**Noname manuscript No.**
(will be inserted by the editor)

# Public Key Authenticated Encryption with Multiple Keywords Search using Mamdani System

**Yang Ma · Hassan Kazemian**

Received: date / Accepted: date

**Abstract** The public cloud environment has attracted massive attackers to exploit insecure ports and access to data, services and other resources. Techniques, such as Public Key Encryption with Keyword Search (PEKS), could be deployed in cloud security to avoid accidents. PEKS allows users to search encrypted documents by a specific keyword without compromising the original data security. The first PEKS scheme was proposed in 2004, since then, PEKS has been experienced a great progress. Recently, Kazemian and Ma firstly incorporated with Fuzzy Logic technique to PEKS scheme, namely "*Public Key Encryption with Multi-keywords Search using Mamdani System (m-PEMKS)*", in order to support Fuzzy Keyword (i.e. "latest", "biggest") Search. However, the m-PEMKS scheme has the ability to prevent Off-line Keyword Guessing Attack (OKGA) but it may suffer Inside Keyword Guessing Attack (IKGA). This paper will revisit the m-PEMKS scheme and propose a robust m-PEMKS mechanism. The proposed scheme has the properties of Ciphertext Indistinguishability, Trapdoor Indistinguishability and User Authentication which can prevent OKGA and IKGA. Besides, the proposed scheme supports both Fuzzy Keyword Search and Multiple Keywords Search and therefore, it is more practical and could be applied to the general public networks.

**Keywords** Public Key Encryption with Keyword Search (PEKS) · Off-line Keyword Guessing Attack (OKGA) · Inside Keyword Guessing Attack (IKGA) · User Authentication · Fuzzy Keyword Search · Multiple Keywords Search

Yang Ma
Underwriters Laboratories, UK Security Lab, Basingstoke, United Kingdom
E-mail: Yang.Ma@ul.com

Hassan Kazemian
School of Computing and Digital Media, London Metropolitan University, London, United Kingdom.

## 1 Introduction

Uploading data into the cloud servers reduces the local memory and retrenches the maintenance cost to some extent, but it may present other adverse impacts. For instance, the adversaries may capture or even counterfeit the user's credentials in order to access to the data. Besides, the attackers could capture the power consumption signals and the electromagnetic emission signals and then launch side channel attacks (i.e. Correlation Power Analysis [1]) to exploit the physical leakage. Finally, they are able to perform Template Attack [2,3] and Deep Learning Analysis [4] to recover the secret assets. The cloud security threats are increased alongside with the massive use of cloud applications. Nevertheless, PEKS is one of the most advanced techniques to protect data security and also provide an efficient way to search encrypted documents from the cloud servers.

A PEKS mechanism contains three participants: a sender, a receiver and an online server. More specially, a sender sends an encrypted document appending with a specific keyword such as "Confidential" to the online server (Searchable encryption = $E(M)||PEKS(Confidential, PK_{receiver})$). When the receiver would like to achieve this document, he/she will send a Trapdoor query to the online server (Trapdoor query = $Trapdoor(Confidential, SK_{receiver})$). Once the online server receives the Searchable encryption and the Trapdoor query, it will execute $Test$ algorithm to check whether the keyword "Confidential" in two ciphertexts is same or not. If so, it will transfer the encrypted document to the receiver. PEKS mechanism has many advantages. For instance, the online server is zero knowledge so that it learns nothing related to the keyword and the encrypted document. Also, if the sender encrypts the same keyword twice by PEKS algorithm, he/she will obtain the two different ciphertexts.

Boneh et al. [5] proposed the blueprint of PEKS scheme in 2004. A secure channel, such as SSL, must be built between the online server and the receiver. However, it might be impossible and consumed huge costs to establish a secure channel. Afterwards, Byun et al. [6] firstly found that the Boneh et al.'s PEKS scheme suffered OKGA.

Baek et al. [7] came up with a new PEKS called "*Secure Channel Free Public Key Encryption with Keyword Search (SCF-PEKS)*". It removed the secure channel from the original PEKS scheme in 2008. However, the server may not be honest, so it may extract the secret assets. Yau et al. [8] pointed out that the SCF-PEKS scheme suffered OKGA in the same year. Afterwards, Tang et al. [9] proposed a PEKS scheme to prevent OKGA, but the encryption algorithm was computational complexity and cost. Soon later, Rhee et al. [10] illustrated a "*Secure Searchable Public Key Encryption scheme with a Designated tester (dPEKS)* to solve OKGA problem. But Yau et al. [11] pointed out that the dPEKS scheme still suffered OKGA. Afterwards, Chen [12] revisited the dPEKS scheme and then defined a secure server-designation PEKS (SPEKS) model in order to resist both OKGA and IKGA. But the encryption algorithm of SPEKS scheme was complex. Zhang et al. [13] came up with a new PEKS scheme in 2018 based on lattice assumption in the standard model with quantum computers resistance, which was a new era for PEKS development.

The PEKS schemes above support single keyword search only rather than multiple keywords search. Baek et al. [7] proposed "*Public Key Encryption with Multi-keywords Search (MPEKS)*" in 2008 to solve multiple keywords search problem. But a secure channel must be required between the server and the receiver. In 2016, "*Secure Channel Free MPEKS (SCF-MPEKS)*" was introduced by Wang et al. [14], which removed the secure channel from the Baek et al.'s MPEKS scheme. Latterly, Ma and Kazemian [15] found that the SCF-MPEKS suffered OKGA and then incorporated with the property of Trapdoor Indistinguishability to SCF-MPEKS (called tSCF-MPEKS) to resist OKGA. However, the tSCF-MPEKS would suffer IKGA, if the malicious server published its private key to the general public network.

Many current PEKS schemes could prevent OKGA but they are vulnerable to IKGA. Hence, Li et al. [16] applied session key establishment to PEKS in order to prevent IKGA. Noroozi et al. [17] found that Li et al.'s PEKS scheme still suffered IKGA in 2018. In the same year, Huang and Li [18] defined a new PEKS model to prevent IKGA. The proposed scheme supported single keyword search only instead of multiple keywords search and therefore, it cannot be deployed in practical. In 2020, Ma and Kazemian [19] came up with a new PEKS scheme, which applied User Authentication Technique to resist IKGA. Besides, the proposed scheme allowed users to search encrypted documents by using multiple keywords. However, the proposed scheme would throw errors, if the keyword for searching was a fuzzy keyword (i.e. "latest", "biggest").

Almost all current PEKS schemes cannot support imprecise keyword search. To solve it, Fuzzy Rule based systems could be applied to the PEKS schemes. In 1973, Lotifi Zadeh's [20] proposed a new fuzzy algorithms to analyse complex systems and decision processes. Afterwards, Ebrahim Mamdani [21] defined a Mamdani fuzzy inference system to control a steam engine and boiler combination by a set of linguistic fuzzy rules ascertained from experienced human operators. But Mamdani-style inference is not computationally efficient. Michio Sugeno [22] then camp up with a new fuzzy inference using a singleton as the rule consequent. Nowadays, Fuzzy sets theory has been deployed in many aspects. Singh et al.[23] found that fuzzy systems could be used in classification, modelling control problems. Lermontov et al. [24] evaluated water quality by fuzzy set and Marchini et al. [25] defined a framework for fuzzy indices of environmental conditions. In 2020, Kazemian and Ma [26] proposed a new PEKS scheme called "Public Key Encryption with Multi-keywords Search using Mamdani System (m-PEMKS)". The m-PEMKS scheme utilised Mamdani fuzzy inference process, that is fuzzification of the input variables, rule evaluations, aggregations of the rule outputs and finally defuzzification [27] in order to solve imprecise keyword search problem.

This paper firstly revisits m-PEMKS scheme proposed by Kazemian and Ma [26] in 2020 and then defines a robust PEKS scheme called *Public Key Authenticated Encryption with Multi-Keywords Search using Mamdani Sys-*

*tem (m-PAEMKS)*. Compared with m-PEMKS scheme, the proposed scheme applies User Authentication technique to prevent IKGA. Besides, m-PAEMKS scheme has the properties of Ciphertext Indistinguishability and Trapdoor Indistinguishability and is proved to be semantic security under random oracle models in order to resist OKGA. Last but not least, the proposed scheme also incorporates with Fuzzy Rule Based Model to support Fuzzy Keyword Search.

## 2 Preliminaries

### 2.1 Bilinear pairings [5]

$G_1$ and $G_2$ are two cyclic groups. $G_1$ is an additive cyclic group and $G_2$ is a multiplicative group, respectively. Also, let $P$ and a prime number $g$ be the generator and the order of the group $G_1$. A bilinear pairing is a map $e : G_1 \times G_1 \to G_2$, which has the following properties:
i. Bilinear: $e(mA, nB) = e(A, B)^{mn}$ for all $A, B \in G_1$ and $x, y \in Z_P$.
ii. Computable: $e(A, B) \in G_2$ is computable in a Probabilistic Polynomial-time (PPT) algorithm, for any $A, B \in G_1$.
iii. Non-degenerate: $e(A, B) \neq 1$.

### 2.2 The Bilinear Diffie-Hellman (BDH) assumption [28]

Given $(g, mP, nP, fP)$ for $m, n, f \in Z_P$, compute the BDH key $e(P, P)^{mnf}$. An algorithm $A$ has an advantage $\varepsilon$ to solve BDH assumption in $G_1$, if $Pr[A(g, mP, nP, fP) = e(P, P)^{mnf}] \geq \varepsilon$. It is known that BDH assumption holds in $G_1$, if no $t$ time algorithm to solve BDH assumption in $G_1$ with the advantage at least $\varepsilon$ .

### 2.3 Public Key Encryption with Keyword Search [5]

Boneh et al. [5] formulated a formal definition of PEKS in 2004, which contained four PPT algorithms as follows:
1.$KeyGen_{Rec}(cp)$: Input the common parameter $cp$, produce a public and private key pair ($pk_{Rec}$, $sk_{Rec}$) of the receiver.
2.$PEKS(cp, pk_{Rec}, W)$: Input the $cp$ and the receiver's public key $pk_{Rec}$, then generate a Searchable encryption S=PEKS($pk_{Rec}$, $w$) of a keyword $w$.
3.$Trapdoor(cp, sk_{Rec}, w)$: Input the $cp$ and the receiver's private key $sk_{Rec}$, then produce a Trapdoor query $T_w$= Trapdoor($sk_{Rec}$, $w^*$) of a keyword $w^*$.
4.$Test(cp, S, T_w)$: Input $cp$, a Searchable encryption S=PEKS(cp,$pk_{Rec}$,$w$) and a Trapdoor query $T_w$ = Trapdoor($sk_{Rec}$,$w^*$). If $w = w^*$, output "Yes". Otherwise, output "No match".

### 2.4 Fuzzy Rule Based Model [29]

The fuzzy rule based model contains the following four stages:
1. *Fuzzification of the input variables*: the aim of this stage is to transform crisp inputs into fuzzy inputs by the membership functions.
2. *Rules evaluation*: the stage combines the antecedents by using the logic operations (“AND”, “OR”, “NOT” ).
3. *Aggregation of the rule outputs*: the stage expresses the consequents as a single fuzzy set.
4. *Defuizzification*: the defuizzification method, such as Center of Gravity (COG), is applied to transform the fuzzy outputs into the crisp outputs.

For discrete membership function, the defuzzified value $x^*$ is defined as $x^* = \frac{\sum_{i=1}^{n} x_i \mu(x_i)}{\sum_{i=1}^{n} \mu(x_i)}$, where $x_i$ represents the sample element, $\mu(x_i)$ is the membership function and $n$ is the number of elements in the sample.

For continuous membership function, the defuzzified value $x^*$ is defined as $x^* = \frac{\int x\mu(x)dx}{\int \mu(x)dx}$.

## 3 Revision of Public Key Encryption with Multi-keywords Search using Mamdani System

Consider a situation: Alice uploads many encrypted financial statements with different dates into the cloud server. Later on, Bob would like to obtain the “Latest” financial statements by sending Trapdoor query $= Trapdoor(Latest, SK_{Bob})$ to the cloud server. Then, the cloud server may witness an issue because “Latest” is a fuzzy keyword. What does “Latest” mean? One week ago, few months ago or even one year ago?

Traditional PEKS schemes will report errors, if the keyword for searching is an imprecise keyword. In 2020, Kazemian and Ma [26] proposed a m-PEMKS scheme to support fuzzy keyword search. The proposed scheme allowed the user to search encrypted document by using a fuzzy keyword without compromising the original data security. However, the server might be honest but curious, it may release its private key to the public networks and also exploit the keyword between the Searchable encryption and the Trapdoor query. Therefore, the m-PEMKS scheme may suffer IKGA. This section will analyse the security of m-PEMKS scheme.

### 3.1 The Concrete Construction for m-PEMKS

The m-PEMKS scheme consists of two main sections: one is Searchable Encryption and Trapdoor Query part. Another one is Fuzzy Encryption and Decryption part. To simplicity, the details below are only described the Searchable Encryption and Trapdoor Query part as this part might be vulnerable to IKGA.

The first algorithm in m-PEMKS scheme is $KeyGen_{Param-PEMKS}(1^{\zeta})$. The main purpose of this algorithm is to achieve the PEMKS common parameter $cp = \{g, P, G_1, G_2, e, H, H^*\}$. More specially, the details of $\{g, P, G_1, G_2, e\}$ can be found in *Section 2.1*. Besides, $\{H, H^*\}$ are two specific hash functions where $H : \{0,1\}^{\circ} \rightarrow G_1$ and $H^* : G_2 \rightarrow \{0,1\}^{\bullet}$.

Once the PEMKS common parameter $cp$ is generated. The next steps are Server's PEMKS key generation ($KeyGen_{Ser-PEMKS}(cp)$) and Receiver's PEMKS key generation ($KeyGen_{Rec-PEMKS}(cp)$).

For $KeyGen_{Ser-PEMKS}(cp)$, the server chooses $a \in Z_P$ uniformly at random and then calculates $A = aP$. Besides, the server randomly selects $B \in G_1$. So, the server's PEMKS public key is $pk_{Ser-PEMKS} = (cp, A, B)$ and the private key is $sk_{Ser-PEMKS} = (cp, a)$. For $KeyGen_{Rec-PEMKS}(cp)$, the receiver selects $c \in Z_P$ uniformly at random and then calculates $C = cP$. Therefore, the receiver's PEMKS public key is $pk_{Rec-PEMKS} = (cp, C)$ and the private key is $sk_{Rec} = (cp, c)$.

Next, the sender will generate a Searchable encryption $E$ by the algorithm $E = Encryption(pk_{Ser-PEMKS}, pk_{Rec-PEMKS}, W)$ and then send the encrypted document appending with the Searchable encryption to the online server. More details are described below.

For $Encryption(pk_{Ser-PEMKS}, pk_{Rec-PEMKS}, W)$, the sender randomly selects $t \in Z_P$ and a keyword-vector $W$=($w_1$,$w_2$,...,$w_n$). Then, the sender calculates a Searchable encryption $E = (M, N_1, N_2, ..., N_n) = (tA, H^*(D_1), H^*(D_2), ..., H^*(D_n))$, where $D_1 = e(H(w_1), C)^t$,$D_2 = e(H(w_2), C)^t$,...,$D_n = e(H(w_n), C)^t$.

When the receiver would like to retrieve the encrypted document, he/she will generate a Trapdoor query $R$ by the algorithm $R = Request(pk_{Ser-PEMKS}, sk_{Rec-PEMKS}, W)$ and then send the Trapdoor query $R$ to the online server. More details are described below.

For $Request(pk_{Ser-PEMKS}, sk_{Rec-PEMKS}, W^*)$, the receiver selects $t^* \in Z_P$ uniformly at random and a keyword-vector $W^*$=($w_1^*$,$w_2^*$,...,$w_m^*$). Then, the receiver calculates a Trapdoor query $R = (Z, T_1, T_2, ..., T_m) = (e(A, t^*B), cH(w_1^*) \oplus e(A, B)^{t^*+c}, cH(w_2^*) \oplus e(A, B)^{t^*+c}, ..., cH(w_m^*) \oplus e(A, B)^{t^*+c})$.

Once the online server receives the Searchable encryption $E$ and the Trapdoor request $R$, it will run $Test$ algorithm ($Test(E, R, sk_{Ser-PEMKS})$) to check whether the keywords in Searchable encryption and Trapdoor query are same or not. Note that, the online server is zero knowledge so that it cannot decrypt the encrypted messages to retrieve the keywords. More details are described below.

$Test(E, R, sk_{Ser-PEMKS}, sk_{Ser-RSA})$: For $i$ and $j$ are the indexes of keyword. Let $i \in \{1, 2, ..., n\}$, $j \in \{1, 2, ..., m\}$ and $j \leq i$.
Firstly, the server computes
$T_{w_1} = T_1 \oplus Z \bullet e(aB, C) = cH(w_1^*), ...,$
$T_{w_j} = T_j \oplus Z \bullet e(aB, C) = cH(w_j^*), ...,$
$T_{w_m} = T_m \oplus Z \bullet e(aB, C) = cH(w_m^*)$
The server then checks whether $H^*[e(T_{w_j}, \frac{M}{a})] = N_i$. If the equation holds, the server will assert the keywords both in the Searchable encryption and

the Trapdoor query are same. Otherwise, the server will reject the receiver's request.

## 3.2 Analysis of m-PEMKS

In general, the online server is zero knowledge so that it cannot achieve any information related to the plaintext of the keywords. However, due to the server might being honest but curious, it may release its private key to the general public networks and also attempt to guess and exploit the keywords between the Searchable encryption and the Trapdoor query.

More specially, the server firstly calculates $T_{w_j} = T_j \oplus Z \bullet e(aB, C) = cH(w_j^*)$. Then, the server could guess a proper keyword $w_i^*$ in order to obtain $H(w_i^*)$. After that, the server will execute a bilinear pairing to check if $e(pk_{Rec-PEMKS}, H(w_i^*)) = e(P, T_{w_j})$. The equation holds for $w_i^* = w_j^*$ as $e(pk_{Rec-PEMKS}, H(w_i^*)) = e(cP, H(w_i^*)) = e(cP, H(w_j^*)) = e(P, T_{w_j})$. In shot, the malicious server could achieve the correct keyword by trial and error as the keyword space is limited. Therefore, the m-PEMKS scheme might be vulnerable to IKGA.

## 4 Public Key Authenticated Encryption with Multi-Keywords Search using Mamdani System

This section defines a robust m-PEMKS scheme called *Public Key Authenticated Encryption with Multi-Keywords Search using Mamdani System (m-PAEMKS)* in order to resist Inside Keyword Guessing Attack (IKGA). The m-PAEMKS scheme incorporates with User Authentication technique so that the malicious server is not able to find the relation between a keyword in Searchable encryption and a keyword in Trapdoor query. Therefore, the proposed scheme resists IKGA.

## 4.1 Formal Definition for m-PAEMKS

The new proposed scheme consists of three parties: a sender, a receiver and an online server.

The sender is a party to generate a Searchable encryption by PEKS algorithm. The receiver is a party to generate a Trapdoor query by Trapdoor algorithm. Once the online server receives these encrypted messages, it will run $Test$ algorithm to check whether these encrypted messages contain the same keyword(s) or not, and reply to the receiver in the end.

The m-PAEMKS scheme contains eight PPT algorithms in the following:
1. $KeyGen_{Param-PAEMKS}(1^{\varsigma})$: Given the parameter $1^{\varsigma}$ and then generate a common parameter $cp$.

2. $KeyGen_{Param-RSA}(k)$: Given the parameter $k$ and then generate a global parameter $gp$.
3. $KeyGen_{Sen-PAEMKS}(cp)$: Given $cp$ and then generate the sender's public and private PAEMKS key pair $(pk_{Sen-PAEMKS},\ sk_{Sen-PAEMKS})$.
4. $KeyGen_{Rec-PAEMKS}(cp)$: Given $cp$ and then generate the receiver's public and private PAEMKS key pair $(pk_{Rec-PAEMKS},\ sk_{Rec-PAEMKS})$.
5. $KeyGen_{Ser-RSA}(gp)$: Given $gp$ and then generate the server's public and private RSA key pair $(pk_{Ser-RSA},\ sk_{Ser-RSA})$.
6. $Encryption(sk_{Sen-PAEMKS}, pk_{Rec-PAEMKS}, pk_{Ser-RSA}, W)$: A Searchable encryption $E$=($E_1$,$E_2$)=SCF-PAEMKS($sk_{Sen-PAEMKS}$,$pk_{Rec-PAEMKS}$, $W_{part1}$)||RSA($pk_{Ser-RSA}$,$W_{part2}$) is created, where $W$=($W_{part1}$,$W_{part2}$)=[($w_1$, $w_2$,...,$w_n$);$w_{n+1}$].
7. $Request(pk_{Sen-PAEMKS}, sk_{Rec-PAEMKS}, pk_{Ser-RSA}, W^*)$: A Trapdoor request $R$=($R_1$,$R_2$)=Trapdoor($pk_{Sen-PAEMKS}$,$sk_{Rec-PAEMKS}$,$W_{part1}$)||RSA( $pk_{Ser-RSA}$,$W_{part2}$) is created, where $W^*$=($W^*_{part1}$,$W^*_{part2}$)=[($w^*_1, w^*_2, ..., w^*_m$); $w_{fuzzy}$].
8. $Test(E, R, pk_{Sen-PAEMKS}, pk_{Rec-PAEMKS}, sk_{Ser-RSA})$: $Test$ algorithm contains two parts: an Exact Match and a Fuzzy Match.
*For Exact Match*: Given $pk_{Sen-PAEMKS}$ and $pk_{Rec-PAEMKS}$, $E_1$ and $R_1$. If $W^*_{part1} \in W_{part1}$, the system will go to Fuzzy Match. Otherwise, the system will be terminated here.
*For Fuzzy Match*: Given $sk_{Ser-RSA}$, $E_2$ and $R_2$. The server decrypts $E_2$ and $R_2$ in order to obtain $W_{part2}$ and $W^*_{part2}$. Let $W^*_{part2}$ and $W_{part2}$ be the conclusion and the condition of the rules in Mamdani system. Then, the encrypted files are filtered by Mamdani system. Finally, the online server will reply to the receiver.

## 4.2 Security Models for m-PAEMKS

The proposed m-PAEMKS scheme has two main cryptographic suites, which are PEKS and RSA. For PEKS algorithm, the security relies on Ciphertext Indistinguishability of Chosen Plaintext Attack (IND-CPA) and Trapdoor Indistinguishability of Chosen Plaintext Attack (Trapdoor-IND-CPA). For RSA, the security relies on factoring the large number. However, PEKS is used to protect the confidentiality of the keywords. Therefore, the security of m-PAEMKS scheme mainly relies on PEKS algorithm. The security models for m-PAEMKS scheme are described below.

Let **A** be an attacker whose running time is bounded by $T$ and **E** be a challenger. The challenger **E** establishes the m-PAEMKS system and accepts the challenges from the attacker **A**. For IND-CPA Game, it defines that the attacker could not recover any one bit keyword from the Searchable encryptions while Trapdoor-IND-CPA Game presents that the attacker could not retrieve any one bit keyword from the Trapdoor queries. More details are illustrated as follows.

### *4.2.1 IND-CPA Game*

**Setup:** **E** firstly runs $KeyGen_{Param-PAEMKS}(1^{\zeta})$, $KeyGen_{Sen-PAEMKS}(cp)$ and $KeyGen_{Rec-PAEMKS}(cp)$ to generate the common parameter *cp*, the sender's public/private PAEMKS keys $(pk_{Sen-PAEMKS},\ sk_{Sen-PAEMKS})$, the receiver's public/private PAEMKS keys $(pk_{Rec-PAEMKS},\ sk_{Rec-PAEMKS})$. Then, **A** receives *cp*, $pk_{Sen-PAEMKS}$, $pk_{Rec-PAEMKS}$ while $sk_{Sen-PAEMKS}$, $sk_{Rec-PAEMKS}$ cannot be sent to **A**.
**Phase 1-1 (Queries):** Adaptively, the attacker **A** can ask the challenger **E** for Trapdoor Oracle $O_T$ and Ciphertext Oracle $O_C$ many times.
**Challenge:** The attacker **A** sends a target keyword-vector pair $(W_0,\ W_1)$ to **E**, where $W_0 = (w_{01}, w_{02}, ..., w_{0n})$ and $W_1 = (w_{11}, w_{12}, ..., w_{1n})$. Note that $W_0$ and $W_1$ cannot be queried in *Phase 1-1.* Once **E** receives the target keyword-vector pair, he/she will execute *Encryption* algorithm to generate a Searchable encryption $E_1 = SCF - PAEMKS(sk_{Sen-PAEMKS}, pk_{Rec-PAEMKS}, W_{\lambda})$, where $\lambda \in \{0, 1\}$. Finally, **E** sends $E_1$ back to **A** .
**Phase 1-2 (Queries):** The attacker **A** can continue to ask **E** for Trapdoor Oracle $O_T$ and Ciphertext Oracle $O_C$ many times as in *Phase 1-1*, as long as $W \neq W_0, W_1$.
**Guess:** The attacker **A** guesses $\lambda^* \in \{0, 1\}$ and wins ***IND-CPA Game***, only if $\lambda^* = \lambda$.

The advantage of **A** winning ***IND-CPA Game*** is as follows:

$$Adv^{IND-CPA}_{m-PAEMKS,A}(k) = |Pr[\lambda^* = \lambda] - 1/2| \tag{1}$$

Therefore, the m-PAEMKS model can be regarded as *IND-CPA* security only if the $Adv^{IND-CPA}_{m-PAEMKS,A}(k)$ is negligible.

### *4.2.2 Trapdoor-IND-CPA Game*

**Setup:** **E** firstly runs $KeyGen_{Param-PAEMKS}(1^{\zeta})$, $KeyGen_{Sen-PAEMKS}(cp)$ and $KeyGen_{Rec-PAEMKS}(cp)$ to generate the common parameter *cp*, the sender's public/private PAEMKS keys$(pk_{Sen-PAEMKS},\ sk_{Sen-PAEMKS})$, the receiver's public/private PAEMKS keys $(pk_{Rec-PAEMKS},\ sk_{Rec-PAEMKS})$. Then, **A** receives *cp*, $pk_{Sen-PAEMKS}$, $pk_{Rec-PAEMKS}$ while $sk_{Sen-PAEMKS}$, $sk_{Rec-PAEMKS}$ cannot be sent to **A**.
**Phase 2-1 (Queries):** Adaptively, the attacker **A** can ask the challenger **E** for Trapdoor Oracle $O_T$ and Ciphertext Oracle $O_C$ many times.
**Challenge:** **A** sends a target keyword-vector pair $(W_0^*,\ W_1^*)$ to **E**, where $W_0^* = (w_{01}^*, w_{02}^*, ..., w_{0m}^*)$ and $W_1^* = (w_{11}^*, w_{12}^*, ..., w_{1m}^*)$. Note that none of $W_0^*$ or $W_1^*$ can be asked in *Phase 2-1.* Once **E** receives the target keyword-vector pair, he/she will execute *Request* algorithm to achieve a Trapdoor request $R_1 = Trapdoor(pk_{Sen-PAEMKS}, sk_{Rec-PAEMKS}, W_{\lambda}^*)$, where $\lambda \in \{0, 1\}$. Finally, **E** sends $R_1$ back to **A**.

**Phase 2-2 (Queries):** The attacker **A** can continue to ask **E** for Trapdoor Oracle $O_T$ and Ciphertext Oracle $O_C$ many times as in *Phase 2-1*, as long as $W^* \neq W_0^*, W_1^*$.
**Guess:** The attacker **A** guesses $\lambda^* \in \{0,1\}$ and wins ***Trapdoor-IND-CPA Game***, only if $\lambda^* = \lambda$.

The advantage of **A** winning ***Trapdoor-IND-CPA Game*** is as follows:

$$Adv_{m-PAEMKS,A}^{Trap-IND-CPA}(k) = |Pr[\lambda^* = \lambda] - 1/2| \qquad (2)$$

Therefore, the m-PAEMKS model can be regarded as *Trapdoor-IND-CPA* security only if the $Adv_{m-PAEMKS,A}^{Trap-IND-CPA}(k)$ is negligible.

## 4.3 The Construction for m-PAEMKS

The m-PAEMKS scheme (Fig.1) consists of two main sections: One is Searchable Encryption and Trapdoor Query part. Another one is Fuzzy Encryption and Decryption part. The formal definition of m-PAEMKS scheme is provided in *Section 4.1* and this section will describe a concrete construction of the m-PAEMKS scheme.

The first algorithm in m-PAEMKS scheme is $KeyGen_{Param-PAEMKS}(1^{\zeta})$. The main purpose of this algorithm is to achieve the PAEMKS common parameter $cp = \{P, g, G_1, G_2, e, H\}$. More specially, the details of $\{P, g, G_1, G_T, e\}$ can be found in *Section 2.1*. Besides, $H$ is a specific hash function that is $H : \{0,1\}^{\circ} \to G_1$.

The second algorithm ($KeyGen_{Param-RSA}(K)$) is used to generate the global parameter $gp$ of RSA which will be used for Fuzzy Encryption and Fuzzy Decryption part. More specially, two prime numbers $u$ and $v$ (where $u \neq v$) are selected uniformly at random. Then, the algorithm calculates $L = u \times v$ and $\phi(L) = (u-1) \times (v-1)$.

Once the sender receives the PAEMKS common parameter $cp$, he/she will execute $KeyGen_{Sen-PAEMKS}(cp)$ to generate the PAEMKS public and private key pair. More specially, the sender randomly selects $a \in Z_P$ and then calculates $A = aP$. So, the sender's PAEMKS public key is $pk_{Sen-PAEMKS} = A$ and the PAEMKS private key is $sk_{Sen-PAEMKS} = a$.

Similarly, the receiver will execute $KeyGen_{Rec-PAEMKS}(cp)$ to generate the PAEMKS public and private key pair by using the PAEMKS common parameter $cp$. More specially, the receiver randomly selects $b \in Z_P$ and then calculates $B = bP$. So, the receiver's PAEMKS public key is $pk_{Rec-PAEMKS} = B$ and the PAEMKS private key is $sk_{Rec-PAEMKS} = b$.

When the server receives the global parameter $gp$ of RSA, it will randomly pick up $x \in Z_q$, where $gcd(\phi(L), x) = 1$ and $1 < x < \phi(L)$. Next, the server computes $y$ by $y \equiv x^{-1}(mod\phi(L))$. So, the server's RSA public key is $pk_{Ser-RSA} = (x, L)$ and the private key is $sk_{Ser-RSA} = (y, L)$.

All of the preparatory work has been finished until now. Then, the sender will send a Searchable encryption $E$ to the online server.

The sender executes $Encryption(sk_{Sen-PAEMKS}, pk_{Rec-PAEMKS}, pk_{Ser-RSA}, W)$ to generate Searchable encryption $E$, where $W = (W_{part1}, W_{part2})$. The Searchable encryption $E$ is made by $E_1$ and $E_2$ while $E_1$ is created by using PAEMKS algorithm and $E_2$ is created by using RSA algorithm. Therefore, for $E_1$ generation, the sender selects $t \in Z_P$ and a keyword-vector $W_{part1} = (w_1, w_2, ..., w_n)$. Then, he/she computes $E_1 = (M, N_1, N_2, ..., N_n) = [t \oplus pk_{Rec-PAEMKS}, e(sk_{Sen-PAEMKS} \bullet H(pk_{Sen-PAEMKS}, pk_{Rec-PAEMKS}, w_1), pk_{Rec-PAEMKS} \bullet t), e(sk_{Sen-PAEMKS} \bullet H(pk_{Sen-PAEMKS}, pk_{Rec-PAEMKS}, w_2), pk_{Rec-PAEMKS} \bullet t), ..., e(sk_{Sen-PAEMKS} \bullet H(pk_{Sen-PAEMKS}, pk_{Rec-PAEMKS}, w_n), pk_{Rec-PAEMKS} \bullet t)]$. After that, the sender calculates $E_2 = N_{n+1} = (w_{n+1})^x mod\, L$, where $W_{part2} = w_{n+1}$ is a keyword. Finally, the sender sends the Searchable encryption $E = (E_1, E_2)$ to the online third party.

When the receiver would like to retrieve the encrypted document, he/she will generate a Trapdoor query $R$ by $Request(pk_{Sen-PAEMKS}, sk_{Rec-PAEMKS}, pk_{Ser-RSA}, W^*)$, where $W^* = (W^*_{part1}, W^*_{part2})$. The Trapdoor query $R$ is made by $R_1$ and $R_2$. For $R_1$, the receiver initially picks up a keyword-vector $W^*_{part1} = (w^*_1, w^*_2, ..., w_{m^*})$. Then, the receiver calculates $R_1 = (Z, T_1, T_2, ..., T_m) = [e(sk_{Rec-PAEMKS} \bullet H(pk_{Sen-PAEMKS}, pk_{Rec-PAEMKS}, w^*_1), pk_{Sen-PAEMKS}), e(sk_{Rec-PAEMKS} \bullet H(pk_{Sen-PAEMKS}, pk_{Rec-PAEMKS}, w^*_2), pk_{Sen-PAEMKS}), ..., e(sk_{Rec-PAEMKS} \bullet H(pk_{Sen-PAEMKS}, pk_{Rec-PAEMKS}, w^*_m), pk_{Sen-PAEMKS})$. After that, the receiver calculates $R_2 = T_{fuzzy} = (w_{fuzzy})^x mod\, L$, where $W^*_{part2} = w_{fuzzy}$ is a fuzzy keyword. Finally, the receiver sends the Trapdoor query $R = (R_1, R_2)$ to the online third party.

Once the online server receives the Searchable encryption $E$ and the Trapdoor query $R$, it will execute $Test(E, R, pk_{Sen-PAEMKS}, pk_{Rec-PAEMKS}, sk_{Ser-RSA})$ to firstly check whether keywords in Searchable encryption and Trapdoor query are same or not (This stage is called *Exact Match*). If the keywords both in Searchable encryption and Trapdoor query are same, the system will go to the next stage that is *Fuzzy Match*. Otherwise, the server will reject the receiver's request. More details are described below.

$Test(E, R, pk_{Sen-PAEMKS}, pk_{Rec-PAEMKS}, sk_{Ser-RSA})$: For $i \in \{1, 2, ..., n\}$ and $j \in \{1, 2, ..., m\}$, where $j \leq i$.
(i) *For Exact Match*: Firstly, the server calculates $M \oplus pk_{Rec}$. Then, the server checks whether $H^*[e(T_j, \frac{M}{a})] = N_i$. If so, the system will go to Fuzzy Match. Otherwise, the system will be terminated here.
(ii) *For Fuzzy Match*: The server decrypts $w_{n+1}$ and $w_{fuzzy}$ from $\{[(w_{n+1})^x mod\, L]^y mod\, L\}$ and $\{[(w_{fuzzy})^x mod\, L]^y mod\, L\}$. Let $w_{fuzzy}$ and $w_{n+1}$ be the conclusion and condition of the rules in Mamdani system.
For simplicity, let $w_{fuzzy}$ and $w_{n+1}$ be the keyword "latest" and a set of "DATE". Hence, three rules could be defined below:
R1: IF DATE is oldest, THEN the encrypted file is unnecessary.
R2: IF DATE is newest, THEN the encrypted file is necessary.
R3: IF DATE is either new or old, THEN the encrypted file may be necessary or unnecessary.

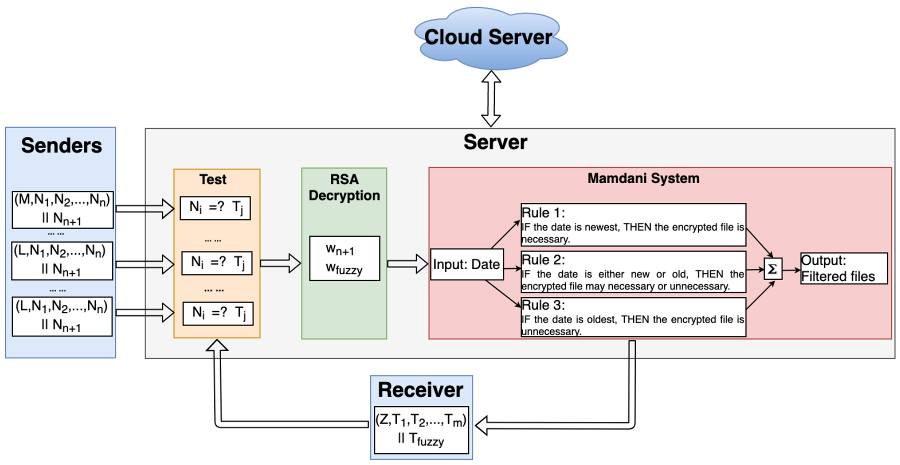
Cloud Server
Server
Senders
(M,N1,N2,...,Nn) || Nn+1
(L,N1,N2,...,Nn) || Nn+1
(L,N1,N2,...,Nn) || Nn+1
Test
Ni =? Tj
Ni =? Tj
Ni =? Tj
RSA Decryption
wn+1
wfuzzy
Mamdani System
Input: Date
Rule 1: IF the date is newest, THEN the encrypted file is necessary.
Rule 2: IF the date is either new or old, THEN the encrypted file may necessary or unnecessary.
Rule 3: IF the date is oldest, THEN the encrypted file is unnecessary.
Σ
Output: Filtered files
Receiver
(Z,T1,T2,...,Tm) || Tfuzzy

**Fig. 1** The concrete construction of m-PAEMKS

Fig.1 shows the process for the m-PAEMKS scheme. Once the preparatory work is finished (i.e. PAEMKS common parameter *cp* generation, RSA global parameter *gp* generation), the sender will generate the Searchable encryption $E$ by the algorithm $Encryption(sk_{Sen-PAEMKS}, pk_{Rec-PAEMKS}, pk_{Ser-RSA}, W)$. Then, the sender sends a Searchable encryption $E = (E_1, E_2) = [(M, N_1, N_2, ..., N_n)||N_{n+1}]$ to the online server. When the receiver wishes to retrieve the encrypted document, he/she will firstly create a Trapdoor query $R$ by the algorithm $Request(pk_{Sen-PAEMKS}, sk_{Rec-PAEMKS}, pk_{Ser-RSA}, W^*)$ and then send a Trapdoor query $R = (R_1, R_2) = [(Z, T_1, T_2, ..., T_m)||T_{fuzzy}]$ to the online server. Once the server receives the Searchable encryption and the Trapdoor query, it will run $Test$ algorithm to firstly check whether the keywords both in Searchable encryption $(N_i)$ and Trapdoor query $(T_j)$ are same or not. If the keywords are same, the server will decrypt $N_{n+1}$ and $T_{fuzzy}$ by RSA algorithm to obtain two keywords ($w_{n+1}$ and $w_{fuzzy}$). Then, let $w_{fuzzy}$ and $w_{n+1}$ be the conclusion and the condition of the rules (see in *Section 4.3*) in Mamdani system. Finally, the server will filter the irrelevant encrypted documents and send the rest encrypted documents back to the receiver.

## 4.4 The Correctness for m-PAEMKS

Suppose $i$ and $j$ are the indexes of keywords in $W_{part1}$ and $W^*_{part1}$, where $i \in \{1, 2, ..., n\}$ and $j \in \{1, 2, ..., m\}$. The proposed scheme is completely correct and the details are described in the following:
*For Exact Match*:

The server initially calculates $M \oplus pk_{Rec} = t \oplus pk_{Rec} \oplus pk_{Rec} = t$.
Then, the server checks whether $T_j^t = N_i$ or not.
For $i \in \{1, 2, ..., n\}$,

$T_j^t$=$e(sk_{Rec-PAEMKS} \bullet H(pk_{Sen-PAEMKS}, pk_{Rec-PAEMKS}, w_j^*),$
$pk_{Sen-PAEMKS})^t$=$e(b \bullet H(pk_{Sen-PAEMKS}, pk_{Rec-PAEMKS}, w_j^*), aP)^t$
$=e(a \bullet H(pk_{Sen-PAEMKS}, pk_{Rec-PAEMKS}, w_j^*), bP)^t$=$e(sk_{Sen-PAEMKS}$
$\bullet H(pk_{Sen-PAEMKS}, pk_{Rec-PAEMKS}, w_j^*), pk_{Rec-PAEMKS} \bullet t) = N_i$
The server confirms $T_j^t = N_i$. So, it is concluded that both the Searchable encryption $N_i$ and the Trapdoor query $T_j$ contains the same keyword.

*For Fuzzy Match*: this algorithm is correct by Mamdani Fuzzy Inference System's properties.

## 5 Security and Performance

### 5.1 Security Analysis for m-PAEMKS

This section verifies the security of m-PAEMKS scheme by **Lemma 1 (Ciphertext Indistinguishability)** and **Lemma 2 (Trapdoor Indistinguishability))**. More specially, **Lemma 1 (Ciphertext Indistinguishability)** confirms that the attacker cannot retrieve any one bit keyword from the Searchable encryptions while **Lemma 2 (Trapdoor Indistinguishability)** confirms that the attacker cannot recover any one bit keyword from the Trapdoor queries. To summary, the m-PAEMKS scheme has the properties of Ciphertext Indistinguishability and Trapdoor Indistinguishability and is proved to semantic security under the random oracle models in order to resist OKGA. Besides, the m-PAEMKS scheme incorporates with User Authentication technique to resist IKGA.

**Lemma 1.** For any PPT adversary **A** against the Ciphertext Indistinguishability of m-PAEMKS scheme, its advantage $ADV_A^T(\lambda)$ will be negligible if BDH assumption holds.

**Proof**: suppose that there is an adversary **A** who breaks the ciphertext privacy of m-PAEMKS scheme with a non-negligible advantage $\varepsilon_C$.

The procedure for verifying the Ciphertext Indistinguishability of m-PAEMKS scheme is based on **IND-CPA Game**. The formal definition of **IND-CPA Game** is described in *Section 4.2.1*. Besides, the security for m-PAEMKS scheme relies on BDH assumption which is defined in *Section 2.2*.

The challenger **E** builds the m-PAEMKS system and accepts the challenges from the adversary **A**. Assume that **E** has $(P, g, G_1, G_2, e, xP, yP, zP)$ as an input of BDH assumption whose running time is bounded by $T$. **E**'s target is to compute a BDH key $e(P, P)^{xyz}$ of $xP$, $yP$ and $zP$ by **A**'s IND-CPA.

**Setup Simulation**
**E** selects $x, y \in Z_P$ uniformly at random. **E** then computes $xP$ and $yP$ as the

PAEMKS public keys ($pk_{Sen-PAEMKS}$,$pk_{Rec-PAEMKS}$) of the sender and the receiver. After that, **E** returns the common parameter $cp = (P, g, G_1, G_2, e, H)$ and sends ($cp$,$pk_{Sen-PAEMKS}$,$pk_{Rec-PAEMKS}$) to **A**.

**Phase 1-1 Simulation (Queries)**

Note that three assumptions are introduced below for simplicity.

1. The adversary **A** issues at most $q_H$, $q_T$, $q_C$ queries from the Hash Oracle $O_H$, the Trapdoor Oracle $O_T$ and the Ciphertext Oracle $O_C$ respectively.
2. The adversary **A** does not repeat any queries from $O_H$, $O_T$ and $O_C$.
3. The adversary **A** cannot send a query ($pk_{Sen-PAEMKS}$,$w$) to $O_T$ nor ($pk_{Rec-PAEMKS}$,$w$) to $O_C$ before issuing ($pk_{Sen-PAEMKS}$,$pk_{Rec-PAEMKS}$,$w$) to $O_H$.

**E** creates the oracles in the following.

For Hash Oracle $O_H$.

When **A** asks a query for a tuple ($pk_{Sen-PAEMKS}$,$pk_{Rec-PAEMKS}$,$w_i$), E will reply as follows:

i. **E** picks up a coin $\theta_i$ uniformly at random and then calculates $Pr[\theta_i = 0] = \frac{1}{h+1}$.

ii. **E** chooses $f_i \in Z_P$ uniformly at random. If $\theta_i = 0$, $F_i = lP + f_iP$ will be returned by **E**. If $\theta_i = 1$, $F_i = f_iP$ will be returned by **E**.

iii. **E** sends $F_i$ to **A** and places $[(pk_{Sen-PAEMKS}, pk_{Rec-PAEMKS}, w_i), F_i, f_i, \theta_i]$ into *H_List*. The *H_List* is initially empty.

For Trapdoor Oracle $O_T$:

When **A** sends a Trapdoor query corresponding to the keyword $w$ to **E**, the challenger **E** will compute $T_w = e(f_i \bullet pk_{Rec-PAEMKS}, pk_{Sen-PAEMKS}) = e(yF_1, pk_{Sen-PAEMKS}) = e(sk_{Rec-PAEMKS} \bullet H(pk_{Sen-PAEMKS}, pk_{Rec-PAEMKS}, w), pk_{Sen-PAEMKS})$. Finally, **E** sends $T_w$ to **A**.

For Ciphertext Oracle $O_C$ :

When **A** send a Ciphertext query corresponding to the keyword $w$ to **E**, the challenger **E** will select $t \in Z_P$ uniformly at random and then calculates $N_w = e(f_1 \bullet pk_{Sen-PAEMKS}, pk_{Rec-PAEMKS} \bullet t) = e(xF_1, pk_{Rec-PAEMKS} \bullet t) = e(sk_{Sen-PAEMKS} \bullet H(pk_{Sen-PAEMKS}, pk_{Rec-PAEMKS}, w), pk_{Rec-PAEMKS} \bullet t)$. Finally, **E** sends $N_w$ to **A**.

**Challenge Simulation**

At some point, the adversary **A** uploads a keyword-vector pair $(W_0^*, W_1^*)$ to **E**, where $W_0^* = (w_{01}, w_{02}, ..., w_{0n})$ and $W_1^* = (w_{11}, w_{12}, ..., w_{1n})$. Note that $(yP, W'^*_0)$ and $(yP, W'^*_1)$ have not been queried to oracle $O_T$ and $(xP, W_0^*)$ and $(xP, W_1^*)$ have not been queried to oracle $O_C$. Then, **E** creates a Searchable encryption $E_1$ as follows:

- **E** chooses $i \in \{1, 2, ..., n\}$ uniformly at random.
- **E** runs all above algorithms to obtain two tuples $(w_{0i}^*, F_{0i}^*, f_{0i}^*, \theta_{0i}^*)$ and $(w_{1i}^*, F_{1i}^*, f_{1i}^*, \theta_{1i}^*)$. If $\theta_0 = \theta_1 = 1$, **E** will stop the system and throw "Suspension".

Otherwise, **E** will create the Searchable encryption below:

- **E** runs all above algorithms for simulating $H$ function at $2(n-1)$ times in order to achieve two vectors $[(w_{01}^*, F_{01}^*, f_{01}^*, \theta_{01}^*), ..., (w_{0i-1}^*, F_{0i-1}^*, f_{0i-1}^*, \theta_{0i-1}^*), (w_{0i+1}^*, F_{0i+1}^*, f_{0i+1}^*, \theta_{0i+1}^*), ..., (w_{0n}^*, F_{0n}^*, f_{0n}^*, \theta_{0n}^*)]$ and $[(w_{11}^*, F_{11}^*, f_{11}^*, \theta_{11}^*), ...,$

$(w^*_{1i-1}, F^*_{1i-1}, f^*_{1i-1}, \theta^*_{1i-1}), (w^*_{1i+1}, F^*_{1i+1}, f^*_{1i+1}, \theta^*_{1i+1}), ..., (w^*_{1n}, F^*_{1n}, f^*_{1n}, \theta^*_{1n})]$.
If $\theta^*_{0j}$ and $\theta^*_{1j}$ are equal to 0 for all $j = 0, ..., i-1, i+1, ..., n$, **E** will stop the system and throw "Suspension". Otherwise, **E** executes the following steps:
– **E** selects $\beta \in \{0,1\}^d$ uniformly at random.
– **E** selects $J_j \in \{0,1\}^d$ uniformly at random and returns a Searchable encryption $E^*_1 = (L^*, N^*_1, N^*_2, ..., N^*_n)$.
So, **E** chooses $t = z$. **E** then computes $E^*_1 = (L^*, N^*_1, ..., N^*_{i-1}, N^*_{i+1}, ..., N^*_n) = [z \oplus yP, e(sk_{Sen-PAEMKS} \bullet H(pk_{Sen-PAEMKS}, pk_{Rec-PAEMKS}, w^*_1), pk_{Rec-PAEMKS} \bullet t), ..., e(sk_{Sen-PAEMKS} \bullet H(pk_{Sen-PAEMKS}, pk_{Rec-PAEMKS}, w^*_{i-1}), pk_{Rec-PAEMKS} \bullet t), e(sk_{Sen-PAEMKS} \bullet H(pk_{Sen-PAEMKS}, pk_{Rec-PAEMKS}, w^*_{i+1}), pk_{Rec-PAEMKS} \bullet t), ..., e(sk_{Sen-PAEMKS} \bullet H(pk_{Sen-PAEMKS}, pk_{Rec-PAEMKS}, w^*_n), pk_{Rec-PAEMKS} \bullet t)]$
It is noticed that $N_\beta = e(sk_{Sen-PAEMKS} \bullet H(pk_{Sen-PAEMKS}, pk_{Rec-PAEMKS}, w^*_\beta), pk_{Rec-PAEMKS} \bullet t) = e(xH(pk_{Sen-PAEMKS}, pk_{Rec-PAEMKS}, w^*_\beta), yP \bullet z) = e(xf_iP, yP \bullet z) = e(f_iP, P)^{xyz}$.
**Phase 1-2 Simulation (Queries)**
**A** can continue to communicate with **E** for the oracle queries. One restriction is that **A** cannot request $[(pk_{Sen-PAEMKS}, W'^*_0), (pk_{Sen-PAEMKS}, W'^*_1)]$ to $O_T$ and $[(pk_{Rec-PAEMKS}, W^*_0), (pk_{Rec-PAEMKS}, W^*_1)]$ to $O_C$.
**Guess**
Finally, **A** shows the guess $\beta^* \in \{0,1\}$. If $\beta = \beta^*$, "Yes". Otherwise, "No match".

**Analysis**
Two events are defined below:
*Event1*: The system does not be stopped during the *Phase 1-1* and the *Phase 1-2*.
*Event2*: The system does not be stopped during the Challenge Simulation.

**Claim 1:**

$$Pr[Event1] \geq (1 - \frac{1}{h+1})^{q_T + q_C} \tag{3}$$

*Proof* : suppose that **A** does not request the same keyword twice in $O_T$ and $O_C$ queries. So, the probability of the m-PAEMKS system being stopped is $\frac{1}{h+1}$. Also, **A** requests at most $q_T$ Trapdoor queries and $q_C$ Ciphertext queries, the probability of the m-PAEMKS system being stopped in all is at least $Pr[Event1] = (1 - \frac{1}{h+1})^{q_T + q_C}$.

**Claim 2:**

$$Pr[Event2] \geq (\frac{1}{h+1}) \bullet (\frac{h}{h+1})^{2(n-1)} \tag{4}$$

*Proof* : if $\theta_0 = \theta_1 = 1$, the m-PAEMKS system will be stopped during the Challenge Simulation. So, the probability that the system does not be stopped is $1 - (1 - \frac{1}{h+1})^2$. Besides, if $\theta^*_{0j}$ and $\theta^*_{1j}$ are equal to 0 for all $j = 0, ..., i-1, i+1, ..., n$, the system will be stopped here. Hence, the probability that the m-PAEMKS system does not be stopped during the Challenge Simulation is at least $(1-\frac{1}{h+1})^{2(n-1)}\{1-(1-\frac{1}{h+1})^2\} \geq (\frac{1}{h+1}) \bullet (\frac{h}{h+1})^{2(n-1)}$.

Suppose *Event* is an event that the m-PAEMKS system does not be stopped during the whole game. Therefore, $Pr[Event] = Pr[Event1] \bullet Pr[Event2] = (1-\frac{1}{h+1})^{q_T+q_C} \bullet (\frac{1}{h+1}) \bullet (\frac{h}{h+1})^{2(n-1)}$. If $h+1 = q_T + q_C$, the $Pr[Event]$ will reach the maximum value. So, $Pr[Event] = \frac{1}{e} \bullet (\frac{1}{q_T+q_C}) \bullet (\frac{q_T+q_C-1}{q_T+q_C})^{2(n-1)}$, which is approximately equal to $\frac{1}{e(q_T+q_C)}$ and thus non-negligible.

Overall, the probability that **E** guesses the correct bit $\beta$ is as follows: $Pr[\beta^{'} = \beta] = Pr[\beta^{'} = \beta \wedge Pr[Event]] + Pr[\beta^{'} = \beta \wedge Pr[\overline{Event}]] = Pr[\beta^{'} = \beta \mid Pr[Event]]Pr[Event] + Pr[\beta^{'} = \beta \mid Pr[\overline{Event}]]Pr[\overline{Event}] = \frac{1}{2} \bullet (1 - Pr[\overline{Event}]) + (\varepsilon_C + \frac{1}{2}) \bullet Pr[\overline{Event}] = \frac{1}{2} + \varepsilon_C \bullet Pr[\overline{Event}]$. If $\varepsilon_C$ is non-negligible, so is $|Pr[\beta^{'} = \beta] - \frac{1}{2}|$.

In shot, the m-PAEMKS scheme satisfies Ciphertext Indistinguishability. And, the attacker cannot retrieve any one bit keyword from the Searchable encryptions.

**Lemma 2.** For any PPT adversary **A** against the Trapdoor Indistinguishability of m-PAEMKS scheme, its advantage $ADV_A^T(\lambda)$ will be negligible if BDH assumption holds.

**Proof**: suppose that there is an adversary **A** who breaks the trapdoor privacy of m-PAEMKS with a non-negligible advantage $\varepsilon_T$.

The procedure for verifying the Trapdoor Indistinguishability of m-PAEMKS scheme is based on **Trapdoor-IND-CPA Game**. The formal definition of **Trapdoor-IND-CPA Game** is described in *Section 4.2.2*. Besides, the security for m-PAEMKS scheme relies on BDH assumption which is defined in *Section 2.2*.

The challenger **E** builds the m-PAEMKS system and accepts the challenges from the adversary **A**. Assume that **E** has $(P, g, G_1, G_2, e, xP, yP, zP)$ as an input of BDH assumption whose running time is bounded by $T$. **E**'s target is to compute a BDH key $e(P,P)^{xyz}$ of $xP$, $yP$ and $zP$ by **A**'s Trapdoor-IND-CPA.

**Setup Simulation**
**E** selects $x, y \in Z_P$ uniformly at random. **E** then computes $xP$ and $yP$ as the PAEMKS public keys ($pk_{Sen-PAEMKS}$,$pk_{Rec-PAEMKS}$) of the sender and the receiver. After that, **E** returns the common parameter $cp = (P, g, G_1, G_2, e, H)$ and sends ($cp$,$pk_{Sen-PAEMKS}$,$pk_{Rec-PAEMKS}$) to **A**.
**Phase 2-1 Simulation (Queries)**
**E** creates the oracles in the same way as the **Phase 1-1 Simulation (Queries)** in **Lemma 1**. So, it is omitted here.

**Challenge Simulation**
At some point, the adversary **A** uploads a keyword-vector pair $(W_0^*, W_1^*)$ to **E**, where $W_0^* = (w_{01}^*, w_{02}^*, ..., w_{0m}^*)$ and $W_1^* = (w_{11}^*, w_{12}^*, ..., w_{1m}^*)$. Note that $(yP, W_0^*)$ and $(yP, W_1^*)$ have not been queried to oracle $O_T$ and $(xP, W'^*_0)$ and $(xP, W'^*_1)$ have not been queried to oracle $O_C$. Then, **E** creates the Challenge trapdoor $R_1 = (T_1, T_2, ..., T_m)$ as follows:
- If $\theta_0 = \theta_1 = 1$, **E** will stop the system and throw "Suspension".
Otherwise, **E** computes the Challenge trapdoor in the following. For simplicity, let $T_\beta$ where $\beta \in (1, 2, ...m)$ be an example:
- $T_\beta = M \bullet e(xP, yP)^{f_i}$. If $M = e(P, P)^{xyz}$, then $T_\beta = e(P, P)^{xy(z+f_i)} = e(F_i, (xy)P)$. If $M$ is a random element of $G_2$, so is $T_\beta$.
- The above calculation will be repeated $m - 1$ times until **E** creates an entire Challenge trapdoor.
**Phase 2-2 Simulation (Queries)**
**A** can continue to communicate with **E** for the oracle queries. One restriction is that **A** cannot request $[(pk_{Sen-PAEMKS}, W_0^*), (pk_{Sen-PAEMKS}, W_1^*)]$ to $O_T$ and $[(pk_{Rec-PAEMKS}, W'^*_0)(pk_{Rec-PAEMKS}, W'^*_1)]$ to $O_C$.
**Guess**
Finally, **A** shows the guess $\beta^* \in \{0, 1\}$. If $\beta = \beta^*$, "Yes". Otherwise, "No match".

**Analysis**
Two events are defined below:
*Event3*: The system does not be stopped during the *Phase 2-1* and the *Phase 2-2*.
*Event4*: The system does not be stopped during the Challenge Simulation.

**Claim 3:**
$$Pr[Event3] \geq (1 - \frac{1}{h+1})^{q_T + q_C} \tag{5}$$
The proof of **Claim 3** is the same as the proof of **Claim 1**, so it is omitted here.

**Claim 4:**
$$Pr[Event4] \geq (\frac{1}{h+1}) \bullet (\frac{h}{h+1})^{2(m-1)} \tag{6}$$
*Proof* : if $\theta_0 = \theta_1 = 1$, the m-PAEMKS system will be stopped during the Challenge Simulation. So, the probability that the system does not be stopped is $1 - (1 - \frac{1}{h+1})^2$. Besides, if $\theta_{0j}^*$ and $\theta_{1j}^*$ are equal to 0 for all $j = 0, ..., i-1, i+1, ..., m$, the system will be stopped here. Hence, the probability that the m-PAEMKS system does not be stopped during the Challenge Simulation is at least $(1 - \frac{1}{h+1})^{2(m-1)}\{1 - (1 - \frac{1}{h+1})^2\} \geq (\frac{1}{h+1}) \bullet (\frac{h}{h+1})^{2(m-1)}$.

Suppose $Event'$ is an event that the m-PAEMKS system does not stopped during the whole game. Therefore, $Pr[Event'] = Pr[Event3] \bullet Pr[Event4] =$

$(1-\frac{1}{h+1})^{q_T+q_C} \bullet (\frac{1}{h+1}) \bullet (\frac{h}{h+1})^{2(m-1)}$. If $h+1 = q_T + q_C$, the $Pr[Event]$ will reach the maximum value. So, $Pr[Event'] = \frac{1}{e} \bullet (\frac{1}{q_T+q_C}) \bullet (\frac{q_T+q_C-1}{q_T+q_C})^{2(m-1)}$, which is approximately equal to $\frac{1}{e(q_T+q_C)}$ and thus non-negligible.

Overall, the probability that **E** guesses the correct bit $\beta$ is as follows:
$Pr[\beta^{'} = \beta] = Pr[\beta^{'} = \beta \wedge Pr[Event']] + Pr[\beta^{'} = \beta \wedge Pr[\overline{Event'}]] = Pr[\beta^{'} = \beta \mid Pr[Event']]Pr[Event'] + Pr[\beta^{'} = \beta \mid Pr[\overline{Event'}]]Pr[\overline{Event'}] = \frac{1}{2} \bullet (1 - Pr[\overline{Event'}]) + (\varepsilon_T + \frac{1}{2}) \bullet Pr[\overline{Event'}] = \frac{1}{2} + \varepsilon_T \bullet Pr[\overline{Event'}]$. If $\varepsilon_T$ is non-negligible, so is $|Pr[\beta^{'} = \beta] - \frac{1}{2}|$.

In shot, the m-PAEMKS scheme satisfies Trapdoor Indistinguishability. And, the attacker cannot retrieve any one bit keyword from the Trapdoor queries.

## 5.2 Performance and Efficiency for m-PAEMKS

This section compares the security, efficiency and performance between the proposed scheme (m-PAEMKS) and its counterpart (m-PEMKS [22]).

**Table 1** Comparison of Functionalities between m-PEMKS and m-PAEMKS schemes

| Scheme | CI Ind | TI Ind | FKS | OKGA | IKGA |
|---|---|---|---|---|---|
| **m-PEMKS** | Satisfied | Satisfied | Supported | Not Suffered | Suffered |
| **m-PAEMKS** | Satisfied | Satisfied | Supported | Not Suffered | Not Suffered |

CT Ind, Trap Ind, FKS, OKGA and IKGA are the abbreviation of Ciphertext Indistinguishability, Trapdoor Indistinguishability, Fuzzy Keyword Search, Offline Keyword Guessing Attack and Inside Keyword Guessing Attack respectively. As it can be seen in Table 1, both of them could resist OKGA due to the properties of CI Ind and TI Ind. Apart from that, they all support Fuzzy Keyword Search. However, the proposed scheme is able to prevent IKGA while its counterpart suffers IKGA.

**Table 2** Comparison of Computation Efficiency between m-PEMKS and m-PAEMKS schemes

| Scheme | PEKS (E1) | Trapdoor (R1) | Test |
|---|---|---|---|
| **m-PEMKS** | E+(E+2H+P)*N | E+P+(2E+H+P)*M | (2E+H+2P)*M |
| **m-PAEMKS** | (2E+H+P)*N | (E+H+P)*M | E*M |

The symbols E, H and P stand for a modular exponentiation, a collision resistant hash function and a bilinear pairing respectively. Besides, M and N denotes the number of keywords in PEKS ciphertext (E1) and Trapdoor query

(R1). Table 2 shows the similar efficiency in $PEKS$ and $Trapdoor$ algorithms between these two PEKS schemes. But the propose scheme has better computation efficiency in $Test$ algorithm than its counterpart.

**Table 3** Comparison of Communication Efficiency between m-PEMKS and m-PAEMKS schemes

| **Scheme** | **\|PEKS PK\|** | **\|E1\|** | **\|R1\|** |
|---|---|---|---|
| **m-PEMKS** | 2\|G1\| | \|G1\|+n*N | \|G2\|+(\|G1\|+\|G2\|)*M |
| **m-PAEMKS** | \|G1\| | \|G1\|+\|G2\|*N | \|G2\|*M |

The symbols of |G1| and |G2| stand for the length of element in group |G1| and |G2|. Besides, n devotes the length of security parameter while M and N are the number of keywords in PEKS ciphertext (E1) and Trapdoor query (R1) respectively. The proposed scheme has better communication efficiency in PEKS public key generation (|PEKS PK|) and Trapdoor generation (|R1|) than its counterpart.

The m-PAEMKS scheme was implemented by JAVA with the JPBC library [30] and the jFuzzyLogic library [31,32].

The m-PAEMKS scheme applies the Single Input Single Output (SISO) Mamdani Fuzzy Inference System instead of Two or More Input Single Out (T/MISO) Mamdani Fuzzy Inference System. The reason is due to the difference between Artificial Intelligence (AI) and Cryptography. For AI, it always uses the plaintext as the input to explore the relation between the different sets. For Cryptography, it uses the ciphertext to protect data security. In general, the more plaintext are explored to the public networks, the more likely the cryptographic system might be vulnerable to some attacks, such as Chosen Plaintext Attack (CPA), Dictionary Attack.

Consider a situation: the UL company's accountant uploads the encrypted UL financial reports with the specific keywords (i.e. $E(M)||PAEMKS(UL)$ $||PAEMKS(Financial)||PAEMKS(Feb2021)$) into the cloud server. When the UL upper manager would like to achieve the "latest" UL financial reports, he/she will send a Trapdoor query ($Trapdoor(UL)||Trapdoor(Financial)||$ $Trapdoor(Latest)$) to the online server. Once the online server receives these encrypted messages, it will run $Test$ algorithm to filter irrelevant documents and finally reply to UL upper manager. Fig. 2 and Fig. 3 show the membership functions and assessed value for each input of this situation, respectively.

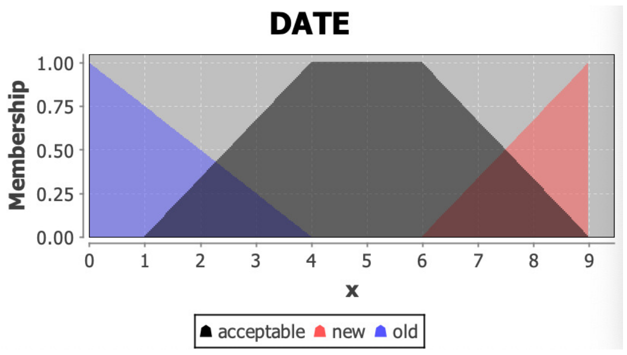

**Fig. 2** Membership functions for an example of searching "latest" financial reports

Fig. 2 shows the membership function. It fuzzifies the input variable "DATE" by "old", "acceptable" and "new".

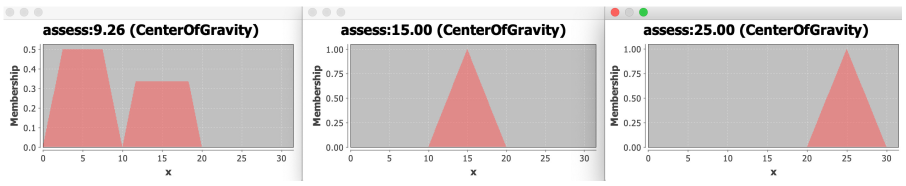

**Fig. 3** Assessed values for an example of searching "latest" financial reports

According to the Fig. 3, the Center Of Gravity (COG) method is applied to the m-PAEMKS scheme to defuzzify the output variable "access". More specially, three UL financial reports are found via "Exact Match" from the online server. By the SISO Mamdani Fuzzy Inference System in m-PAEMKS scheme, the first report partially belongs to "old" and "acceptable" report, the second one belongs to "acceptable" UL financial report and the third one completely belongs to the "latest" UL financial report. Finally, the online server will return the second and the third reports to the UL upper manager.

## 6 Conclusion

The paper firstly revisited Kazemian and Ma's m-PEMKS scheme [22] and then pointed out that the m-PEMKS scheme may suffer the Inside Keyword Guessing Attack, if the server is malicious. The paper then defined a robust m-PEMKS called *Public Key Authenticated Encryption with Multi-Keywords Search using Mamdani System (m-PAEMKS)*. The m-PAEMKS scheme incorporates with User Authentication technique so that it is able to prevent Inside Keyword Guessing Attack. Besides, the proposed scheme has the properties of Ciphertext Indistinguishability and Trapdoor Indistinguishability and is proved to be semantic security under the random oracle model. Hence, the proposed scheme is able to resist OKGA. Furthermore, Mamdani Fuzzy Inference System is applied to the proposed scheme, which can support users to search the encrypted documents by both multiple keywords and a fuzzy keyword without compromising the original data security.

## References

1. Brier, E., Clavier, C. and Olivier, F., 2004. Correlation Power Analysis with a Leakage Model. Lecture Notes in Computer Science, pp.16-29.
2. S. Chari, J. Rao, and P. Rohatgi, ?Template Attacks?, CHES 2002, Springer, 2003, LNCS 2523, pp 51?62.
3. Brier, E., Clavier, C. and Olivier, F., 2004. Correlation Power Analysis with a Leakage Model. Lecture Notes in Computer Science, pp.16-29.
4. Maghrebi, H., 2019. Deep Learning based Side Channel Attacks in Practice. Cryptology ePrint Archive, Report 2019/578.
5. Boneh, D., Di Crescenzo, G., Ostrovsky, R. and Persiano, G. (2004). Public Key Encryption with Keyword Search. Advances in Cryptology - EUROCRYPT 2004, pp.506-522.
6. Byun, J., Rhee, H., Park, H. and Lee, D. (2006). Off-Line Keyword Guessing Attacks on Recent Keyword Search Schemes over Encrypted Data. Lecture Notes in Computer Science, pp.75-83.
7. Baek, J., Safavi-Naini, R. and Susilo, W. (n.d.). Public Key Encryption with Keyword Search Revisited. Computational Science and Its Applications ? ICCSA 2008, pp.1249-1259.
8. 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.
9. 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).
10. Rhee, H., Park, J., Susilo, W. and Lee, D. (2010). Trapdoor security in a searchable public-key encryption scheme with a designated tester. Journal of Systems and Software, 83(5), pp.763-771.
11. Yau, W., Phan, R., Heng, S. and Goi, B., 2013. Keyword guessing attacks on secure searchable public key encryption schemes with a designated tester.International Journal of Computer Mathematics, 90(12), pp.2581-2587.
12. Chen YC (2015) SPEKS: secure server-designation public key encryption with keyword search against keyword guessing attacks. Computing J 58(4):922-933.
13. 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.
14. 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).
15. 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 - SIN18.
16. Li, C.-T., Lee, C.-W., Shen, J.-J.: An extended chaotic maps-based keyword search scheme over encrypted data resist outside and inside keyword guessing attacks in cloud storage services. Nonlinear Dyn. 80(3), 1601-1611 (2015).
17. Noroozi, M., Eslami, Z., and Pakniat, N. (2018). Comments on a chaos-based public key encryption with keyword search scheme. Nonlinear Dynamics, 94(2), 1127-1132.
18. 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.
19. Kazemian, H and Ma, Y. (2020). A Secure and Efficient Public Key Authenticated Encryption with Multi-keywords Search Scheme against Inside Keyword Guessing Attack. International Journal of Cyber-Security and Digital Forensics (IJCSDF), Volume 9, Issue 2, pp.90-101.

20. 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.
21. 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.
22. Takagi, T.; Sugeno, M. Fuzzy identification of systems and its applications to modeling and control. IEEE Trans. Sys. Man. Cybern. 1985, 15, 116-132.
23. 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.
24. 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.
25. Marchini, A.; Facchinetti, T.; Mistri, M. F-IND: A framework to design fuzzy indices of environmental conditions. Eco. Indic. 2009, 9, 485-496.
26. Kazemian, H. and Ma, Y., 2020. Fuzzy Logic Application to Searchable Cryptography. Proceedings of the 21st EANN (Engineering Applications of Neural Networks) 2020 Conference, pp.410-422.
27. Kazemian, H., 1998. Study of MIMO learning fuzzy controllers for dynamic system applications.
28. Boneh, D. and Boyen, X.: Secure Identity Based Encryption Without Random Oracles, Advances in Cryptology, CRYPTO 2004, vol 3152, 443-459 (2004).
29. Voskoglou, M., n.d. Fuzzy Sets, Fuzzy Logic and Their Applications.
30. De Caro, A. and Iovino, V. (2011). jPBC: Java pairing based cryptography. 2011 IEEE Symposium on Computers and Communications (ISCC).
31. Cingolani, Pablo, and Jess Alcal-Fdez. jFuzzyLogic: a Java Library to Design Fuzzy Logic Controllers According to the Standard for Fuzzy Control Programming.International Journal of Computational Intelligence Systems, Vol. 6, Supplement 1 (2013), 61-75.
32. Cingolani, Pablo, and Jesus Alcala-Fdez. jFuzzyLogic: a robust and flexible Fuzzy-Logic inference system language implementation. Fuzzy Systems (FUZZ-IEEE), 2012 IEEE International Conference on. IEEE, 2012.