Secure Channel Free Public Key Encryption with Multiple Keywords Search

A DISSERTATION

SUBMITTED TO THE SCHOOL OF COMPUTING AND DIGITAL MEDIA

AND THE COMMITTEE ON RESEARCH DEGREES

OF LONDON METROPOLITAN UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

YANG MA

April 2020

Copyright

Abstract

With a further exploration of modern cryptography, people realize that Public Key Infrastructure (PKI) is not perfect but has its limitations. One of the limitations is that PKI completely depends on the Certification Authority (CA) to obtain a digital certificate, but this online trusted third party might be compromised by the cyber attacks. Therefore, Shamir proposed a definition of Identity Based Encryption (IBE) which only relies on user’s identity to generate the public key instead of CA. Although the blueprint of IBE was presented in 1984, the first secure and reliable IBE system has been introduced until 2001. Meanwhile, some applications of IBE, such as Public Key Encryption with Keyword Search (PEKS), have been came up with since then.

PEKS is one of the most technologically advanced crypto-systems to address searchable encryption. It enables individuals to search encrypted documents appending with a keyword without deriving any information. The first PEKS scheme was formalized by BDOP in 2004, but a secure channel must be established in order to transfer the Trapdoor query to the third party. Comparing with the original PEKS scheme, the later PEKS approaches remove secure channels and become much secure and efficient as time goes by. However, no matter what happened, Multiple Keywords Search and Keyword Guessing Attack are still two main research interests for the consideration. This PhD research aims to propose a few PEKS schemes in order to solve both Single and Multiple Keyword(s) Search issues and resist Off-line Keyword Guessing Attack (OKGA) and/or Inside Keyword Guessing Attack (IKGA). The focus of this research is listed on the three following parts:

Many current Public Key Encryption with Multiple Keywords Search (MPEKS) schemes suffers OKGA. Therefore, this research firstly defines a formal MPEKS scheme to solve OKGA, which is called “*Trapdoor-indistinguishable Secure Channel*

Free Public Key Encryption with Multi-keywords Search (tSCF-MPEKS)". More specially, the new scheme allows users to search both Single and Multiple Keyword(s) and also has the characters of Ciphertext Indistinguishability and Trapdoor Indistinguishability so that it proves to be semantic secure under Random Oracle Models by Bilinear Diffie-Hellman (BDH) and 1-Bilinear Diffie-Hellman Inversion(1-BDHI) assumptions for preventing OKGA. Besides, the efficiency and performance of *tSCF-MPEKS* is presented from both the theoretical analysis and the practical analysis.

IKGA in MPEKS schemes is still an intractable problem up to now. But, this phd research solves IKGA by applying User Authentication technique. More specially, the second proposed PEKS scheme, namely "*Robust Secure Channel Free Public Key Encryption with Multi-keywords Search (rSCF-MPEKS)",* not only addresses both the Single and Multiple Keyword(s) Search problems but also satisfies Ciphertext Indistinguishability and Trapdoor Indistinguishability properties and incorporates with User Authentication, therefore, it proves to be semantic secure under Random Oracle Models by BDH assumption for resisting IKGA. Besides, OKGA is also resisted in the proposed scheme. In addition, the performance of *rSCF-MPEKS* is also analyzed by the theoretical analysis and the computer simulation.

Thirdly, almost all current PEKS and MPEKS schemes cannot deal with imprecise keywords, such as "latest", "newest", etc. The research incorporates with Fuzzy Logic (Artificial Intelligence) technique to PEKS and then proposes a formal statement of "*Public Key Encryption with Multi-keywords Search using Mamdani System (m-PEMKS)".* Its concrete construction, correctness and security verification are then proposed in the following section of the thesis. The new approach solves Fuzzy Keyword Search problem and proves to be semantic secure under Random Oracle Models by BDH and 1-BDHI assumptions so that it could resist OKGA. Besides, the

performance of *m-PEMKS* is presented by the theoretical analysis and the computer simulation.

Acknowledgments

Firstly, I would like to thank you my supervisors (Prof Hassan Kazemian, Dr Jianming Cai) to support my PhD study during the last four years.

PhD is a tough job. You may suffer substantial difficulties when you pursue the PhD. So do I. Prof Hassan Kazemian is a great supervisor, who not only teaches me how to conduct research but also cheers me up when I witness some obstacles. During the study, Prof Hassan invited me to take part in some seminars and conferences to broaden my horizon and enrich my knowledge, etc.

In the first year of PhD, I have done a lot of literature reviews and think out some ideals of my PEKS systems. Based on the first year study and research, I define few PEKS systems and also propose the concrete constructions of these systems, verify the correctness and security of these PEKS approaches, and finally analyze the efficiency and the performance of the proposed schemes by the theoretical analysis and the computer simulation. In the final year of PhD, I published two papers on 11th International Conference On Security Of Information and Networks (SIN'2018) and 21st International Conference on Engineering Applications of Neural Networks (EANN 2020) and also prepare the PhD thesis.

As a full time student, I could not afford the tuition fee and the living fee during my PhD study. Therefore, I would like to thank you my parents to support my study in the end.

Tables of Contents

List of Figures

List of Tables

Glossary

1. **Cryptography:** The study of encryption principles and procedures.
2. **Cryptanalysis:** The study of cipher-breaking without knowing key.
3. **Cryptology:** A field which contains both cryptanalysis and cryptography.
4. **Cryptographic algorithm:** An algorithm that makes data unreadable or vice versa.
5. **Plaintext:** The original information or data.
6. **Ciphertext:** The enciphered information or data.
7. **Encryption:** Transfer the plaintext into the ciphertext.
8. **Decryption:** Transfer the ciphertext into the plaintext.
9. **Secret key:** The input of cryptographic algorithm, which is only known between the sender and the receiver.
10. **Public key and Private key:** A pair key used in asymmetric key cipher. The sender applies the public key for encryption while the receiver applies the private key for decryption.
11. **Symmetric key cipher:** The cryptographic algorithm applies the same secret key for encryption and decryption.
12. **Asymmetric key cipher:** The cryptographic algorithm applies the different keys for encryption and decryption.
13. **Challenger:** The challenger is able to establish the crypto-system and also encrypts the message.
14. **Adversary:** The attacker who tries to break the crypto-system.
15. **Hash function:** The mathematical algorithm maps information into a fixed size.
16. **Oracle:** It’s an abstract machine which is able to reply after each query.
17. **Semantic Secure:** The adversary cannot drive one bit plaintext even though he/she intercepts the whole ciphertext.

Abbreviations

1. **CBC:** Cipher Block Chaining
2. **CFB:** Cipher Feedback
3. **OFB:** Output Feedback
4. **PEKS:** Public Key Encryption with Keyword Search
5. **SC:** Secure Channel
6. **SCF-PEKS:** Secure Channel Free Public Key Encryption with Keyword Search
7. **SPA:** Simple Power Analysis
8. **CPA:** Correlation Power Analysis
9. **DPA**: Differential Power Analysis
10. **DL:** Deep Learning
11. **IND:** Indistinguishability
12. **CPA:** Chosen Plaintext Attack
13. **CCA:** Chosen Ciphertext Attack
14. **CPA2:** Adaptive Chosen Ciphertext Attack
15. **CI:** Ciphertext Indistinguishability
16. **TI:** Trapdoor Indistinguishability
17. **BDH assumption:** The Bilinear Diffie-Hellman (BDH) assumption
18. **1-BDHI assumption:** The 1-Bilinear Diffie-Hellman Inversion (1-BDHI) assumption
19. **KGA:** Keyword Guessing Attack
20. **OKGA:** Off-line Keyword Guessing Attack
21. **IKGA:** Inside Keyword Guessing Attack
22. **PPT:** Probabilistic Polynomial Time
23. **FCL**: Fuzzy Control Language

1. Introduction

Chapter one starts with the motivation of the PhD research and then introduces the aims and the objectives of this research. After that, the evaluations of the proposed schemes are described and the contributions of this research are also presented in the following part. Finally, the overview of the thesis is introduced in the end of this chapter.

1.1 Motivation

With the rapid development of the Internet, more and more individuals or companies store and manage the amount of sensitive information into the online trusted third parties (i.e. cloud storage) for reducing local storages, decreasing overheads and supplying backups, etc. Although it carries out plenty of benefits, it should not be overlooked that uploading information into the networked servers may lead in some adverse effects. More specifically, unfriendly hackers are able to launch port scanning in order to search the backdoors of the system. Then, they bypass the firewall and invade the victim's host without user authentication for stealing sensitive information (i.e. personal information, bank card details, etc.) and finally break the victim's operating system. Besides, many crackers may intercept the data packets transmitting on the public network and then recover these packets in order to achieve the information. In addition, attackers may exploit the physical leakage to disclose secrets during the processing of a cryptographic operation by the side channel analysis. For instance, attackers intercept the electromagnetic emission traces and power consumption traces so that they could apply Simple Power Analysis (SPA), Differential Power Analysis (DPA), Correlation Power Analysis (CPA) or even Deep Learning (DL) to derive the secret key. Hacktivism is immoral and carries out great loss in many aspects, such as time and money, etc. To keep the confidentiality, integrity and availability (CIA) of sensitive information from both inside and outside attackers, cryptographic techniques must be applied during the whole data transmission. There is no doubt that Public Key Encryption with Keyword Search (PEKS) is a significant approach in cryptographic techniques to provide a secure data transmission. Although PEKS schemes takes substantial advantages, it should not be overlooked that they carry about some negative effects. For instance, PEKS schemes may vulnerable to Off-line Keyword Guessing Attack (OKGA) and/or Inside Keyword Guessing Attack (IKGA) and some of them are only able to support Single Keyword Search instead of Multiple Keywords Search so that these schemes may not be applied to the general public networks. Therefore, the PhD thesis pays more attention to PEKS research and then proposes few strengthen PEKS schemes to reverse these problems.

1.2 Aims and Objectives

This research has three aims. These include (1) come up with three PEKS schemes to solve both Single and Multiple Keyword(s) Search problem, (2) verify the security of the proposed schemes in order to resist OKGA and/or IKGA, and (3) propose a powerful PEKS system incorporating with Artificial Intelligence (Fuzzy Logic) to address Fuzzy Keyword Search problem.

The objectives are listed in the following.

i. Propose the first PEKS scheme to support both Single and Multiple Keyword(s) Search and also resist OKGA.

ii. Propose the second PEKS statement, which incorporates with User Authentication technique to resist IKGA. Apart from that, the proposed has the ability to resist OKGA. In addition, this scheme also allows users to search encrypted documents by Single or Multiple Keyword(s).

iii. Propose the third PEKS scheme by applying Mamdani Fuzzy Inference System so that it solves Fuzzy Keyword Search problem. Besides, the proposed scheme also resists OKGA.

1.3 Evaluation

Based on the previous part, the evaluation of the proposed PEKS schemes contains three main steps. The first step is to ensure the correctness of the proposed schemes. Even one bit error is not accepted in the proposed PEKS systems. The second step is about security verification. The proposed schemes are proved to be semantic secure under Random Oracle Models which means the adversary cannot break any one bit of ciphertext. The third step of evaluation is to analyze the performance and the efficiency of the proposed schemes by the theoretical analysis (Mathematical computation) and the computer simulation (JAVA).

1.4 Contribution of the Research to the Knowledge

Many current Public Key Encryption with Multiple Keywords Search (MPEKS) schemes suffers OKGA. Therefore, the research firstly defines a MPEKS scheme, namely "*Trapdoor-indistinguishable Secure Channel Free Public Key Encryption with Multi-keywords Search (tSCF-MPEKS)*", to resist OKGA. More specially, the proposed MPEKS scheme contains the properties of Ciphertext Indistinguishability and Trapdoor Indistinguishability and is proved to be semantic secure under Random Oracle Models so that it is able to resist OKGA. Besides, the proposed scheme allows users to exploit encrypted messages by multiple keywords instead of single keyword only, therefore, *tSCF-MPEKS* is much more practical and could be applied in the general public networks.

Secondly, IKGA in MPEKS schemes is still an intractable problem up to now. The research then defines the strengthen MPEKS scheme called "*Robust Secure Channel Free Public Key Encryption with Multi-keywords Search (rSCF-MPEKS)*" to reverse IKGA. Some times, the third party is honest-but-curious and therefore, it may exploit the secret key during data transmission. Therefore, the proposed scheme incorporates with User Authentication techniques and also satisfies Ciphertext Indistinguishability and Trapdoor Indistinguishability properties so that it is able to prevent IKGA. Besides, the proposed scheme could also prevent OKGA. In addition, similar to the *tSCF-MPEKS scheme*, the *rSCF-MPEKS* also has the ability to address both Single and Multiple Keyword(s) Search issues.

Last but not least, almost all current PEKS and MPEKS schemes cannot deal with imprecise keywords, such as "latest", "newest", etc. Therefore, the research also formalizes the third MPEKS statement, namely "*Public Key Encryption with Multi-keywords Search using Mamdani System (m-PEMKS)*", which is able to solve Fuzzy Keyword Search issue. In shot, the proposed scheme applies artificial intelligence (Mamdani Fuzzy Inference System in Fuzzy Logic) technique to solve Fuzzy Keyword Search problem. Besides, Ciphertext Indistinguishability and Trapdoor Indistinguishability properties are also providing in *m-PEMKS* scheme so that it perfectly stands up to OKGA.

1.5 The Outline of the Thesis

This PhD thesis includes seven chapters starting with the Introduction part. This chapter introduces the motivation, aims and objectives, and the contributions of the PhD research.

Chapter two briefly introduces the background and the development of crypto-systems from the ancient time to the modern time. Then, it revisits the current researches on Public Key Encryption with Keyword Search (PEKS) and also analyzes the advantages and disadvantages of these schemes.

Chapter three provides the preliminaries and the methodology of PEKS. These include bilinear pairing, BDH and 1-BDHI assumptions and so on, which will be used to establish the construction of the proposed PEKS schemes and also provide the security of the proposed schemes.

Chapter four comes up with the *tSCF-MPEKS* scheme. The proposed system addresses both Single and Multiple Keyword(s) Search issues and resists OKGA. The correctness, security verification and performance and efficiency of the proposed approach are also provided in this chapter.

Chapter five indicates that nearly all of previous PEKS approaches are vulnerable to IKGA and then proposes *rSCF-MPEKS* scheme, which incorporates with User Authentication technique to resist IKGA. Besides, the proposed system also solves Single and Multiple Keyword(s) Search issues and resists OKGA. In addition, the correctness, security verification and performance and efficiency of the proposed approach are also provided in this chapter.

Chapter six introduces the third proposed PEKS scheme. The new system called *m-PEMKS* incorporates with the advantages of Mamdani Fuzzy Inference System (Fuzzy logic) so that it is able to address Fuzzy Keyword Search issue (i.e. “latest”, “biggest”, etc.). Apart from that, the proposed scheme has the properties of Ciphertext Indistinguishability and Trapdoor Indistinguishability and therefore, it is able to prevent OKGA as well. The correctness, security verification and performance and efficiency of the proposed approach are also provided in this chapter.

Chapter seven is the final chapter which will conclude the whole PhD thesis.

2. Background and Literature Review

2.1 Introduction

The chapter briefly introduces the history and development of cryptography from the ancient time, the medieval time until the modern time. In the modern time, cryptography witnesses a huge development. More specially, the cryptographic algorithms become much more secure and advanced than before. There is no doubt that Public Key Encryption with Keyword Search (PEKS) is one of the typical cryptographic techniques that is able to secure data transmission. Thus, the rest of the chapter will review the current PEKS researches.

2.2 The History and Development of Cryptography

What is cryptography? It is a method used to protect electronic message security by changing the message into unreadable characters. Cryptography plays a pivotal role in information security and starts at a couple of thousand years ago.

The first known use of cryptography was found in the chamber from Khnumhotep II's tomb in Egypt around 1900 BC. It is the hieroglyphic symbols engraved into the wall of the tomb. Around 500 to 600 BC, Hebrew encrypted the message by using few simple mono-alphabetic substitution ciphers such as Atbash cipher. Later, "Scytale Transposition Cipher (Figure 1)" was applied by ancient Greeks for hiding the information. For instance, the soldier in Spartan military firstly prepared a tape wrapping on a stick so that he/she could depict the information on this wood tape. After that, a courier sent out the tape into the allied forces. Once received the tape, the allies found a stick which had the same diameter as the original one and wrapped the tape around the stick again for recovering the original message. Interestingly, this transposition cipher is an efficient and advanced way to protect information security in the remote antiquity mainly because the secret will be unreadable, if the tape is unwound.

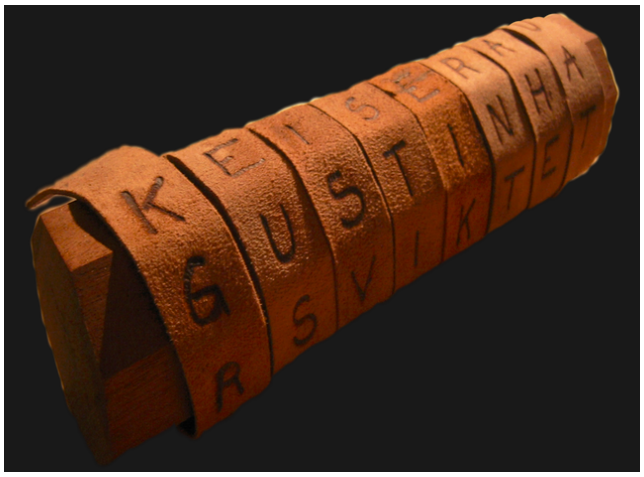


FIGURE 1. SCYTALE TRANSPOSITION CIPHER (WIKIPEDIA, 2019)

In the medieval time, the classical encryption techniques rose blowout. These ciphers follow two rules: one is called substitution which means the characters in the plaintext are replaced by alternative letters, numbers or even symbols. Another rule is called transposition which is shifting the positions of the plaintext by a regular procedure so that the message will be hided. Caesar Shift cipher is a typical substitution cipher whose characters in plaintext are changed by 3rd letters on (Figure 2). However, this cipher is vulnerable to exhaustive search that is a trial and error method to enumerate every possible combination until breaking the cipher. Mono-alphabetic cipher is then provided to solve the disadvantages of the Caesar Shift cipher. More specially, it maps the plaintext into one of the several possible cipher-texts by the fixed substitutions and vice versa. Although the Mono-alphabetic cipher possesses many keys, it should not be overlooked that frequency analysis may compromise it. Besides, Rail Fence cipher and Row Transposition cipher are two representative transposition ciphers.

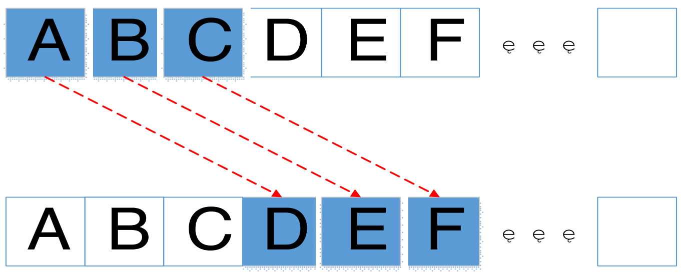


FIGURE 2. CAESAR SHIFT CIPHER

In the end of the Second World War, Arthur Scherbius designed the Enigma Machine and therefore, cryptography witnesses a huge quantum jump. The Enigma Machine (Figure 3) is an electro-mechanical rotor cipher machine with mechanical and electrical subsystems. The Enigma Machine is comprised of several parts such as a keyboard, etc. For encryption or decryption, the user only needs to press keys on the keyboard and records the lighting letters as the plaintext or the ciphertext. The substitution keys will be automatically changed after each letter pressing.

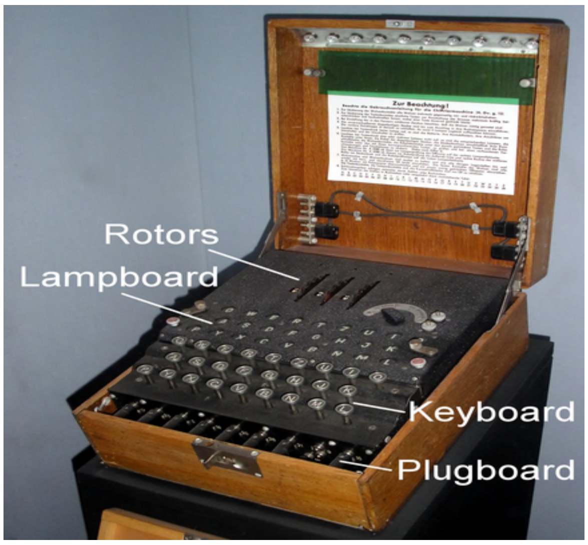


FIGURE 3. ENIGMA MACHINE (WIKIPEDIA, 2019)

Let Enigma Machine be having three rotors. It is able to generate 17576 substitution keys so that brute-force search seems unrealistic. Nothing is impossible. Marian Rejewski found that the cycles will be appeared twice in the beginning of the ciphertext and then broke the Enigma Machine via the cycles. Later on, Alan Turing designed a Bletchley Park Bombe (Figure 4) to speed up the calculations required in order to break the codes.

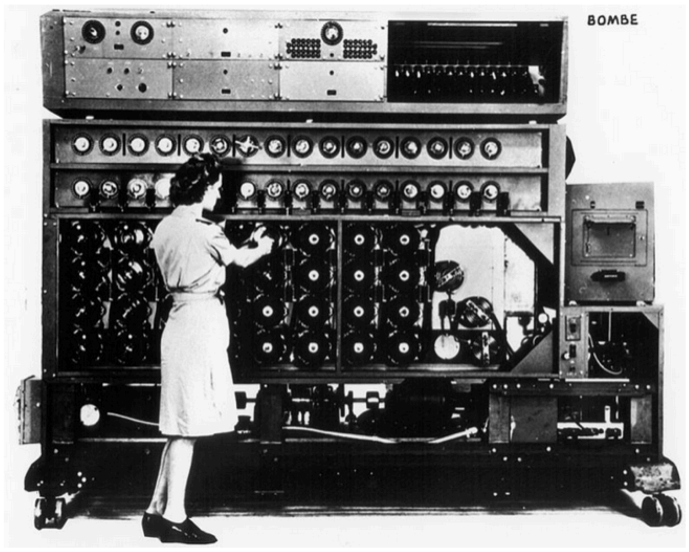


FIGURE 4. BLETCHLEY PARK BOMBE (WIKIPEDIA, 2019)

With the development of cryptanalysis, individuals are not discontented with encrypting the data by simple methods such as Enigma Machine. Plenty of experts embark on designing the safe, simple and efficient cryptographic algorithms to ensure the confidentiality of data.

Symmetric and asymmetric ciphers have been proposed during the modern time in order to strengthen the information security. Symmetric cipher (Figure 5) allows the sender and the receiver using the same single key to encrypt the message. A perfect symmetric cipher satisfies two rules: the first rule is that the security of symmetric cipher relies on a strong and complex encryption algorithm. The second one is that the secret key is not able to be captured by anyone except the sender and the receiver. In 1949, Claude Shannon proposed the blueprint of SP network to make the symmetric ciphers as complicated as possible.

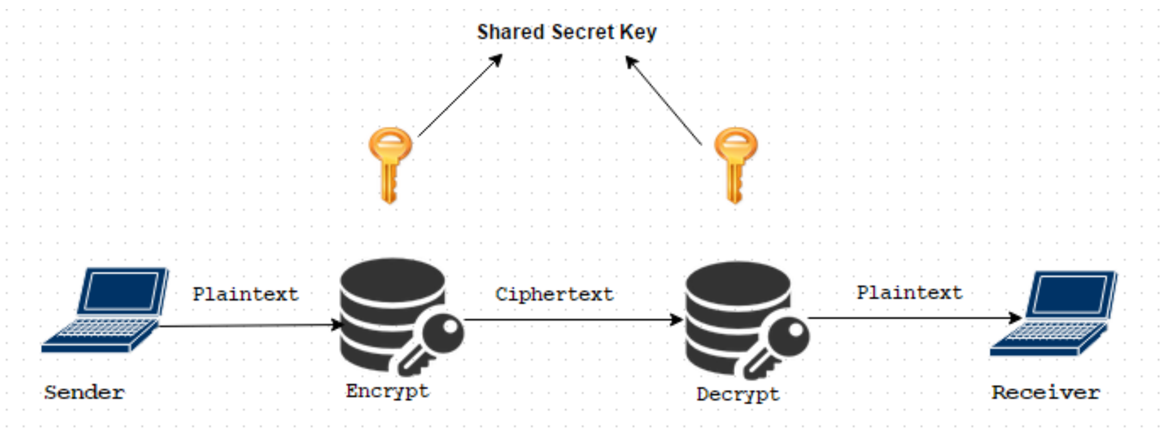


FIGURE 5. SYMMETRIC CIPHER MODEL

IBM designed Data Encryption Standard (DES) in the year of 1977, which is the most typical block cipher in the world. DES relies on a Feistel network with 16 rounds encryption to encrypt 8 bytes data by 7 bytes key. Therefore, the ciphertext becomes much sophisticated because DES shows the strong avalanche so that one bit error may affect half output result. Nowadays, it is possible to break DES by brute force attack and therefore, Triple DES and CBC DES are designed to solve the weaknesses appearing in single DES.

Comparing with the Feistel structure, Advanced Encryption Standard (AES) uses iterative for operations. The outline of AES is comprised of four steps, which is described in Figure 6. It should be clear that the last round of AES contains three steps without mix columns.

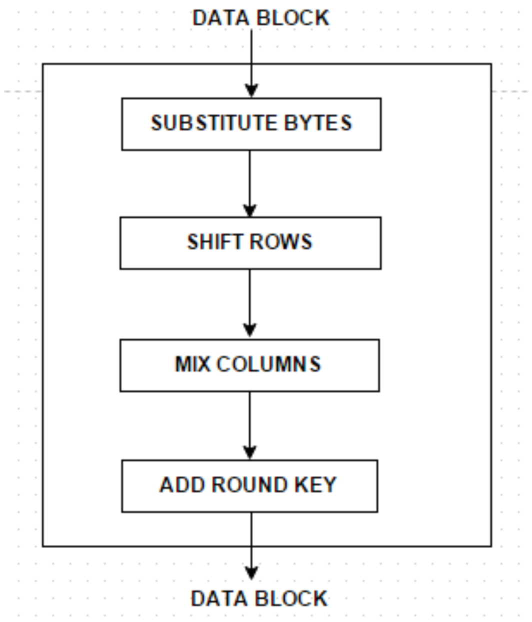


FIGURE 6. ONE INTERNAL ROUND OF AES

However, people is not satisfied the symmetric key ciphers mainly because the shared key will be compromised for a long time use. Social engineering is an easy and efficient way to compromise the shared key. In addition, shared key management is also a thorny problem. Therefore, a new cipher called asymmetric key cryptography is proposed to reverse the intractable problems that the single key cryptography witnesses. Asymmetric key cryptography is the complements rather than replacing the single key cryptography. Compared with the single key cipher, the enormous difference in asymmetric cipher (Figure 7) is that it has two keys. Public key is applied for encryption and private key is applied for decryption.

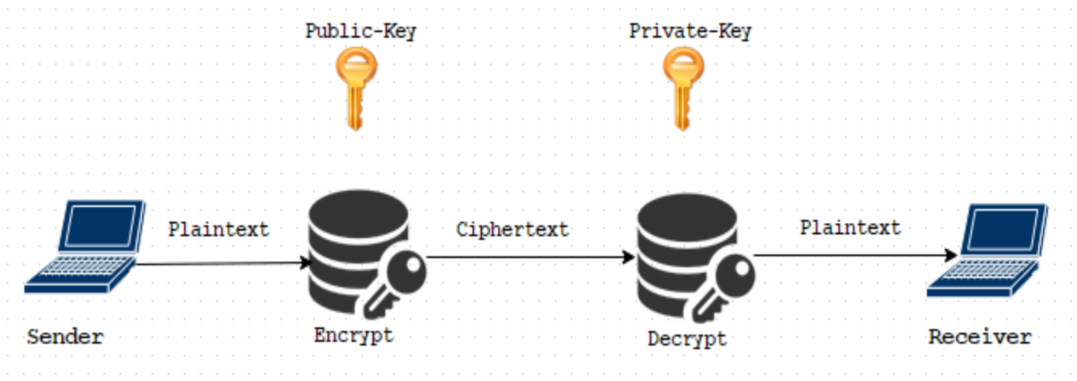
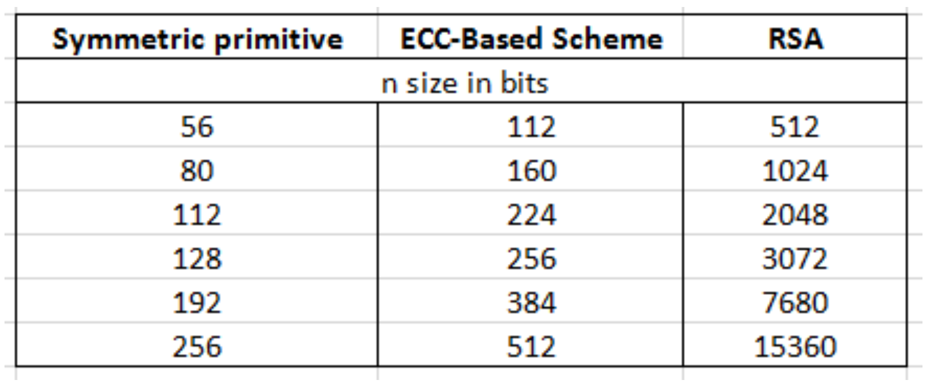


FIGURE 7. ASYMMETRIC CIPHER MODEL

For public key ciphers, theirs contributions are used in three different applications. The first application is for encryption and decryption to protect data confidentiality. The second one is digital signatures to provide individual, entity and message authentications. The last application is key exchange, which is fairly used in session keys. Almost all public key ciphers rely on a Trapdoor one-way function and theirs security depend on the difficulty of factoring large numbers or computing discrete logarithms. More specially, Rivest, Shamir and Adleman from MIT were proposed RSA in 1977, which is an advanced cipher to resist many attacks, such as brute-force key search and mathematical attacks, etc. But, RSA suffers Chosen Ciphertext Attack (CCA) as adversaries are able to select any ciphertext to derive the characters of RSA and then recover parts of plaintext via cryptanalysis, side channel analysis and a lot more besides.

Diffie and Hellman came up with the first public key approach in 1976, namely Diffie-Hellman Key Exchange Primitive. The proposed scheme provides a practical way for key exchange to the general public. Its security is based on the difficulty of computing discrete logarithms, but it is compromising to Man-in-the-Middle Attack. Thus, key authentication should be considered between both ends. ElGamal cipher is another typical public key cryptographic algorithm which has the similar technique as Diffie-Hellman Key Exchange Primitive. The security of ElGamal cipher also relies on the difficulty of computing discrete logarithms. Recently, Elliptic Curve Cryptography (ECC) takes more consideration than other public key ciphers mainly because it provides same security but requires short key sizes (Table 1).

TABLE 1. SAME SECURITY WITH DIFFERENT KEY SIZES



Symmetric primitive	ECC-Based Scheme	RSA
n size in bits		
56	112	512
80	160	1024
112	224	2048
128	256	3072
192	384	7680
256	512	15360

With time goes by, individuals are dissatisfied with the security of cryptographic algorithms only relying on the hard mathematic problems. They focus on analyzing the characters of quantum. Therefore, quantum cryptography is then invented by the experts. More specially, quantum cryptography is a technique to exploit the properties of quantum for designing unbreakable ciphers. There is no doubt that quantum cryptography is completely different with the classical cryptography and the modern cryptography, whose security is based on physics instead of mathematics. Interestingly, quantum cryptography cannot be broken mainly because it seems impossible to assess the quantum states in any existing system. Hence, quantum cryptography is regarded as a secure system because of the quantum properties. So far, there are some quantum key distribution protocols (i.e. BB84, E91, etc.) could be used in the general public and the adversaries are not able to capture the quantum states during data transmission.

According to the length of input, the cipher is separated into two parties. The first one is stream cipher (such as RC4) which encrypts only one bit of input at a time with a bit pseudorandom key stream in order to obtain one bit ciphertext stream. On the contrary, another party is called block cipher, which applies the deterministic algorithm and a secret key to encrypt the blocks of input. For instance, DES is a typical block cipher that could encrypt 8 bytes blocks of message as an input. Apart from that, the block chain ciphers and stream chain ciphers are designed with time goes on.

Cipher Block Chaining (CBC) (Figure 8) cipher is one of the typical block chain ciphers which mainly applies in bulk data encryption and authentication. However, it has its limitations. For instance, a block relies on all ciphertext blocks before it, therefore, one bit error in the block may bring about negative influences to all following ciphertext blocks.

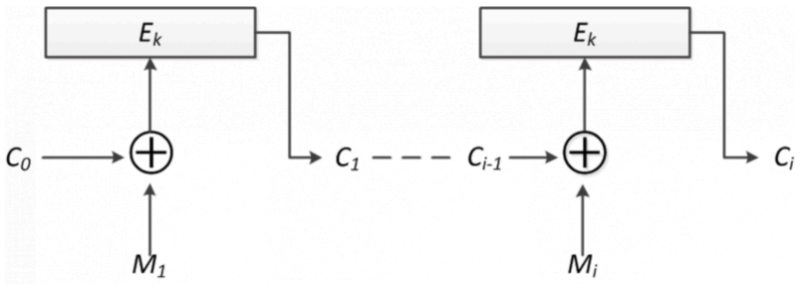


FIGURE 8. CIPHER BLOCK CHAINING CIPHER (CRYPTOWIKI, 2019)

Compared with CBC, Cipher Feedback (CFB) cipher and Output Feedback (OFB) cipher are two typical stream ciphers whose input of the message is regarded as a stream of bits. For CFB (Figure 9), it is able to apply in stream data encryption and authentication. However, it also has its limitations, such as errors. Once the error is generated, it may propagate for several blocks. In terms of OFB (Figure 10), it could apply in stream encryption on the noisy channels so that the error does not propagate during the whole encryption. But, it requires the sender and the receiver remaining in synchronism.

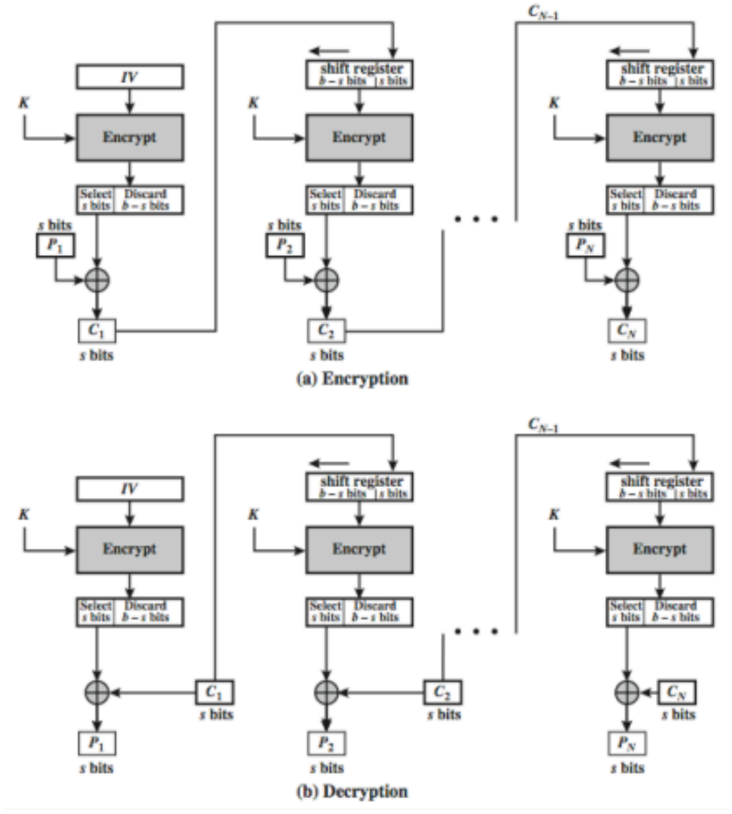


FIGURE 9. CIPHER FEEDBACK CIPHER (STALLINGS, 2017)

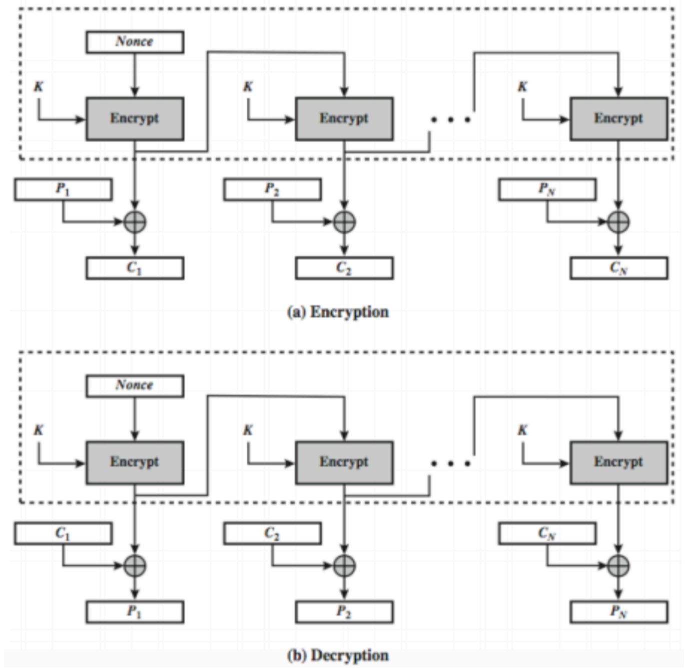


FIGURE 10. OUTPUT FEEDBACK CIPHER (STALLINGS, 2017)

The above section briefly introduces some typical cryptographic algorithms in the world over a few thousand years. However, there are other algorithms which are also important and useful, but they are omitted here. For instance, MACAlgo3 and CRT-RSA are the most powerful ciphers embedding in smart card for transaction.

2.3 Literature Review on PEKS

Alongside with the development of modern cryptography, a new cryptography called Identity-based Encryption (IBE) (Shamir, 1984) has been initially proposed by Shamir in 1984. However, the first secure IBE scheme has been introduced until 2001. Comparing with Public Key Infrastructure (PKI), IBE is independent of the online trusted third parties and is able to work independently. Provable security and efficiency (Boneh and Boyen, 2004; Waters, 2005; Gentry, 2006), key escrow (Boneh and Franklin, 2001; Paterson et al., 2003; Goyal, 2007) and anonymous problem (Boyen and Waters, 2006; Brandt and Sandholm, 2005; Clarkson et al., 2008) are the current main research topics of IBE and some applications based on IBE has also been introduced recently, such as Public Key Encryption with Keyword Search (PEKS).

PEKS plays a pivotal role in cryptography to secure data transmission between two different networks. It offers a secure and efficient method for users to search encrypted messages from the online third parties by a specific keyword. In the year of 2004, Boneh et al. formalized the first PEKS scheme (Boneh et al., 2004), which satisfies Indistinguishability under Chosen Plaintext Attack (IND-CPA) secure. Although the first PEKS scheme has IND-CPA secure, it should not be overlooked that it carries out some functional issues. For instance, it is only utilised to encrypt the keyword described in the file. If the user wishes to encrypt the whole message, he/she has to apply the traditional Public Key Encryption (PKE) algorithms to encrypt it. Later, Abdalla et al. camp up with a new encryption (E) (Abdalla et al., 2005) which is based on the original PEKS scheme. The new approach defines the consistency. For instance, the hash functions in PEKS schemes should be assumed as collision resistance. Otherwise, the PEKS scheme is not perfect consistency and then may suffer Off-line Keyword Guessing Attack (OKGA).

Both PEKS and PKE schemes require the secure channels between the receiver and the online third party to transmit the Trapdoor. However, it consumes huge human and material resources to build a secure channel and therefore, these PEKS schemes cannot be applied in the general public. In the year of 2008, Baek et al. presented a new method, namely "Secure Channel Free Public Key Encryption with Keyword Search (SCF-PEKS)" (Baek et al., 2008). It deletes the secure channel and becomes a cost-saving system comparing with BDOP's PEKS. Sometimes, the server is honest-but-curious so that it may exploit some details related to the keyword. Besides, the Trapdoor is transferred to the online third party via the general public network so that it can be intercepted by anyone and therefore, the outside attackers are able to explore the private interests and keyword from the Trapdoor. Hence, SCF-PEKS scheme seems to be subjected to OKGA. More specially, Byun et al. discovered, for the first time, that PEKS is vulnerable to OKGA in 2006 (Byun et al., 2006). They pointed out that the user always picked up the proverbial keyword (low entropy) so that the keyword for searching is in a narrow space. Consequently, the adversary has the abilities to guess the keyword. Later, Jeong et al. presented an open problem that is establishing provably secure and coherent PEKS approach against KGA is impossible (Jeong et al., 2009). Meanwhile, Yau et al. proposed that some PEKS schemes (i.e. SCF-PEKS) are also vulnerable to OKGA and those approaches are compromised by inside attackers (Yau et al., 2008). This is because that outside adversaries could capture the Trapdoor from the public network and then derive the keyword by OKGA. Afterwards, Tang et al. presented a new method to prevent Off-line KGA (Tang and Chen, 2010). However, the encryption algorithm of the proposed scheme is much complex. Soon later, Rhee et al. formalized a new SCF-PEKS scheme which firstly incorporates with the advantage of Trapdoor Indistinguishability (Rhee et al., 2010) to solve OKGA. In 2013, Zhao et al. introduced a new SCF-PEKS scheme satisfying Trapdoor Indistinguishability (Zhao et al., 2013). Comparing with Rhee et al's model, it is much more efficient. From then on, people concentrate on constructing much secure PEKS and SCF-PEKS schemes to resist OKGA (Yau et al., 2008; Hu and Liu, 2011; Chen, 2014; Sun et al., 2017; Mao et al., 2018; Noroozi and Eslami, 2019; Huang and Li, 2018).

The security of substantial PEKS and SCF-PEKS schemes relies on the difficulties of bilinear mapping with CDH, DDH, BDH and 1-BDHI assumptions. In 1994, Shor effectively solved the discrete logarithm problem by quantum computing and pointed out that quantum computers (Shor, 1994) could compromise the security of these PKI systems. After that, lattice-based cryptographic systems have undertaken a rapid development. Ajtai firstly provided a way to proof the difficulty of lattice problems (Ajtai, 1996). Later on, many schemes and applications with lattices have been proposed since then. These include ID-based Encryption systems (Gentry et al', 2008; Agrawal et al., 2010), hash functions (Ajtai, 1996; Micciancio, 2002), fully homomorphic encryption systems (Gentry, 2009; Gentry, 2010), PEKS (Gu et al., 2013; Hou et al., 2013) schemes and a lot more besides. In 2018, Zhang et al. proposes a new PEKS approach from lattice assumption in the base model with quantum computer resistance (Zhang et al., 2018), which is the mile stone in the post-quantum cryptographic communication era.

The PEKS schemes above only concentrate on “precise” keyword retrieve rather than solving format error (“etc” and “etc.”) and/or spell inconsistent (“common” and “comon”). In the year of 2010, Li et al. presented the definition of “Fuzzy Keyword Search” (Li et al., 2010) to PEKS scheme for supporting formation error and/or spell inconsistent. However, Li et al.’s scheme is significant but suffers Off-line KGA. Later on, Xu et al. presented a new mechanism with Fuzzy Keyword Search in 2013, which is able to prevent Off-line KGA (Xu et al., 2013).

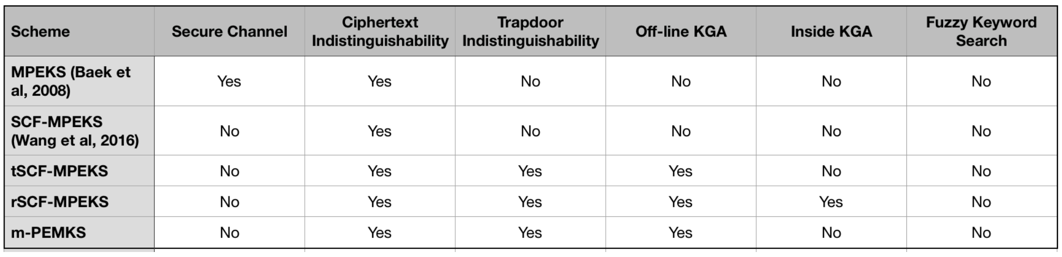
Many PEKS mechanisms are specialized in solving single keyword search problem rather than multiple keywords search issue. Golle et al. firstly came up with a PEKS model with conjunctive keyword search (Golle et al., 2004) in order to solve multiple keywords search. Soon later, Boneh et al. revisited the Golle et al’s model and then strengthened the PEKS model to stand by “conjunctive”, “subset” and range requests on the keywords (Boneh et al., 2007). In the meantime, Baek et al. defined “Public Key Encryption with Multi-keywords Search (MPEKS)” to address multiple keywords retrieve issue (Baek et al., 2008). However, a secure channel has to be considered to transmit Trapdoor in MPEKS scheme, which is similar as PEKS and PKE approaches. In 2009, Camenisch et al.’s proposed PEKS with oblivious keyword search (Camenisch et al., 2009) which enables the individual to achieve the Trapdoor from the secret key holder so that it blinds the key extraction and strengthens keyword privacy to against adversaries. Later, Cao et al. proposed PEKS with ranked multiple keywords retrieve on encrypted cloud data and set rules for privacy requirements (Cao et al., 2014). In 2016, Wang et al. designed a new SCF-MPEKS approach removing the secure channel (Wang et al., 2016). But SCF-MPEKS might be subjected to OKGA, if the infected server or receiver releases his/her secret key to the network. However, the majority of PEKS schemes may suffer IKGA, if the inside adversary executes one extra bilinear pairing operation. Therefore, Huang and Li came up with an efficient PEKS approach with User Authentication in 2018 that is able to resist IKGA but the scheme only aims for solving single keyword search problem (Huang and Li, 2018).

Apart from that, some representative PEKS schemes are also introduced as time goes by. For instance, PEKS with Delegated Search was formalized by Ibraimi et al., which is applied on detecting encrypted malicious code (Ibraimi et al., 2011). In 2014, Dual-Server PEKS was formalized by Chen et al. that is able to resist inherent insecurity (Chen et al., 2014). In the meaning time, He et al. came with up an approach in mobile social networks (He et al., 2016). It allows individuals to share contents and subscribes services.

This PhD thesis formalizes "*Trapdoor-indistinguishable Secure Channel Free Public Key Encryption with Multi-keywords Search (tSCF-MPEKS)*" and "*Robust Secure Channel Free Public Key Encryption with Multi-keywords search (rSCF-MPEKS)*" systems and subsequently proposes the concrete constructions of these models. The proposed schemes are able to solve both Single and Multiple Keyword(s) Search problems. Besides, the security models of *tSCF-MPEKS* and *rSCF-MPEKS* are also presented in the thesis. The proposed schemes are proved to be semantic secure in the security models under BDH and 1-BDHI assumptions so that both of them are able to resist OKGA. In addition, *rSCF-MPEKS* scheme applies User Authentication technique and therefore, IKGA has been avoided in this scheme. Last but not least, the thesis finally proposed a scheme called "*Public Key Encryption with Multi-keywords Search using Mamdani System (m-PEMKS)*", which incorporates with the advantages of Mamdani System (Fuzzy Logic) to solve Fuzzy Keyword (i.e. "latest") Search problem. It also should be noted that m-PEMKS has the ability to resist OKGA as well.

More specially, Table 2 below compares security and functionality between the typical MPEKS schemes (MPEKS, Baek et al., 2008; SCF-MPEKS, Wang et al., 2016) and three proposed schemes (tSCF-MPEKS, rSCF-MPEKS, m-PEMKS).

TABLE 2. A COMPARISON BETWEEN TYPICAL MPEKS SCHEMES AND THREE PROPOSED SCHEMES



Scheme	Secure Channel	Ciphertext Indistinguishability	Trapdoor Indistinguishability	Off-line KGA	Inside KGA	Fuzzy Keyword Search
MPEKS (Baek et al, 2008)	Yes	Yes	No	No	No	No
SCF-MPEKS (Wang et al, 2016)	No	Yes	No	No	No	No
tSCF-MPEKS	No	Yes	Yes	Yes	No	No
rSCF-MPEKS	No	Yes	Yes	Yes	Yes	No
m-PEMKS	No	Yes	Yes	Yes	No	No

According to the Table 2, the typical MPEKS (Baek et al., 2008) and SCF-MPEKS (Wang et al., 2016) schemes are vulnerable to Off-line KGA, but the proposed PEKS schemes, called tSCF-MPEKS, rSCF-MPEKS and m-PEMKS, satisfy Trapdoor Indistinguishability and Ciphertext Indistinguishability properties to resist Off-line KGA. Besides, rSCF-MPEKS incorporating with User Authentication technique is much strengthen and therefore, it could prevent Inside KGA. Apart from that, m-PEMKS scheme which applies Mamdani Fuzzy Inference System (Fuzzy Logic, Artificial Intellgence) has the powerful functionality to support Fuzzy Keyword Search, such as “latest”, “fastest”, etc.

2.4 Literature Review on Fuzzy Logic

In PEKS system, the user may search encrypted document by using imprecise keyword, such as “latest”, “biggest” etc. Due to PEKS ciphertext that contains fuzzy keyword leading to searching errors, therefore, Mamdani’s Fuzzy Inference method is able to be perfectly utilized in solving fuzzy keyword search problem. In 1973, Lotifi Zadeh came up with new fuzzy algorithms (Zadeh, 1973) to analyse complex systems and decision processes. Later, Ebrahim Mamdani revisited Lotifi’s approach and then proposed an inference system to control a steam engine and boiler combination based on linguistic rules from human knowledge (Mamdani, 1975). However, Mamdani-style inference is not computationally efficient, Michio Sugeno proposed a new fuzzy inference (Sugeno, 1985) using a single spike (a singleton) as the rule consequent. Recently, Fuzzy sets theory has been applied successfully in many areas. Singh et al. pointed out fuzzy systems could applied to classification, modelling control problems (Sing et al., 2006). Lermontov et al. analysed water quality using fuzzy set (Lermontov et al., 2009). Meanwhile, Marchini et al. proposed a framework for fuzzy indices of environmental conditions (Marchini et al., 2009).

3. Preliminary for the PEKS Research

3.1 Introduction

This chapter describes the preliminaries for the PEKS research, such as Number Theory (i.e. Bilinear pairing) and Game Theory , etc.

3.2 Number Theory

3.2.1 Bilinear Pairings (Boneh and Boyen, 2004)

Suppose G_1 is an additive cyclic group and G_T is a multiplicative cyclic group. Let P be a random generator of G_1 and a prime number g be the order of G_1. Suppose α and β are the components of Z_P. A bilinear pairing is considered to be a map $e : G_1 \times G_1 \to G_T$, which has the characters below:

i. Bilinearity: $e(\alpha X, \beta Y) = e(X, Y)^{\alpha\beta}$ for all $X, Y \in G_1$ and all $x, y \in Z_P$.

ii. Computability: $e(X, Y) \in G_T$ for any $X, Y \in G_1$.

iii. Non-degenerate: If P is a generator of G_1 then $e(P, P)$ is a generator of G_T.

3.2.2 The Bilinear Diffie-Hellman (BDH) assumption (Boneh and Boyen, 2004)

Let P, αP, βP and γP be the inputs (where $\alpha, \beta, \gamma \in Z_P$) and then calculate $e(P, P)^{\alpha\beta\gamma} \in G_T$. ξ is an advantage of the algorithm A that could solve BDH assumption in the group G_1, if $Pr[A(P, \alpha P, \beta P, \gamma P) = e(P, P)^{\alpha\beta\gamma}] \geq \xi$. Therefore, it is known that if no Probabilistic Polynomial Time (PPT) algorithm takes at least ξ advantage in addressing BDH assumption in the group G_1, BDH assumption will be held in G_1.

3.2.3 The 1-Bilinear Diffie-Hellman Inversion (1-BDHI) assumption (Boneh and Boyen, 2004)

Let P and αP be the inputs (where $\alpha \in Z_P$) and then compute $e(P,P)^{\frac{1}{\alpha}}$. ξ is an advantage of the algorithm A that could solve 1-BDHI assumption in the group G_1, if $Pr[A(P,\alpha P) = e(P,P)^{\frac{1}{\alpha}}] \geq \xi$. Therefore, it is known that if no PPT algorithm takes at least ξ advantage in addressing 1-BDHI assumption in the group G_1, 1-BDHI assumption will be held in G_1.

3.3 Public Key Encryption with Keyword Search (PEKS)

Let sender, server and receiver be three parties in PEKS scheme. The sender is a party who runs PEKS algorithm to create a Searchable ciphertext. Besides, the receiver is a party who executes Trapdoor algorithm to create a Trapdoor query. Once the server receives the encrypted messages from the sender and the receiver, he/she will run Test algorithm to estimate whether two ciphertexts contain the same keyword or not.

In 2004, BDOP proposed the first PEKS approach in order to address keyword search issue (Boneh et al., 2004). The PEKS scheme has five algorithms as follows:

1. $KeyGen_{Param}(1^n)$: Import 1^n, a common parameter cp is then created.

2. $KeyGen_{Receiver}(cp)$: Import cp, a public and a private keys (pk_{Rec}, sk_{Rec}) of the receiver are then created.

3. $PEKS(pk_{Rec}, w)$: Import the receiver's public key pk_{Rec} and a keyword w, a Searchable ciphertext $S = PEKS(pk_{Rec}, w)$ is then generated by the sender.

4. $Trapdoor(sk_{Rec}, w*)$: Import the receiver's private key sk_{Rec} and a keyword $w*$, a Trapdoor query $T_{w*} = Trapdoor(sk_{Rec}, w*)$ is then generated by the receiver.

5. $Test(pk_{Rec}, S, T_{w*})$: Import the receiver's public key pk_{Rec}, a Searchable encryption $S = PEKS(pk_{Rec}, w)$ and a Trapdoor query $T_{w*} = Trapdoor(sk_{Rec}, w*)$. Then, the server will test whether $w = w*$. If so, output "yes" and "no" otherwise.

3.4 Secure Channel Free Public Key Encryption with Multiple Keywords Search (SCF-MPEKS)

PEKS scheme has its limitations. It not only requires a secure channel to deliver Trapdoor query to the server, but also cannot search multiple keywords. So, Wang et al. proposed SCF-MPEKS scheme (Wang et al., 2016) to solve these problems, which contains six polynomial time algorithms as follows:

1. $KeyGen_{Param}(1^n)$: Import 1^n to obtain a common parameter cp.

2. $KeyGen_{Server}(cp)$: Import cp to obtain a public and private keys (pk_{Ser}, sk_{Ser}) of the server.

3. $KeyGen_{Receiver}(cp)$: Import cp to obtain a public and a private keys (pk_{Rec}, sk_{Rec}) of the receiver.

4. $SCF-MPEKS(pk_{Ser}, pk_{Rec}, W)$: Import the server's public key pk_{Ser} and the receiver's public key pk_{Rec} in order to obtain a Searchable ciphertext $S = SCF-MPEKS(pk_{Ser}, pk_{Rec}, W)$ of multi-keywords $W = (w_1, w_2, \ldots, w_\eta)$.

5. $Trapdoor(sk_{Rec}, w)$: Import the receiver's private key sk_{Rec} in order to produce a Trapdoor query $T_w = Trapdoor(sk_{Rec}, w)$ of a keyword w.

6. $Test(sk_{Ser}, S, T_w)$: Import the server's private key sk_{Ser}, a Searchable encryption $S = SCF-MPEKS(pk_{Ser}, pk_{Rec}, W)$ and a Trapdoor query $T_w = Trapdoor(sk_{Rec}, w)$. Then, if W includes w, export "yes" and "no" otherwise.

3.5 Security Verification Models

Goldwasser and Micali proposed the semantic security in the year of 1984. The semantic security (Goldwasser and Micali, 1984) is the prototype for provable security. That is an attacker cannot obtain any one bit plaintext even though he/she intercepts the whole ciphertext. Therefore, semantic secure is used for versifying the security of PEKS schemes.

Consider **A** is an adversary who may break the crypto-system. While, **E** is a challenger who sets up the system and accepts the challenge from the adversary **A**.

3.5.1 Indistinguishable Choose Plaintext Attack (IND-CPA) Game

1. **Setup**: The challenger **E** establishes the PEKS system ξ and the attacker **A** obtains the public key of the system ξ.
2. **Challenge**: The attacker **A** sends a plaintext pair (M_0, M_1) to the challenger **E**. Then, **E** chooses $b \in \{0,1\}$ uniformly at random and also encrypts one of the above plaintext. Finally, **E** sends the ciphertext C_b to **A**.
3. **Guess**: **A** guesses $b^* \in \{0,1\}$ and wins **IND-CPA Game**, if $b^* = b$.

For any Probabilistic Polynomial Time (PPT) adversary **A** against the **IND-CPA Game**, its advantage $Adv_{\xi,A}^{IND-CPA}(k)$ is negligible.

3.5.2 Indistinguishable Choose Ciphertext Attack (IND-CCA) Game

However, **IND-CPA** has its limitation. For instance, it is vulnerable to the deterministic cryptography algorithms, such as RSA, Rabin, etc. On the contrary, **IND-CPA** is able to resist passive attacks (i.e. Monitor) in the probabilistic cryptography algorithms (i.e. ElGamal, etc.) but cannot prevent active attacks (i.e. Fault injection) in these algorithms. Therefore, Naor and Yung came up with the concept of **Chosen**

Ciphertext Attack (CCA) (Naor and Yung, 1990) in 1990. It allows the attacker **A** querying Oracle many times before the **Challenge** step.

1. **Setup**: The challenger **E** establishes the PEKS system ξ and the attacker **A** obtains the public key of the system ξ.
2. **Training**: The attacker **A** uploads the encrypted message C to **E** as many times as possible. **E** decrypts the ciphertext C and then sends the corresponding plaintext M back to **A**.
3. **Challenge**: The attacker **A** sends a plaintext pair (M_0, M_1) to the challenger **E**. Then, **E** chooses $b \in \{0,1\}$ uniformly at random and also encrypts one of the above plaintext. Finally, **E** sends the ciphertext C_b to **A**.
4. **Guess**: **A** guesses $b^* \in \{0,1\}$ and wins **IND-CCA Game**, if $b^* = b$.

For any Probabilistic Polynomial Time (PPT) adversary **A** against the **IND-CCA Game**, its advantage $Adv_{\xi,A}^{IND-CCA}(k)$ is negligible.

3.5.3 Indistinguishable Adaptive Choose Ciphertext Attack (IND-CCA2) Game

In 1991, Rackoff and Simon proposed the concept of **Adaptive Choose Ciphertext Attack** (CCA2) (Rackoff and Simon, 1991), which enables the adversary **A** to query Oracle many times after the **Challenge** step.

1. **Setup**: The challenger **E** builds the PEKS system ξ and the attacker **A** obtains the public key of the system ξ.
2. **Training**: The attacker **A** sends the encrypted message C to **E** as many times as possible. **E** decrypts the ciphertext C and then sends the corresponding plaintext M back to **A**.

3. **Challenge**: The attacker **A** sends a plaintext pair (M_0, M_1) to the challenger **E**. Then, **E** chooses $b \in \{0,1\}$ uniformly at random and also encrypts one of the above plaintext. Finally, **E** sends the ciphertext C_b to **A**.

4. **Training**: The attacker **A** sends the encrypted message C $(C \neq C_b)$ to **E** as many times as possible. **E** decrypts the ciphertext C and then sends the corresponding plaintext M back to **A**.

5. **Guess**: **A** guesses $b^* \in \{0,1\}$ and wins **IND-CCA2 Game**, if $b^* = b$.

For any Probabilistic Polynomial Time (PPT) adversary **A** against the **IND-CCA2 Game**, its advantage $Adv_{\xi,A}^{IND-CCA2}(k)$ is negligible.

3.6 The Procedure of PEKS Verification

Suppose system 2 is completely secure which has BDH and 1-BDHI assumptions and system 1 is a designed PEKS system. Consider the Challenger establishes the system 2 and the adversary **E** challenges the system 2. Meanwhile, the adversary **E** could also be regarded as a challenger who establishes the system 1. Therefore, **E** is able to accept the challenges from the adversary **A** and sends the responses back to **A**. In order to break the system 2, **E** is able to train **A** and then applies **A**'s results to the system 2. However, due to system 2 being a secure system, **E** cannot break the system 2 and consequently, **A** could not break the system 1. Thus, the designed PEKS system is proved to be a secure system (Figure 11).

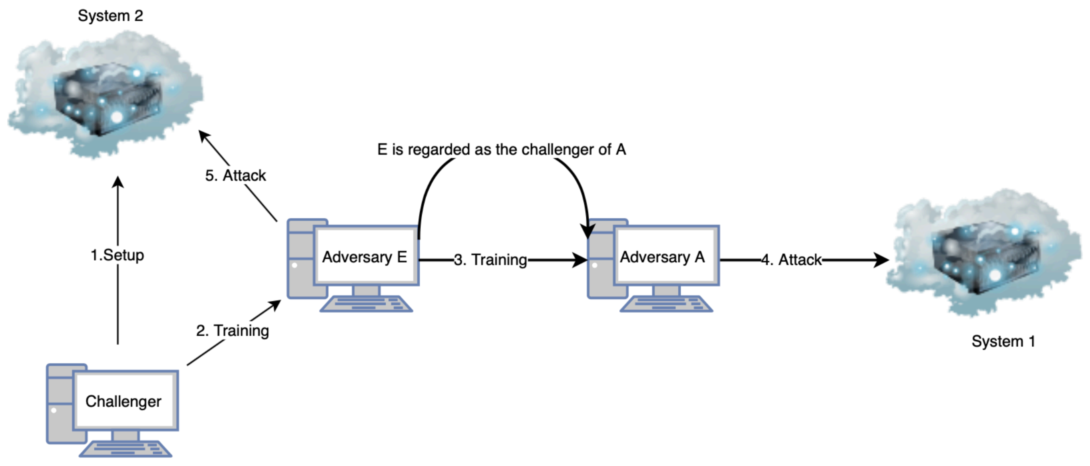


FIGURE 11. THE PROCEDURE OF PEKS VERIFICATION

3.7 Fuzzy Logic

3.7.1 Fuzzy Rule Based Model

The fuzzy rule based model (Figure 12) is based on Mamdani Fuzzy Inference System and consists four steps as follows:

1. **Fuzzification of the input variables:** The aim of this step is transforming crisp inputs into fuzzy inputs by the membership functions. Although there are substantial curves can be used in fuzzification process, Gaussian, triangular and trapezoidal membership functions are the most widely used in it.

2. **Rules evaluation:** The fuzzified inputs are applied to the antecedents of the fuzzy rules and then apply fuzzy logic operations (AND, OR, NOT) to these rule antecedents.

3. **Aggregation of the rule outputs:** The membership functions of all rule consequents previously clipped or scaled are combined into a single fuzzy set.

4. **Defuizzification:** Some defuzzification methods, such as Center of Gravity (COG), Mean Max, etc., can be utilized to transformed fuzzy outputs into crisp outputs.

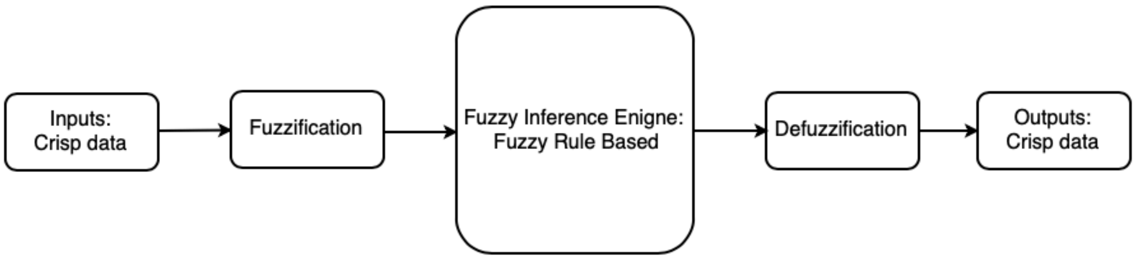


FIGURE 12. FUZZY RULE BASED MODEL

3.7.2 The Process of Mamdani-Type Fuzzy Inference System

The Mamdani-Type Fuzzy Inference System contains five stages (Wang et al., 2015) in the following:

1. Fuzzify the input variables (crisp data)
2. Apply fuzzy operator
3. Apply implication method
4. Apply aggregation method
5. Defuzzification

4. Trapdoor-indistinguishable Secure Channel Free Public Key Encryption with Multi-keywords Search

4.1 Introduction

According to the section 1.4, it is known that many current Public Key Encryption with Multiple Keywords Search (MPEKS) schemes suffer Off-line Keyword Guessing Attack (OKGA). Therefore, this chapter gives a formal definition of MPEKS scheme, which incorporates with Trapdoor Indistinguishability so that it has the ability to resist OKGA. More specially, it firstly defines a new PEKS scheme namely "*Trapdoor-indistinguishable Secure Channel Free Public Key Encryption with Multi-keywords Search (tSCF-MPEKS)*" and the security verification models as well. After that, a concrete construction of *tSCF-MPEKS* is proposed following by the correctness analysis, security verification and efficiency and performance analysis.

4.2 The Outline of tSCF-MPEKS

In 2008, Baek et al. defined PEKS scheme with multiple keywords search to address multi-keywords search problem (Baek et al., 2008). However, a secure channel between the receiver and the online third party is required to transmit the Trapdoor request. There is no doubt that establishing a secure channel consumes huge human and material resources, which seems impossible in reality. Later on, a Secure Channel Free PEKS with Multiple Keywords Search approach (Wang et al., 2016) was introduced by Wang et al. in 2016. Although the new method removes the secure channel, it might suffer OKGA, if the server's or receiver's private key is compromised and released to the public networks. The PhD thesis defines a new PEKS scheme (Figure 13) called "*Trapdoor-indistinguishable Secure Channel Free Public Key Encryption with Multi-*

keywords Search (tSCF-MPEKS)" and it incorporates with Trapdoor indistinguishability to deal with both Single and Multiple Keyword(s) Search issues and OKGA. The *tSCF-MPEKS* contains six following algorithms:

1. $KeyGen_{Param}(1^n)$: Import 1^n, a common parameter cp is then created.

2. $KeyGen_{Server}(cp)$: Import cp, a public and a private keys (pk_{Ser}, sk_{Ser}) of the server are then created.

3. $KeyGen_{Receiver}(cp)$: Import cp, a public and a private keys (pk_{Rec}, sk_{Rec}) of the receiver are then created.

4. $SCF-MPEKS(pk_{Ser}, pk_{Rec}, W)$: Import the server's public key pk_{Ser} and the receiver's public key pk_{Rec}, a Searchable ciphertext $S = SCF-MPEKS(pk_{Ser}, pk_{Rec}, W)$ is then generated by the sender, where $W = (w_1, w_2, \ldots, w_\eta)$.

5. $Trapdoor(pk_{Ser}, sk_{Rec}, w)$: Import the server's public key pk_{Ser} and the receiver's private key sk_{Rec}, a Trapdoor query $T_w = Trapdoor(pk_{Ser}, sk_{Rec}, w)$ is then generated by the receiver.

6. $Test(sk_{Ser}, S, T_w)$: Import the server's private key sk_{Ser}, a Searchable encryption $S = SCF-MPEKS(pk_{Ser}, pk_{Rec}, W)$ and a Trapdoor query $T_w = Trapdoor(pk_{Ser}, sk_{Rec}, w)$. The server checks whether the W includes w. If so, export "yes". Otherwise, export "no".

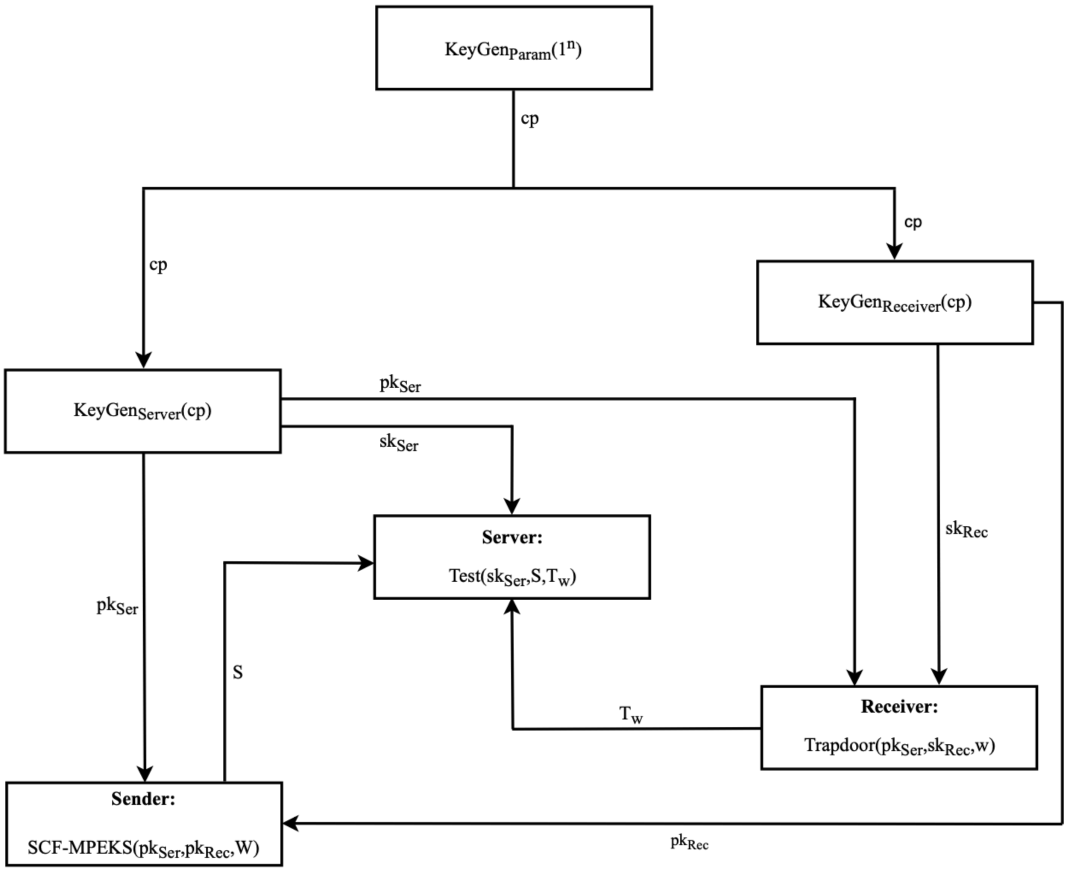


FIGURE 13. THE OUTLINE OF TSCF-MPEKS

4.3 The Security Models of tSCF-MPEKS

As discussed in (Baek et al.; Wang et al.), *tSCF-MPEKS* is **IND-CPA** and **Trapdoor-IND-CPA**.

The definition of **IND-CPA** security means that the untrusted server may not determine which Searchable ciphertext has which encrypted keyword, if the Trapdoor query that contains the given keyword has not been obtained by the server (***Game1***). Besides, if the server's private key has not been obtained by the untrusted receiver, he/she could not estimate whether the SCF-MPEKS ciphertext and the Trapdoor request contain the same keyword or not, even though all Trapdoors for any keyword are intercepted (***Game2***).

The definition of **Trapdoor-IND-CPA** security means that an outside adversary cannot observe any difference between Trapdoors for any two distinct keywords (***Game3***).

Therefore, the *tSCF-MPEKS's* **IND-CPA** security and **Trapdoor-IND-CPA** security are formalized as follows: Suppose **A** is an adversary and **E** is a challenger.

***Game1*: Let A suppose to be an untrusted server.**

Stage1 (Setup): $KeyGen_{Param}(1^n)$, $KeyGen_{Server}(cp)$ and $KeyGen_{Receiver}(cp)$ are called by **E** in order to generate a common parameter cp, the key pairs (pk_{Ser}, sk_{Ser}) and (pk_{Rec}, sk_{Rec}) of the server and the receiver. Then, **E** sends cp, pk_{Ser}, sk_{Ser} and pk_{Rec} to **A**.

Stage2 (Trapdoor queries): Adaptably, **E** is able to return any Trapdoor query T_w for any keyword w to **A**.

Stage3 (Challenge simulation): A target keyword-vector pair $[W_0 = (w_{01}, \dots, w_{0\eta}),\ W_1 = (w_{11}, \dots, w_{1\eta})]$ is sent from **A** to **E**. It is known that W_0 and W_1 cannot be requested in **Stage2 (*Game1*)**. Once **E** obtains the pair, the $SCF-MPEKS$ algorithm will be called by **E** for creating a Searchable ciphertext $C = SCF-MPEKS(pk_{Ser}, pk_{Rec}, W_\xi)$, where $\xi \in \{0,1\}$. Finally, C will be sent back from **E** to **A**.

Stage4 (Trapdoor queries): **E** can continue return any Trapdoor query T_w for any keyword w to **A** as in **Stage2 (*Game1*)**, only if $w \notin W_0, W_1$.

Stage5 (Guess): **A** guesses $\xi^* \in \{0,1\}$ and wins ***Game1***, if $\xi^* = \xi$.

Game2: **Let A suppose to be an untrusted receiver.**

Stage1 (Setup): $KeyGen_{Param}(1^n)$, $KeyGen_{Server}(cp)$ and $KeyGen_{Receiver}(cp)$ are called by **E** in order to generate a common parameter cp, the key pairs (pk_{Ser}, sk_{Ser}) and (pk_{Rec}, sk_{Rec}) of the server and the receiver. Then, **E** sends cp, pk_{Rec}, sk_{Rec} and pk_{Ser} to **A**.

Stage2 (Challenge simulation): A target keyword-vector pair $[W_0 = (w_{01}, ..., w_{0\eta}),\ W_1 = (w_{11}, ..., w_{1\eta})]$ is sent from **A** to **E**. It is known that $T_{w_{0i}}$ and $T_{w_{1i}}$ are not able to be requested during *Test* algorithm on which $i = 1,...,\eta$. Once **E** obtains the pair, the $SCF-MPEKS$ algorithm will be called by **E** for creating a Searchable ciphertext $C = SCF-MPEKS(pk_{Ser}, pk_{Rec}, W_\xi)$, where $\xi \in \{0,1\}$. Finally, C will be sent back from **E** to **A**.

Stage3 (Guess): **A** guesses $\xi^* \in \{0,1\}$ and wins ***Game2***, if $\xi^* = \xi$.

A's advantage to win ***Game1*** and ***Game2*** is listed below:

$$Adv^{IND-CPA}_{tSCF-MPEKS,A_i}(k) = |Pr[\xi^* = \xi] - 1/2|. \quad (i = 1,2)$$

So, the *tSCF-MPEKS* system is considered to be **IND-CPA** secure as long as the $Adv^{IND-CPA}_{tSCF-MPEKS,A_i}(k)$ is trivial.

***Game3*:** **Let A suppose to be an outside attacker.**

Stage1 (Setup): $KeyGen_{Param}(1^n)$, $KeyGen_{Server}(cp)$ and $KeyGen_{Receiver}(cp)$ are called by **E** in order to generate a common parameter cp, the key pairs (pk_{Ser}, sk_{Ser}) and (pk_{Rec}, sk_{Rec}) of the server and the receiver. Then, **E** sends cp, pk_{Ser}, sk_{Rec} to **A** and keeps sk_{Ser}, sk_{Rec} from **A**.

Stage2 (Trapdoor queries): Adaptably, **E** is able to return any Trapdoor query T_w for any keyword w to **A**.

Stage3 (Challenge simulation): A target keyword pair (w_0, w_1) is sent from **A** to **E.** It is known that w_0 and w_1 cannot be requested in **Stage2 (*Game3*)**. Once **E** obtains the keyword pair, the $Trapdoor$ algorithm will be called by **E** for creating a Trapdoor query $T_w = Trapdoor(pk_{Ser}, sk_{Rec}, w_\xi)$, where $\xi \in \{0,1\}$. Finally, T_w will be sent back from **E** to **A**.

Stage4 (Trapdoor queries): E can continue return any Trapdoor query T_w for any keyword w to **A** as in **Stage2 (*Game3*)**, only if $w \neq w_0, w_1$.

Stage5 (Guess): A guesses $\xi^* \in \{0,1\}$ and wins ***Game3***, if $\xi^* = \xi$.

A's advantage to win ***Game3*** is listed below:

$$Adv_{tSCF-MPEKS,A_3}^{Trap-IND-CPA}(k) = |Pr[\xi^* = \xi] - 1/2|.$$

So, the *tSCF-MPEKS* system is considered to be **Trapdoor-IND-CPA** secure as long as the $Adv_{tSCF-MPEKS,A_3}^{Trap-IND-CPA}(k)$ is trivial.

4.4 The Concrete Construction of tSCF-MPEKS

1. $KeyGen_{Param}(k)$: Suppose G_1 is an additive cyclic group and G_T is a multiplicative cyclic group. Let P be a random generator of G_1 and a prime number $g \geq 2^k$ be the order of G_1. A bilinear pairing is considered to be a map $e : G_1 \times G_1 \rightarrow G_T$. Suppose $H : \{0,1\}^\star \rightarrow G_1$ and $H^* : G_T \rightarrow \{0,1\}^*$ are two particular hash functions. Therefore, a common parameter $cp = \{g, P, G_1, G_T, e, H, H^*\}$ can be achieved by the $KeyGen_{Param}(k)$ algorithm.

2. $KeyGen_{Server}(cp)$: The server selects $m \in Z_P$ uniformly at random and subsequently calculates $M = mP$. In addition, the server also randomly selects $K \in G_1$. So, $pk_{Ser} = (cp, M, K)$ and $sk_{Ser} = (cp, m)$ are the server's public and private keys.

3. $KeyGen_{Receiver}(cp)$: The receiver selects $n \in Z_P$ uniformly at random and subsequently calculates $N = nP$. So, $pk_{Rec} = (cp, N)$ and $sk_{Rec} = (cp, n)$ are the receiver's public and private keys.

4. $SCF - MPEKS(pk_{Ser}, pk_{Rec}, W)$: The sender selects $t \in Z_P$ uniformly at random and $W = (w_1, w_2, \ldots w_\eta)$, and then calculates a Searchable ciphertext $C = (X, Y_1, Y_2, \ldots, Y_\eta) = (tM, H^*(V_1), H^*(V_2), \ldots, H^*(V_\eta))$, where $V_1 = e(H(w_1), N)^t$, $V_2 = e(H(w_2), N)^t$,..., $V_\eta = e(H(w_\eta), N)^t$.

5. $Trapdoor(pk_{Ser}, sk_{Rec}, w^*)$: The receiver selects $t^* \in Z_P$ uniformly at random and subsequently calculates $T_w = (T_1, T_2)$, where $T_1 = nH(w^*) \oplus e(M, K)^{t^*+n}$ and $T_2 = e(M, t^*K)$.

6. $Test(C, T_w, sk_{Ser})$: For $i \in \{1,2,...,\eta\}$, the online server firstly computes $T = T_1 \oplus T_2 \bullet e(mK, N) = nH(w^*)$. After that, the server tests whether $H^*[e(T, \frac{X}{m})] = Y_i$ or not. If so, output "yes"; otherwise, output "no".

4.5 The Correctness of tSCF-MPEKS

Suppose W and w^* are keyword-vector and keyword respectively in $SCF - MPEKS$ and $Trapdoor$ algorithms. This PEKS approach is considered to be corrected as long as W includes w^*. The correctness verification is listed below.

Note that $\bullet$ stands for Multiplication and $\oplus$ stands for Exclusive Or.

According to Bilinear pairing, note also that $e(M,K) = e(K,M)$ and $e(M,K)^{t^*+n} = e(t^*M, nK) = e(nM, t^*K)$.

Therefore, for $i \in \{1,2,...,\eta\}$,

Firstly,

$$\begin{aligned} T &= T_1 \oplus T_2 \bullet e(mK, N) \\ &= nH(w^*) \oplus e(M,K)^{t^*+n} \oplus e(M, t^*K) \bullet e(mK, nP) \\ &= nH(w^*) \oplus e(M,K)^{t^*+n} \oplus e(M, t^*K) \bullet e(K, mP)^n \\ &= nH(w^*) \oplus e(M,K)^{t^*+n} \oplus e(M,K)^{t^*} \bullet e(M,K)^n \\ &= nH(w^*) \oplus e(M,K)^{t^*+n} \oplus e(M,K)^{t^*+n} \\ &= nH(w^*) \end{aligned}$$

Secondly,

$$\begin{aligned} H^*[e(T, \frac{X}{m})] &= H^*[e(nH(w^*), \frac{tM}{m})] \\ &= H^*[e(nH(w^*), tP)] \\ &= H^*[e(H(w^*), N)^t] \\ &= Y_i \end{aligned}$$

Therefore, the algorithm is completely correct.

4.6 The Security Analysis of tSCF-MPEKS

The *tSCF-MPEKS* approach possesses the characters of Ciphertext Indistinguishability and Trapdoor Indistinguishability against Chosen Plaintext Attack (CPA) whose security relies on BDH and 1-BDHI assumptions (Boneh and Boyen, 2004).

The proposed approach above could be regarded as **IND-CPA** secure in ***Game1*** under the random oracle model, if the BDH assumption (Boneh and Boyen, 2004) is completely difficult.

Game1: **Let A suppose to be an untrusted server.**

Consider that the challenger **E** is able to achieve the input $(g, P, G_1, G_T, e, \alpha P, \beta P, \gamma P)$ of BDH assumption (Boneh and Boyen, 2004). **E** sets up the computation of a BDH key $e(P, P)^{\alpha\beta\gamma}$ of αP, βP and γP using **A**'s **IND-CPA** as a goal. Apart from that, **A** queries at most h and h^* times hash function requests.

Stage1 (Setup)

E chooses $N = \alpha P$ in the beginning. Then, **E** chooses $m \in Z_P$ uniformly at random and also computes $M = mP$. In addition, **E** randomly selects $K \in G_1$. Finally, the following parameters are returned by **E**, which are the common parameter $(g, P, G_1, G_T, e, H, H^*)$, the server's public/private keys (cp, M, K) and (cp, m), and the receiver's public key (cp, N). Besides, two specific hash functions H and H^* are selected by **E** in the following:

- **A** is able to request a keyword w_i to H function at any time. After that, **E** traverses a tuple $(w_i, \mu_i, \nu_i, \varepsilon_i)$ from H_List that is initially empty. If the tuple exists, **E** will reply $H(w_i) = \mu_i$ to **A**. Otherwise, the challenger **E** executes the operations below:

i. The challenger **E** randomly selects a coin ε_i and then computes $Pr[\varepsilon_i = 0] = \frac{1}{h+1}$.

ii. The challenger **E** randomly chooses $\nu_i \in Z_P$. If $\varepsilon_i = 0$, $\mu_i = \beta P + \nu_i P$ will be computed by **E**. Similarly, $\mu_i = \nu_i P$ will be computed by **E** once $\varepsilon_i = 1$.

iii. **A** receives μ_i from **E**. Meanwhile, **E** adds $(w_i, \mu_i, \nu_i, \varepsilon_i)$ into H_List.

- **A** is able to request V_i to H^* function at any time. Later on, **E** traverses a tuple (V_i, Y_i) from H^*_List. If the tuple exists, **E** will return Y_i to **A**. Otherwise, **E** randomly selects $Y_i \in \{0,1\}^{\bullet}$ and replies Y_i to **A**. Finally, **E** adds (V_i, Y_i) into H^*_List.

Stage2 （Trapdoor queries)

If **A** queries a Trapdoor request with a specific keyword w_i, **E** will do the operations below:

- The challenger **E** recalls the above algorithms in order to simulate H function for generating a tuple $(w_i, \mu_i, \nu_i, \varepsilon_i)$. If $\varepsilon_i = 0$, **E** will output "Suspension" and also terminate the system. Otherwise, he/she executes the following step.

- The challenger **E** randomly chooses $t^* \in Z_P$ and sequently calculates $T_1 = \nu_i N \oplus e(M,K)^{t^*+\alpha} = \nu_i \alpha P \oplus e(M,K)^{t^*+\alpha} = \alpha\mu_i \oplus\ e(M,K)^{t^*+\alpha} = \alpha H(w_i) \oplus e(M,K)^{t^*+\alpha}$ and $T_2 = e(M, t^*K)$. So, $T_w = (T_1, T_2)$.

Stage3 (Challenge simulation)

The adversary **A** sends a keyword-vector pair $[W_0 = (w_{01}, \ldots., w_{0n})$, $W_1 = (w_{11}, \ldots., w_{1n})]$ to **E**. Once **E** achieves the pair, he/she will conduct the following steps:

- The challenger **E** chooses $i \in \{1,2,...,\eta\}$ uniformly at random.

- The challenger **E** recalls the above algorithms in order to simulate H function for obtaining two tuples $(w^*_{0i}, \mu^*_{0i}, \nu^*_{0i}, \varepsilon^*_{0i})$ and $(w^*_{1i}, \mu^*_{1i}, \nu^*_{1i}, \varepsilon^*_{1i})$. If ε^*_{0i} and ε^*_{1i} are equal to 1, **E** will output "Suspension" and also terminate the system. Otherwise, **E** executes the following step.

i. The challenger **E** recalls the above algorithms again to simulate H function at $2(\eta - 1)$ times so that **E** is able to create two tuples' vectors $\{(w^*_{01}, \mu^*_{01}, \nu^*_{01}, \varepsilon^*_{01}), \ldots,$

$(w^*_{0i-1}, \mu^*_{0i-1}, \nu^*_{0i-1}, \varepsilon^*_{0i-1})$, $(w^*_{0i+1}, \mu^*_{0i+1}, \nu^*_{0i+1}, \varepsilon^*_{0i+1}), \ldots,$ $(w^*_{0\eta}, \mu^*_{0\eta}, \nu^*_{0\eta}, \varepsilon^*_{0\eta})\}$ and $\{(w^*_{11}, \mu^*_{11}, \nu^*_{11}, \varepsilon^*_{11}), \ldots,$ $(w^*_{1i-1}, \mu^*_{1i-1}, \nu^*_{1i-1}, \varepsilon^*_{1i-1}), (w^*_{1i+1}, \mu^*_{1i+1}, \nu^*_{1i+1}, \varepsilon^*_{1i+1}), \ldots,$ $(w^*_{1\eta}, \mu^*_{1\eta}, \nu^*_{1\eta}, \varepsilon^*_{1\eta})\}$. If $\varepsilon^*_{0j} = \varepsilon^*_{1j} = 0$ for all $j = 0,...,i-1,i+1,...,\eta$, **E** will export "Suspension" and consequently terminate the system. Otherwise, **E** executes the steps below:

— The challenger **E** randomly selects $\delta \in \{0,1\}$.

— The challenger **E** randomly selects $Y_i \in \{0,1\}^{\bullet}$ for generating a target $SCF-MPEKS$ encryption $C^* = (X^*,\ Y^*_1, Y^*_2, \ldots, Y^*_\eta) = (\gamma M, H^*[B_1],$ $H^*[B_2], \ldots, H^*[B_\eta])$.

So,

$C^* = (X^*, Y^*_1, \ldots, Y^*_{i-1}, Y^*_{i+1}, \ldots, Y^*_\eta) = (\gamma M, H^*[e(H(w_{\delta_1}), N)^\gamma], \ldots, H^*[e(H(w_{\delta_{i-1}}),$ $N)^\gamma], H^*[e(H(w_{\delta_i+1}), N)^\gamma], \ldots, H^*[e(H(w_{\delta_\eta}), N)^\gamma])$.

Note that,

$B_i = e(H(w_{\delta_i}), N)^\gamma = e(\beta P + \nu_{\delta_i} P, \alpha P)^\gamma = e(\beta P, \alpha P)^\gamma \bullet e(\nu_{\delta_i} P, \alpha P)^\gamma = e(P, P)^{\alpha\beta\gamma}$ $\bullet\, e(\gamma P, \alpha P)^{\nu_{\delta i}}$

Note also that $e(\nu_{\delta_i} P, \alpha P)^\gamma = e(\nu_{\delta_i} P, N)^\gamma = e(H(w_{\delta_i}), N)^\gamma$

Stage4 (Trapdoor queries)

E can continue return any Trapdoor query T_{w_i} for any keyword w_i to **A** as in **Stage2 (*Game1*)**, only if $w_i \notin W_0, W_1$.

Stage5 (Guess)

A outputs $\delta^* \in \{0,1\}$ as the guess. Then, **E** chooses d from H^* function and returns the guessed BDH key $\frac{d_{\delta^*_i}}{e(\gamma P, \alpha P)^{\nu_{\delta^*_i}}}$.

Analysis of *Game1*

Stage1-5 describes the procedure and operations of the challenger **E**. It remains to show that BDH assumption (Boneh and Boyen, 2004) is satisfied in ***Game1***. To do so, the first thing is to analyze that the challenger **E** does not stop during the simulation. Therefore, three events are formalized below:

Event1: The challenger **E** does not stop during **Stage2** (**Trapdoor queries)** and **Stage4 (Trapdoor queries)**.

Event2: The challenger **E** does not stop during **Stage3 (Challenge simulation)**.

Event3: The adversary **A** is not able to request either $H^*(e(H(w_{0i}^*), N)^\gamma)$ or $H^*(e(H(w_{1i}^*), N)^\gamma)$.

Claim 1: $Pr[Event1] \geq \frac{1}{e}$

Proof: Consider that **A** cannot request the same keyword twice in **Stage2** and **Stage4**. So, $\frac{1}{h+1}$ is the probability causing **E** for suspension. From the previous definition, **A** queries at most h ($h > 0$) Trapdoor requests so that the probability that the system which does not be terminated by **E** in all Trapdoor queries is at least $(1 - \frac{1}{h+1})^h \geq \frac{1}{e}$.

Claim 2: $Pr[Event2] \geq (\frac{1}{h+1}) \bullet (\frac{h}{h+1})^{2(\eta-1)}$

Proof: If $\varepsilon_0 = \varepsilon_1 = 1$, the system will be terminated by **E** during **Stage3**. So, $1 - (1 - \frac{1}{h+1})^2$ is the probability that **E** does not suspend it. In addition, if $\varepsilon_{0j}^* = \varepsilon_{1j}^* = 0$ for all $j = 0,...,i-1,i+1,...,\eta$, the system will be terminated by **E**. Overall, the probability that the system which does not be terminated by **E** during **Stage3** is at least $(1 - \frac{1}{h+1})^{2(\eta-1)}\{1 - (1 - \frac{1}{h+1})^2\} \geq (\frac{1}{h+1}) \bullet (\frac{h}{h+1})^{2(\eta-1)}$.

Claim 3: $Pr[Event3] \geq 2\xi$

Proof: As discussed in (Baek et al., 2008), let $Hybrid_r$ ($r \in \{1,2,...,\eta\}$) be an **event** that the adversary **A** can correctly guess the keyword of the left part of a "hybrid" $SCF - MPEKS$ encryption formed with r, coordinates from w_δ followed by $(\eta - r)$ coordinates from $w_{1-\delta}$.

So, $Pr[Event3] = 2\Sigma_{j=1}^{\eta}(Pr[Hybrid_r] - Pr[Hybrid_{r-1}]) = 2(Pr[Hybrid_r] - Pr[Hybrid_0]) = 2\xi$.

Overall, due to **A** queries either $H^*(e(H(w_{0i}^*), N)^\gamma)$ or $H^*(e(H(w_{1i}^*), N)^\gamma)$ being at least 2ξ, the probability that **A** querying $H^*(e(H(w_{ji}^*), N)^\gamma)$ is at least ξ. Therefore, the success probability ξ^* achieved by **E** is $(\frac{h}{h+1})^{2(\eta-1)} \bullet \frac{\xi}{e(h+1)h^*}$, which is negligible.

The proposed scheme above could be regarded as **IND-CPA** secure in ***Game2*** under the random oracle model, if the 1-BDHI assumption (Boneh and Boyen, 2004) is completely difficult.

Game2: **Let A suppose to be an untrusted receiver.**

Consider that the challenger **E** is able to achieve the input $(g, P, G_1, G_T, e, \alpha P)$ of 1-BDHI assumption (Boneh and Boyen, 2004). **E** sets up the computation of a 1-BDHI key $e(P, P)^{\frac{1}{\alpha}}$ of αP using **A**'s **IND-CPA** as a goal. Apart from that, **A** queries at most h and h^* times hash function requests.

Stage1 (Setup)

E selects $M = \alpha P$ and $K \in G_1$ in the beginning. Then, **E** randomly chooses $n \in Z_P$ and also computes $N = nP$. After that, the following parameters are returned by **E**, which are the common parameter $(g, P, G_1, G_T, e, H, H^*)$, the server's public key

(cp, M, K), and the receiver's public/private keys (cp, N) and (cp, n). Besides, two specific hash functions H and H^* are selected by **E** in the following:

— **A** is able to request a keyword w_i to H function at any time. Later on, **E** traverses a tuple (w_i, μ_i, ν_i) from H_List. If the tuple exists, **E** will return μ_i to **A**. Otherwise, **E** randomly chooses $\nu_i \in Z_P$ and computes $\mu_i = \nu_i P$. After that, **E** returns μ_i to **A**.

— **A** is able to request V_i to H^* function at any time. Later on, **E** traverses a tuple (V_i, Y_i) from H^*_List. If the tuple exists, **E** will return Y_i to **A**. Otherwise, **E** randomly selects $Y_i \in \{0,1\}^\bullet$ and replies Y_i to the adversary **A**. Finally, **E** adds (V_i, Y_i) into H^*_List.

Stage2 (Challenge simulation)

A sends a keyword-vector pair $[(W^*_{0i}, \mu^*_{0i}, \nu^*_{0i}, \varepsilon^*_{0i}), (W^*_{1i}, \mu^*_{1i}, \nu^*_{1i}, \varepsilon^*_{1i})]$ to **E**, where $W^*_0 = (w_{01}, w_{02}, \ldots, w_{0\eta})$ and $W^*_1 = (w_{11}, w_{12}, \ldots, w_{1\eta})$. Once the challenger **E** achieves the pair, he/she will do the following steps:

— The challenger **E** randomly selects $Y_i \in \{0,1\}^\bullet$ and $\delta \in \{0,1\}$.

— The challenger **E** recalls the $SCF - MPEKS$ algorithm for generating the Searchable ciphertext $C^* = (X^*, Y^*_1, Y^*_2, \ldots, Y^*_\eta) = (\psi \alpha P, H^*[B_1], H^*[B_2], \ldots, H^*[B_\eta])$.

So,

$C^* = (X^*, Y^*_1, Y^*_2, \ldots, Y^*_\eta) = (\psi \alpha P, H^*(e(H(w_{\delta_1}), N)^\psi), H^*(e(H(w_{\delta_2}), N)^\psi), \ldots, H^*(e(H(w_{\delta_\eta}), N)^\psi))$

It is known that $B_i = e(H(w_{\delta^*_i}), N)^\psi) = e(\nu_i P, nP)^\psi = e(P, P)^{\psi \cdot \nu_i n}$.

Stage3 (Guess)

The adversary **A** exports $\delta^* \in \{0,1\}$ as the guess. Later on, **E** returns the guessed 1-BDHI key $\psi = \frac{1}{\alpha \cdot \nu_i n}$.

Analysis of *Game2*

Stage1-3 describes the procedure and operations of the challenger **E**. It remains to show that 1-BDHI assumption (Boneh and Boyen, 2004) is satisfied in ***Game2***. To do so, the first thing is to analyze that the challenger **E** does not stop during the simulation. Therefore, two events are formalized below:

Event4: The challenger **E** does not stop during **Stage2 (Challenge simulation)**.

Event5: The adversary **A** does not request either $H^*(e(H(w_{0i}^*), N)^{\psi})$ or $H^*(e(H(w_{1i}^*), N)^{\psi})$.

Claim 4: $Pr[Event4] = 1$

Proof: There is no limitation to illustrate that the system will be terminated by the challenger **E** during **Stage2**. Thus, it is clear that $Pr[Event4] = 1$.

Claim 5: $Pr[\neg Event5] \geq 2\xi$

Proof: If $Event5$ happens, it will show that the bit $j \in \{0,1\}$ pointing out whether the Searchable encryption contains w_{0i} or w_{1i} separates of **A**'s view. Hence, the probability that the adversary **A**'s exporting j^* which satisfies $j = j^*$ is at most $\frac{1}{2}$.

By the concept of Bayes's rule,

$Pr[j = j^*] = Pr[j = j^* | Event5]Pr[Event5] + Pr[j = j^* | Event5]Pr[\neg Event5]$

$\leq Pr[j = j^* | Event5]Pr[Even5] + Pr[\neg Event5] = \frac{1}{2} \bullet Pr[Event5] + Pr[\neg Event5] =$

$\frac{1}{2} + \frac{1}{2} \bullet Pr[\neg Event5]$**.**

By definition, it should be known that $|Pr[j = j^*] - \frac{1}{2}| \geq \xi$. And then, $\xi \leq Pr[j = j^*] - \frac{1}{2} \leq \frac{1}{2} \bullet Pr[\neg Event5]$. Thus, $Pr[\neg Event5] \geq 2\xi$.

Overall, due to **A** requests either $H^*(e(H(w^*_{0i}), N)^{\Psi})$ or $H^*(e(H(w^*_{1i}), N)^{\Psi})$ being at least 2ξ, the probability that the adversary **A**requesting $H^*(e(H(w^*_{ji}), N)^{\Psi})$ is at least ξ. However, according to the previous definition that **A** requests at most h^* hash function queries, $\frac{1}{h^*}$ is the probability that the challenger **E** chooses the correct solution**.** Overall**,** the success probability ξ^* achieved by **E** is $\frac{\xi}{h^*}$, which is negligible.

The proposed scheme above could be regarded as **Trapdoor-IND-CPA** secure in ***Game3*** under the random oracle model, if the BDH assumption (Boneh and Boyen, 2004) is completely difficult.

Game3: Let A suppose to be an untrusted outside attacker.

Consider that the challenger **E** is able to achieve the input $(g, P, G_1, G_T, e, \alpha P, \beta P, \gamma P)$ of BDH assumption (Boneh and Boyen, 2004). **E** sets up the computation of a BDH key $e(P, P)^{\alpha\beta\gamma}$ of αP, βP and γP using **A**'s **IND-CPA** as a goal. Apart from that, **A** queries at most h and h^* times hash function requests.

Stage1 (Setup)

E selects $M = \alpha P$, $K = \beta P$ and $N = \gamma P$ in the beginning. Then, the following parameters are returned by **E**, which are the common parameter $(g, P, G_1, G_T e, H, H^*)$, the server's public key (cp, M, K), and the receiver's public key (cp, N). Apart from that, two specific hash functions H and H^* are randomly selected by **E**.

Stage2 （Trapdoor queries)

If **A** queries a Trapdoor request with a specific keyword w_i, **E** will randomly pick up $t^* \in Z_P$ and subsequently compute $T_1 = \gamma H(w_i) \oplus e(\beta P, \alpha P)^{t^*+\gamma}$ and $T_2 = e(t^*\beta P, \alpha P)$. So, $T_w = (T_1, T_2)$. After that, **E** sends T_w back to the adversary **A**

Stage3 (Challenge simulation)

A uploads a keyword pair (w_0^*, w_1^*) to **E**. Once the challenger **E** receives the keyword pair, he/she will do the following steps:

— The challenger **E** randomly selects $\delta \in \{0,1\}$.

— The challenger **E** recalls the $Trapdoor$ algorithm for generating $T_1 = \gamma H(w_{\delta^*}) \oplus e(\beta P, \alpha P)^{t^*+\gamma} = \gamma H(w_{\delta^*}) \oplus e(P,P)^{\alpha\beta\gamma} \bullet e(P,P)^{\alpha\beta t^*}$ and $T_2 = e(t^*\beta P, \alpha P)$.

Stage4 (Trapdoor queries)

E can continue return any Trapdoor query T_{w_i} for any keyword w_i to **A** as in **Stage2 (*Game3*)**, only if $w_i \neq w_0, w_1$.

Stage5 (Guess)

A outputs $\delta^* \in \{0,1\}$ as the guess. If $\delta = \delta^*$, **E** outputs "yes" and "no" otherwise.

Analysis of *Game3*

According to **A** is an untrusted outside adversary, he/she is not able to observe any difference between two Trapdoor queries even if these queries contain the same keyword. This is because **E** randomly picks up $t^* \in Z_P$ and t^* changes in every calculation so that $T_1 = nH(w_i) \oplus e(M,K)^{t^*+n}$ changes in every calculation. Consider two Trapdoor queries contain the same keyword, but the calculation results are different mainly because of the value $t*$. Hence, the core portion of **Trapdoor-IND-CPA** secure in the proposed scheme is the confidentiality of $e(M, K)^{t*+n}$.

Consider that if **A** has $e(M, K)^{t*+n}$, he/she could estimate whether two Trapdoor queries have the same keyword or not. More specially, **A** computes one extra XOR as follows: $T_1 = nH(w_i) \oplus e(M, K)^{t*+n} \oplus e(M, K)^{t*+n} = nH(w_i)$. So, **A** is able to know that $T_{w_0} = nH(w_0)$ and $T_{w_1} = nH(w_1)$ are equal, only if $w_0 = w_1$.

By **Stage3** in ***Game3***, it shows that $e(M, K)^{t*+n} = e(P, P)^{\alpha\beta\gamma} \bullet e(P, P)^{\alpha\beta t*}$, which meets BDH assumption (Boneh and Boyen, 2004). Therefore, **A** is not able to computes $e(M, K)^{t*+n}$ so that he/she cannot calculate $T_1 = nH(w_i) \oplus e(M, K)^{t*+n}$ either.

4.7 The Efficiency and Performance of tSCF-MPEKS

This part describes the security comparison between the proposed approach (*tSCF-MPEKS*) and another two PEKS approaches [MPEKS (Baek et al., 2008) and SCF-MPEKS (Wang et al., 2016)]. Besides, the performance and efficiency of these three PEKS schemes are also presented in the following.

TABLE 3. A COMPARISON OF THE SECURITY ASSUMPTION AND PROPERTIES

Scheme	CT Ind	Trap Ind	SC	OKGA
MPEKS	Satisfied	Not satisfied	Required	Vulnerable to OKGA
SCF-MPEKS	Satisfied	Not satisfied	Not required	Vulnerable to OKGA
Proposed scheme	Satisfied	Satisfied	Not required	Not vulnerable to OKGA

CT Ind, Trap Ind, SC and OKGA are the abbreviation of Ciphertext Indistinguishability, Trapdoor Indistinguishability, Secure Channel and Off-line Keyword Guessing Attack respectively.

The proposed approach does not rely on the secure channel to transmit Trapdoor. In addition, it has the characters of CT Ind and Trap Ind so that it prevents OKGA. To conclude, comparing with MPEKS and SCF-MPEKS methods, the proposed approach has better efficiency and performance.

The proposed approach is programmed by applying type A pairing in JPBC Library (Angelo and Vincenzo, 2011) and the platform details are presented in Table 4.

TABLE 4. THE SIMULATION PLATFORM FOR tSCF-MPEKS

OS	macOS Sierra 10.12.5
CPU	2.5 GHz Intel Core i7
Memory	16 GB 1600 MHz DDR3
Hard disk	512GB
Programming language	JAVA

The performance and efficiency of proposed approach is also presented by the theoretical analysis and 1000 times computer simulations. So, Table 5 illustrates the comparison of computation efficiency between MPEKS, SCF-MPEKS and proposed schemes and the time cost of proposed approach is listed in Table 6.

TABLE 5. A COMPARISON OF THE COMPUTATION EFFICIENCY

Scheme	**PEKS**	**Trapdoor**	**Test**
MPEKS	$2E + 2H + P$	$E + H$	$H + P$
SCF-MPEKS	$2E + 2H + P$	$E + H$	$E + H + P$
Proposed Scheme	$2E + 2H + P$	$3E + H + 2P$	$2E + 2P + H$

According to Table 5, the symbols E, H and P are the abbreviation of a modular exponentiation, a collision resistant hash function and a bilinear pairing respectively. Due to MPEKS and SCF-MPEKS suffering OKGA, the proposed approach has better performance than its counterparts.

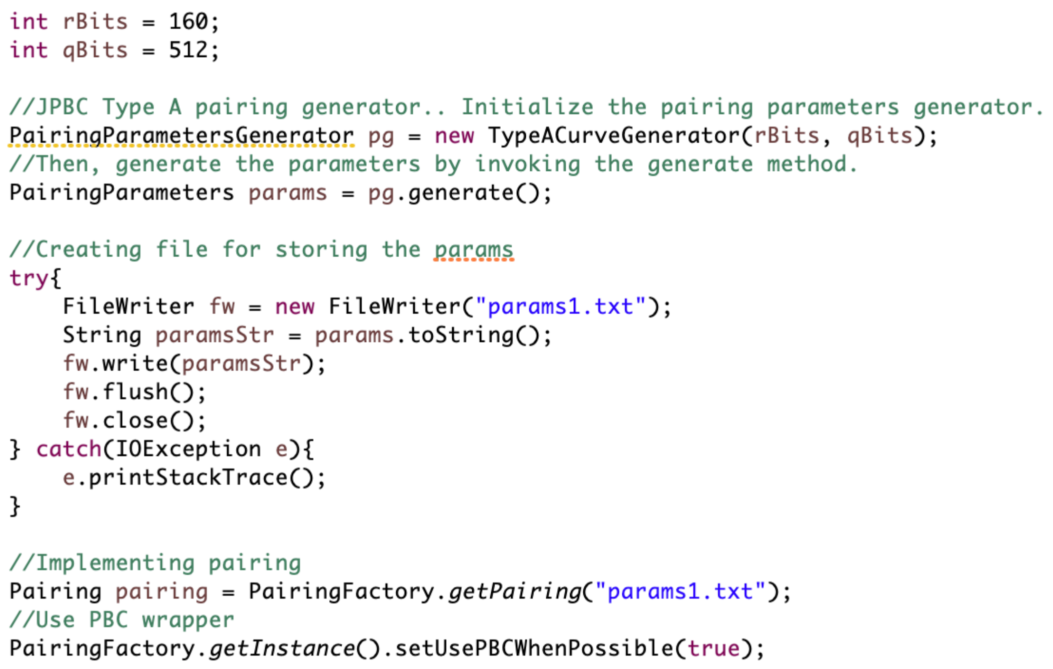
TABLE 6. PERFORMANCE BASED ON 1000 TIMES COMPUTER SIMULATION (n=3)

tSCF-MPEKS	KeyGen_Ser	KeyGen_Rec	SCF-MPEKS	Trapdoor	Test
Average time	0.017s	0.012s	0.088s	0.045s	0.019s

4.8 The Key Code of tSCF-MPEKS

This part shows the key codes of the proposed scheme from parameters initialisation, Server's and Receiver's key pairs generations, Searchable ciphertext (SCF-MPEKS) generation, Trapdoor request and Test algorithm.

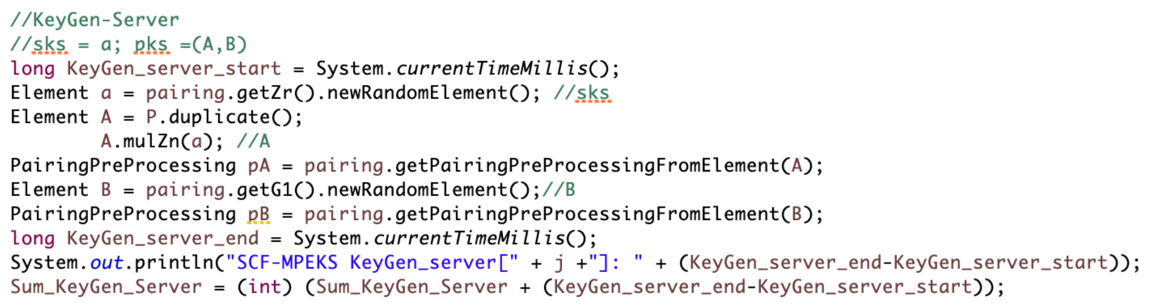
The proposed approach is programmed by applying type A pairing in JPBC Library (Angelo and Vincenzo, 2011). Besides, the pairing parameters initialization is described in Figure 14.



```
int rBits = 160;
int qBits = 512;

//JPBC Type A pairing generator.. Initialize the pairing parameters generator.
PairingParametersGenerator pg = new TypeACurveGenerator(rBits, qBits);
//Then, generate the parameters by invoking the generate method.
PairingParameters params = pg.generate();

//Creating file for storing the params
try{
    FileWriter fw = new FileWriter("params1.txt");
    String paramsStr = params.toString();
    fw.write(paramsStr);
    fw.flush();
    fw.close();
} catch(IOException e){
    e.printStackTrace();
}

//Implementing pairing
Pairing pairing = PairingFactory.getPairing("params1.txt");
//Use PBC wrapper
PairingFactory.getInstance().setUsePBCWhenPossible(true);
```

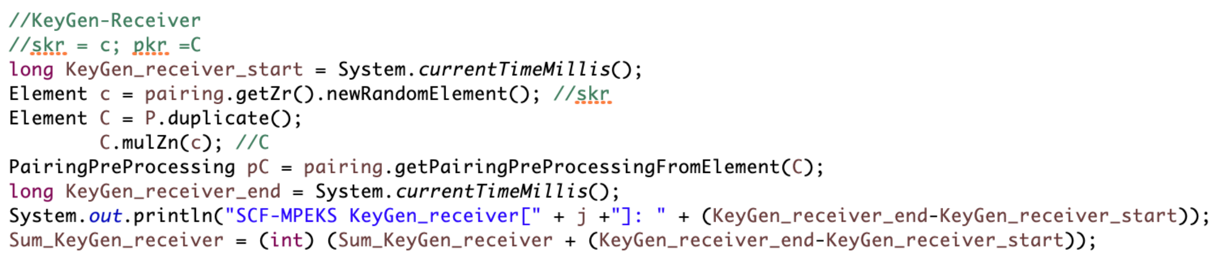
FIGURE 14. THE PARAMETERS INITIALIZATION IN TSCF-MPEKS

The server's key pair generation and receiver's key pair generation are described in Figure 15 and Figure 16.



```
//KeyGen-Server
//sks = a; pks =(A,B)
long KeyGen_server_start = System.currentTimeMillis();
Element a = pairing.getZr().newRandomElement(); //sks
Element A = P.duplicate();
        A.mulZn(a); //A
PairingPreProcessing pA = pairing.getPairingPreProcessingFromElement(A);
Element B = pairing.getG1().newRandomElement();//B
PairingPreProcessing pB = pairing.getPairingPreProcessingFromElement(B);
long KeyGen_server_end = System.currentTimeMillis();
System.out.println("SCF-MPEKS KeyGen_server[" + j +"]: " + (KeyGen_server_end-KeyGen_server_start));
Sum_KeyGen_Server = (int) (Sum_KeyGen_Server + (KeyGen_server_end-KeyGen_server_start));
```

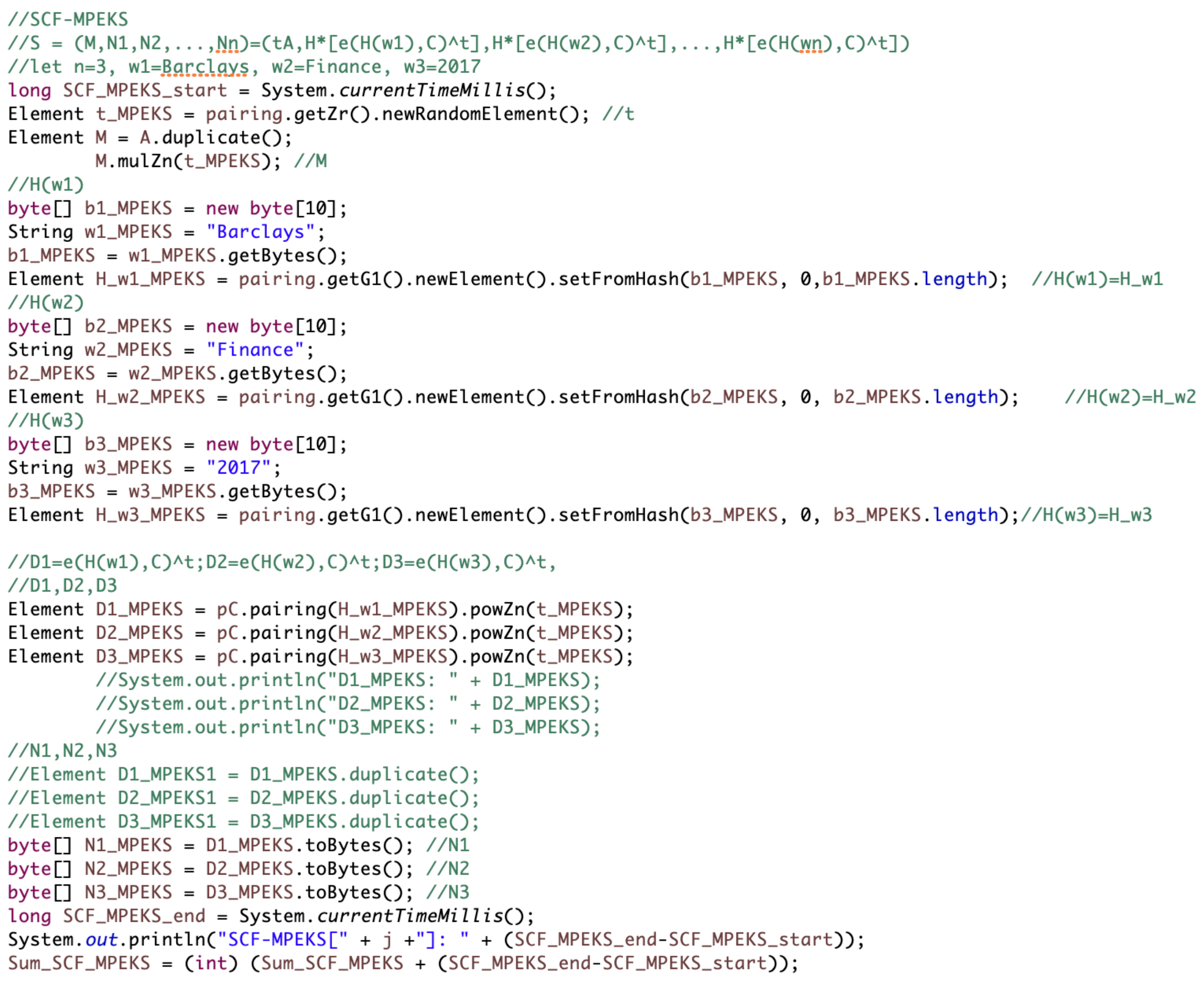
FIGURE 15. SERVER'S KEY PAIR GENERATION IN TSCF-MPEKS



```
//KeyGen-Receiver
//skr = c; pkr =C
long KeyGen_receiver_start = System.currentTimeMillis();
Element c = pairing.getZr().newRandomElement(); //skr
Element C = P.duplicate();
        C.mulZn(c); //C
PairingPreProcessing pC = pairing.getPairingPreProcessingFromElement(C);
long KeyGen_receiver_end = System.currentTimeMillis();
System.out.println("SCF-MPEKS KeyGen_receiver[" + j +"]: " + (KeyGen_receiver_end-KeyGen_receiver_start));
Sum_KeyGen_receiver = (int) (Sum_KeyGen_receiver + (KeyGen_receiver_end-KeyGen_receiver_start));
```

FIGURE 16. RECEIVER'S KEY PAIR GENERATION IN TSCF-MPEKS

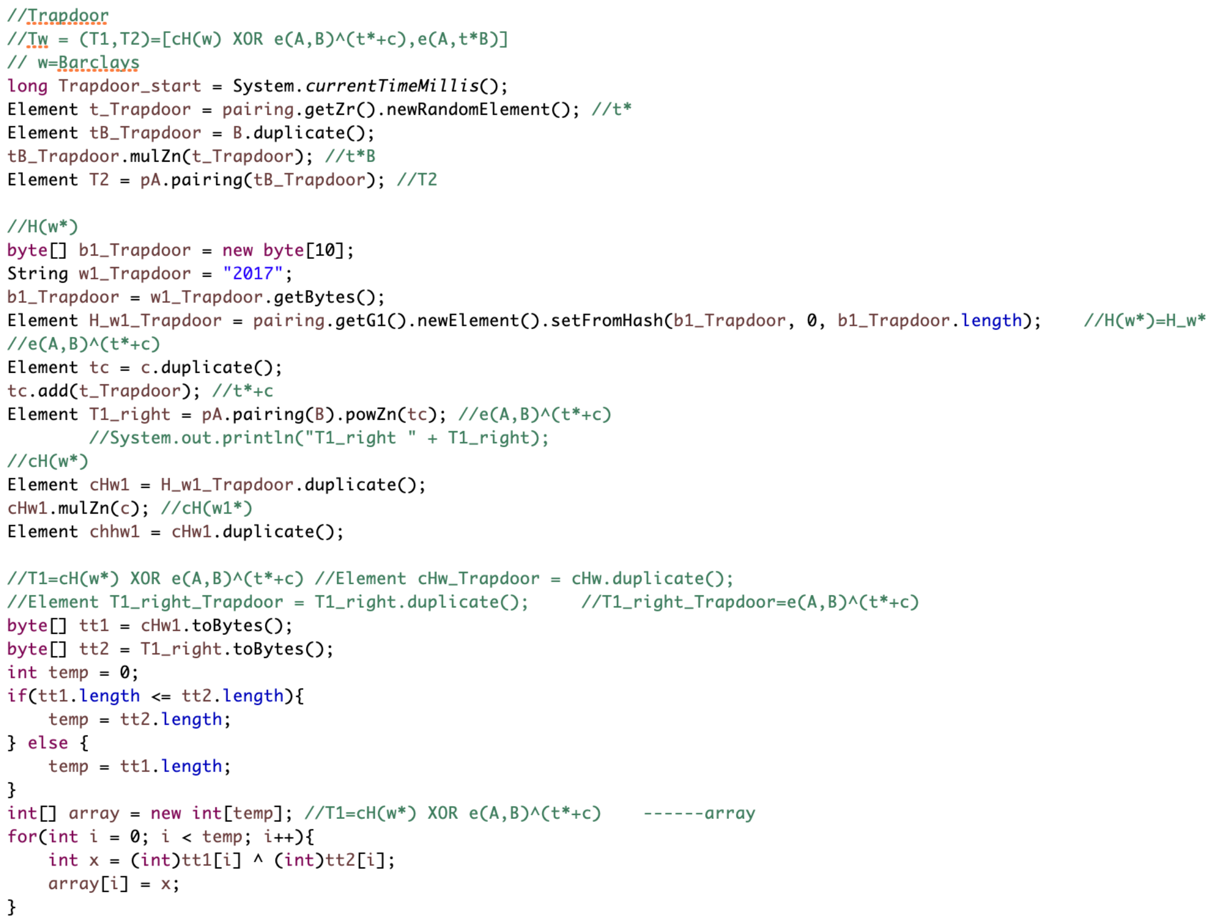
The following figure (Figure 17) describes the Searchable ciphertext (SCF-MPEKS) generation. In order to simplify the code, let the number of keywords be three.



```
//SCF-MPEKS
//S = (M,N1,N2,...,Nn)=(tA,H*[e(H(w1),C)^t],H*[e(H(w2),C)^t],...,H*[e(H(wn),C)^t])
//let n=3, w1=Barclays, w2=Finance, w3=2017
long SCF_MPEKS_start = System.currentTimeMillis();
Element t_MPEKS = pairing.getZr().newRandomElement(); //t
Element M = A.duplicate();
        M.mulZn(t_MPEKS); //M
//H(w1)
byte[] b1_MPEKS = new byte[10];
String w1_MPEKS = "Barclays";
b1_MPEKS = w1_MPEKS.getBytes();
Element H_w1_MPEKS = pairing.getG1().newElement().setFromHash(b1_MPEKS, 0,b1_MPEKS.length);  //H(w1)=H_w1
//H(w2)
byte[] b2_MPEKS = new byte[10];
String w2_MPEKS = "Finance";
b2_MPEKS = w2_MPEKS.getBytes();
Element H_w2_MPEKS = pairing.getG1().newElement().setFromHash(b2_MPEKS, 0, b2_MPEKS.length);    //H(w2)=H_w2
//H(w3)
byte[] b3_MPEKS = new byte[10];
String w3_MPEKS = "2017";
b3_MPEKS = w3_MPEKS.getBytes();
Element H_w3_MPEKS = pairing.getG1().newElement().setFromHash(b3_MPEKS, 0, b3_MPEKS.length);//H(w3)=H_w3

//D1=e(H(w1),C)^t;D2=e(H(w2),C)^t;D3=e(H(w3),C)^t,
//D1,D2,D3
Element D1_MPEKS = pC.pairing(H_w1_MPEKS).powZn(t_MPEKS);
Element D2_MPEKS = pC.pairing(H_w2_MPEKS).powZn(t_MPEKS);
Element D3_MPEKS = pC.pairing(H_w3_MPEKS).powZn(t_MPEKS);
        //System.out.println("D1_MPEKS: " + D1_MPEKS);
        //System.out.println("D2_MPEKS: " + D2_MPEKS);
        //System.out.println("D3_MPEKS: " + D3_MPEKS);
//N1,N2,N3
//Element D1_MPEKS1 = D1_MPEKS.duplicate();
//Element D2_MPEKS1 = D2_MPEKS.duplicate();
//Element D3_MPEKS1 = D3_MPEKS.duplicate();
byte[] N1_MPEKS = D1_MPEKS.toBytes(); //N1
byte[] N2_MPEKS = D2_MPEKS.toBytes(); //N2
byte[] N3_MPEKS = D3_MPEKS.toBytes(); //N3
long SCF_MPEKS_end = System.currentTimeMillis();
System.out.println("SCF-MPEKS[" + j +"]: " + (SCF_MPEKS_end-SCF_MPEKS_start));
Sum_SCF_MPEKS = (int) (Sum_SCF_MPEKS + (SCF_MPEKS_end-SCF_MPEKS_start));
```

FIGURE 17. SEARCHABLE CIPHERTEST GENERATION IN TSCF-MPEKS

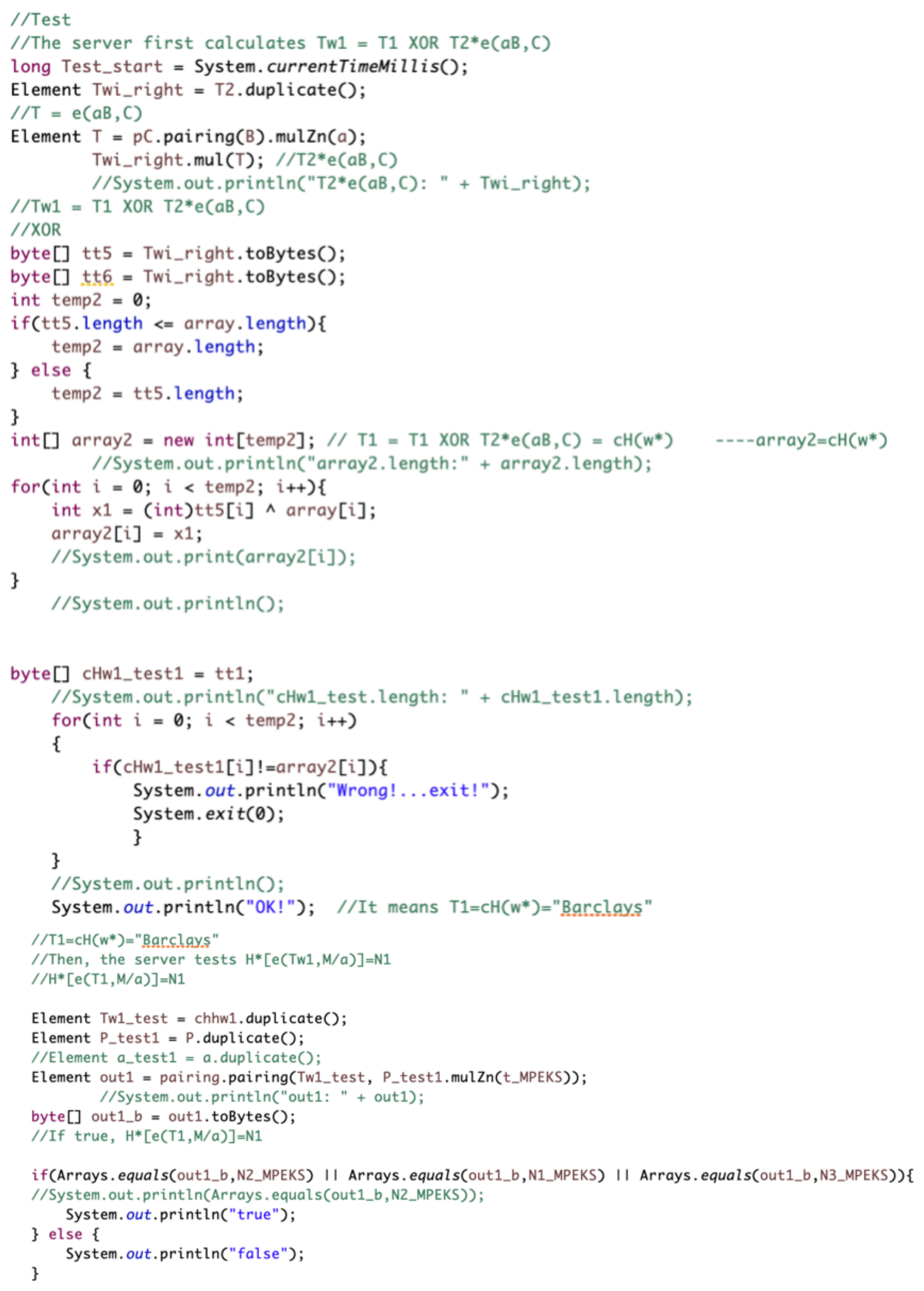
Figure 18 illustrates the Trapdoor request generation by using server's public key pk_{Ser}, receiver's private key sk_{Rec} and a keyword w*.



```
//Trapdoor
//Tw = (T1,T2)=[cH(w) XOR e(A,B)^(t*+c),e(A,t*B)]
// w=Barclays
long Trapdoor_start = System.currentTimeMillis();
Element t_Trapdoor = pairing.getZr().newRandomElement(); //t*
Element tB_Trapdoor = B.duplicate();
tB_Trapdoor.mulZn(t_Trapdoor); //t*B
Element T2 = pA.pairing(tB_Trapdoor); //T2

//H(w*)
byte[] b1_Trapdoor = new byte[10];
String w1_Trapdoor = "2017";
b1_Trapdoor = w1_Trapdoor.getBytes();
Element H_w1_Trapdoor = pairing.getG1().newElement().setFromHash(b1_Trapdoor, 0, b1_Trapdoor.length);    //H(w*)=H_w*
//e(A,B)^(t*+c)
Element tc = c.duplicate();
tc.add(t_Trapdoor); //t*+c
Element T1_right = pA.pairing(B).powZn(tc); //e(A,B)^(t*+c)
        //System.out.println("T1_right " + T1_right);
//cH(w*)
Element cHw1 = H_w1_Trapdoor.duplicate();
cHw1.mulZn(c); //cH(w1*)
Element chhw1 = cHw1.duplicate();

//T1=cH(w*) XOR e(A,B)^(t*+c) //Element cHw_Trapdoor = cHw.duplicate();
//Element T1_right_Trapdoor = T1_right.duplicate();     //T1_right_Trapdoor=e(A,B)^(t*+c)
byte[] tt1 = cHw1.toBytes();
byte[] tt2 = T1_right.toBytes();
int temp = 0;
if(tt1.length <= tt2.length){
    temp = tt2.length;
} else {
    temp = tt1.length;
}
int[] array = new int[temp]; //T1=cH(w*) XOR e(A,B)^(t*+c)    ------array
for(int i = 0; i < temp; i++){
    int x = (int)tt1[i] ^ (int)tt2[i];
    array[i] = x;
}
```

FIGURE 18. TRAPDOOR REQUEST GENERATION IN TSCF-MPEKS

The final figure (Figure 19) shows Test algorithm, which describes the keywords comparison between the Searchable ciphertext and the Trapdoor request.



```java
//Test
//The server first calculates Tw1 = T1 XOR T2*e(aB,C)
long Test_start = System.currentTimeMillis();
Element Twi_right = T2.duplicate();
//T = e(aB,C)
Element T = pC.pairing(B).mulZn(a);
        Twi_right.mul(T); //T2*e(aB,C)
        //System.out.println("T2*e(aB,C): " + Twi_right);
//Tw1 = T1 XOR T2*e(aB,C)
//XOR
byte[] tt5 = Twi_right.toBytes();
byte[] tt6 = Twi_right.toBytes();
int temp2 = 0;
if(tt5.length <= array.length){
    temp2 = array.length;
} else {
    temp2 = tt5.length;
}
int[] array2 = new int[temp2]; // T1 = T1 XOR T2*e(aB,C) = cH(w*)    ----array2=cH(w*)
        //System.out.println("array2.length:" + array2.length);
for(int i = 0; i < temp2; i++){
    int x1 = (int)tt5[i] ^ array[i];
    array2[i] = x1;
    //System.out.print(array2[i]);
}
    //System.out.println();


byte[] cHw1_test1 = tt1;
    //System.out.println("cHw1_test.length: " + cHw1_test1.length);
    for(int i = 0; i < temp2; i++)
    {
        if(cHw1_test1[i]!=array2[i]){
            System.out.println("Wrong!...exit!");
            System.exit(0);
            }
    }
    //System.out.println();
    System.out.println("OK!");  //It means T1=cH(w*)="Barclays"

  //T1=cH(w*)="Barclays"
  //Then, the server tests H*[e(Tw1,M/a)]=N1
  //H*[e(T1,M/a)]=N1

  Element Tw1_test = chhw1.duplicate();
  Element P_test1 = P.duplicate();
  //Element a_test1 = a.duplicate();
  Element out1 = pairing.pairing(Tw1_test, P_test1.mulZn(t_MPEKS));
          //System.out.println("out1: " + out1);
  byte[] out1_b = out1.toBytes();
  //If true, H*[e(T1,M/a)]=N1

  if(Arrays.equals(out1_b,N2_MPEKS) || Arrays.equals(out1_b,N1_MPEKS) || Arrays.equals(out1_b,N3_MPEKS)){
  //System.out.println(Arrays.equals(out1_b,N2_MPEKS));
      System.out.println("true");
  } else {
      System.out.println("false");
  }
```

FIGURE 19. TEST ALGORITHM IN TSCF-MPEKS

5. Robust Secure Channel Free Public Key Encryption with Multi-keywords Search

5.1 Introduction

According to the section 1.4, it is known that almost all current Public Key Encryption with Multiple Keywords Search (MPEKS) schemes suffers Inside Keyword Guessing Attack (IKGA). Therefore, this chapter gives a formal definition of MPEKS scheme, which incorporates with Trapdoor Indistinguishability and User Authentication technique so that it has the ability to resist both IKGA and OKGA. More specially, it firstly defines a new PEKS scheme namely "*Robust Secure Channel Free Public Key Encryption with Multi-keywords Search (rSCF-MPEKS)*" and the security verification models. In addition, a concrete construction of *rSCF-MPEKS* is proposed following by the correctness analysis, security verification and efficiency and performance analysis.

5.2 The Outline of rSCF-MPEKS

Huang and Li pointed out that all current PEKS schemes suffer IKGA in 2018 and then proposed a PEKS scheme, namely "Public Key Authenticated Encryption with Keyword Search (PAEKS)", to resist IKGA (Huang and Li, 2018). Although PAEKS is able to address IKGA, it aims for solving Single Keyword Search only instead of Multiple Keywords Search and therefore, it may not be applied to the general public network. However, this PhD thesis defines a new PEKS scheme (Figure 20) called "*Robust Secure Channel Free Public Key Encryption with Multi-keywords Search (rSCF-MPEKS)"* which not only deals with both IKGA and OKGA but also solves both Single and Multiple Keyword(s) Search issues. The proposed PEKS scheme contains six PPT algorithms as follows:

1. $KeyGen_{Param}(1^n)$: Import 1^n, a common parameter cp is then created.

2. $KeyGen_{Sender}(cp)$: Import cp, a public and a private keys (pk_{Sen}, sk_{Sen}) of the sender are then created.

3. $KeyGen_{Receiver}(cp)$: Import cp, a public and a private keys (pk_{Rec}, sk_{Rec}) of the receiver are then created.

4. $SCF-MPEKS(pk_{Rec}, sk_{Sen}, W)$: Import the receiver's public key pk_{Rec} and the sender's private key sk_{Sen}, a Searchable ciphertext $S = SCF-MPEKS(pk_{Rec}, sk_{Sen}, W)$ is then generated by the sender, where $W = (w_1, w_2, \ldots, w_\eta)$.

5. $Trapdoor(pk_{Sen}, sk_{Rec}, w)$: Import the sender's public key pk_{Sen} and the receiver's private key sk_{Rec}, a Trapdoor query $T_w = Trapdoor(pk_{Sen}, sk_{Rec}, w)$ is then created by the receiver.

6. $Test(pk_{Sen}, pk_{Rec}, S, T_w)$: Import the sender's public key pk_{Sen}, the receiver's public key pk_{Rec}, a Searchable ciphertext $S = SCF-MPEKS(pk_{Rec}, sk_{Sen}, W)$ and a Trapdoor query $T_w = Trapdoor(pk_{Sen}, sk_{Rec}, w*)$. The server then estimates whether the W includes w. If so, output "yes" and "no", otherwise.

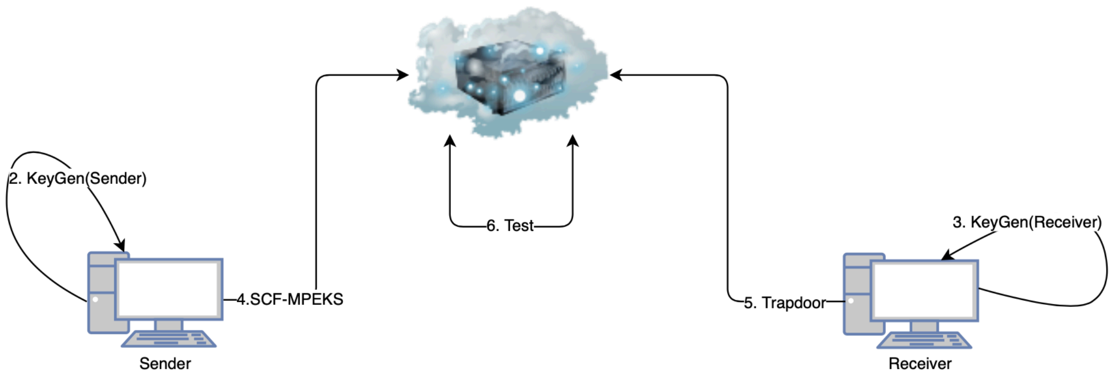


FIGURE 20. THE OUTLINE OF RSCF-MPEKS

5.3 The Security Models of rSCF-MPEKS

As discussed in tSCF-MPEKS (Ma and Kazemian, 2018) and PAEKS (Huang and Li, 2018), the proposed scheme is **Indistinguishability under Chosen Plaintext Attack** (**IND-CPA**) and **Trapdoor-IND-CPA**. The **IND-CPA** (***Game4***) and **Trapdoor-IND-CPA** (***Game5***) for *rSCF-MPEKS* are formalized below. Suppose **A** is an adversary and **E** is a challenger.

***Game4*: Ciphertext Indistinguishability**

Stage1 (Setup): $KeyGen_{Param}(1^n)$, $KeyGen_{Sender}(cp)$ and $KeyGen_{Receiver}(cp)$ are called by **E** in order to generate a common parameter cp, the key pairs (pk_{Sen}, sk_{Sen}) and (pk_{Rec}, sk_{Rec}) of the sender and the receiver. Then, **E** sends cp, pk_{Sen} and pk_{Rec} to **A** while keeps sk_{Sen} and sk_{Rec} from **A**.

Stage2 (Queries): Adaptably, **E** is able to return any Trapdoor query O_T and Ciphertext query O_C for any keyword w to **A**.

Stage3 (Challenge simulation): A target keyword-vector pair $[W_0 = (w_{01}, ..., w_{0\eta}),\ W_1 = (w_{11}, ..., w_{1\eta})]$ is sent from **A** to **E**. It is known that W_0 and W_1 cannot be requested in **Stage2 (*Game4*)**. Once the challenger **E** achieves the pair, the $SCF - MPEKS$ algorithm will be called by the challenger **E** for generating a Searchable ciphertext $S = SCF - MPEKS(sk_{Sen}, pk_{Rec}, W_\xi)$, where $\xi \in \{0,1\}$. Finally, S will be sent back to **A**.

Stage4 (Queries): E can continue return any Trapdoor query O_T and Ciphertext query O_C many times for any keyword w to **A** as in **Stage2 (*Game4*)**, only if $W \neq W_0, W_1$.

Stage5 (Guess): A guesses $\xi^* \in \{0,1\}$ and wins ***Game4***, if $\xi^* = \xi$.

A's advantage to win ***Game4*** is listed as follows:

$$Adv_{rSCF-MPEKS,A_4}^{IND-CPA}(k) = |Pr[\xi^* = \xi] - 1/2|.$$

Hence, the *rSCF-MPEKS* system is considered to be **IND-CPA** secure as long as the $Adv_{rSCF-MPEKS,A_4}^{IND-CPA}(k)$ is negligible.

Game5: **Trapdoor Indistinguishability**

Stage1 (Setup): $KeyGen_{Param}(1^n)$, $KeyGen_{Sender}(cp)$ and $KeyGen_{Receiver}(cp)$ are called by **E** in order to generate a common parameter cp, the key pairs (pk_{Sen}, sk_{Sen}) and (pk_{Rec}, sk_{Rec}) of the sender and the receiver. Then, **E** sends cp, pk_{Sen} and pk_{Rec} to **A** while keeps sk_{Sen} and sk_{Rec} from **A**.

Stage2 (Queries): Adaptably, **E** is able to return any Trapdoor query O_T and Ciphertext query O_C for any keyword w to **A**.

Stage3 (Challenge simulation): A target keyword pair (w_0, w_1) is sent from **A** to **E.** It is known that w_0 and w_1 cannot be requested in **Stage2 (*Game5*)**. Once **E** obtains the keyword pair, the $Trapdoor$ algorithm will be called by the challenger **E** in order to generate a Trapdoor query $T_w = Trapdoor(sk_{Rec}, pk_{Sen}, w_\xi)$, where $\xi \in \{0,1\}$. Finally, T_w will be sent back to **A**.

Stage4 (Queries): E can continue return any Trapdoor query O_T and Ciphertext query O_C many times for any keyword w to **A** as in **Stage2 (*Game5*)**, only if $w \neq w_0, w_1$.

Stage5 (Guess): A guesses $\xi^* \in \{0,1\}$ and wins ***Game5***, if $\xi^* = \xi$.

A's advantage to win ***Game5*** is listed as follows:

$$Adv^{IND-CPA}_{rSCF-MPEKS,A_5}(k) = |Pr[\xi^* = \xi] - 1/2|.$$

Therefore, the *rSCF-MPEKS* system is considered to be **Trapdoor-IND-CPA** secure as long as the $Adv^{IND-CPA}_{rSCF-MPEKS,A_5}(k)$ is negligible.

5.4 The Concrete Construction of rSCF-MPEKS

1. $KeyGen_{Param}(k)$: Suppose G_1 is an additive cyclic group and G_T is a multiplicative cyclic group. Let P be a random generator of G_1 and a prime number $g \geq 2^k$ be the order of G_1. A bilinear pairing is considered to be a map $e: G_1 \times G_1 \rightarrow G_T$. Suppose $H: \{0,1\}^\star \rightarrow G_1$ is a particular hash function. Therefore, a common parameter $cp = \{g, P, G_1, G_T, e, H\}$ can be achieved by the $KeyGen_{Param}(k)$ algorithm.

2. $KeyGen_{Sender}(cp)$: The sender selects $m \in Z_P$ uniformly at random and then calculates $M = mP$. So, the $pk_{Sen} = M$ and $sk_{Sen} = m$ are the server's public and private keys.

3. $KeyGen_{Receiver}(cp)$: The receiver selects $n \in Z_P$ uniformly at random and then calculates $N = nP$. So, $pk_{Rec} = N$ and $sk_{Rec} = n$ are the receiver's public and private keys.

4. $SCF-MPEKS(pk_{Rec}, sk_{Sen}, W)$: The sender selects $t \in Z_P$ uniformly at random and $W = (w_1, w_2, \ldots, w_\eta)$. Then, he/she calculates a Searchable ciphertext $C = (X, Y_1, Y_2, \ldots, Y_\eta) = [t \oplus pk_{Rec}, \quad e(sk_{Sen} \bullet \quad H(pk_{Sen}, pk_{Rec}, w_1), pk_{Rec} \bullet t),$ $e(sk_{Sen} \bullet H(pk_{Sen}, pk_{Rec}, w_2), pk_{Rec} \bullet t), \ldots, e(sk_{Sen} \bullet H(pk_{Sen}, pk_{Rec}, w_\eta), pk_{Rec} \bullet t)].$

5. $Trapdoor(pk_{Sen}, sk_{Rec}, w)$: The receiver computes $T_w = e(sk_{Rec} \bullet H(pk_{Sen}, pk_{Rec}, w), pk_{Sen})$.

6. $Test(pk_{Sen}, pk_{Rec}, C, T_w)$: For $i \in \{1,2,...,\eta\}$, the online server firstly computes $t = X \oplus pk_{Rec}$ and then checks whether $T_w^t = Y_i$ or not.

5.5 The Correctness of rSCF-MPEKS

Suppose W and $w*$ are keyword-vector and keyword in $SCF - MPEKS$ and $Trapdoor$ algorithms. The *rSCF-MPEKS* approach is considered to be corrected as long as W includes $w*$. The correctness verification is in the following:

Note that $\bullet$ stands for Multiplication and $\oplus$ stands for Exclusive Or.

The server initially computes $X \oplus pk_{Rec} = t \oplus pk_{Rec} \oplus pk_{Rec} = t$. And then, it will check whether $T_w^t = Y_i$ or not.

For $i \in \{1,2,...,\eta\}$,

$$\begin{aligned} T_w^t &= e(sk_{Rec} \bullet H(pk_{Sen}, pk_{Rec}, w*), pk_{Sen})^t \\ &= e(n \bullet H(pk_{Sen}, pk_{Rec}, w*), mP)^t \\ &= e(m \bullet H(pk_{Sen}, pk_{Rec}, w*), nP)^t \\ &= e(sk_{Sen} \bullet H(pk_{Sen}, pk_{Rec}, w*), pk_{Rec} \bullet t) \\ &= Y_i \end{aligned}$$

Therefore, the algorithm is completely correct.

5.6 The Security Analysis of rSCF-MPEKS

The *rSCF-MPEKS* approach possesses the characters of Ciphertext Indistinguishability and Trapdoor Indistinguishability against Chosen Plaintext Attack (CPA) whose security relies on the BDH assumption (Boneh and Boyen, 2004).

The proposed approach above is Ciphertext Indistinguishability in ***Game4*** under the random oracle model, if the BDH assumption (Boneh and Boyen, 2004) is completely difficult.

Game4: Ciphertext Indistinguishability of rSCF-MPEKS

Consider that the challenger **E** is able to achieve the input $(g, P, G_1, G_T, e, \alpha P, \beta P, \gamma P)$ of BDH assumption (Boneh and Boyen, 2004). **E** sets up the computation of a BDH key $e(P, P)^{\alpha\beta\gamma}$ of αP, βP and γP using **A**'s **IND-CPA** as a goal.

Stage1 (Setup)

E randomly selects $\alpha, \beta \in Z_P$ and subsequently returns αP and βP as the public keys (pk_{Sen}, pk_{Rec}) of the sender and the receiver. After that, **E** generates the common parameter $cp = (g, P, G_1, G_T, e, H)$ and transmits (cp, pk_{Sen}, pk_{Rec}) to **A**.

Stage2 (Queries):

For simplicity, three assumptions are proposed in the following:

1. **A** requests at most R_H, R_T, R_C to the Hash function query Q_H, the Trapdoor query Q_T and the Ciphertext query Q_C respectively.

2. **A** does not repeat any query.

3. **A** is not able to request a query (pk_{Sen}, w) to Q_T nor (pk_{Rec}, w) to Q_C before requesting (pk_{Sen}, pk_{Rec}, w) to Q_H.

The queries are simulated by **E** below.

For Hash function query Q_H.

When **A** issues a query for a tuple $(pk_{Sen}, pk_{Rec}, w_i)$. To respond,

i. **E** randomly selects a coin ε_i and then computes $Pr[\varepsilon_i = 0] = \frac{1}{h+1}$.

ii. **E** randomly chooses $\nu_i \in Z_P$. If $\varepsilon_i = 0$, $\mu_i = \beta P + \nu_i P$ will be computed by **E**. Similarly, $\mu_i = \nu_i P$ will be computed by **E** once $\varepsilon_i = 1$.

iii. The adversary **A** obtains μ_i from the challenger **E**. In addition, **E** uploads $[(pk_{Sen}, pk_{Rec}, w_i), \mu_i, \nu_i, \varepsilon_i]$ into H_List, which is initially empty.

For Trapdoor query Q_T:

If **A** queries a Trapdoor request with a specific keyword w, **E** will calculate $T_w = e(\nu_i \bullet pk_{Rec}, pk_{Sen}) = e(\beta\mu_i, pk_{Sen}) = e(sk_{Rec} \bullet H(pk_{Sen}, pk_{Rec}, w), pk_{Sen})$, where T_w is a correct Trapdoor under the sender's public key and the receiver's private key. After that, **E** returns T_w to **A**.

For Ciphertext query Q_C:

If **A** queries a Ciphertext request with a specific keyword w, **E** will randomly selects $t \in Z_P$ and calculates $Y_w = e(\nu_i \bullet pk_{Sen}, pk_{Rec} \bullet t) = e(\alpha\mu_i, pk_{Rec} \bullet t) = e(sk_{Sen} \bullet H(pk_{Sen}, pk_{Rec}, w), pk_{Rec} \bullet t)$, where Y_w is a correct Ciphertext under the sender's private key and the receiver's public key. After that, **E** returns Y_w to **A**.

Stage3 (Challenge simulation)

A sends a keyword-vector pair $[W_0 = (w_{01}, \ldots., w_{0\eta}), W_1 = (w_{11}, \ldots., w_{1\eta})]$ to **E**. According to **Stage2 (Queries)**, it should be known that $(\beta P, w_0^*)$ and $(\beta P, w_1^*)$ cannot be required by Q_T and similarly, $(\alpha P, W_0^*)$ and $(\alpha P, W_1^*)$ cannot be required by Q_C. Then, **E** returns a Searchable ciphertext C as follows:

- The challenger **E** chooses $i \in \{1,2,\ldots,\eta\}$ uniformly at random.

- The challenger **E** recalls the above algorithms in order to obtain two tuples $(w_{0i}^*, \mu_{0i}^*, \nu_{0i}^*, \varepsilon_{0i}^*)$ and $(w_{1i}^*, \mu_{1i}^*, \nu_{1i}^*, \varepsilon_{1i}^*)$. If $\varepsilon_{0i} = \varepsilon_{1i} = 1$, **E** will output "Suspension" and also terminate the system. Otherwise, the challenger **E** does the following step:

- **E** recalls the above algorithms in order to simulate H function at $2(\eta - 1)$ times for obtaining two tuples' vectors $\{(w_{01}^*, \mu_{01}^*, \nu_{01}^*, \varepsilon_{01}^*), \ldots, (w_{0i-1}^*, \mu_{0i-1}^*, \nu_{0i-1}^*, \varepsilon_{0i-1}^*), (w_{0i+1}^*, \mu_{0i+1}^*, \nu_{0i+1}^*, \varepsilon_{0i+1}^*), \ldots, (w_{0\eta}^*, \mu_{0\eta}^*, \nu_{0\eta}^*, \varepsilon_{0\eta}^*)\}$ and $\{(w_{11}^*, \mu_{11}^*, \nu_{11}^*, \varepsilon_{11}^*), \ldots, (w_{1i-1}^*, \mu_{1i-1}^*, \nu_{1i-1}^*, \varepsilon_{1i-1}^*),$

$(w_{1i+1}^*, \mu_{1i+1}^*, \nu_{1i+1}^*, \varepsilon_{1i+1}^*), \ldots, (w_{1\eta}^*, \mu_{1\eta}^*, \nu_{1\eta}^*, \varepsilon_{1\eta}^*)\}$. If $\varepsilon_{0j}^* = \varepsilon_{1j}^* = 0$ for all $j = 0,...,i-1,i+1,...,\eta$, **E** will export "Suspension" and consequently terminate the system. Otherwise, the challenger **E** executes the operations below:

— The challenger **E** randomly picks up $\delta \in \{0,1\}$.

— The challenger **E** randomly picks up $Y_i \in \{0,1\}^{\bullet}$ for generating a target $SCF-MPEKS$ encryption $C^* = (X^*, Y_1^*, Y_2^*, \ldots, Y_\eta^*)$. So, **E** selects $t = \gamma$. Then, **E** calculates $C^* = (X^*, Y_1^*, \ldots, Y_{i-1}^*, Y_{i+1}^*, \ldots, Y_\eta^*) = [\gamma \oplus \beta P$, $e(sk_{Sen} \bullet H(pk_{Sen}, pk_{Rec}, w_1^*), pk_{Rec} \bullet \gamma)$, ..., $e(sk_{Sen} \bullet H(pk_{Sen}, pk_{Rec}, w_{i-1}^*)$, $pk_{Rec} \bullet \gamma), e(sk_{Sen} \bullet H(pk_{Sen}, pk_{Rec}, w_{i+1}^*), pk_{Rec} \bullet \gamma), \ldots,$ $e(sk_{Sen} \bullet H(pk_{Sen}, pk_{Rec}, w_\eta^*), pk_{Rec} \bullet \gamma)]$.

Note that

$Y_\delta = e(sk_{Sen} \bullet H(pk_{Sen}, pk_{Rec}, w_\delta^*), pk_{Rec} \bullet \gamma) = e(\alpha H(pk_{Sen}, pk_{Rec}, w_\delta^*),$ $\beta P \bullet \gamma) = e(\alpha \nu_i P, \beta P \bullet \gamma) = e(\nu_i P, P)^{\alpha\beta\gamma}$.

Stage4 (Queries):

E can continue return any queries for any keyword w to **A** as in **Stage2 (*Game4*)**, only if **A** is not able to request $[(pk_{Sen}, w_0^*), (pk_{Sen}, w_1^*)]$ to Q_T and $[(pk_{Rec}, W_0^*), (pk_{Rec}, W_1^*)]$ to Q_C.

Stage5 (Guess)

A outputs $\delta^* \in \{0,1\}$ as the guess. If $\delta^* = \delta$, **E** outputs "yes" and "no" otherwise.

Analysis of *Game4*:

Stage1-5 describes the procedure and operations of the challenger **E**. It remains to show that BDH assumption (Boneh and Boyen, 2004) is satisfied in ***Game4***. To do so, the first thing is to analyze that the challenger **E** does not stop during the simulation. Therefore, two events are formalized below:

Event6: The challenger **E** does not stop during **Stage2** (**Queries)** and **Stage4 (Queries)**.

Event7: The challenger **E** does not stop during **Stage3 (Challenge simulation)**.

Claim 6: $Pr[Event6] \geq (1 - \frac{1}{h+1})^{R_T + R_C}$

Proof: Consider that **A** is not able to request the same keyword twice in Q_T and Q_C. So, $\frac{1}{h+1}$ is the probability causing **E** for suspension. From the previous definition, **A** requests at most R_T Trapdoor queries and R_C Ciphertext queries so that the system which does not be terminated by **E** in all queries is at least $(1 - \frac{1}{h+1})^{R_T + R_C}$.

Claim 7: $Pr[Event7] \geq (\frac{1}{h+1}) \bullet (\frac{h}{h+1})^{2(\eta - 1)}$

Proof: If $\varepsilon_0 = \varepsilon_1 = 1$, the system will be terminated by **E** during **Stage3 (Challenge simulation)**. So, the $1 - (1 - \frac{1}{h+1})^2$ is the probability that **E** does not suspend. Apart from that, if $\varepsilon^*_{0j} = \varepsilon^*_{1j} = 0$ for all $j = 0,...,i-1,i+1,...,\eta$, the system will be terminated by **E** again. Overall, the probability that the challenger **E** who does not terminate the system during **Stage3** is at least $(1 - \frac{1}{h+1})^{2(\eta-1)}\{1 - (1 - \frac{1}{h+1})^2\}$ $\geq (\frac{1}{h+1}) \bullet (\frac{h}{h+1})^{2(\eta-1)}$.

Let $Event$ be an event that **E** does not terminate in the whole game. Therefore, it is known that $Pr[Event] = Pr[Event6] \bullet Pr[Event7] =$

$(1-\frac{1}{h+1})^{R_T+R_C} \bullet (\frac{1}{h+1}) \bullet (\frac{h}{h+1})^{2(\eta-1)}$. $Pr[Event]$ will obtain the maximum value, if $h+1 = R_T + R_C$.

So, $Pr[Event] = \frac{1}{e} \bullet (\frac{1}{R_T+R_C}) \bullet (\frac{R_T+R_C-1}{R_T+R_C})^{2(\eta-1)}$, which is around equal to $\frac{1}{e(R_T+R_C)}$ and therefore non-negligible.

Overall, the probability that the bit δ correctly guessing by **A** is listed below:

$Pr[\delta^{'} = \delta] = Pr[\delta^{'} = \delta \wedge Pr[Event]] + Pr[\delta^{'} = \delta \wedge Pr[\overline{Event}]] = Pr[\delta^{'} = \delta \mid Pr[Event]]Pr[Event] + Pr[\delta^{'} = \delta \mid Pr[\overline{Event}]]Pr[\overline{Event}] = \frac{1}{2} \bullet (1 - Pr[\overline{Event}]) + (\epsilon_C + \frac{1}{2}) \bullet Pr[\overline{Event}] = \frac{1}{2} + \epsilon_C \bullet Pr[\overline{Event}]$.

If ϵ_C is non-negligible, so is $|Pr[\delta^{'} = \delta] - \frac{1}{2}|$.

Therefore, rSCF-MPEKS scheme based on BDH assumption (Boneh and Boyen, 2004) satisfies Ciphertext Indistinguishability.

The proposed scheme above is Trapdoor Indistinguishability in ***Game5*** under the random oracle model, if the BDH assumption (Boneh and Boyen, 2004) is completely difficult.

Game5: Trapdoor Indistinguishability of rSCF-MPEKS

Consider that the challenger **E** is able to achieve the input $(g, P, G_1, G_T, e, \alpha P, \beta P, \gamma P)$ of BDH assumption (Boneh and Boyen, 2004). **E** sets up the computation of a BDH key $e(P,P)^{\alpha\beta\gamma}$ of $\alpha P, \beta P$ and γP using **A**'s **Trapdoor-IND-CPA** as a goal.

Stage1 (Setup)

E randomly selects $\alpha, \beta \in Z_P$ and subsequently returns αP and βP as the public keys (pk_{Sen}, pk_{Rec}) of the sender and the receiver. After that, **E** generates the common parameter $cp = (g, P, G_1, G_T, e, H)$ and transmits (cp, pk_{Sen}, pk_{Rec}) to **A**.

Stage2 (Queries)

1. **A** requests at most R_H, R_T, R_C to the Hash function query Q_H, the Trapdoor query Q_T and the Ciphertext query Q_C respectively.

2. **A** does not repeat any query.

3. **A** is not able to request a query (pk_{Sen}, w) to Q_T nor (pk_{Rec}, w) to Q_C before requesting (pk_{Sen}, pk_{Rec}, w) to Q_H.

The queries are simulated by **E** in the following.

For Trapdoor query Q_T and Ciphertext query Q_C, **E**'s responses are the same as in the proof of Ciphertext Indistinguishability of *rSCF-MPEKS* scheme.

For Hash function query Q_H.

When **A** issues a query for a tuple $(pk_{Sen}, pk_{Rec}, w_i)$. To respond,

i. **E** randomly selects a coin ε_i and then computes $Pr[\varepsilon_i = 0] = \frac{1}{h+1}$.

ii. **E** randomly chooses $\nu_i \in Z_P$. If $\varepsilon_i = 0$, $\mu_i = \gamma P + \nu_i P$ will be computed by **E**. Similarly, $\mu_i = \nu_i P$ will be computed by **E** once $\varepsilon_i = 1$.

iii. **A** achieves μ_i from **E**. In addition, **E** uploads $[(pk_{Sen}, pk_{Rec}, w_i), \mu_i, \nu_i, \varepsilon_i]$ into H_List, which is initially empty.

Stage3 (Challenge simulation)

A uploads the keyword pair (w_0^*, w_1^*) to **E**. According to **Stage2 (Queries)**, it should be known that $(\beta P, w_0^*)$ and $(\beta P, w_1^*)$ cannot be required by Q_T and similarly,

$(\alpha P, W_0^*)$ and $(\alpha P, W_1^*)$ cannot be required by Q_C. Then, **E** returns a challenge Trapdoor T_w as follows:

- If $\varepsilon_0 = \varepsilon_1 = 1$, **E** will output "Suspension" and also terminate the system.

Otherwise, **E** calculates the Trapdoor in the following:

- $T_\delta = Z \bullet e(\alpha P, \beta P)^{\nu_i}$. Let Z be an element in G_T. Therefore, $T_\delta = e(P,P)^{\alpha\beta(\gamma+\nu_i)} = e(\mu_i, (\alpha\beta)P)$ once $Z = e(P,P)^{\alpha\beta\gamma}$.

Stage4 (Queries)

E can continue return any queries for any keyword w to **A** as in **Stage2 (*Game5*)**, only if **A** is not able to request $[(pk_{Sen}, w_0^*), (pk_{Sen}, w_1^*)]$ to Q_T and $[(pk_{Rec}, W_0^*), (pk_{Rec}, W_1^*)]$ to Q_C.

Stage5 (Guess)

A outputs $\delta^* \in \{0,1\}$ as the guess. If $\delta^* = \delta$, **E** outputs "yes" and "no" otherwise.

Analysis of *Game5*

Stage1-5 describes the procedure and operations of the challenger **E**. It remains to show that BDH assumption (Boneh and Boyen, 2004) is satisfied in ***Game5***. To do so, the first thing is to analyze that the challenger **E** does not stop during the simulation. Therefore, two events are formalized below:

Event8: The challenger **E** does not stop during **Stage2** (**Queries)** and **Stage4 (Queries)**.

Event9: The challenger **E** does not stop during **Stage3 (Challenge simulation)**.

Claim 8: $Pr[Event8] \geq (1 - \frac{1}{h+1})^{R_T+R_C}$

The proof of **Claim 8** is same as the proof of **Claim 6**, so it is omitted here.

Claim 9: $Pr[Event9] \geq 1 - (1 - \frac{1}{h+1})^2$

Proof: If $\varepsilon_0 = \varepsilon_1 = 1$, the system will be terminated by **E** during **Stage3 (Challenge simulation)**. So, the probability that the system does not be terminated by **E** in **Stage3** is $1 - (1 - \frac{1}{h+1})^2$.

Let $Event'$ be an event that **E** does not terminate in the whole game. Therefore, it is known that

$Pr[Event'] = Pr[Event8] \bullet Pr[Event9] = (1 - \frac{1}{h+1})^{R_T+R_C} \bullet (1 - (1 - \frac{1}{h+1})^2)$

$Pr[Event']$ will reach the maximum value if $\frac{1}{h+1} = 1 - \sqrt{\frac{R_T + R_C}{R_T + R_C + 2}}$.

So, $Pr[Event'] = (\frac{R_T + R_C}{R_T + R_C + 2})^{(R_T+R_C)/2} \bullet \frac{2}{R_T + R_C + 2}$,

which is around equal to $\frac{2}{(R_T + R_C)e}$ and therefore non-negligible.

Overall, the probability that the bit δ correctly guessing by **A** is listed below:

$Pr[\delta^{'} = \delta] = Pr[\delta^{'} = \delta \wedge Pr[Event']] + Pr[\delta^{'} = \delta \wedge Pr[\overline{Event'}]] = Pr[\delta^{'} = \delta \mid Pr[Event']]Pr[Event'] + Pr[\delta^{'} = \delta \mid Pr[\overline{Event'}]]Pr[\overline{Event'}] = \frac{1}{2} \bullet (1 - Pr[\overline{Event'}]) + (\epsilon_T + \frac{1}{2}) \bullet Pr[\overline{Event'}] = \frac{1}{2} + \epsilon_T \bullet Pr[\overline{Event'}]$.

If ϵ_T is non-negligible, so is $|Pr[\delta^{'} = \delta] - \frac{1}{2}|$.

Therefore, rSCF-MPEKS scheme based on BDH assumption (Boneh and Boyen, 2004) satisfies Trapdoor Indistinguishability.

5.7 The Efficiency and Performance of rSCF-MPEKS

This part describes the security comparison between the proposed PEKS scheme (*rSCF-MPEKS*) and the other several approaches [PEKS (Boneh et al., 2004); SCF-PEKS (Beak et al., 2008); dPEKS (Rhee et al., 2010); PAEKS (Huang and Li, 2018); MPEKS (Beak et al., 2008); SCF-MPEKS (Wang et al., 2016); tSCF-MPEKS (Ma and Hassan, 2018)]. Besides, the performance and efficiency of these PEKS approaches are presented in the following part. Table 7 below shows the functionalities in different PEKS mechanisms.

TABLE 7. A COMPARISON OF THE FUNCTIONALITIES

Scheme	**CT Ind**	**Trap Ind**	**MS**	**IKGA**
PEKS	Satisfied	Not satisfied	No	Suffered
SCF-PEKS	Satisfied	Not satisfied	No	Suffered
dPEKS	Satisfied	Satisfied	No	Suffered
PAEKS	Satisfied	Satisfied	No	Not suffered
MPEKS	Satisfied	Not satisfied	Yes	Suffered
SCF-MPEKS	Satisfied	Not satisfied	Yes	Suffered
tSCF-MPEKS	Satisfied	Satisfied	Yes	Suffered
Proposed Scheme	Satisfied	Satisfied	Yes	Not suffered

CT Ind, Trap Ind, MS and IKGA are the abbreviation of Ciphertext Indistinguishability, Trapdoor Indistinguishability, Multi-keywords Search and Inside Keyword Guessing Attack respectively. As seen from Table 7, the proposed scheme is much secure compared with the others. More specially, all of them except the proposed scheme and PAEKS are vulnerable to IKGA. Although PAEKS scheme prevents IKGA, it only aims for solving Single Keyword Search problem instead of supporting Multiple Keywords Search so that it may not be applied to the general public. To conclude, the proposed scheme is more secure and has better performance than its counterparts.

Table 8 below provides a comparison of computation efficiency between the proposed approach (*rSCF-MPEKS*) and the others.

TABLE 8. A COMPARISON OF THE COMPUTATION EFFICIENCY

Scheme	PEKS	Trapdoor	Test
PEKS	$2E+2H+P$	$E+H$	$H+P$
SCF-PEKS	$3E+2H+2P$	$E+H$	$E+H+P$
dPEKS	$2E+2H+P$	$3E+2H$	$2E+2H+P$
PAEKS	$3E+H$	$E+H+P$	$2P$
MPEKS	$2E+2H+P$	$E+H$	$H+P$
SCF-MPEKS	$2E+2H+P$	$E+H$	$E+H+P$
tSCF-MPEKS	$2E+2H+P$	$3E+H+2P$	$2E+2P+H$
Proposed Scheme	$2E+H+P$	$E+H+P$	E

According to Table 8, the symbols E, H and P are the abbreviation of a modular exponentiation, a collision resistant hash function and a bilinear pairing respectively. The PAEKS and proposed schemes which resist IKGA have the similar efficiency in PEKS algorithm. But the proposed scheme has better computation efficiency in Test algorithm than PAEKS scheme mainly because it only executes one XOR operation and one modular exponentiation in Test stage.

Table 9 shows the communication efficiency between the proposed scheme and its counterparts.

TABLE 9. A COMPARISON OF THE COMMUNICATION EFFICIENCY

Scheme	$\|PK\|$	$\|C\|$	$\|T_w\|$
PEKS	$\|G_1\|$	$\|G_1\|+n$	$\|G_1\|$
SCF-PEKS	$2\|G_1\|$	$\|G_1\|+n$	$\|G_1\|$
dPEKS	$2\|G_1\|$	$\|G_1\|+n$	$2\|G_1\|$
PAEKS	$\|G_1\|$	$2\|G_1\|$	$\|G_T\|$
MPEKS	$\|G_1\|$	$\|G_1\|+n$	$\|G_1\|$
SCF-MPEKS	$\|G_1\|$	$\|G_1\|+n$	$\|G_1\|$
tSCF-MPEKS	$2\|G_1\|$	$\|G_1\|+n$	$\|G_1\|+2\|G_T\|$
Proposed Scheme	$\|G_1\|$	$\|G_T\|+\|Z_P\|$	$\|G_T\|$

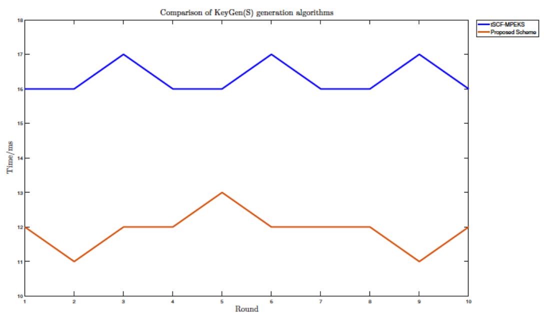
According to Table 9, the symbols of $|G_1|$, $|G_T|$ and $|Z_P|$ devote the length of element in group G_1, G_T and Z_P. Besides, n denotes the length of security parameters. It is clear that the proposed approach has better communication efficiency than some of its counterparts. For instance, comparing with dPEKS and *tSCF-MPEKS* schemes, the proposed scheme is efficient in $|PK|$ and $|T_w|$.

Table 10 below illustrates the simulation platform of *rSCF-MPEKS* scheme. Note that the proposed scheme is programmed by JAVA and JPBC Library (Angelo and Vincenzo, 2011).

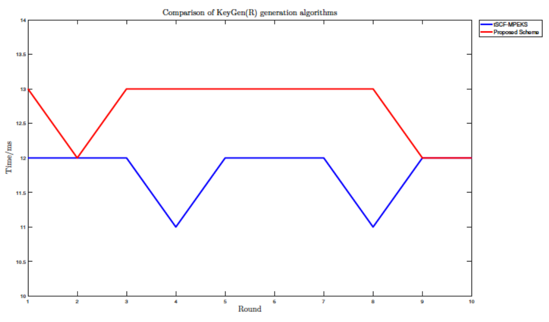
TABLE 10. THE SIMULATION PLATFORM FOR rSCF-MPEKS

OS	macOS Sierra 10.12.5
CPU	2.5 GHz Intel Core i7
Memory	16 GB 1600 MHz DDR3
Hard disk	512GB
Programming language	JAVA

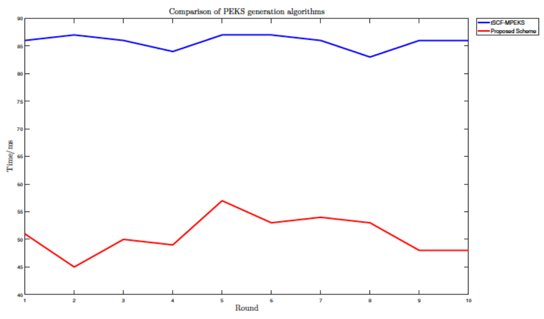
Figure 21 below compares $KeyGen_S$, $KeyGen_R$, $PEKS$, $Trapdoor$ and $Test$ generation algorithms between the *tSCF-MPEKS* and the proposed approaches by 1000 times computer simulation. Every 100 times computer simulation is called one round. In $KeyGen_S$ generation algorithm, the proposed approach is slightly efficient than the *tSCF-MPEKS* system. In $KeyGen_R$ generation algorithm, these two schemes are similar. However, the proposed scheme witnesses a high efficiency in $PEKS$, $Trapdoor$ and $Test$ generation algorithms comparing with the *tSCF-MPEKS* scheme.



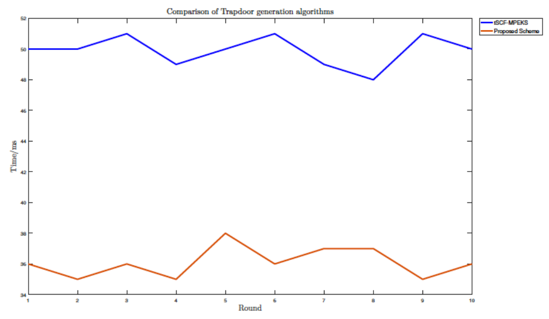
(a) Comparison of KeyGen(S) generation algorithms



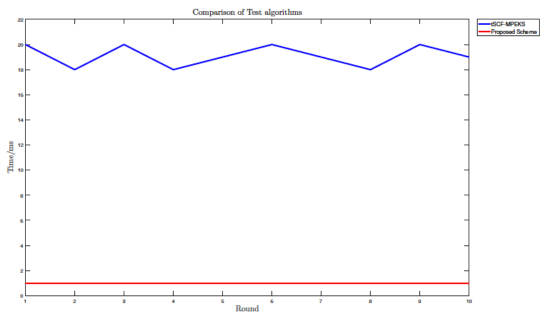
(b) Comparison of KeyGen(R) generation algorithms



(c) Comparison of PEKS generation algorithms



(d) Comparison of Trapdoor generation algorithms



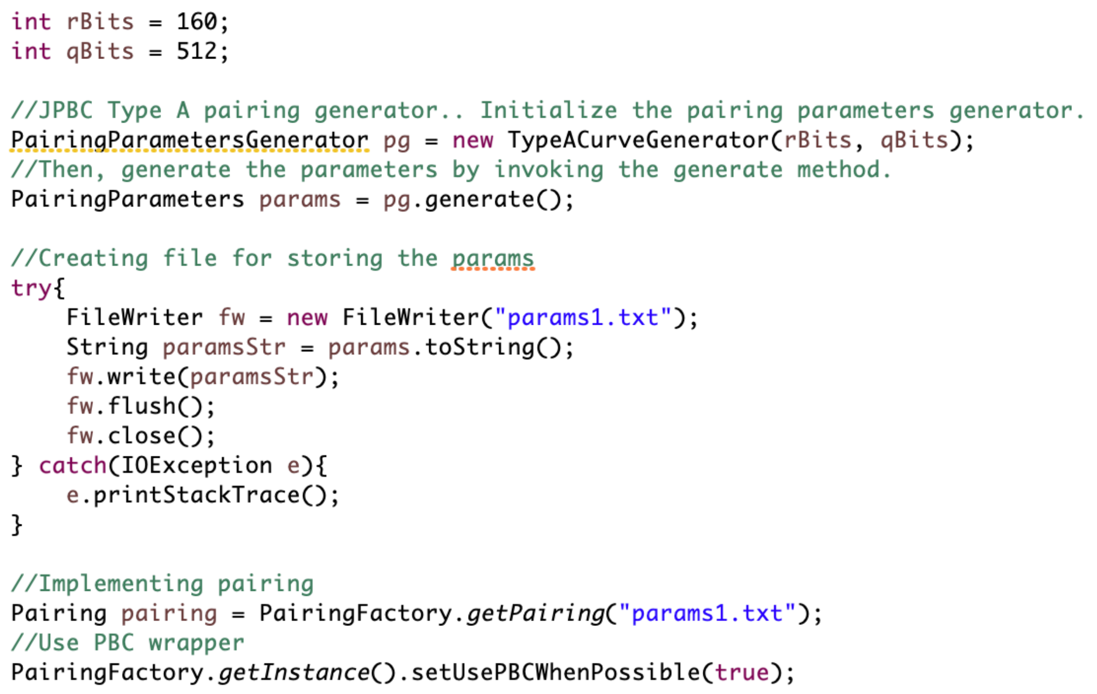
(e) Comparison of Test generation algorithms

FIGURE 21. A COMPARISON BETWEEN TSCF-MPEKS AND RSCF-MPEKS

5.8 The Key Code of rSCF-MPEKS

This part shows the key codes of the proposed scheme from parameters initialisation, Sender's and Receiver's key pairs generations, Searchable ciphertext (PEKS) generation, Trapdoor request and Test algorithm.

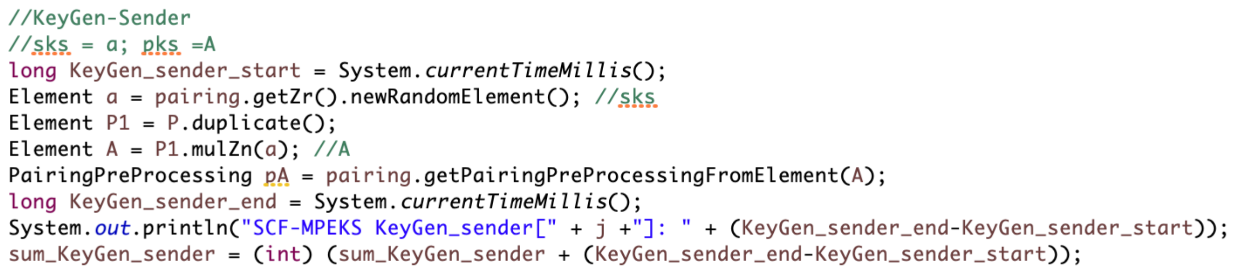
The proposed approach is programmed by JAVA using type A pairing in JPBC Library (Angelo and Vincenzo, 2011) and the pairing parameters initialization is described in Figure 22.



```java
int rBits = 160;
int qBits = 512;

//JPBC Type A pairing generator.. Initialize the pairing parameters generator.
PairingParametersGenerator pg = new TypeACurveGenerator(rBits, qBits);
//Then, generate the parameters by invoking the generate method.
PairingParameters params = pg.generate();

//Creating file for storing the params
try{
    FileWriter fw = new FileWriter("params1.txt");
    String paramsStr = params.toString();
    fw.write(paramsStr);
    fw.flush();
    fw.close();
} catch(IOException e){
    e.printStackTrace();
}

//Implementing pairing
Pairing pairing = PairingFactory.getPairing("params1.txt");
//Use PBC wrapper
PairingFactory.getInstance().setUsePBCWhenPossible(true);
```

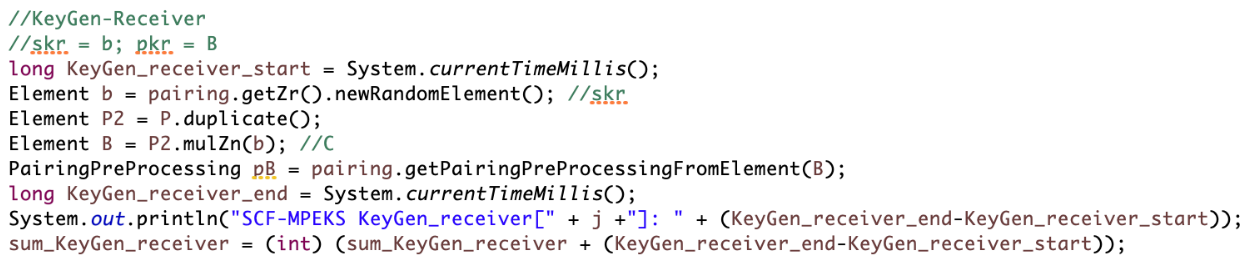
FIGURE 22. THE PARAMETERS INITIALIZATION IN RSCF-MPEKS

The sender's key pair generation and receiver's key pair generation are described in Figure 23 and Figure 24.



```java
//KeyGen-Sender
//sks = a; pks =A
long KeyGen_sender_start = System.currentTimeMillis();
Element a = pairing.getZr().newRandomElement(); //sks
Element P1 = P.duplicate();
Element A = P1.mulZn(a); //A
PairingPreProcessing pA = pairing.getPairingPreProcessingFromElement(A);
long KeyGen_sender_end = System.currentTimeMillis();
System.out.println("SCF-MPEKS KeyGen_sender[" + j +"]: " + (KeyGen_sender_end-KeyGen_sender_start));
sum_KeyGen_sender = (int) (sum_KeyGen_sender + (KeyGen_sender_end-KeyGen_sender_start));
```

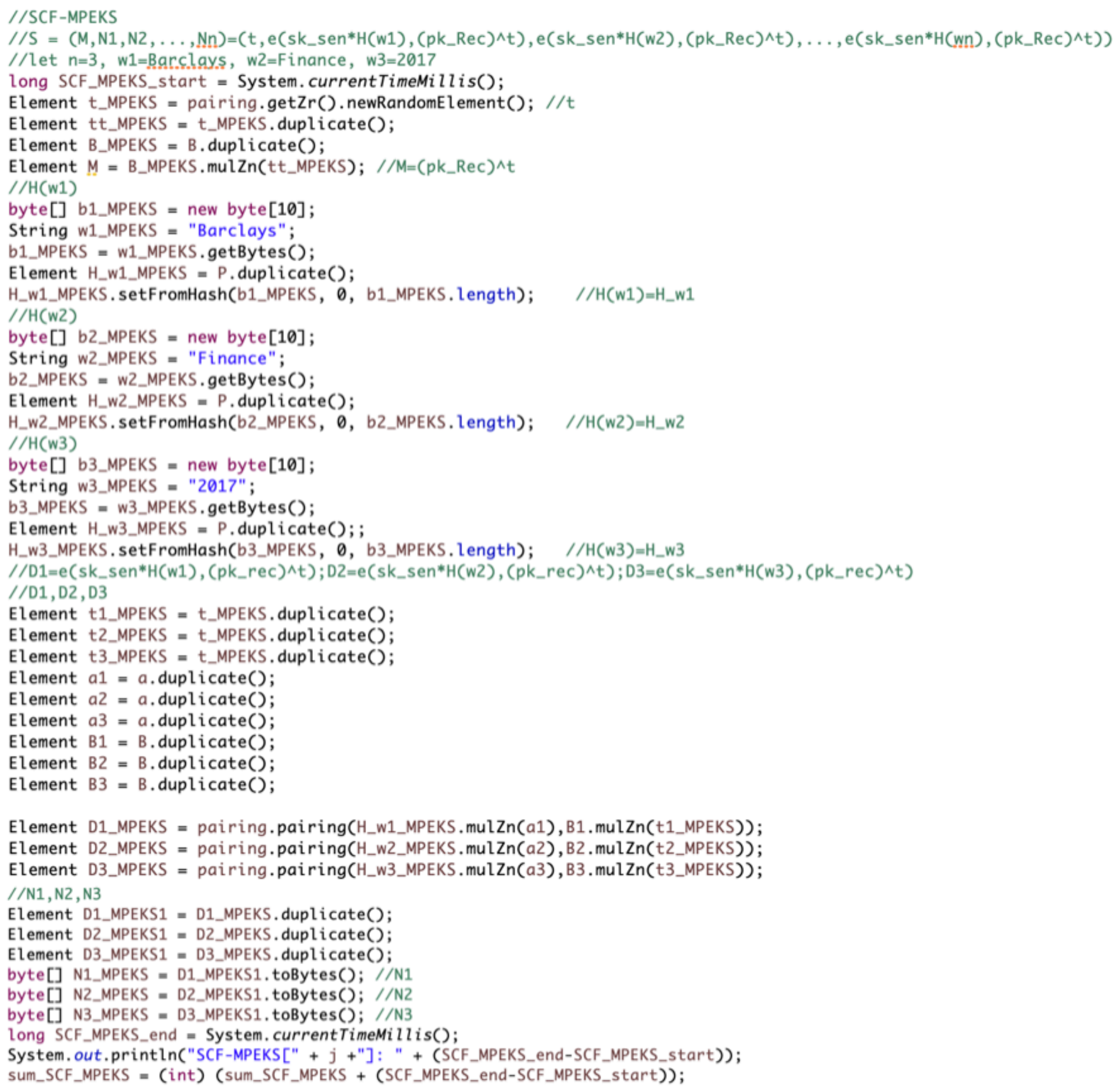
FIGURE 23. SENDER'S KEY PAIR GENERATION IN RSCF-MPEKS



```java
//KeyGen-Receiver
//skr = b; pkr = B
long KeyGen_receiver_start = System.currentTimeMillis();
Element b = pairing.getZr().newRandomElement(); //skr
Element P2 = P.duplicate();
Element B = P2.mulZn(b); //C
PairingPreProcessing pB = pairing.getPairingPreProcessingFromElement(B);
long KeyGen_receiver_end = System.currentTimeMillis();
System.out.println("SCF-MPEKS KeyGen_receiver[" + j +"]: " + (KeyGen_receiver_end-KeyGen_receiver_start));
sum_KeyGen_receiver = (int) (sum_KeyGen_receiver + (KeyGen_receiver_end-KeyGen_receiver_start));
```

FIGURE 24. RECEIVER'S KEY PAIR GENERATION IN RSCF-MPEKS

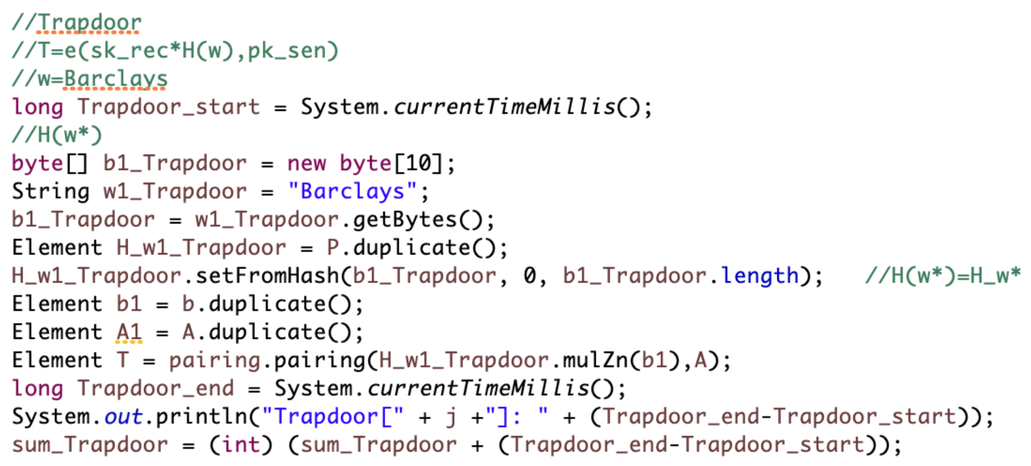
The following Figure 25 describes the Searchable ciphertext (SCF-MPEKS) generation. In order to simplify the code, let the number of keywords be three.



```
//SCF-MPEKS
//S = (M,N1,N2,...,Nn)=(t,e(sk_sen*H(w1),(pk_Rec)^t),e(sk_sen*H(w2),(pk_Rec)^t),...,e(sk_sen*H(wn),(pk_Rec)^t))
//let n=3, w1=Barclays, w2=Finance, w3=2017
long SCF_MPEKS_start = System.currentTimeMillis();
Element t_MPEKS = pairing.getZr().newRandomElement(); //t
Element tt_MPEKS = t_MPEKS.duplicate();
Element B_MPEKS = B.duplicate();
Element M = B_MPEKS.mulZn(tt_MPEKS); //M=(pk_Rec)^t
//H(w1)
byte[] b1_MPEKS = new byte[10];
String w1_MPEKS = "Barclays";
b1_MPEKS = w1_MPEKS.getBytes();
Element H_w1_MPEKS = P.duplicate();
H_w1_MPEKS.setFromHash(b1_MPEKS, 0, b1_MPEKS.length);    //H(w1)=H_w1
//H(w2)
byte[] b2_MPEKS = new byte[10];
String w2_MPEKS = "Finance";
b2_MPEKS = w2_MPEKS.getBytes();
Element H_w2_MPEKS = P.duplicate();
H_w2_MPEKS.setFromHash(b2_MPEKS, 0, b2_MPEKS.length);   //H(w2)=H_w2
//H(w3)
byte[] b3_MPEKS = new byte[10];
String w3_MPEKS = "2017";
b3_MPEKS = w3_MPEKS.getBytes();
Element H_w3_MPEKS = P.duplicate();;
H_w3_MPEKS.setFromHash(b3_MPEKS, 0, b3_MPEKS.length);   //H(w3)=H_w3
//D1=e(sk_sen*H(w1),(pk_rec)^t);D2=e(sk_sen*H(w2),(pk_rec)^t);D3=e(sk_sen*H(w3),(pk_rec)^t)
//D1,D2,D3
Element t1_MPEKS = t_MPEKS.duplicate();
Element t2_MPEKS = t_MPEKS.duplicate();
Element t3_MPEKS = t_MPEKS.duplicate();
Element a1 = a.duplicate();
Element a2 = a.duplicate();
Element a3 = a.duplicate();
Element B1 = B.duplicate();
Element B2 = B.duplicate();
Element B3 = B.duplicate();

Element D1_MPEKS = pairing.pairing(H_w1_MPEKS.mulZn(a1),B1.mulZn(t1_MPEKS));
Element D2_MPEKS = pairing.pairing(H_w2_MPEKS.mulZn(a2),B2.mulZn(t2_MPEKS));
Element D3_MPEKS = pairing.pairing(H_w3_MPEKS.mulZn(a3),B3.mulZn(t3_MPEKS));
//N1,N2,N3
Element D1_MPEKS1 = D1_MPEKS.duplicate();
Element D2_MPEKS1 = D2_MPEKS.duplicate();
Element D3_MPEKS1 = D3_MPEKS.duplicate();
byte[] N1_MPEKS = D1_MPEKS1.toBytes(); //N1
byte[] N2_MPEKS = D2_MPEKS1.toBytes(); //N2
byte[] N3_MPEKS = D3_MPEKS1.toBytes(); //N3
long SCF_MPEKS_end = System.currentTimeMillis();
System.out.println("SCF-MPEKS[" + j +"]: " + (SCF_MPEKS_end-SCF_MPEKS_start));
sum_SCF_MPEKS = (int) (sum_SCF_MPEKS + (SCF_MPEKS_end-SCF_MPEKS_start));
```

FIGURE 25. SEARCHABLE CIPHERTEXT (SCF-MPEKS) GENERATION IN RSCF-MPEKS

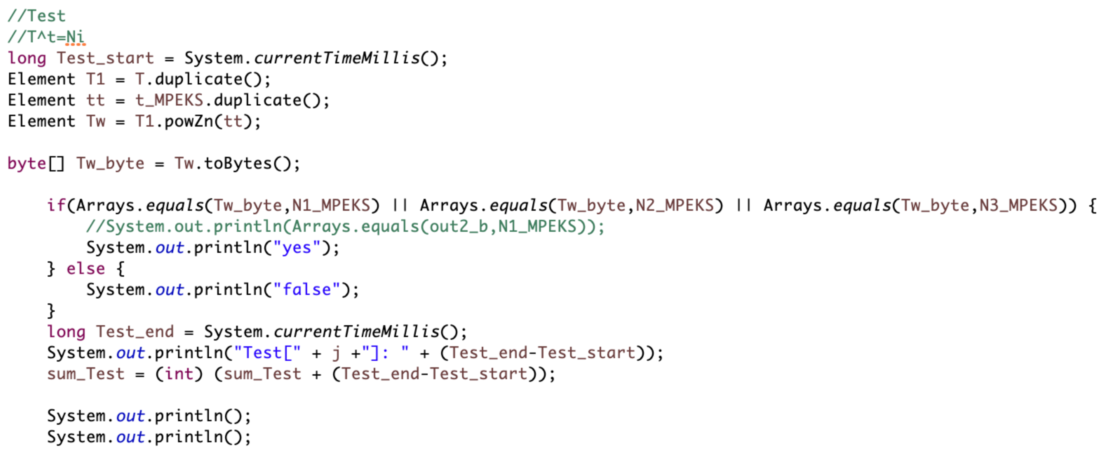
Figure 26 illustrates the Trapdoor request generation by using sender's public key pk_{Sen}, receiver's private key sk_{Rec} and a keyword w.



```
//Trapdoor
//T=e(sk_rec*H(w),pk_sen)
//w=Barclays
long Trapdoor_start = System.currentTimeMillis();
//H(w*)
byte[] b1_Trapdoor = new byte[10];
String w1_Trapdoor = "Barclays";
b1_Trapdoor = w1_Trapdoor.getBytes();
Element H_w1_Trapdoor = P.duplicate();
H_w1_Trapdoor.setFromHash(b1_Trapdoor, 0, b1_Trapdoor.length);   //H(w*)=H_w*
Element b1 = b.duplicate();
Element A1 = A.duplicate();
Element T = pairing.pairing(H_w1_Trapdoor.mulZn(b1),A);
long Trapdoor_end = System.currentTimeMillis();
System.out.println("Trapdoor[" + j +"]: " + (Trapdoor_end-Trapdoor_start));
sum_Trapdoor = (int) (sum_Trapdoor + (Trapdoor_end-Trapdoor_start));
```

FIGURE 26. TRAPDOOR REQUEST GENERATION IN RSCF-MPEKS

The final figure (Figure 27) is Test algorithm, which describes the keywords comparison between the Searchable ciphertext and the Trapdoor request.



```
//Test
//T^t=Ni
long Test_start = System.currentTimeMillis();
Element T1 = T.duplicate();
Element tt = t_MPEKS.duplicate();
Element Tw = T1.powZn(tt);

byte[] Tw_byte = Tw.toBytes();

    if(Arrays.equals(Tw_byte,N1_MPEKS) || Arrays.equals(Tw_byte,N2_MPEKS) || Arrays.equals(Tw_byte,N3_MPEKS)) {
        //System.out.println(Arrays.equals(out2_b,N1_MPEKS));
        System.out.println("yes");
    } else {
        System.out.println("false");
    }
    long Test_end = System.currentTimeMillis();
    System.out.println("Test[" + j +"]: " + (Test_end-Test_start));
    sum_Test = (int) (sum_Test + (Test_end-Test_start));

    System.out.println();
    System.out.println();
```

FIGURE 27. TEST ALGORITHM IN RSCF-MPEKS

6. Public Key Encryption with Multi-keywords Search using Mamdani System

6.1 Introduction

According to the section 1.4, it is known that almost all current PEKS and MPEKS schemes cannot deal with imprecise keywords, such as “latest”, “newest”, etc. Therefore, this chapter gives a formal definition of MPEKS scheme, which incorporates with Mamdani Fuzzy Inference System to solve Fuzzy Keyword Search problem. Besides, the proposed scheme is able to resist Off-line Keyword Guessing Attack (OKGA). More specially, this chapter firstly defines a new PEKS approach namely “*Public Key Encryption with Multi-keywords Search using Mamdani System (m-PEMKS)*” and the security verification models. Then, a concrete construction of *m-PEMKS* is proposed following by the correctness analysis, security verification and efficiency and performance analysis.

6.2 The Outline of m-PEMKS

Consider a situation: a bank manager would like to search the “latest” financial report. But, what is the “latest”? A week ago? A month ago? A year ago? Therefore, almost all current PEKS systems are not able to solve imprecise keyword search. This thesis provides a way that incorporates with Fuzzy Logic method into Searchable Cryptography to solve Fuzzy Keyword Search issue, called “*Public Key Encryption with Multi-keywords Search using Mamdani System (m-PEMKS)"*. The *m-PEMKS* contains eight polynomial time algorithms as follows:

1. $KeyGen_{Param-PEKS}(1^n)$: Import 1^n, a common parameter cp is then created.
2. $KeyGen_{Param-RSA}(k)$: Import k, a global parameter gp is then created.

3. $KeyGen_{Ser-PEKS}(cp)$: Import cp, a public and a private PEKS keys $(pk_{Ser-PEKS}, sk_{Ser-PEKS})$ of the server are then created.

4. $KeyGen_{Ser-RSA}(k)$: Import gp, a public and private RSA keys $(pk_{Ser-RSA}, sk_{Ser-RSA})$ of the server are then created.

5. $KeyGen_{Rec-PEKS}(cp)$: Import cp, a public and private PEKS keys $(pk_{Rec-PEKS}, sk_{Rec-PEKS})$ of the receiver are then created.

6. $Encryption(pk_{Ser-PEKS}, pk_{Rec-PEKS}, pk_{Ser-RSA}, W)$: Import the server's PEKS public key $pk_{Ser-PEKS}$, the receiver's PEKS public key $pk_{Rec-PEKS}$, the server's RSA public key $pk_{Ser-RSA}$ and a keyword-vector $W = (W_{part-1}; W_{part-2}) = [(w_1, w_2, \ldots, w_{\eta-1}); w_\eta]$. An encryption is subsequently produced in the following: $E = (E1, E2) = SCF - MPEKS(pk_{Ser-PEKS}, pk_{Rec-PEKS}, W_{part-1}) \,||\, RSA(pk_{Ser-RSA}, W_{part-2})$.

7. $Request(pk_{Ser-PEKS}, sk_{Rec-PEKS}, pk_{Ser-RSA}, W*)$: Import the server's PEKS public key $pk_{Ser-PEKS}$, the receiver's PEKS private key $sk_{Rec-PEKS}$, the server's RSA public key $pk_{Ser-RSA}$ and a keyword-vector $W* = (W*_{part-1}; W*_{part-2}) = [(w_1^*, w_2^*, \ldots, w_{\iota-1}^*); w_\iota^*]$. A request is subsequently created in the following: $R = (R1, R2) = Trapdoor(pk_{Ser-PEKS}, sk_{Rec-PEKS}, W*_{part-1}) \,||\, RSA(pk_{Ser-RSA}, W*_{part-2})$.

8. $Test(E, R, sk_{Ser-MPEKS}, sk_{Ser-RSA})$: The algorithm contains two steps, which are called Searchable Match and Fuzzy Match.

- For Searchable Match, input the server's PEKS private key $sk_{Ser-PEKS}$, the Searchable ciphertext $E1 = SCF - MPEKS(pk_{Ser-PEKS}, \; pk_{Rec-PEKS}, W_{part-1})$ and the Trapdoor query $R1 = Trapdoor(pk_{Ser-PEKS}, sk_{Rec-PEKS}, W^*_{part-1})$. If $W^*_{part-1} \subseteq W_{part-1}$, mark it as the Fuzzy Match input. Then, the server repeats Searchable Match until traversing all the encrypted messages stored in its database.

- If the server has the marked input(s), it will run Fuzzy Match search. Otherwise, the system will be terminated by the server.

- For Fuzzy Match, input the server's RSA private key $sk_{Ser-RSA}$, the RSA encryption $E2 = RSA(pk_{Ser-RSA}, W_{part-2})$ and the RSA request $R2 = RSA(pk_{Ser-RSA}, W^*_{part-2})$. Then, the server firstly decrypts $E2$ and $R2$ to obtain W_{part-2} and W^*_{part-2}. Let W_{part-2} and W^*_{part-2} be the condition and the conclusion of the rules in Mamdani system. After running Mamdani system, the server sends a response to the receiver.

6.3 The Security Models of m-PEMKS

The security of *m-PEMKS* system relies on two parts. The first one is **IND-CPA** and **Trapdoor-IND-CPA** applying in Exact keywords search. Another one is the difficulty of factoring large integers, which is used in Fuzzy keyword search.

Due to the properties of Mamdani system, the input for Mamdani system must be plaintext. However, it is no meaningful to verify the security on the Fuzzy Keyword Search part. More specially, suppose the fuzzy keyword is “highest”. Even though the cracker knows that the fuzzy keyword is “highest”, he/she could not know more details (such as “highest building”, “highest person”, “highest temperature”, etc.).

This part only focuses on the security of Exact Keywords Search in *m-PEMKS* system and the proposed security models are listed below:

As discussed in (Baek et al., 2008; Wang et al., 2016; Ma and Kazemian, 2018), *m-PEMKS* is **IND-CPA** and **Trapdoor-IND-CPA**.

The definition of **IND-CPA** security means that the untrusted server may not determine which Searchable ciphertext has which encrypted keyword, if the Trapdoor query that contains the given keyword has not been obtained by the server (***Game6)***. Besides, if the server's private key has not been obtained by the untrusted receiver, he/she could not estimate whether the SCF-MPEKS ciphertext ($E1$) and the Trapdoor request ($R1$) contain the same keyword or not, even though all Trapdoors for any keyword are intercepted (***Game7)***.

The definition of **Trapdoor-IND-CPA** security means that an outside adversary cannot observe any difference between Trapdoors for any two distinct keywords (***Game8)***.

Therefore, the **IND-CPA** and **Trapdoor-IND-CPA** for *m-PEMKS* are formalized as follows: Suppose **A** is an adversary and **E** is a challenger.

***Game6*: Let A suppose to be an untrusted server.**

Stage1 (Setup): $KeyGen_{Param-PEKS}(1^n)$, $KeyGen_{Ser-PEKS}(cp)$ and $KeyGen_{Rec-PEKS}(cp)$ are called by **E** in order to generate a common parameter cp, the key pairs $(pk_{Ser-PEKS}, sk_{Ser-PEKS})$ and $(pk_{Rec-PEKS}, sk_{Rec-PEKS})$ of the server and the receiver. Then, **E** sends $cp, pk_{Ser-PEKS}, sk_{Ser-PEKS}$ and $pk_{Rec-PEKS}$ to **A**.

Stage2 (Trapdoor queries): Adaptively, **E** is able to return any Trapdoor query T^*_W for any keyword-vector $W* = (w^*_1, \dots, w^*_l)$.

Stage3 (Challenge simulation): A target keyword-vector pair $[W_0 = (w_{01}, \dots, w_{0\eta}),\ W_1 = (w_{11}, \dots, w_{1\eta})]$ is sent from **A** to **E**. It is known that W_0

and W_1 cannot be requested in **Stage2 (*Game6*)**. Once **E** obtains the keyword-vector pair, the $SCF-MPEKS$ algorithm will be called by **E** for generating a Searchable ciphertext $C = SCF-MPEKS(pk_{Ser-PEKS}, pk_{Rec-PEKS}, W_\xi)$, where $\xi \in \{0,1\}$. Finally, C will be sent back to **A**.

Stage4 (Trapdoor queries): **E** can continue return any Trapdoor query T^*_W for any keyword-vector W^* to **A** as in **Stage2 (*Game6*)**, only if $W^* \neq W^*_0, W^*_1$.

Stage5 (Guess): **A** guesses $\xi^* \in \{0,1\}$ and wins ***Game6***, if $\xi^* = \xi$.

Game7: **Let A suppose to be an untrusted receiver.**

Stage1 (Setup): $KeyGen_{Param-PEKS}(1^n)$, $KeyGen_{Ser-PEKS}(cp)$ and $KeyGen_{Rec-PEKS}(cp)$ are called by **E** in order to generate a common parameter cp, the key pairs $(pk_{Ser-PEKS}, sk_{Ser-PEKS})$ and $(pk_{Rec-PEKS}, sk_{Rec-PEKS})$ of the server and the receiver. Then, **E** sends $cp, pk_{Rec-PEKS}, sk_{Rec-PEKS}$ and $pk_{Ser-PEKS}$ to **A**.

Stage2 (Challenge simulation): A target keyword-vector pair $[W_0 = (w_{01}, \ldots, w_{0\eta}),\ W_1 = (w_{11}, \ldots, w_{1\eta})]$ is sent from **A** to **E**. It is known that $T_{w_{0i}}$ and $T_{w_{1i}}$ are not able to be requested during *Test* algorithm, on which $i = 1,...,\iota$. Once the challenger **E** obtains the pair, the $SCF-MPEKS$ algorithm will be called by **E** for generating a Searchable ciphertext $C = SCF-MPEKS(pk_{Ser-PEKS}, pk_{Rec-PEKS}, W_\xi)$, where $\xi \in \{0,1\}$. Finally, C will be sent back to **A**.

Stage3 (Guess): **A** guesses $\xi^* \in \{0,1\}$ and wins ***Game7***, if $\xi^* = \xi$.

A's advantage to win ***Game6*** and ***Game7*** is listed below:

$$Adv_{m-PEMKS,A_i}^{IND-CPA}(k) = |Pr[\xi^* = \xi] - 1/2|. \quad (i = 6,7)$$

So, the *m-PEMKS* model is considered to be **IND-CPA** secure as long as $Adv_{m-PEMKS,A_i}^{IND-CPA}(k)$ is trivial.

***Game8*: Let A suppose to be an outside attacker.**

Stage1 (Setup): $KeyGen_{Param-PEKS}(1^n)$, $KeyGen_{Ser-PEKS}(cp)$ and $KeyGen_{Rec-PEKS}(cp)$ are called by **E** in order to generate a common parameter cp, the key pairs $(pk_{Ser-PEKS}, sk_{Ser-PEKS})$ and $(pk_{Rec-PEKS}, sk_{Rec-PEKS})$ of the server and the receiver. Then, **E** sends cp, $pk_{Ser-PEKS}$, $pk_{Rec-PEKS}$ to **A** and keeps $sk_{Ser-PEKS}$, $sk_{Rec-PEKS}$ from **A**.

Stage2 (Trapdoor queries): Adaptively, **E** is able to return any Trapdoor query T_W^* for any keyword-vector $W^* = (w_1^*, ..., w_l^*)$ to **A**.

Stage3 (Challenge simulation): A target keyword-vector pair $[W_0^* = (w_{01}^*, ..., w_{0l}^*),\ W_1^* = (w_{11}^*, ..., w_{1l}^*)]$ is sent from **A** to **E.** It is known that W_0^* and W_1^* cannot be requested in **Stage2 (*Game8*).** Once **E** obtains the keyword-vector pair, the $Trapdoor$ algorithm will be called by the challenger **E** for generating a Trapdoor query $T_W = Trapdoor(pk_{Ser-PEKS}, sk_{Rec-PEKS}, W_\xi^*)$, where $\xi \in \{0,1\}$. Finally, T_W will be sent back to **A**.

Stage4 (Trapdoor queries): E can continue return any Trapdoor query T_W^* for any keyword-vector W^* to **A** as in **Stage2 (*Game8*)**, only if $W^* \neq W_0^*, W_1^*$.

Stage5 (Guess): A guesses $\xi^* \in \{0,1\}$ and wins ***Game8***, if $\xi^* = \xi$.

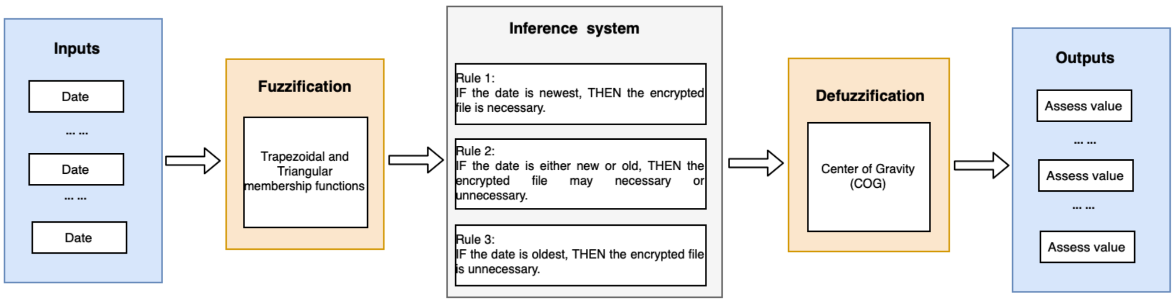
A's advantage to win ***Game8*** is listed below:

$$Adv_{m-PEMKS,A_8}^{Trap-IND-CPA}(k) = |Pr[\xi^* = \xi] - 1/2|.$$

Therefore, the *m-PEMKS* model is considered to be **Trapdoor-IND-CPA** secure as long as $Adv_{m-PEMKS,A_8}^{Trap-IND-CPA}(k)$ is trivial.

6.4 The Fuzzy Inference System of m-PEMKS

Almost all current PEKS schemes will report errors, if the keyword for searching is blur. On the contrary, *m-PEMKS* scheme incorporates with the fuzzy logic technique to solve fuzzy keyword search problem. To simplicity, let's take an example about searching "latest" financial reports. So, Figure 28 illustrates the fuzzy inference system structure of this example that is used in m-PEMKS scheme.



Inputs
Date
... ...
Date
... ...
Date
Fuzzification
Trapezoidal and Triangular membership functions
Inference system
Rule 1: IF the date is newest, THEN the encrypted file is necessary.
Rule 2: IF the date is either new or old, THEN the encrypted file may necessary or unnecessary.
Rule 3: IF the date is oldest, THEN the encrypted file is unnecessary.
Defuzzification
Center of Gravity (COG)
Outputs
Assess value
... ...
Assess value
... ...
Assess value

FIGURE 28. THE STRUCTURE OF FUZZY INFERENCE SYSTEM

Note that the fuzzy system in m-PEMKS scheme is implemented by JAVA using jFuzzyLogic (Cingolani et al., 2012) package. From Figure 28, it can be seen that the inputs and outputs for the fuzzy system are the crisp values and the membership functions for both fuzzification and defuzzification are defined for every linguistic term using TERM statement that is followed by a function definition. Functions are defined as piece-wise linear functions using a series of points $(A_0, B_0), (A_1, B_1), \ldots, (A_m, B_m)$. According to Section 6.2, the bank manger would like to search the "latest" financial statements so that the fuzzification could apply trapezoidal and triangular membership functions, for instance, TERM $Date := \{(1,0),(4,1),(6,1),(9,0)\}$ defines the trapezoidal membership function. Apart from that, the defuzzification method applies Center of Gravity (COG) and each rule used in this fuzzy inference system is defined by "IF condition THEN conclusion". More details can be found in Figure 38.

Also note that the PhD thesis is mainly on cryptography area and the design of fuzzy inference system (such as why defuzzification method uses COG? Why use trapezoidal membership functions? etc.) is not the key point. The purpose of m-PEMKS scheme is to show that the searchable cryptography (PEKS) could connect and apply with fuzzy logic to solve the fuzzy keyword search.

6.5 The Concrete Construction of m-PEMKS

1. $KeyGen_{Param-PEKS}(1^n)$: Suppose G_1 is an additive cyclic group and G_T is a multiplicative cyclic group. Let P be a random generator of G_1 and a prime number $g \geq 2^k$ be the order of G_1. A bilinear pairing is considered to be a map $e : G_1 \times G_1 \to G_T$. Suppose $H : \{0,1\}^\star \to G_1$ and $H* : G_T \to \{0,1\}^*$ are two particular hash functions. Therefore, a common parameter $cp = \{g, P, G_1, G_T, e, H, H*\}$ can be achieved by the $KeyGen_{Param-PEKS}(1^n)$ algorithm.

2. $KeyGen_{Param-RSA}(k)$: Randomly select prime numbers $\mathbb{P}$, $\mathbb{V}$ where $\mathbb{P} \neq \mathbb{V}$. Then, calculate $\mathbb{Z} = \mathbb{P} \times \mathbb{V}$ and $\phi(\mathbb{Z}) = (\mathbb{P} - 1) \times (\mathbb{V} - 1)$.

3. $KeyGen_{Ser-PEKS}(cp)$: The server selects $m \in Z_P$ uniformly at random and then calculates $M = mP$. In addition, the server also randomly selects $K \in G_1$. So, $pk_{Ser-PEKS} = (pk_{Ser-PEKS1}, pk_{Ser-PEKS2}) = (cp, M, N)$ and $sk_{Ser-PEKS} = (cp, m)$ are the server's public and private PEKS keys.

4. $KeyGen_{Ser-RSA}(k)$: The server randomly selects $f \in Z_I$, where $gcd(\phi(\mathbb{Z}), f) = 1, 1 < f < \phi(\mathbb{Z})$. Next, the server calculates l by $l \equiv f^{-1}(mod\phi(\mathbb{Z}))$. $pk_{Ser-RSA} = (f, \mathbb{Z})$ and $sk_{Ser-RSA} = (l, \mathbb{Z})$ are the server's public and private RSA keys.

5. $KeyGen_{Rec-PEKS}(cp)$: The receiver selects $n \in Z_P$ uniformly at random and then calculates $N = nP$. So, $pk_{Rec-PEKS} = (cp, N)$ and $sk_{Rec-PEKS} = (cp, n)$ are the receiver's public and private PEKS keys.

6. $Encryption(pk_{Ser-PEKS}, pk_{Rec-PEKS}, pk_{Ser-RSA}, W)$: The sender randomly picks up $t \in Z_P$, $W = (W_{part-1}; W_{part-2}) = [(w_1, w_2, \ldots, w_{\eta-1}); w_\eta]$ and then calculates an encryption $E = (E_1, E_2) = [(X, Y_1, Y_2, \ldots Y_{\eta-1});\ Y_\eta] = [(tM, H^*(V_1), H^*(V_2), \ldots, H^*(V_{\eta-1}));\ (w_\eta)^f mod\, \mathbb{Z}]$, where $V_1 = e(H(w_1), N)^t$, $V_2 = e(H(w_2), N)^t$, $\ldots, V_{\eta-1} = e(H(w_{\eta-1}), N)^t$.

7. $Request(pk_{Ser-PEKS}, sk_{Rec-PEKS}, pk_{Ser-RSA}, W^*)$: The receiver randomly selects $t^* \in Z_P, W^* = (W^*_{part-1}; W^*_{part-2}) = [(w^*_1, w^*_2, \ldots, w^*_{\iota-1}); w_\iota]$ and then calculates $R = (R_1, R_2) = [(Q, T_1, T_2, \ldots, T_{\iota-1}), T_\iota] = [(e(M, t^*K), nH(w^*_1) \oplus e(M, K)^{t^*+n}, nH(w^*_2) \oplus e(M, K)^{t^*+n}, \ldots, nH(w^*_{\iota-1}) \oplus e(M, K)^{t^*+n}); (w^*_\iota)^f mod\, \mathbb{Z}]$.

8. $Test(E, R, sk_{Ser-MPEKS}, sk_{Ser-RSA})$: For $i \in \{1,2,...,\eta\}$ and $j \in \{1,2,...,\iota\}$, where $j \le i$.

i. For Searchable Match:

Firstly, the server computes

$$T_{w_1} = T_1 \oplus Q \bullet e(mK, N) = nH(w^*_1),$$

$$T_{w_2} = T_2 \oplus Q \bullet e(mK, N) = nH(w^*_2), \ldots,$$

$$T_{w_j} = T_j \oplus Q \bullet e(mK, N) = nH(w^*_j), \ldots,$$

$$T_{w_{\iota-1}} = T_{\iota-1} \oplus Q \bullet e(mK, N) = nH(w^*_{\iota-1}).$$

Then, the server tests whether $H^*[e(T_{w_j}, \frac{X}{m})] = Y_i$ or not. If "yes", mark it as the Fuzzy Match input. Next, the server repeats Searchable Match until traversing all the encrypted messages stored in its database.

ii. If the server obtains the marked input(s), it will run Fuzzy Match search. Otherwise, the system will be terminated by the server.

iii. For Fuzzy Match (More details are in section 6.4): the server firstly decrypts w_η and w^*_ι as $\{[(w_\eta)^f mod\ \mathbb{Z}]^l mod\ \mathbb{Z}\}$ and $\{[(w^*_\iota)^f mod\ \mathbb{Z}]^l mod\ \mathbb{Z}\}$ respectively. Let w_η and w^*_ι be the condition and the conclusion of the rules in Mamdani Fuzzy Inference System. After running Mamdani Fuzzy Inference System, the server replies to the receiver in the following.

Without loss of generality, suppose "w_η" stands for a set of DATE while "w^*_ι" is the keyword "latest" . Therefore, three rules can be defined as follows:

Rule1: IF DATE is oldest, THEN the encrypted file is unnecessary.

Rule2: IF DATE is newest, THEN the encrypted file is necessary.

Rule3: IF DATE is either new or old, THEN the encrypted file may necessary or may unnecessary.

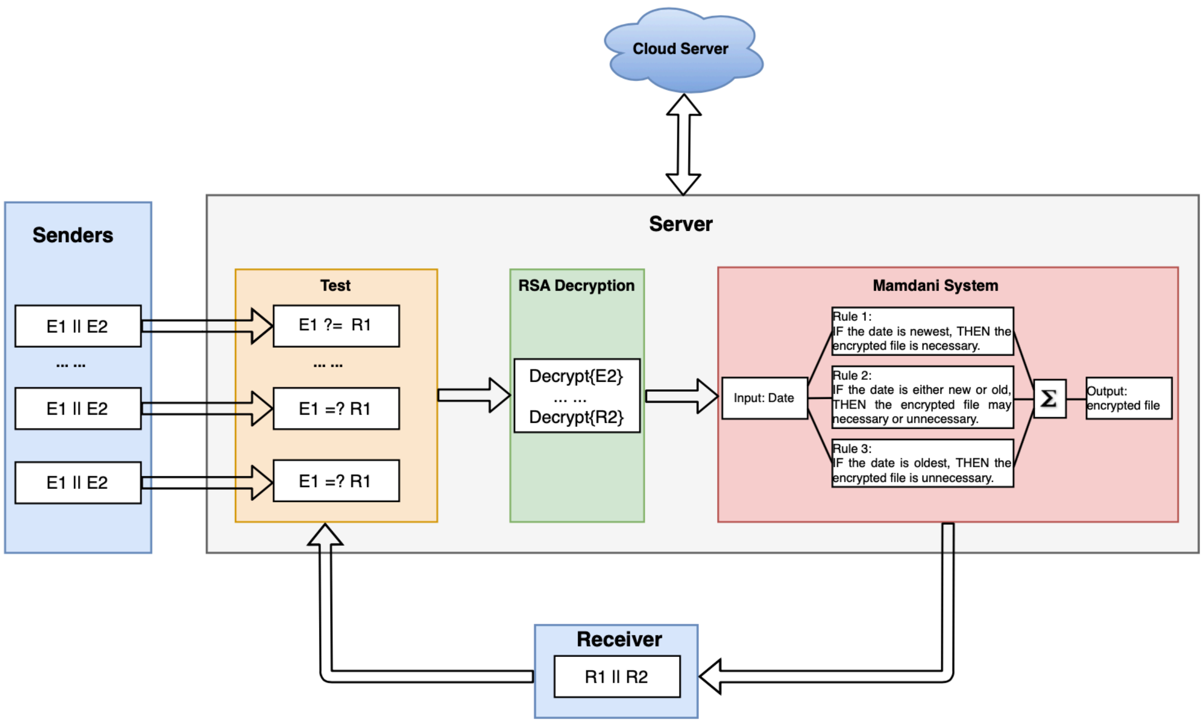


FIGURE 29. THE STRUCTURE OF M-PEMKS

6.6 The Correctness of m-PEMKS

i. For Searchable Match:

For $i \in \left\{1,2,\ldots,\eta - 1\right\}$ and $j \in \{1,2,\ldots,\iota - 1\}$, the correctness of the proposed approach is easily verified as follows:

Note that $\bullet$ stands for Multiplication and $\oplus$ stands for Exclusive Or.

According to Bilinear pairing, note also that $e(M,K) = e(K,M)$ and $e(M,K)^{t^*+n} = e(t^*M, nK) = e(nM, t^*K)$.

Therefore, firstly,

$$T_{w_j} = T_j \oplus Q \bullet e(mK, N) = nH(w_j^*)$$

$$= nH(w_j^*) \oplus e(M,K)^{t^*+n} \oplus e(M, t^*K) \bullet e(mK, N)$$

$$= nH(w_j^*) \oplus e(M,K)^{t^*+n} \oplus e(M, t^*K) \bullet e(M, nK)$$

$$= nH(w_j^*) \oplus e(M, K)^{t^*+n} \oplus e(M, K)^{t^*+n}$$

$$= nH(w_j^*)$$

Secondly, $H^*[e(T_{w_j}, \frac{X}{m})] = H^*[e(nH(w_j^*), \frac{tM}{m})]$

$$= H^*[e(nH(w_j^*), \frac{ntP}{n})]$$

$$= H^*[e(H(w_j^*), N)^t]$$

$$= Y_i$$

ii. For Fuzzy Match:

This algorithm is still correct due to the properties of Mamdani system.

6.7 The Security Analysis of m-PEMKS

The *m-PEMKS* approach possesses the characters of Ciphertext Indistinguishability and Trapdoor Indistinguishability against Chosen Plaintext Attack (CPA) whose security relies on BDH and 1-BDHI assumptions (Boneh and Boyen, 2004).

The proposed approach above could be regarded as **IND-CPA** secure in ***Game6*** under the random oracle model, if the BDH assumption (Boneh and Boyen, 2004) is completely difficult.

Game6: **Let A suppose to be an untrusted server.**

Consider that the challenger **E** is able to achieve the input $(g, P, G_1, G_T, e, \alpha P, \beta P, \gamma P)$ of BDH assumption (Boneh and Boyen, 2004). **E** sets up the computation of a BDH key $e(P, P)^{\alpha\beta\gamma}$ of αP, βP and γP using **A**'s **IND-CPA** as a goal. Apart from that, **A** requests at most h and h^* times hash function requests.

Stage1 (Setup)

E chooses $N = \alpha P$ in the beginning. Then, **E** chooses $m \in Z_P$ uniformly at random and also computes $M = mP$. In addition, **E** randomly selects $K \in G_1$. Finally, the following parameters are returned by **E**, which are the common parameter $(g, P, G_1, G_T, e, H, H^*)$, the server's public/private PEKS key pair (cp, M, K) and (cp, m), and the receiver's public PEKS key (cp, N). Apart from that, two particular hash functions H and H^* are selected by **E** in the following:

- **A** is able to request a keyword w_i to H function at any time. After that, **E** traverses a tuple $(w_i, \mu_i, \nu_i, \varepsilon_i)$ from H_List that is initially empty. If the tuple exists, **E** will return $H(w_i) = \mu_i$ to **A**. Otherwise, the challenger **E** executes the details below:

i. The challenger **E** randomly selects a coin ε_i and then computes $Pr[\varepsilon_i = 0] = \frac{1}{h+1}$.

ii. The challenge **E** randomly chooses $\nu_i \in Z_P$. If $\varepsilon_i = 0$, $\mu_i = \beta P + \nu_i P$ will be computed by **E**. Similarly, $\mu_i = \nu_i P$ will be computed by **E** once $\varepsilon_i = 1$.

iii. **A** receives μ_i from **E**. Meanwhile, **E** adds $(w_i, \mu_i, \nu_i, \varepsilon_i)$ into H_List.

- **A** is able to request V_i to H^* function at any time. Later on, **E** traverses a tuple (V_i, Y_i) from H^*_List. If the tuple exists, **E** will return Y_i to **A**. Otherwise, **E** randomly selects $Y_i \in \{0,1\}^\bullet$ and replies Y_i to **A**. Finally, **E** adds (V_i, Y_i) into H^*_List.

Stage2 (Trapdoor queries)

If **A** queries a Trapdoor request with a specific keyword-vector $W_i = (w_1, w_2, \ldots w_l)$, **E** will do the operations below:

- The challenger **E** recalls the above algorithms in order to simulate H function for generating a tuple $(w_i, \mu_i, \nu_i, \varepsilon_i)$. If $\varepsilon_i = 0$, **E** will output "Suspension" and also terminate the system. Otherwise, the challenger **E** executes the following steps.

- **E** randomly chooses $t^* \in Z_P$ and calculates $Z = e(M, t^*K)$.

- **E** then computes $T_1 = \nu_1 N \oplus e(M,K)^{t^*+\alpha} = \nu_1 \alpha P \oplus e(M,K)^{t^*+\alpha}$ $= x\mu_1 \oplus e(M,K)^{t^*+\alpha} = \alpha H(w_1) \oplus e(M,K)^{t^*+\alpha}$, $T_2 = \alpha H(w_2) \oplus e(M,K)^{t^*+\alpha}$ $,\ldots, T_\iota = \alpha H(w_\iota) \oplus e(M,K)^{t^*+\alpha}$. So, $T_W = (Q, T_1, T_2, \ldots, T_\iota)$.

Stage3 (Challenge simulation)

The challenger **A** sends a keyword-vector pair $[W_0 = (w_{01}, \ldots., w_{0\eta})$, $W_1 = (w_{11}, \ldots., w_{1\eta})]$ to **E**. Once the challenger **E** obtains the keyword-vector pair, he/she will do the following steps:

- **E** chooses $i \in \{1,2,...,\eta\}$ uniformly at random.

- **E** recalls the above algorithms in order to simulate H function for obtaining two tuples $(w^*_{0i}, \mu^*_{0i}, \nu^*_{0i}, \varepsilon^*_{0i})$ and $(w^*_{1i}, \mu^*_{1i}, \nu^*_{1i}, \varepsilon^*_{1i})$. If ε^*_{0i} and ε^*_{1i} are equal to 1, **E** will output "Suspension" and also terminate the system. Otherwise, the challenger **E** does the following operations:

i. **E** recalls above algorithms again in order to simulate H function at $2(\eta - 1)$ times for searching two tuples' vectors $\{(w^*_{01}, \mu^*_{01}, \nu^*_{01}, \varepsilon^*_{01}), \ldots,$ $(w^*_{0i-1}, \mu^*_{0i-1}, \nu^*_{0i-1}, \varepsilon^*_{0i-1})$, $(w^*_{0i+1}, \mu^*_{0i+1}, \nu^*_{0i+1}, \varepsilon^*_{0i+1})$ $,\ldots, (w^*_{0\eta}, \mu^*_{0\eta}, \nu^*_{0\eta}, \varepsilon^*_{0\eta})\}$ and $\{(w^*_{11}, \mu^*_{11}, \nu^*_{11}, \varepsilon^*_{11}), \ldots,$ $(w^*_{1i-1}, \mu^*_{1i-1}, \nu^*_{1i-1}, \varepsilon^*_{1i-1}), (w^*_{1i+1}, \mu^*_{1i+1}, \nu^*_{1i+1}, \varepsilon^*_{1i+1}), \ldots,$ $(w^*_{1\eta}, \mu^*_{1\eta}, \nu^*_{1\eta}, \varepsilon^*_{1\eta})\}$. If $\varepsilon^*_{0j} = \varepsilon^*_{1j} = 0$ for all $j = 0,...,i-1,i+1,...,\eta$, the challenger **E** will export "Suspension" and terminate the system. Otherwise, the challenger **E** does the following operations:

— The challenger **E** randomly picks up $\delta \in \{0,1\}$.

— The challenger **E** randomly picks up $Y_i \in \{0,1\}^\bullet$ and then generates a target $SCF-MPEKS$ ciphertext $C^* = (X^*, Y^*_1, Y^*_2, \ldots, Y^*_\eta) = (\gamma M, H^*[B_1], H^*[B_2], \ldots,$ $H^*[B_\eta])$.

So,

$C^* = (X^*, Y_1^*, \ldots, Y_{i-1}^*, Y_{i+1}^*, \ldots, Y_\eta^*) = (\gamma M, H^*[e(H(w_{\delta_1}), N)^\gamma], \ldots,$

$H^*[e(H(w_{\delta_{i-1}}), N)^\gamma], H^*[e(H(w_{\delta_i+1}), N)^\gamma], \ldots, H^*[e(H(w_{\delta_\eta}), N)^\gamma])$.

Note that

$B_i = e(H(w_{\delta_i}), N)^\gamma = e(\beta P + \nu_{\delta_i} P, \alpha P)^\gamma = e(\beta P, \alpha P)^\gamma \bullet e(\nu_{\delta_i} P, \alpha P)^\gamma =$

$e(P, P)^{\alpha\beta\gamma} \bullet e(\gamma P, \alpha P)^{\nu_{\delta i}}$.

Note also that $e(\nu_{\delta_i} P, \alpha P)^\gamma = e(\nu_{\delta_i} P, N)^\gamma = e(H(w_{\delta_i}), N)^\gamma$

Stage4 (Trapdoor queries)

E can continue return any Trapdoor query T_W^* for any keyword-vector W^* to **A** as in **Stage2 (*Game6*)**, only if $W^* \neq W_0^*, W_1^*$.

Stage5 (Guess)

A outputs $\delta^* \in \{0,1\}$ as the guess. Then, **E** chooses s from H^* function and replies the guessed BDH key $\frac{s_{\delta_i^*}}{e(\gamma P, \alpha P)^{\nu_{\delta_i^*}}}$.

Analysis of *Game6*

Stage1-5 describes the procedure and operations of the challenger **E**. It remains to show that BDH assumption (Boneh and Boyen, 2004) is satisfied in ***Game6***. To do so, the first thing is to analyze that the challenger **E** does not stop during the simulation. Therefore, three events are formalized below:

Event10: The challenger **E** does not stop during **Stage2 (Trapdoor queries)** and **Stage4 (Trapdoor queries)**.

Event11: The challenger **E** does not stop during **Stage3 (Challenge simulation)**.

Event12: The adversary **A** is not able to request either$H^*(e(H(w^*_{0i}), N)^\gamma)$ or $H^*(e(H(w^*_{1i}), N)^\gamma)$.

Claim 10: $Pr[Event10] \geq \frac{1}{e^\iota}$

Proof: Consider that **A** cannot request the same keyword twice in **Stage2** and **Stage4**. So, $\frac{1}{h+1}$ is the probability causing **E** for suspension. From the previous definition, **A** queries at most h Trapdoor requests and the keyword-vector in Trapdoor has ι elements so that the probability that the system which does not be terminated by **E** in all Trapdoor queries is at least $[(1-\frac{1}{h+1})^h]^\iota \geq \frac{1}{e^\iota}$.

Claim 11: $Pr[Event11] \geq (\frac{1}{h+1}) \bullet (\frac{h}{h+1})^{2(\eta-1)}$

Proof: If $\varepsilon_0 = \varepsilon_1 = 1$, the system will be terminated by **E** during **Stage3 (Challenge simulation)**. So, the $1-(1-\frac{1}{h+1})^2$ is the probability that **E** does not suspend. In addition, if $\varepsilon^*_{0j} = \varepsilon^*_{1j} = 0$ for all $j = 0,...,i-1,i+1,...,\eta$, the system will be terminated by **E**. Overall, the probability that the system which does not be terminated by **E** during **Stage3** is at least$(1-\frac{1}{h+1})^{2(\eta-1)}\{1-(1-\frac{1}{h+1})^2\} \geq (\frac{1}{h+1}) \bullet (\frac{h}{h+1})^{2(\eta-1)}$.

Claim 12: $Pr[Event12] \geq 2\xi$

Proof: As discussed in (Baek et al., 2008), let $Hybrid_r$ for $r \in \{1,2,...,\eta\}$ be an **event** that the adversary **A** can correctly guess the keyword of the left part of a "hybrid" $SCF-MPEKS$ encryption formed with r, coordinates from w_β followed by $(\eta - r)$ coordinates from $w_{1-\beta}$. So, $Pr[Event12] = 2\Sigma^{\eta}_{j=1}(Pr[Hybrid_r] - Pr[Hybrid_{r-1}])$ $=2(Pr[Hybrid_r] - Pr[Hybrid_0]) = 2\xi$.

Overall, due to **A** queries either $H^*(e(H(w_{0i}^*), N)^\gamma)$ or $H^*(e(H(w_{1i}^*), N)^\gamma)$ being at least 2ξ, the probability that **A** querying $H^*(e(H(w_{ji}^*), N)^\gamma)$ is at least ξ. Therefore, the success probability ξ^* achieved by **E** is $(\frac{h}{h+1})^{2(\eta-1)} \bullet \frac{\xi}{e^\iota(h+1)h^*}$, which is negligible.

The proposed scheme above could be regarded as **IND-CPA** secure in ***Game7*** under the random oracle model, if the 1-BDHI assumption (Boneh and Boyen, 2004) is completely difficult.

Game7: **Let A suppose to be an untrusted receiver.**

Consider that **E** is able to achieve the input $(g, P, G_1, G_T, e, \alpha P)$ of 1-BDHI assumption (Boneh and Boyen, 2004). **E** sets up the computation of a 1-BDHI key $e(P, P)^{\frac{1}{\alpha}}$ of αP using **A**'s **IND-CPA** as a goal. Apart from that, **A** requests at most h and h^* times hash function requests.

Stage1 (Setup)

E selects $M = \alpha P$ and $K \in G_1$ in the beginning. Then, **E** randomly chooses $n \in Z_P$ and also computes $N = nP$. After that, the following parameters are returned by **E**, which are the common parameter $(g, P, G_1, G_T, e, H, H^*)$, the server's public PEKS key (cp, M, K), and the receiver's public/private PEKS key pair (cp, N) and (cp, n). Apart from that, two specific hash functions H and H^* are selected by **E** in the following:

— **A** is able to request a keyword w_i to H function at any time. Later on, **E** traverses a tuple (w_i, μ_i, v_i) from H_List. If the tuple exists, **E** will return μ_i to **A**. Otherwise, **E** randomly chooses $\nu_i \in Z_P$ and computes $\mu_i = \nu_i P$. After that, **E** responds μ_i to **A**.

— **A** is able to request V_i to H^* function at any time. Later on, **E** traverses a tuple (V_i, Y_i) from H^*_List. If the tuple exists, **E** will return Y_i to **A**. Otherwise,**E** randomly selects $Y_i \in \{0,1\}^\bullet$ and replies Y_i to **A**. Finally, **E** adds (V_i, Y_i) into H^*_List.

Stage2 (Challenge simulation)

A uploads a keyword-vector pair $[(W^*_{0i}, F^*_{0i}, f^*_{0i}, \theta^*_{0i}),(W^*_{1i}, F^*_{1i}, f^*_{1i}, \theta^*_{1i})]$ to **E**, where $W^*_0 = (w_{01}, w_{02}, \ldots, w_{0\eta})$ and $W^*_1 = (w_{11}, w_{12}, \ldots, w_{1\eta})$. Once the challenger **E** obtains the pair, he/she will do the following steps:

— The challenger **E** randomly picks up $Y_i \in \{0,1\}^\bullet$ and $\delta \in \{0,1\}$.

— The challenger **E** recalls the $SCF-MPEKS$ algorithm for generating the Searchable ciphertext $C^* = (X^*, Y^*_1, Y^*_2, \ldots, Y^*_\eta) = (\psi\alpha P, H^*[B_1], H^*[B_2], \ldots, H^*[B_\eta])$.

So,

$C^* = (X^*, Y^*_1, Y^*_2, \ldots, Y^*_\eta) = (\psi\alpha P, H^*(e(H(w_{\delta_1}), N)^\psi), H^*(e(H(w_{\delta_2}), N)^\psi), \ldots, H^*(e(H(w_{\delta_\eta}), N)^\psi))$.

It is known that $B_i = e(H(w_{\delta^*_i}), N)^\psi) = e(\nu_i P, nP)^\psi = e(P, P)^{\psi \cdot \nu_i n}$.

Stage3 (Guess)

The adversary **A** exports $\delta^* \in \{0,1\}$ as the guess. Then, **E** returns the guessed 1-BDHI key $\psi = \frac{1}{\alpha \cdot \nu_i n}$.

Analysis of *Game7*

Stage1-3 describes the procedure and operations of the challenger **E**. It remains to show that 1-BDHI assumption (Boneh and Boyen, 2004) is satisfied in ***Game7***. To do so, the first thing is to analyze that the challenger **E** does not stop during the simulation. Therefore, two events are formalized below:

Event13: The challenger **E** does not stop during **Stage2 (Challenge simulation)**.

Event14: The adversary **A** is not able to request either $H^*(e(H(w_{0i}^*), N)^{\psi})$ or $H^*(e(H(w_{1i}^*), N)^{\psi})$.

Claim 13: $Pr[Event13] = 1$

Proof: There is no limitation to illustrate that the system will be terminated by the challenger **E** during **Stage2**. Thus, it is clear that $Pr[Event13] = 1$.

Claim 14: $Pr[\neg Event14] \geq 2\xi$

Proof: If $Event14$ happens, it will show that the bit $j \in \{0,1\}$ pointing out whether the Searchable encryption contains w_{0i} or w_{1i} separates of **A**'s view. Hence, the probability that the adversary **A**'s exporting j^* which satisfies $j = j^*$ is at most $\frac{1}{2}$.

By the concept of Bayes's rule,

$Pr[j = j^*] = Pr[j = j^* | Event14]Pr[Event14] + Pr[j = j^* | Event14]$
$Pr[\neg Event14] \leq Pr[j = j^* | Event14]Pr[Even14] + Pr[\neg Event14] =$
$\frac{1}{2} \bullet Pr[Event14] + Pr[\neg Event14] = \frac{1}{2} + \frac{1}{2} \bullet Pr[\neg Event14]$.

By definition, it should be known that $|Pr[j = j^*] - \frac{1}{2}| \geq \xi$. Then, $\xi \leq Pr[j = j^*] - \frac{1}{2} \leq \frac{1}{2} \bullet Pr[\neg Event14]$. Thus, $Pr[\neg Event14] \geq 2\xi$.

Overall, due to **A** requests either $H^*(e(H(w_{0i}^*), N)^{\psi'})$ or $H^*(e(H(w_{1i}^*), N)^{\psi'})$ being at least 2ξ, the probability that **A** requests $H^*(e(H(w_{ji}^*), N)^{\psi'})$ is at least ξ. However, according to the previous definition that **A** requests at most h^* hash function queries, $\frac{1}{h^*}$ is the probability that the challenger **E** chooses the correct solution. Overall**,** the success probability ξ^* achieved by **E** is $\frac{\xi}{h^*}$, which is negligible.

The proposed scheme above could be regarded as **Trapdoor-IND-CPA** secure in ***Game8*** under the random oracle model, if the BDH assumption (Boneh and Boyen, 2004) is completely difficult.

Game8: Let A suppose to be an untrusted outside attacker.

Consider that the challenger **E** is able to achieve the input $(g, P, G_1, G_T, e, \alpha P, \beta P, \gamma P)$ of BDH assumption (Boneh and Boyen, 2004). **E** sets up the computation of a BDH key $e(P, P)^{\alpha\beta\gamma}$ of αP, βP and γP using **A**'s **IND-CPA** as a goal. Apart from that, **A** requests at most h and h^* hash function queries.

Stage1 (Setup)

E selects $M = \alpha P$, $K = \beta P$ and $N = \gamma P$ in the beginning. Then, the following parameters are returned by **E**, which are the common parameter $(g, P, G_1, G_T, e, H, H^*)$, the server's public PEKS key (cp, M, K), and the receiver's public PEKS key (cp, N). In addition, two specific hash functions H and H^* are randomly selected by **E**.

Stage2 （Trapdoor queries)

If **A** queries a Trapdoor request with a specific keyword-vector $W_i = (w_1, w_2, \ldots w_l)$, **E** will randomly choose $t^* \in Z_P$ and subsequently calculate $Q = e(t^*\beta P, \alpha P)$**.** After that, **E** also computes $T_1, T_2, \ldots, T_l$ in the following:

$T_1 = \gamma H(w_1) \oplus e(\beta P, \alpha P)^{t^*+\gamma}$, $T_2 = \gamma H(w_2) \oplus e(\beta P, \alpha P)^{t^*+\gamma}, \ldots,$ $T_\iota = \gamma H(w_\iota) \oplus e(\beta P, \alpha P)^{t^*+\gamma}$. So, $T_W = (Q, T_1, T_2, \ldots, T_\iota)$. After that, **E** returns T_W to **A**.

Stage3 (Challenge simulation)

A uploads a keyword-vector pair $(W_{0i}^*, \mu_{0i}^*, \nu_{0i}^*, \varepsilon_{0i}^*)$ and $(W_{1i}^*, \mu_{1i}^*, \nu_{1i}^*, \varepsilon_{1i}^*)$ to **E**, where $W_0^* = (w_{01}, w_{02}, \ldots, w_{0\iota})$ and $W_1^* = (w_{11}, w_{12}, \ldots, w_{1\iota})$. Once **E** obtains the pair, he/she will do the following steps:

— The challenger **E** randomly chooses $\delta^* \in \{0,1\}$.

—The challenger **E** recalls the *Trapdoor* algorithm for searching the Challenge Trapdoor $T_W^* = (Q^*, T_1^*, T_2^*, \ldots, T_\iota^*) = (e(t^*\beta P, \alpha P), B_1, B_2, \ldots, B_\iota)$.

So,

$$T_1 = \gamma H(w_{\delta_1^*}) \oplus e(\beta P, \alpha P)^{t^*+\gamma} = \gamma H(w_{\delta_1^*}) \oplus e(P, P)^{\alpha\beta\gamma} \bullet e(P, P)^{\alpha\beta t^*},$$

$$T_2 = \gamma H(w_{\delta_2^*}) \oplus e(\beta P, \alpha P)^{t^*+\gamma} = \gamma H(w_{\delta_2^*}) \oplus e(P, P)^{\alpha\beta\gamma} \bullet e(P, P)^{\alpha\beta t^*}, \ldots,$$

$$T_\iota = \gamma H(w_{\delta_\iota^*}) \oplus e(\beta P, \alpha P)^{t^*+\gamma} = \gamma H(w_{\delta_\iota^*}) \oplus e(P, P)^{\alpha\beta\gamma} \bullet e(P, P)^{\alpha\beta t^*}.$$

Stage4 (Trapdoor queries)

E can continue return any Trapdoor query T_W^* for any keyword-vector W^* to **A** as in **Stage2 (*Game8*)**, only if $W^* \neq W_0^*, W_1^*$.

Stage5 (Guess)

A exports $\delta^* \in \{0,1\}$ as the guess. If $\delta = \delta^*$, **E** outputs “yes” and “no” otherwise.

Analysis of *Game8*

According to **A** is an untrusted outside attacker, he/she is not able to observe any difference between two Trapdoor queries even if these two queries contain the same keyword. This is because **E** selects $t^* \in Z_P$ uniformly at random and t^* changes in every calculation so that $T_i = nH(w_i) \oplus e(M, K)^{t^*+n}$ changes in every calculation. Consider two Trapdoor queries contain the same keyword, but the calculation results are different mainly because of the value t^*. Hence, the core part of **Trapdoor-IND-CPA** secure in the proposed scheme is the confidentiality of $e(M, K)^{t^*+n}$.

Consider that if **A** has $e(M, K)^{t^*+n}$, he/she could estimate whether two Trapdoor queries have the same keyword or not. More specially, **A** computes one extra XOR as follows: $T_i = nH(w_i) \oplus e(M, K)^{t^*+n} \oplus e(M, K)^{t^*+n} = nH(w_i)$. So, **A** is able to know that $T_{w_{0i}} = nH(w_{0i})$ and $T_{w_{1i}} = nH(w_{1i})$ are equal, only if $w_0 = w_1$.

By **Stage3** in ***Game8***, it shows that $e(M, K)^{t^*+n} = e(P, P)^{\alpha\beta\gamma} \bullet e(P, P)^{\alpha\beta t^*}$, which meets BDH assumption. Therefore, **A** is not able to computes $e(M, K)^{t^*+n}$ so that he/she cannot calculate $T_i = nH(w_i) \oplus e(M, K)^{t^*+n}$ either.

6.8 The Efficiency and Performance of m-PEMKS

This scheme is implemented by JAVA requiring two libraries: JPBC (Angelo and Vincenzo, 2011) and jFuzzyLogic (Cingolani et al., 2012). The flow chart is described in Figure 30. More specially, sender, receiver and server are implemented by JAVA socket programming. The specific keywords are encrypted by PEKS (JPBC) while the fuzzy keyword is encrypted by RSA (java.security). All encrypted messages are kept in server's file system and the file indexes are stored in Mysql database. Mamdani Fuzzy Inference system is implemented by JAVA Fuzzy Control Language (jFuzzyLogic).

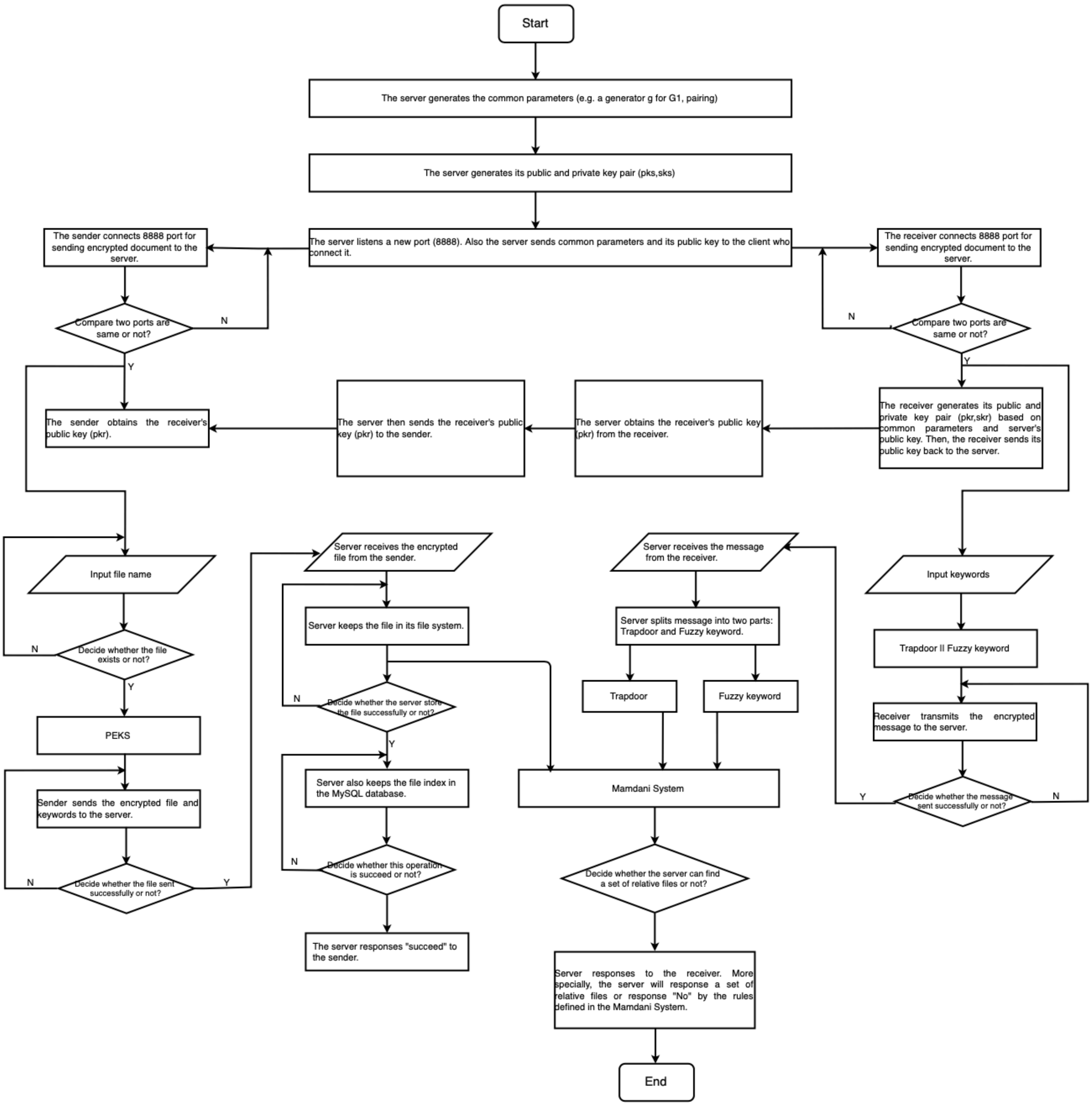

Start
The server generates the common parameters (e.g. a generator g for G1, pairing)
The server generates its public and private key pair (pks,sks)
The server listens a new port (8888). Also the server sends common parameters and its public key to the client who connect it.
The sender connects 8888 port for sending encrypted document to the server.
The receiver connects 8888 port for sending encrypted document to the server.
Compare two ports are same or not?
N
Y
The sender obtains the receiver's public key (pkr).
The server then sends the receiver's public key (pkr) to the sender.
The server obtains the receiver's public key (pkr) from the receiver.
The receiver generates its public and private key pair (pkr,skr) based on common parameters and server's public key. Then, the receiver sends its public key back to the server.
Input file name
Decide whether the file exists or not?
PEKS
Sender sends the encrypted file and keywords to the server.
Decide whether the file sent successfully or not?
Server receives the encrypted file from the sender.
Server keeps the file in its file system.
Decide whether the server store the file successfully or not?
Server also keeps the file index in the MySQL database.
Decide whether this operation is succeed or not?
The server responses "succeed" to the sender.
Server receives the message from the receiver.
Server splits message into two parts: Trapdoor and Fuzzy keyword.
Trapdoor
Fuzzy keyword
Mamdani System
Decide whether the server can find a set of relative files or not?
Server responses to the receiver. More specially, the server will response a set of relative files or response "No" by the rules defined in the Mamdani System.
End
Input keywords
Trapdoor II Fuzzy keyword
Receiver transmits the encrypted message to the server.
Decide whether the message sent successfully or not?

FIGURE 30. FLOW CHART OF M-PEMKS

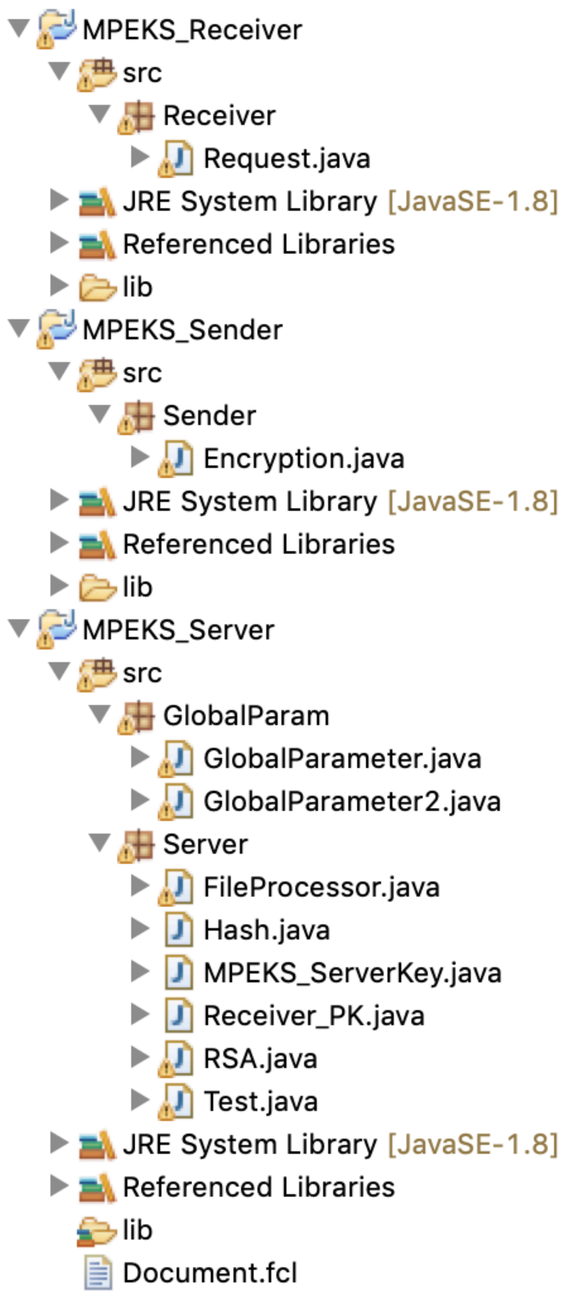
Table 11 below illustrates the simulation platform of *m-PEMKS* scheme. Note that the proposed scheme is programmed by JAVA and JPBC Library (Angelo and Vincenzo, 2011) .

TABLE 11. THE SIMULATION PLATFORM FOR m-PEMKS

OS	macOS Sierra 10.12.5
CPU	2.5 GHz Intel Core i7
Memory	16 GB 1600 MHz DDR3
Hard disk	512GB
Programming language	JAVA

6.9 The Key Code of m-PEMKS

The *m-PEMKS* scheme is programmed by JAVA using JPBC Library (Angelo and Vincenzo, 2011). The pairing parameters are generated by Type A curve. Figure 31 shows all java files used in this proposed scheme.

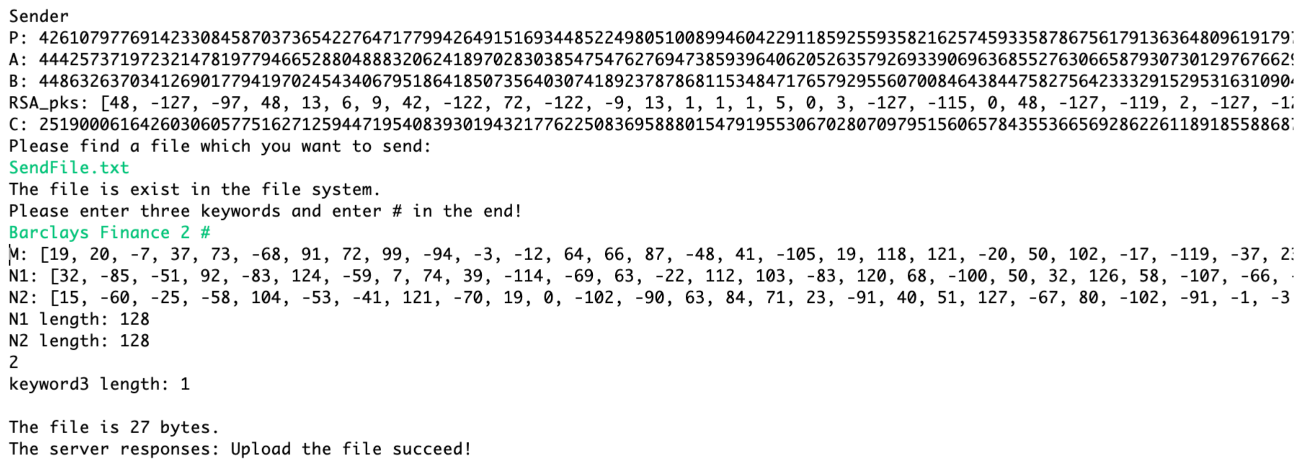


```
▼ MPEKS_Receiver
    ▼ src
        ▼ Receiver
            ▶ Request.java
    ▶ JRE System Library [JavaSE-1.8]
    ▶ Referenced Libraries
    ▶ lib
▼ MPEKS_Sender
    ▼ src
        ▼ Sender
            ▶ Encryption.java
    ▶ JRE System Library [JavaSE-1.8]
    ▶ Referenced Libraries
    ▶ lib
▼ MPEKS_Server
    ▼ src
        ▼ GlobalParam
            ▶ GlobalParameter.java
            ▶ GlobalParameter2.java
        ▼ Server
            ▶ FileProcessor.java
            ▶ Hash.java
            ▶ MPEKS_ServerKey.java
            ▶ Receiver_PK.java
            ▶ RSA.java
            ▶ Test.java
    ▶ JRE System Library [JavaSE-1.8]
    ▶ Referenced Libraries
      lib
      Document.fcl
```

FIGURE 31. JAVA FILES FOR M-PEMKS

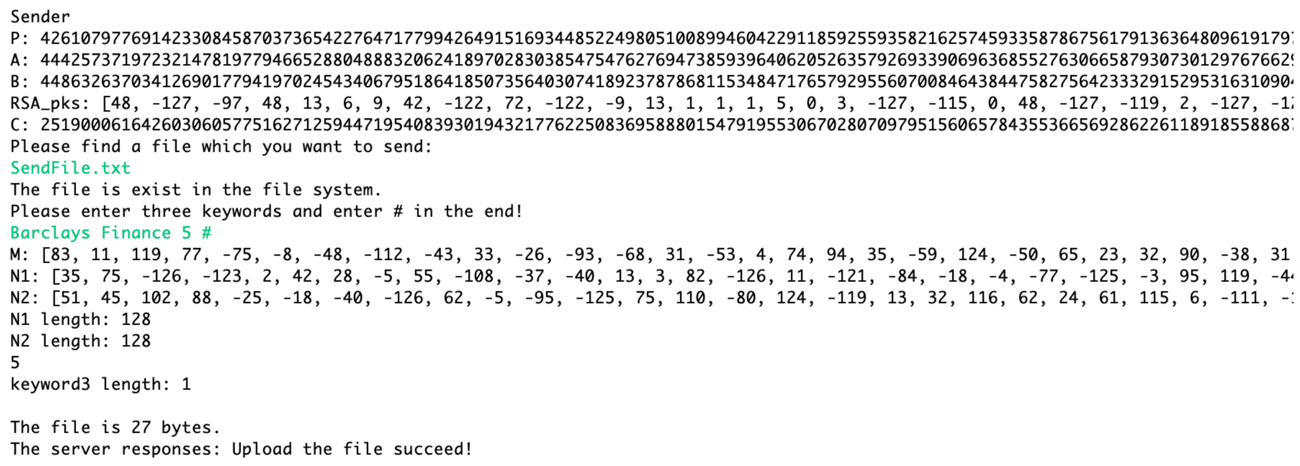
6.9.1 For senders' site

Many senders (employees) wish to send the emails appending with keywords to the receiver (manager). The emails and keywords should be encrypted before sending to the third party. With out loss generality, let the sender's number be three and the keyword's number be three. So, the computer simulation can be found as follows: three senders encrypt the same keywords ("barclays", "finance", a number stands for the date) to the online third party as shown in Figure 32, Figure 33 and Figure 34 respectively. Note that due to the property of Ciphertext Indistinguishability (CI), the encryption results are different even though the keywords are same.



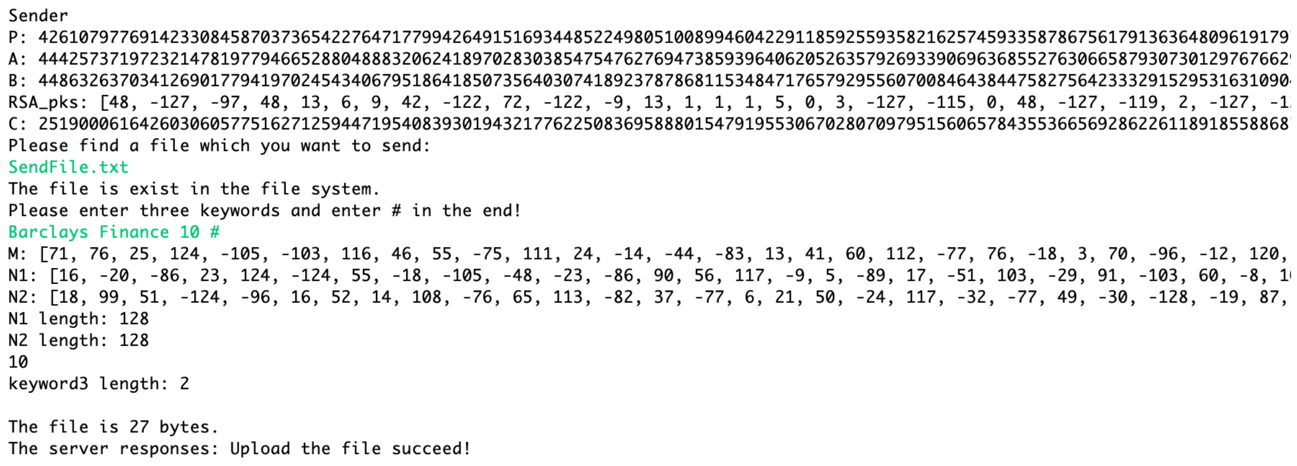
```
Sender
P: 426107977691423308458703736542276471779942649151693448522498051008994604229118592559358216257459335878675617913636480961917
A: 4442573719723214781977946652880488832062418970283038547547627694738593964062052635792693390696368552763066587930730129767662
B: 4486326370341269017794197024543406795186418507356403074189237878681153484717657929556070084643844758275642333291529531631090
RSA_pks: [48, -127, -97, 48, 13, 6, 9, 42, -122, 72, -122, -9, 13, 1, 1, 1, 5, 0, 3, -127, -115, 0, 48, -127, -119, 2, -127, -1
C: 2519000616426030605775162712594471954083930194321776225083695888015479195530670280709795156065784355366569286226118918558868
Please find a file which you want to send:
SendFile.txt
The file is exist in the file system.
Please enter three keywords and enter # in the end!
Barclays Finance 2 #
M: [19, 20, -7, 37, 73, -68, 91, 72, 99, -94, -3, -12, 64, 66, 87, -48, 41, -105, 19, 118, 121, -20, 50, 102, -17, -119, -37, 2
N1: [32, -85, -51, 92, -83, 124, -59, 7, 74, 39, -114, -69, 63, -22, 112, 103, -83, 120, 68, -100, 50, 32, 126, 58, -107, -66,
N2: [15, -60, -25, -58, 104, -53, -41, 121, -70, 19, 0, -102, -90, 63, 84, 71, 23, -91, 40, 51, 127, -67, 80, -102, -91, -1, -3
N1 length: 128
N2 length: 128
2
keyword3 length: 1

The file is 27 bytes.
The server responses: Upload the file succeed!
```

FIGURE 32. ENCRYPTION RESULT OF SENDER1 IN M-PEMKS



```
Sender
P: 426107977691423308458703736542276471779942649151693448522498051008994604229118592559358216257459335878675617913636480961917
A: 4442573719723214781977946652880488832062418970283038547547627694738593964062052635792693390696368552763066587930730129767662
B: 4486326370341269017794197024543406795186418507356403074189237878681153484717657929556070084643844758275642333291529531631090
RSA_pks: [48, -127, -97, 48, 13, 6, 9, 42, -122, 72, -122, -9, 13, 1, 1, 1, 5, 0, 3, -127, -115, 0, 48, -127, -119, 2, -127, -1
C: 2519000616426030605775162712594471954083930194321776225083695888015479195530670280709795156065784355366569286226118918558868
Please find a file which you want to send:
SendFile.txt
The file is exist in the file system.
Please enter three keywords and enter # in the end!
Barclays Finance 5 #
M: [83, 11, 119, 77, -75, -8, -48, -112, -43, 33, -26, -93, -68, 31, -53, 4, 74, 94, 35, -59, 124, -50, 65, 23, 32, 90, -38, 31
N1: [35, 75, -126, -123, 2, 42, 28, -5, 55, -108, -37, -40, 13, 3, 82, -126, 11, -121, -84, -18, -4, -77, -125, -3, 95, 119, -4
N2: [51, 45, 102, 88, -25, -18, -40, -126, 62, -5, -95, -125, 75, 110, -80, 124, -119, 13, 32, 116, 62, 24, 61, 115, 6, -111, -
N1 length: 128
N2 length: 128
5
keyword3 length: 1

The file is 27 bytes.
The server responses: Upload the file succeed!
```

FIGURE 33. ENCRYPTION RESULT OF SENDER2 IN M-PEMKS

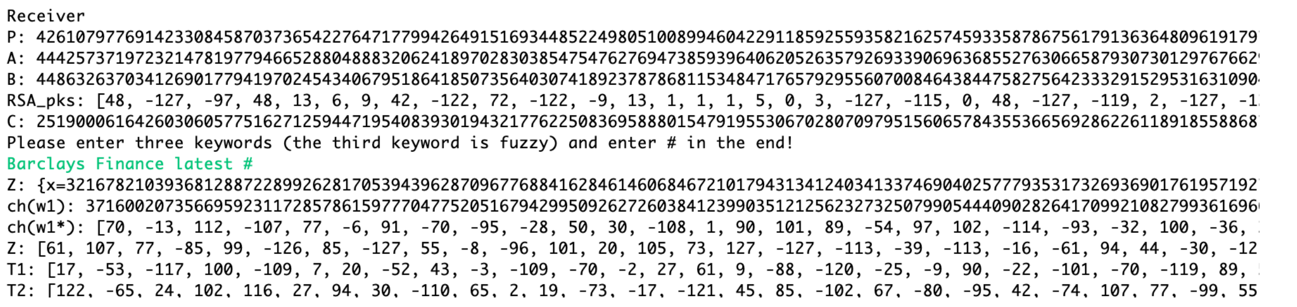


```
Sender
P: 42610797769142330845870373654227647177994264915169344852249805100899460422911859255935821625745933587867561791363648096191 79
A: 44425737197232147819779466528804888320624189702830385475476276947385939640620526357926933906963685527630665879307301297676 62
B: 44863263703412690177941970245434067951864185073564030741892378786811534847176579295560700846438447582756423332915295316310 90
RSA_pks: [48, -127, -97, 48, 13, 6, 9, 42, -122, 72, -122, -9, 13, 1, 1, 1, 5, 0, 3, -127, -115, 0, 48, -127, -119, 2, -127, -1
C: 25190006164260306057751627125944719540839301943217762250836958880154791955306702807097951560657843553665692862261189185588 68
Please find a file which you want to send:
SendFile.txt
The file is exist in the file system.
Please enter three keywords and enter # in the end!
Barclays Finance 10 #
M: [71, 76, 25, 124, -105, -103, 116, 46, 55, -75, 111, 24, -14, -44, -83, 13, 41, 60, 112, -77, 76, -18, 3, 70, -96, -12, 120,
N1: [16, -20, -86, 23, 124, -124, 55, -18, -105, -48, -23, -86, 90, 56, 117, -9, 5, -89, 17, -51, 103, -29, 91, -103, 60, -8, 1
N2: [18, 99, 51, -124, -96, 16, 52, 14, 108, -76, 65, 113, -82, 37, -77, 6, 21, 50, -24, 117, -32, -77, 49, -30, -128, -19, 87,
N1 length: 128
N2 length: 128
10
keyword3 length: 2

The file is 27 bytes.
The server responses: Upload the file succeed!
```

FIGURE 34. ENCRYPTION RESULT OF SENDER3 IN M-PEMKS

6.9.2 For receiver's site

Later on, if the receiver (manager) wishes to obtain the "latest" emails, he/she should send a Trapdoor request to the third party. The keywords in Trapdoor request should be encrypted by Trapdoor and RSA algorithms (See in Figure 35). For instance, if the manager wishes to obtain "Barclays Bank latest financial statements", he/she will only send the Trapdoor request with three keywords ("barclays", "finance", "latest").

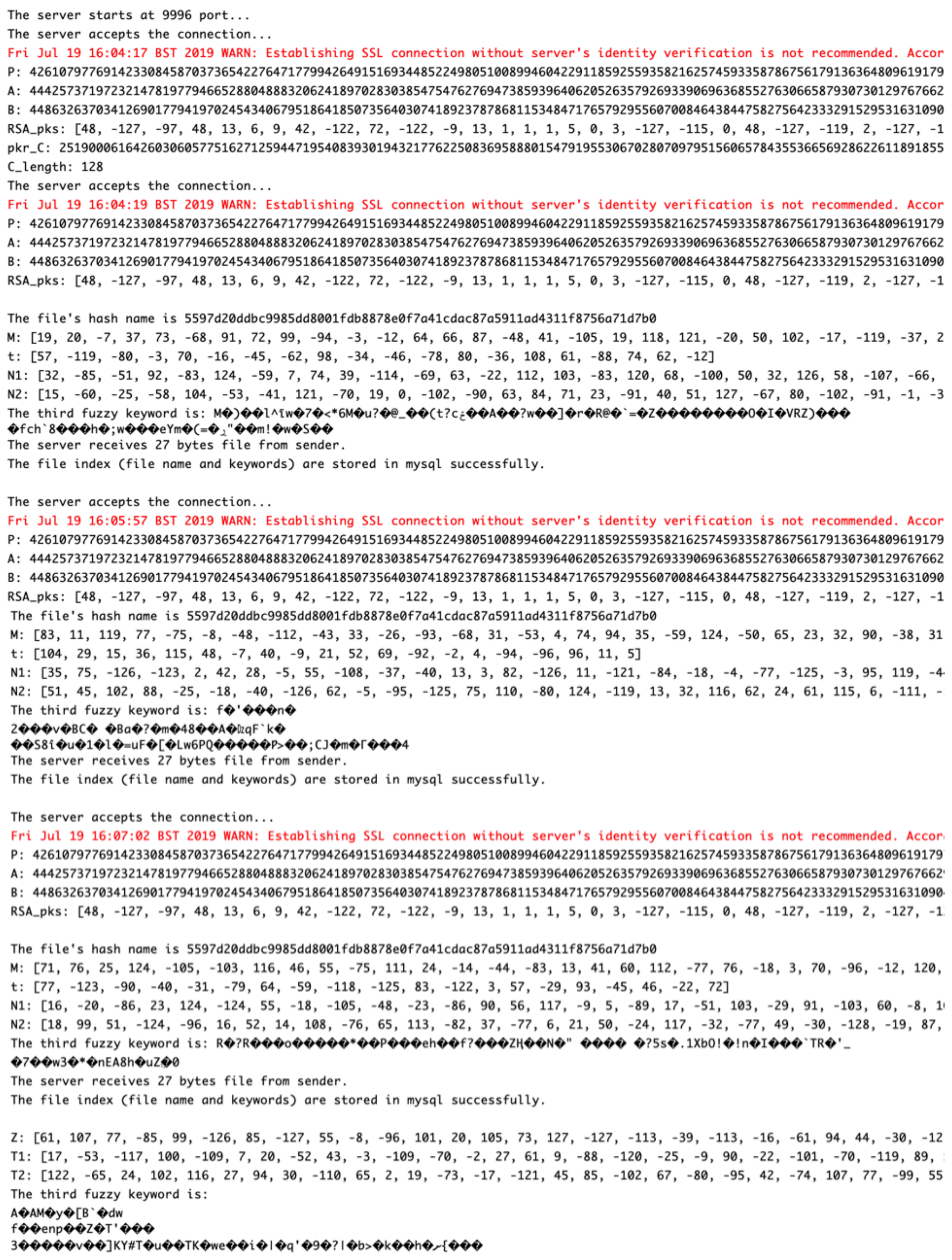


```
Receiver
P: 42610797769142330845870373654227647177994264915169344852249805100899460422911859255935821625745933587867561791363648096191 79
A: 44425737197232147819779466528804888320624189702830385475476276947385939640620526357926933906963685527630665879307301297676 62
B: 44863263703412690177941970245434067951864185073564030741892378786811534847176579295560700846438447582756423332915295316310 90
RSA_pks: [48, -127, -97, 48, 13, 6, 9, 42, -122, 72, -122, -9, 13, 1, 1, 1, 5, 0, 3, -127, -115, 0, 48, -127, -119, 2, -127, -1
C: 25190006164260306057751627125944719540839301943217762250836958880154791955306702807097951560657843553665692862261189185588 68
Please enter three keywords (the third keyword is fuzzy) and enter # in the end!
Barclays Finance latest #
Z: {x=32167821039368128872289926281705394396287096776884162846146068467210179431341240341337469040257779353173269369017619571 92
ch(w1): 37160020735669592311728578615977704775205167942995092627260384123990351212562327325079905444090282641709921082799361696
ch(w1*): [70, -13, 112, -107, 77, -6, 91, -70, -95, -28, 50, 30, -108, 1, 90, 101, 89, -54, 97, 102, -114, -93, -32, 100, -36,
Z: [61, 107, 77, -85, 99, -126, 85, -127, 55, -8, -96, 101, 20, 105, 73, 127, -127, -113, -39, -113, -16, -61, 94, 44, -30, -12
T1: [17, -53, -117, 100, -109, 7, 20, -52, 43, -3, -109, -70, -2, 27, 61, 9, -88, -120, -25, -9, 90, -22, -101, -70, -119, 89,
T2: [122, -65, 24, 102, 116, 27, 94, 30, -110, 65, 2, 19, -73, -17, -121, 45, 85, -102, 67, -80, -95, 42, -74, 107, 77, -99, 55
```

FIGURE 35. REQUEST RESULT OF RECEIVER IN M-PEMKS

6.9.3 For server's site

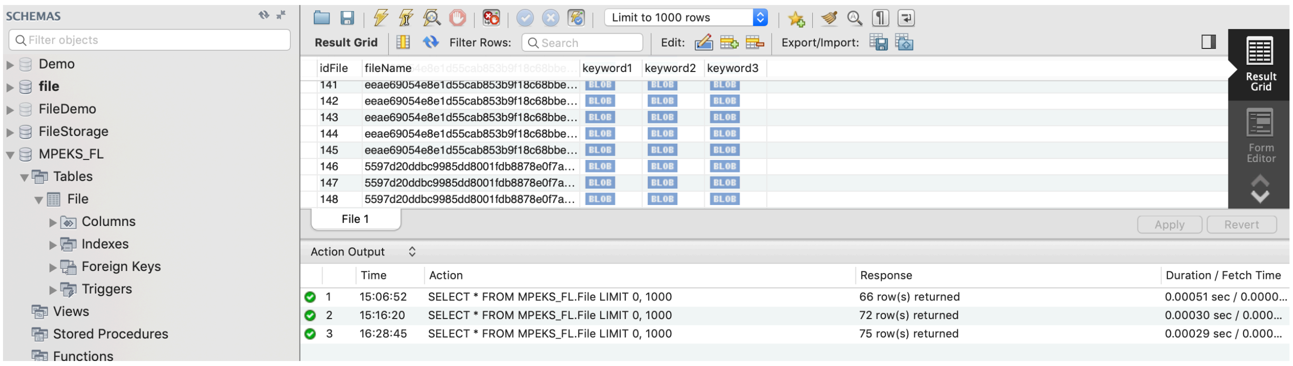
After receiving Searchable ciphertext and Trapdoor request, the server will call Test algorithm to estimate whether they have the same keywords or not. However, due to the server storing millions of encrypted documents, if the receiver wishes to obtains a specific file, it will impossible for the server to decrypt all of the encrypted documents and then compare the keywords both in Searchable ciphertext and Trapdoor request before making a response. In this scheme, the server does not execute decryption operation but only compares the hash values of the results (ciphertext) between PEKS ciphertext and Trapdoor (Figure 36). If matched, the server will run Fuzzy Match algorithm and then reply to the receiver.



```
The server starts at 9996 port...
The server accepts the connection...
Fri Jul 19 16:04:17 BST 2019 WARN: Establishing SSL connection without server's identity verification is not recommended. Accor
P: 4261079776914233084587037365422764717799426491516934485224980510089946042291185925593582162574593358786756179136364809619179
A: 4442573719723214781977946652880488832062418970283038547547627694738593964062052635792693390696368552763066587930730129767662
B: 4486326370341269017794197024543406795186418507356403074189237878681153484717657929556070084643844758275642333291529531631090
RSA_pks: [48, -127, -97, 48, 13, 6, 9, 42, -122, 72, -122, -9, 13, 1, 1, 1, 5, 0, 3, -127, -115, 0, 48, -127, -119, 2, -127, -1
pkr_C: 251900061642603060577516271259447195408393019432177622508369588801547919553067028070979515606578435536656928622611891855
C_length: 128
The server accepts the connection...
Fri Jul 19 16:04:19 BST 2019 WARN: Establishing SSL connection without server's identity verification is not recommended. Accor
P: 4261079776914233084587037365422764717799426491516934485224980510089946042291185925593582162574593358786756179136364809619179
A: 4442573719723214781977946652880488832062418970283038547547627694738593964062052635792693390696368552763066587930730129767662
B: 4486326370341269017794197024543406795186418507356403074189237878681153484717657929556070084643844758275642333291529531631090
RSA_pks: [48, -127, -97, 48, 13, 6, 9, 42, -122, 72, -122, -9, 13, 1, 1, 1, 5, 0, 3, -127, -115, 0, 48, -127, -119, 2, -127, -1

The file's hash name is 5597d20ddbc9985dd8001fdb8878e0f7a41cdac87a5911ad4311f8756a71d7b0
M: [19, 20, -7, 37, 73, -68, 91, 72, 99, -94, -3, -12, 64, 66, 87, -48, 41, -105, 19, 118, 121, -20, 50, 102, -17, -119, -37, 2
t: [57, -119, -80, -3, 70, -16, -45, -62, 98, -34, -46, -78, 80, -36, 108, 61, -88, 74, 62, -12]
N1: [32, -85, -51, 92, -83, 124, -59, 7, 74, 39, -114, -69, 63, -22, 112, 103, -83, 120, 68, -100, 50, 32, 126, 58, -107, -66,
N2: [15, -60, -25, -58, 104, -53, -41, 121, -70, 19, 0, -102, -90, 63, 84, 71, 23, -91, 40, 51, 127, -67, 80, -102, -91, -1, -3
The third fuzzy keyword is: M�)��l^ĩw�7�<*6M�u?�@_��(t?c¿��A��?w��]�r�R@�`=�Z���������O�I�VRZ)���
�fch`8���h�;w���eYm�(=�˛"��m!�w�S��
The server receives 27 bytes file from sender.
The file index (file name and keywords) are stored in mysql successfully.

The server accepts the connection...
Fri Jul 19 16:05:57 BST 2019 WARN: Establishing SSL connection without server's identity verification is not recommended. Accor
P: 4261079776914233084587037365422764717799426491516934485224980510089946042291185925593582162574593358786756179136364809619179
A: 4442573719723214781977946652880488832062418970283038547547627694738593964062052635792693390696368552763066587930730129767662
B: 4486326370341269017794197024543406795186418507356403074189237878681153484717657929556070084643844758275642333291529531631090
RSA_pks: [48, -127, -97, 48, 13, 6, 9, 42, -122, 72, -122, -9, 13, 1, 1, 1, 5, 0, 3, -127, -115, 0, 48, -127, -119, 2, -127, -1
 The file's hash name is 5597d20ddbc9985dd8001fdb8878e0f7a41cdac87a5911ad4311f8756a71d7b0
 M: [83, 11, 119, 77, -75, -8, -48, -112, -43, 33, -26, -93, -68, 31, -53, 4, 74, 94, 35, -59, 124, -50, 65, 23, 32, 90, -38, 31
 t: [104, 29, 15, 36, 115, 48, -7, 40, -9, 21, 52, 69, -92, -2, 4, -94, -96, 96, 11, 5]
 N1: [35, 75, -126, -123, 2, 42, 28, -5, 55, -108, -37, -40, 13, 3, 82, -126, 11, -121, -84, -18, -4, -77, -125, -3, 95, 119, -4
 N2: [51, 45, 102, 88, -25, -18, -40, -126, 62, -5, -95, -125, 75, 110, -80, 124, -119, 13, 32, 116, 62, 24, 61, 115, 6, -111, -
 The third fuzzy keyword is: f�'���n�
 2���v�BC� �Ba�?�m�48��A�␀qF`k�
 ��S8î�u�1�l�=uF�[�Lw6PQ�����P>��;CJ�m�Γ���4
 The server receives 27 bytes file from sender.
 The file index (file name and keywords) are stored in mysql successfully.

 The server accepts the connection...
 Fri Jul 19 16:07:02 BST 2019 WARN: Establishing SSL connection without server's identity verification is not recommended. Accor
 P: 4261079776914233084587037365422764717799426491516934485224980510089946042291185925593582162574593358786756179136364809619179
 A: 4442573719723214781977946652880488832062418970283038547547627694738593964062052635792693390696368552763066587930730129767662
 B: 4486326370341269017794197024543406795186418507356403074189237878681153484717657929556070084643844758275642333291529531631090
 RSA_pks: [48, -127, -97, 48, 13, 6, 9, 42, -122, 72, -122, -9, 13, 1, 1, 1, 5, 0, 3, -127, -115, 0, 48, -127, -119, 2, -127, -1

 The file's hash name is 5597d20ddbc9985dd8001fdb8878e0f7a41cdac87a5911ad4311f8756a71d7b0
 M: [71, 76, 25, 124, -105, -103, 116, 46, 55, -75, 111, 24, -14, -44, -83, 13, 41, 60, 112, -77, 76, -18, 3, 70, -96, -12, 120,
 t: [77, -123, -90, -40, -31, -79, 64, -59, -118, -125, 83, -122, 3, 57, -29, 93, -45, 46, -22, 72]
 N1: [16, -20, -86, 23, 124, -124, 55, -18, -105, -48, -23, -86, 90, 56, 117, -9, 5, -89, 17, -51, 103, -29, 91, -103, 60, -8, 1
 N2: [18, 99, 51, -124, -96, 16, 52, 14, 108, -76, 65, 113, -82, 37, -77, 6, 21, 50, -24, 117, -32, -77, 49, -30, -128, -19, 87,
 The third fuzzy keyword is: R�?R���o�����*��P���eh��f?���ZĦ��N�" ���� �?5s�.1XbO!�!n�I���`TR�'_
 �7��w3�*�nEA8h�uZ�0
 The server receives 27 bytes file from sender.
 The file index (file name and keywords) are stored in mysql successfully.

 Z: [61, 107, 77, -85, 99, -126, 85, -127, 55, -8, -96, 101, 20, 105, 73, 127, -127, -113, -39, -113, -16, -61, 94, 44, -30, -12
 T1: [17, -53, -117, 100, -109, 7, 20, -52, 43, -3, -109, -70, -2, 27, 61, 9, -88, -120, -25, -9, 90, -22, -101, -70, -119, 89,
 T2: [122, -65, 24, 102, 116, 27, 94, 30, -110, 65, 2, 19, -73, -17, -121, 45, 85, -102, 67, -80, -95, 42, -74, 107, 77, -99, 55
 The third fuzzy keyword is:
 A�AM�y�[B`�dw
 f��enp��Z�T'���
 3�����v��]KY#T�u��TK�we��i�|�q'�9�?|�b>�k��h�⌐{���
```

FIGURE 36. PEKS CIPHERTEXT AND TRAPDOOR REQUEST COMPARISON IN M-PEMKS

In addition, the online third party keeps the encrypted messages in its file system and stores the file indexes in the Mysql database (See in Figure 37)



idFile	fileName	keyword1	keyword2	keyword3
141	eeae69054e8e1d55cab853b9f18c68bbe…	BLOB	BLOB	BLOB
142	eeae69054e8e1d55cab853b9f18c68bbe…	BLOB	BLOB	BLOB
143	eeae69054e8e1d55cab853b9f18c68bbe…	BLOB	BLOB	BLOB
144	eeae69054e8e1d55cab853b9f18c68bbe…	BLOB	BLOB	BLOB
145	eeae69054e8e1d55cab853b9f18c68bbe…	BLOB	BLOB	BLOB
146	5597d20ddbc9985dd8001fdb8878e0f7a…	BLOB	BLOB	BLOB
147	5597d20ddbc9985dd8001fdb8878e0f7a…	BLOB	BLOB	BLOB
148	5597d20ddbc9985dd8001fdb8878e0f7a…	BLOB	BLOB	BLOB

		Time	Action	Response	Duration / Fetch Time
✓	1	15:06:52	SELECT * FROM MPEKS_FL.File LIMIT 0, 1000	66 row(s) returned	0.00051 sec / 0.0000…
✓	2	15:16:20	SELECT * FROM MPEKS_FL.File LIMIT 0, 1000	72 row(s) returned	0.00030 sec / 0.000…
✓	3	16:28:45	SELECT * FROM MPEKS_FL.File LIMIT 0, 1000	75 row(s) returned	0.00029 sec / 0.000…

FIGURE 37. THE FILE INDEXES STORING IN MYSQL DATABASE OF M-PEMKS SYSTEM

Mamdani Fuzzy Inference System is the key tool in Fuzzy Match, which could be regarded as a router to filter irrelative documents. The example of Mamdani Fuzzy Inference System using Fuzzy Control Language (FCL) which calculates the assessed value by DATE is implemented by jFuzzyLogic. Figure 38 points out that the trapezoidal and triangular membership functions are applied for fuzzification in m-PEMKS system while the defuzzification is defined by triangular membership function only and the Center of Gravity (COG) is selected as defuzzification method. Meanwhile, Figure 39 illustrates the corresponding JAVA code to execute FCL code. In addition, Figure 40 describes the membership functions of Inputs and Outputs for this example.

```
FUNCTION_BLOCK Document

VAR_INPUT               // Define input variables
    DATE : REAL;
END_VAR

VAR_OUTPUT              // Define output variable
    assess : REAL;
END_VAR

FUZZIFY DATE            // Fuzzify input variable 'DATE': {'old', 'acceptable', 'new' }
    TERM old := (0, 1) (4, 0) ;
    TERM acceptable := (1, 0) (4,1) (6,1) (9,0);
    TERM new := (6, 0) (9, 1);
END_FUZZIFY

DEFUZZIFY assess            // Defzzzify output variable 'access' : {'oldest', 'average', 'latest' }
    TERM oldest := (0,0) (5,1) (10,0);
    TERM average := (10,0) (15,1) (20,0);
    TERM latest := (20,0) (25,1) (30,0);
    METHOD : COG;       // Use 'Center Of Gravity' defuzzification method
    DEFAULT := 0;       // Default value is 0 (if no rule activates defuzzifier)
END_DEFUZZIFY

RULEBLOCK No1
    AND : MIN;          // Use 'min' for 'and' (also implicit use 'max' for 'or' to fulfill DeMorgan's Law)
    ACT : MIN;          // Use 'min' activation method
    ACCU : MAX;         // Use 'max' accumulation method

    RULE 1 : IF DATE IS old THEN assess IS oldest;
    RULE 2 : IF DATE IS acceptable THEN assess IS average;
    RULE 3 : IF DATE IS new THEN assess IS latest;
END_RULEBLOCK

END_FUNCTION_BLOCK
```

FIGURE 38. FCL CODE IN M-PEMKS

```
// Fuzzy Logic || Mamdani system
String fileName = "Document.fcl";
FIS fis = FIS.load(fileName, true); // Load from 'FCL'
                                    // file
if (fis == null) { // Error while loading?
    System.err.println("Can't load file: '" + fileName + "'");
    return;
}

// Show ruleset
FunctionBlock functionBlock = fis.getFunctionBlock(null);
JFuzzyChart.get().chart(functionBlock);
// Set inputs
functionBlock.setVariable("DATE", Integer.parseInt(new String(decrypted, 0, decrypted.length)));
// Evaluate
functionBlock.evaluate();
// Show output variable's chart
Variable assess = functionBlock.getVariable("assess");
JFuzzyChart.get().chart(assess, assess.getDefuzzifier(), true);
// functionBlock.getVariable("tip").defuzzify();
// Print ruleSet
System.out.println(functionBlock);
System.out.println("ASSESS:" + functionBlock.getVariable("assess").getValue());
```

FIGURE 39. JAVA API TO EXECUTE FCL CODE IN M-PEMKS

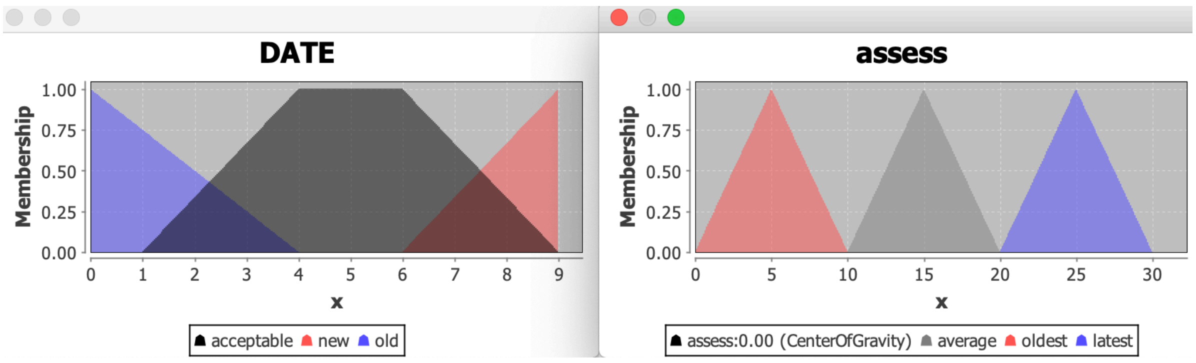


FIGURE 40. MEMBERSHIP FUNCTIONS OF INPUTS AND OUTPUTS FOR M-PEMKS SYSTEM

It is apparent that the proposed scheme applies the Single Input Single Output (SISO) Mamdani Fuzzy Inference System. The reason is due to the properties of Artificial Intelligence and Cryptography. Artificial Intelligence explores and analyzes the data for discovering the relationships between the different data sets. On the contrary, the purpose of cryptography is hiding information as much as possible. In addition, the input value of Mamdani system is plaintext. Therefore, if m-PEMKS applies Two or More Input Single Out (T/MISO) Mamdani Fuzzy Inference System, sufficient information will be exposed to the general public network so that crackers may break the ciphertext to some extent. However, SISO Mamdani system has less accuracy than T/MISO Mamdani system. In order to reverse low accuracy problem, the proposed system will firstly execute Searchable Match and then execute Fuzzy Match. More specially, the server will select the encrypted emails containing “barclays” and “finance” keywords by Exact Match. Then, the server will decrypt the keyword of “DATE” (in PEKS ciphertext) and the keyword “latest” (in Trapdoor). Note that “DATE” and “latest” are the condition and the conclusion of the rules in Mamdani System respectively. Finally, the server will execute Fuzzy Match to distill the most related documents via the assessed values by SISO Mamdani Fuzzy Inference System and reply to the receiver in the end.

In Figure 41, it is obvious that Mamdani system will calculate an assessed value for each input. More specifically, the third keyword of sender1 is “2” and the assessed value is 9.26. According to Figure 38, the value “2” contains two portions, which partly belongs to “old” and “acceptable”. However, the value “2” takes more percentage in “old” part than “acceptable” part. Therefore, the trapezium of “old” in Figure 39 is bigger than the trapezium of “acceptable”. Similarly, the third keyword of sender2 is “5” and the assessed value is 15.00, which belongs to “acceptable” part. In terms of the sender3, the third keyword is “10” and the assessed value is 25.00, which fully belongs to “new” part.

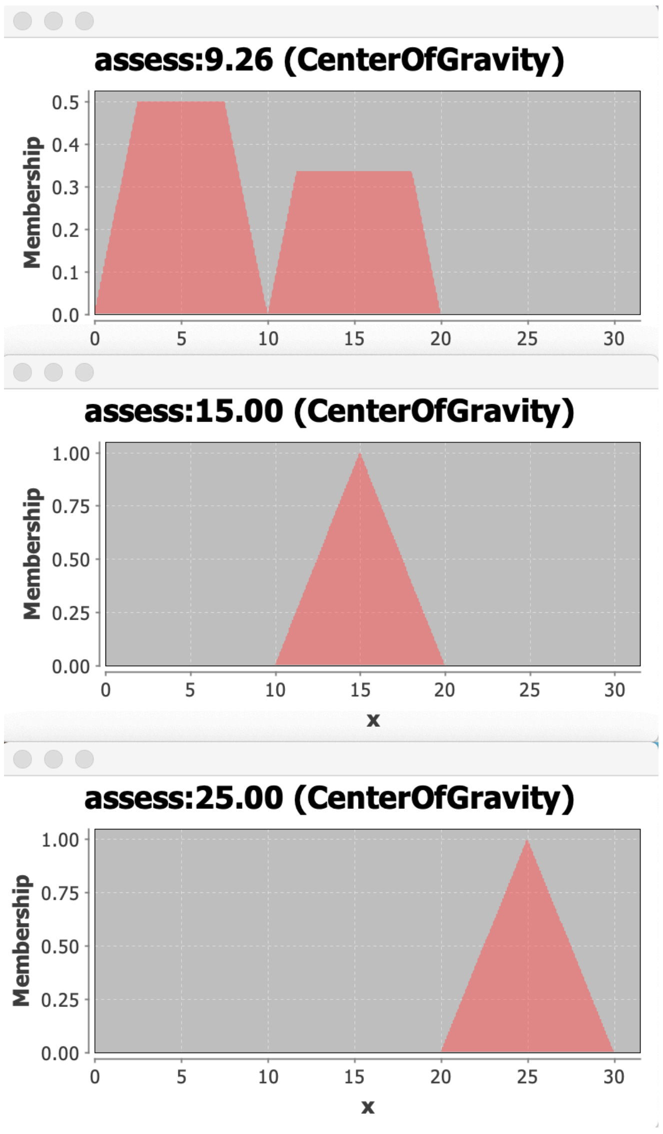


FIGURE 41. THE ASSESSED VALUES OF THREE DIFFERENT DATE INPUT IN M-PEMKS

6.10 The Comparison between Three Proposed Schemes

Table 12 provides the comparison of security and functionality between three proposed MPEKS schemes.

TABLE 12. A COMPARISON BETWEEN THREE PROPOSED SCHEMES

Scheme	Secure Channel	Ciphertext Indistinguishability	Trapdoor Indistinguishability	User Authentication	Off-line KGA	Inside KGA	Fuzzy Keyword Seach
tSCF-MPEKS	No	Yes	Yes	No	Yes	No	No
rSCF-MPEKS	No	Yes	Yes	Yes	Yes	Yes	No
m-PEMKS	No	Yes	Yes	No	Yes	No	Yes

It can be seen that all of these three proposed schemes do not rely on a secure channel to transmit the trapdoor queries. Apart from that, all of these three proposed schemes satisfy the properties of Ciphertext Indistinguishability and Trapdoor Indistinguishability so that they have an ability to resist Off-line Keyword Guessing Attack. However, rSCF-MPEKS scheme incorporates with User Authentication technique and therefore, it could prevent Inside Keyword Guessing Attack. Last but not least, m-PEMKS scheme applies Fuzzy Logic technique, which is able to solve the fuzzy and imprecise keyword search, such as “latest”, etc.

To conclude, these three proposed schemes are much secure and strengthen and have powerful functionalities comparing with theirs counterparts.

7. Conclusion

Public Key Encryption with Keyword Search (PEKS) is one of the most powerful crypto-systems to solve Single Keyword Search problem. Compared with the traditional Public Key Infrastructure (PKI), PEKS based on Identity Based Encryption (IBE) is independent of an online trusted third party (such as Certificate Authority) to authorize the public key.

Although PEKS carries out a lot of merits, it should not be overlooked that PEKS has its weaknesses. Firstly, the original PEKS schemes require secure channels between the sever and the receiver to transmit Trapdoor queries. However, building secure channel consumes huge human and material resources and seems impossible in some cases. Secondly, many PEKS schemes are able to solve Single Keyword Search problem but do not support Multiple Keywords Search and therefore, these PEKS approaches may not be applied to the general public networks. Last but not least, due to the online third party and/or the receiver in PEKS system may honest but curious, he/she may release the private key to the public networks so that the PEKS schemes could suffer Off-line Keyword Guessing Attack (OKGA). Although the later PEKS systems incorporate with Trapdoor indistinguishability to resist OKGA, their security still need to improve. For instance, almost all current PEKS schemes are vulnerable to Inside Keyword Guessing Attack (IKGA), etc.

This PhD thesis concentrates on proposing three secure and efficient PEKS schemes to solve both Single and Multiple Keyword(s) Search problems, and also resist OKGA and/or IKGA.

Many current Public Key Encryption with Multiple Keywords Search (MPEKS) schemes suffers OKGA. Therefore, the thesis firstly defines a MPEKS scheme incorporating with Trapdoor indistinguishability to resist OKGA, which is called *"Trapdoor-indistinguishable Secure Channel Free Public Key Encryption with Multi-keywords Search (tSCF-MPEKS)"*. More specially, the proposed scheme is proved to be semantic secure under the Random Oracles Models with BDH and 1-BDHI assumptions so that it is able to resist OKGA. Besides, it has the ability to address both Single and Multiple Keyword(s) Search problems. Comparing with its counterparts, the efficiency and performance of tSCF-MPEKS scheme are affordable by the mathematical calculation and the computer simulation.

Secondly, IKGA in MPEKS schemes is still an intractable problem up to now. The research then defines the strengthen and powerful MPEKS scheme called *"Robust Secure Channel Free Public Key Encryption with Multi-keywords Search (rSCF-MPEKS)"* to prevent IKGA. More specially, the *r*SCF-MPEKS system has the characters of Ciphertext Indistinguishability and Trapdoor Indistinguishability and also incorporates with User Authentication technique so that it is not only able to resist OKGA but also prevents IKGA. In addition, rSCF-MPEKS scheme has the ability to solve both Single and Multiple Keyword(s) Search problems. Comparing with some typical MPEKS schemes (such as MPEKS and SCF-MPEKS, etc.), the rSCF-MPEKS approach is much more secure and also has high efficiency and better performance as well.

Last but not least, almost all current PEKS and MPEKS schemes cannot deal with imprecise keywords, such as "latest", "newest", etc. For instance, if the keyword is fuzzy (i.e. "latest, biggest"), these current PEKS/MPEKS schemes will be terminated and report errors. Therefore, the research formalizes the third MPEKS statement, namely *"Public Key Encryption with Multi-keywords Search using Mamdani System (m-PEMKS)"*, to address Fuzzy Keyword Search problem. More specially, the proposed

MPEKS scheme applies Mamdani Fuzzy Inference System (Fuzzy Logic) in Artificial Intelligence to solve Fuzzy Keyword Search problem. The m-PEMKS scheme is verified to be semantic secure under the Random Oracles Models with BDH and 1-BDHI assumptions and therefore, it is also able to resists OKGA.

Furthermore, the performance and efficiency of the proposed schemes are analyzed by the theoretical analysis based on mathematical calculations and the practical analysis based on programming with JAVA, JPBC Library and jFuzzylogic Library. For practical analysis, the proposed approaches are called by 1000 times computer simulations and every 100 times computer simulations is considered to be one round. To conclude, these proposed schemes consume less computing time and resources and have better performance and functionalities comparing with theirs counterparts.

References

En.wikipedia.org. (2019). *Scytale*. [online] Available at: https://en.wikipedia.org/wiki/Scytale [Accessed 20 Jun. 2019].

En.wikipedia.org. (2019). *Enigma machine*. [online] Available at: https://en.wikipedia.org/wiki/Enigma_machine [Accessed 20 Jun. 2019].

En.wikipedia.org. (2019). *Bombe*. [online] Available at: https://en.wikipedia.org/wiki/Bombe [Accessed 20 Jun. 2019].

A. Shamir. Identity-based Crypto-systems and Signature Schemes. In: G. R. Blakley, D.C. Chaum (ed.), Advances in Cryptology-Proceedings of CRYPTO'84. California: Springer-Verlag, LNCS, Vol. 196, 1985. 48~53.

Boneh, D., Di Crescenzo, G., Ostrovsky, R. and Persiano, G.: Public Key Encryption with Keyword Search, Advances in Cryptology, EUROCRYPT 2004, vol 3024, 506-522 (2004).

Koblitz, Neal; Menezes, Alfred (2005). Pairing-Based Cryptography at High Security Levels. *LNCS*. 3796.

Baek, J., Safavi-Naini, R. and Susilo, W.: Public Key Encryption with Keyword Search Revisited, Computational Science and Its Applications, ICCSA 2008, vol 5072, 1249-1259 (2008).

Rhee, H., Park, J., Susilo, W. and Lee, D.: Trapdoor Security in a Searchable Public-key Encryption Scheme with a Designated Tester, Journal of Systems and Software, vol 83(5), 763-771 (2010).

Zhao, Y. J., Chen, X. F., Ma, H., Tang, Q., and Zhu, H.: A New Trapdoor indistinguishable Public Key Encryption with Keyword Search, Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, vol 3, 72-81

(2013).

Wang, T., Au, M. andWu,W.: An Efficient Secure Channel Free Searchable Encryption Scheme with Multiple Keywords, Network and System Security, v9955, 251-265 (2016).

Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P. and Shi, H.: Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions, Advances in Cryptology, CRYPTO 2005, vol 3621, 205-222 (2005).

Byun J.W., Rhee H.S., Park HA., Lee D.H.: Off-Line Keyword Guessing Attacks on Recent Keyword Search Schemes over Encrypted Data, Secure Data Management 2006, vol 4165, 75-83 (2006).

Yau, W., Heng, S. and Goi, B. (n.d.). Off-Line Keyword Guessing Attacks on Recent Public Key Encryption with Keyword Search Schemes. Lecture Notes in Computer Science, pp.100-105.

Huang, Q. and Li, H. (2017). An Efficient Public-key Searchable Encryption Scheme Secure against Inside Keyword Guessing Attacks. Information Sciences, 403-404, pp.1-14.

Jeong, I., Kwon, J., Hong, D. and Lee, D. (2009). Constructing PEKS Schemes Secure against Keyword Guessing Attacks is Possible? Computer Communications, 32(2), pp.394-396.

Ma, Y. and Kazemian, H. (2018). Trapdoor-indistinguishable Secure Channel Free Public Key Encryption with Multi-Keywords Search (Student Contributions). Proceedings of the 11th International Conference on Security of Information and Networks - SIN'18.

Tang, Q. and Chen, L.: Public-Key Encryption with Registered Keyword Search, Public Key Infrastructures, Services and Applications, EuroPKI 2009, vol 6391, 163-178 (2010).

Zhang, X., Xu, C., Xie, R. and Jin, C. (2018). Designated Cloud Server Public Key Encryption with Keyword Search from Lattice in the Standard Model. Chinese Journal of Electronics, 27(2), pp.304-309.

Li, J., Wang, Q., Wang, C., Cao, N., Ren, K. and Lou, W.: Fuzzy Keyword Search over Encrypted Data in Cloud Computing, 2010 Proceedings IEEE INFOCOM, 257-266 (2010).

Xu, P., Jin, H., Wu, Q. and Wang, W.: Public-Key Encryption with Fuzzy Keyword Search: A Provably Secure Scheme under Keyword Guessing Attack, IEEE Transactions on Computers, vol 62(11), 2266-2277 (2013).

Ibraimi, L., Nikova, S., Hartel, P. and Jonker, W. (2011). Public-Key Encryption with Delegated Search. Applied Cryptography and Network Security, pp.532-549.

Chen, R.,Mu, Y., Yang, G., Guo, F. andWang, X. (2015). Dual-Server Public-Key Encryption with Keyword Search for Secure Cloud Storage. IEEE Transactions on Information Forensics and Security, pp.1-1.

He, Z., Cai, Z., Han, Q., Tong, W., Sun, L. and Li, Y. (2016). An Energy Efficient Privacy-preserving Content Sharing Scheme in Mobile Social Networks. Personal and Ubiquitous Computing, 20(5), pp.833-846.

Angelo De Caro and Vincenzo Iovino. 2011. jPBC: Java Pairing Based Cryptography. In Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011. Kerkyra, Corfu, Greece, June 28 - July 1, 850–855.

Huang, Q. and Li, H. (2017). An Efficient Public-key Searchable Encryption Scheme Secure against Inside Keyword Guessing Attacks. *Information Sciences*, 403-404, pp.1-14.

Goldwasser, S. and Micali, S. (1984). Probabilistic Encryption. *Journal of Computer and System Sciences*, 28(2), pp.270-299.

Moni Naor and Moti Yung. Public-key Crypto-systems Provably Secure against Chosen Ciphertext Attacks. *Proceedings 21st Annual ACM Symposium on Theory of Computing*: 427–437. 1990.

Rackoff, C. and D. Simon (1991). Non-interactive Zero-knowledge Proof of Knowledge and Chosen Ciphertext Attack. *Advances in Cryptology—CRYPTO'91*, Lecture Notes in Computer Science, vol. 576, ed. J. Feigenbaum. Springer-Verlag, Berlin, 433–444.

X. Boyen and B. Waters. Anonymous Hierarchical Identity-Based Encryption. In:Dwork, Cynthia (ed.), Advances in Cryptology-Crypto 2006. California:Springer-Verlag, LNCS, Vol. 4117, 2006. 290~307.

D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. In: J.Kilian (ed.), Advances in Cryptology-Crypto 2001. California: Springer-Verlag, LNCS,Vol.2139, 2001. 231~229.

S. Al-Riyami, K. Paterson. Certificateless Public Key Cryptography. In: Chi-Sung Laih (ed.), Advances in Cryptology-Asiacrypt’2003. Taiwan: Springer-Verlag, LNCS, Vol. 2332, 2003. 452-473.

V. Goyal. Reducing Trust in the PKG in Identity-Based Cryptosystems. In: A. Menezes (ed.), Advances in Cryptology-Crypto 2007. California: Springer-Verlag, LNCS, Vol. 4622, 2007. 430~447.

V. Goyal. Reducing Trust in the PKG in Identity-Based Cryptosystems. In: A. Menezes (ed.), Advances in Cryptology-Crypto 2007. California: Springer-Verlag, LNCS, Vol.4622, 2007. 430~447.

Felix Brandt and Tuomas Sandholm. Efficient Privacy-Preserving Protocols for

Multi-unit Auctions. In A.S. Patrick and M. Yung (ed.), FC 2005, LNCS, Vol. 3570, Springer-Verlag, 2005. 298~312.

M. R. Clarkson, S. Chong and A.C. Myers. Civitas: Toward a Secure Voting System. In Proceeding of SP 2008. 354~368.

D. Boneh and X. Boyen. Secure Identity Based Encryption Without Random Oracles. In: M. K. Franklin (ed.), Advances in Cryptology-Crypto 2004. California: Springer-Verlag, LNCS, Vol. 3152, 2004. 443~459.

D. Boneh and X. Boyen. Efficient Selective-ID Identity Based Encryption Without Random Oracles. In: C. Cachin, J. Camenisch (ed.), Advances in Cryptology EUROCRYPT'2004. Switzerland: Springer-Verlag, LNCS, Vol. 3027, 2004. 223~238.

B. Waters. Efficient Identity-Based Encryption Without Random Oracles. In: R.Cramer (ed.), Advances in Cryptology-EUROCRYPT'2005. Denmark: Springer-Verlag, LNCS, Vol. 3494, 2005. 114~127.

C.Gentry. Practical Identity-Based Encryption Without Random Oracles. In: S. Vaudenay (ed.), Advances in Cryptology-EUROCRYPT'2006. Russia: Springer-Verlag, LNCS, Vol. 4004, 2006. 445~464.

W. C. Yau, S. H. Heng, and B. M. Goi. Off-line Keyword Guessing Attacks on Recent Public Key Encryption with Keyword Search Schemes. in Autonomic and Trusted Computing, vol. 5060 of Lecture Notes in Computer Science, pp. 100–105, Oslo, Norway, 2008. Springer Berlin/ Heidelberg.

C. Hu and P. Liu. A Secure Searchable Public Key Encryption Scheme with a Designated Tester against Keyword Guessing Attacks and its Extension. In Advances in Computer Science, Environment, Ecoinformatics, and Education, vol. 215 of Communications in Computer and Information Science, pp. 131– 136, Wuhan, China, 2011. Springer Berlin/ Heidelberg.

Chen, Y. (2014). SPEKS: Secure Server-Designation Public Key Encryption with Keyword Search against Keyword Guessing Attacks. *The Computer Journal*, 58(4), pp. 922-933.

Sun, L., Xu, C., Zhang, M., Chen, K. and Li, H. (2017). Secure Searchable Public Key Encryption against Inside Keyword Guessing Attacks from Indistinguishability Obfuscation. *Science China Information Sciences*, 61(3).

Mao, Y., Fu, X., Guo, C. and Wu, G. (2018). Public Key Encryption with Conjunctive Keyword Search Secure against Keyword Guessing Attack from Lattices. *Transactions on Emerging Telecommunications Technologies*, p.e3531.

Noroozi, M. and Eslami, Z. (2019). Public Key Encryption with Keyword Search: A Generic Construction Secure against Online and Offline Keyword Guessing Attacks. *Journal of Ambient Intelligence and Humanized Computing*.

P. W. Shor. Algorithms for Quantum Computation: Discrete Logarithms and Factoring. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science (SFCS '94)*, pp. 124–134, IEEE, 1994.

M. Ajtai. Generating Hard Instances of Lattice Problems (extended abstract). In STOC '96: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, pages 99-108, New York, NY, USA, 1996. ACM.

C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for Hard Lattices and New Cryptographic Constructions. In Richard E. Ladner and Cynthia Dwork, editors, STOC, pages 197-206. ACM, 2008.

S. Agrawal, D. Boneh, and X. Boyen. Efficient Lattice HIBE in the Standard Model. In Advances in EUROCRYPT 2010, volume 6110 of LNCS, pages 553-572. Springer, 2010.

S. Agrawal, D. Boneh, and X. Boyen. Lattice Basis Delegation in Fixed

Dimension and Shorter Ciphertext Hierarchical IBE. In Advances in CRYPTO 2010, volume 6223 of LNCS, pages 98-115. Springer, 2010.

D. Micciancio. Generalized Compact Knapsacks, Cyclic Lattices and Efficient One-way Functions from Worst-case Complexity Assumptions. In FOCS, pages 356-365, 2002.

C. Gentry. Fully Homomorphic Encryption using Ideal Lattices. In Proceedings of the 41th Annual ACM Symposium on Theory of Computing (STOC '09), pp. 169–178, ACM, 2009.

C. Gentry. Toward Basing Fully Homomorphic Encryption on Worst-case Hardness. In *Annual Cryptology Conference: CRYPTO 2010: Advances in Cryptology—CRYPTO 2010*, vol. 6223 of *Lecture Notes in Computer Science*, pp. 116–137, Springer, Berlin, Germany, 2010.

C. Gu, Y. Guang, Y. Zhu, and Y. Zheng. Public Key Encryption with Keyword Search from Lattices. *International Journal of Information Technology*, vol. 19, no. 1, 2013.

C. Hou, F. Liu, H. Bai, and L. Ren. Public Key Encryption with Keyword Search from Lattice. In Proceedings of the 8th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC '13), pp. 336–339, IEEE, Compiegne, France, October 2013.

P. Golle, J. Staddon, and B. Waters. Secure Conjunctive Keyword Search over Encrypted Data. In Applied Cryptography and Network Security: Second International Conference, ACNS 2004, Yellow Mountain, China, June 8–11, 2004. Proceedings, vol. 3089 of Lecture Notes in Computer Science, pp. 31–45, Springer, Berlin, Germany, 2004.

D. Boneh and B. Waters. Conjunctive, Subset, and Range Queries on Encrypted

Data. In *Theory of Cryptography Conference: TCC 2007: Theory of Cryptography*, vol. 4392 of *Lecture Notes in Computer Science*, pp. 535–554, Springer, Berlin, Germany, 2007.

J. Camenisch, M. Kohlweiss, A. Rial, and C. Sheedy. Blind and Anonymous Identity-based Encryption and Authorised Private Searches on Public Key Encrypted Data. In *Public Key Cryptography-PKC 2009*, pp. 196–214, Springer, Berlin, Germany, 2009.

N. Cao, C. Wang, M. Li, K. Ren, and W. Lou. Privacy-preserving Multi-keyword Ranked Search over Encrypted Cloud Data. *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 1, pp. 222–233, 2014.

Cryptowiki.net. (2019). *Cipher Block Chaining (CBC) - CryptoWiki*. [online] Available at: http://cryptowiki.net/index.php?title=Cipher_Block_Chaining_(CBC) [Accessed 5 Aug. 2019].

Stallings, W. (2017). Cryptography and Network Security. Boston, Mass: Pearson.

Angelo De Caro and Vincenzo Iovino. 2011. jPBC: Java Pairing Based Cryptography. In Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011. Kerkyra, Corfu, Greece, June 28 - July 1, 850–855.

Zadeh, L. (1973). Outline of a New Approach to the Analysis of Complex Systems and Decision Processes. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3(1), pp.28-44.

Mamdani, E.H. and Assilian, S. (1975). An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller. International Journal of Man–Machine Studies, 7(1), 1–13.

Takagi, T.; Sugeno, M. Fuzzy Identification of Systems and its Applications to Modeling and Control. IEEE Trans. Sys. Man. Cybern. 1985, 15, 116–132.

Singh, J., Singh, N. and Sharma, J. K. 2006. Fuzzy Modeling and Identification of Intelligent Control for Refrigeration Compressor. *J. Sci. Ind. Res.,* 65: 22-30.

Lermontov, A.; Yokoyama, L.; Lermontov, M.; Machado, M.A.S. River Quality Analysis using Fuzzy Water Quality Index: Ribeira do Iguape river watershed, Brazil. Ecol. Indic. 2009, 9, 1188–1197.

Marchini, A.; Facchinetti, T.; Mistri, M. F-IND: A Framework to Design Fuzzy Indices of Environmental Conditions. Eco. Indic. 2009, 9, 485–496.

Wang, C., Chew, M. and Harlow, D., 2015. A Study Of Membership Functions On Mamdani-Type Fuzzy Inference System For Industrial Decision-Making.

Appendix

Publication

Ma, Y. and Kazemian, H. (2018). Trapdoor-indistinguishable Secure Channel Free Public Key Encryption with Multi-Keywords Search (Student Contributions). Proceedings of the 11th International Conference on Security of Information and Networks - SIN'18.

Kazemian, H. and Ma, Y. (2020). Fuzzy Logic Application to Searchable Cryptography. Proceedings of the 21st International Conference on Engineering Applications of Neural Networks - EANN 2020.