures, Services and Applications*, 163–178. https://doi.org/10.1007/978-3-642-16441-5_11.

Wen, M., Lu, R., Zhang, K., Lei, J., Liang, X., & Shen, X. (2013). PaRQ: A privacy-preserving range query scheme over encrypted metering data for smart grid. *IEEE Transactions on Emerging Topics in Computing*, *1*, 178–191. https://uwaterloo.ca/broadband-communications-research-lab/publications/parq-privacy-preserving-range-query-scheme-over-encrypted

Yang, Y. (2015). Attribute-based data retrieval with semantic keyword search for e-health cloud. *Journal of Cloud Computing*, *4*(1). https://doi.org/10.1186/s13677-015-0034-8

Yanguo, P. & Jiangtao, C., Changgen, P. & Zuobin, Y. (2014). Certificateless public key encryption with keyword search, *China Communications*, **11**(11): 100–113.

Zhang, X., Xu, C., Wang, H., Zhang, Y., & Wang, S. (2019). FS-PEKS: Lattice-based Forward Secure Public-key Encryption with Keyword Search for Cloud-assisted Industrial Internet of Things. *IEEE Transactions on Dependable and Secure Computing*, 1–1. https://doi.org/10.1109/tdsc.2019.2914117

Zhang, Y. & Lu, S. (2014). POSTER: efficient method for disjunctive and conjunctive keyword search over encrypted data, in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, Scottsdale A., **57**: 278-288.