Fuzzy Logic Application to Searchable Cryptography

Hassan B.Kazemian$^{1[0000-0003-3568-0715]}$ and Yang Ma$^{2[0000-0003-2445-3226]}$

^{1} School of Computing and Digital Media, London Metropolitan University, London, United Kingdom.
h.kazemian@londonmet.ac.uk

^{2} Underwriters Laboratories, UK Security Lab, Basingstoke, United Kingdom.
Yang.Ma@ul.com

Abstract. Public Key Encryption with Keyword Search (PEKS) allows users to search encrypted files by a specific keyword without compromising the original data security. Almost all current PEKS schemes enable users to search exact keyword only instead of imprecise keyword (such as "latest", "biggest", etc.). Therefore, if the keyword is fuzzy, these PEKS schemes will be terminated and then report errors. Besides, some PEKS schemes are not secure mainly because they are vulnerable to Off-line Keyword Guessing Attack (OKGA). This research paper incorporates with Mamdani Fuzzy Inference System to PEKS for supporting Fuzzy Keyword Search. Secondly, the proposed scheme is proved to be semantic secure under the random oracle models so that it is able to resist OKGA. In addition, the new scheme allows users to search multiple keywords and therefore, it could be applied to the general public networks.

Keywords: Public Key Encryption with Keyword Search (PEKS) · Off-line Keyword Guessing Attack (OKGA) · Mamdani Fuzzy Inference System.

1 Introduction

The rising popularity of cloud computing attracts companies and individuals to upload their data into the online trusted servers (i.e. cloud servers). It brings about substantial merits, such as saving local memory and reducing maintenance fee, etc. How to keep data security becomes an intractable problem. Interestingly, Public Key Encryption with Keyword Search (PEKS) protects information security and data transmission security.

Boneh et al. [1] defined the first PEKS scheme in 2004 which requires a secure channel (i.e. Secure Sockets Layer) between the server and the receiver. But building a secure channel is much expensive and unrealistic in some cases. Besides, Byun et al. [3] pointed out that the first PEKS was compromising from Off-line Keyword Guessing Attack (OKGA). In 2008, Baek et al. [2] proposed a new PEKS scheme to remove the secure channel from the first PEKS system.

But, Yau et al.[4] also found that Baek et al.'s PEKS suffers OKGA. Tang et al. [5] introduced a new PEKS scheme resisting OKGA, but the encryption algorithm is complex. Soon later, Rhee et al. [6] defined the concept of Trapdoor Indistinguishability to PEKS (called dPEKS) for preventing OKGA. However, dPEKS scheme is able to search single keyword only instead of multiple keywords so that it may not be applied to the general public networks. Meanwhile, Baek et al.'s proposed MPEKS [2] scheme to solve multiple keywords search problem. However, MPEKS also needs a secure channel. Later on, Wang et al. [7] came up with a Secure Channel Free MPEKS system to remove the secure channel and support multiple keywords search, but it suffers OKGA. Recently, PEKS witnesses a dramatic development and becomes much security and functionalities.

In practice, several keywords are distilled to represent the whole document instead of one keyword only. Besides, the user may type imprecise keyword for searching, such as "latest", "biggest", etc. Due to PEKS ciphertext may contain fuzzy keyword leading to system errors, therefore, Mamdani Fuzzy Inference method could be perfectly applied to PEKS scheme in order to solve fuzzy keyword search problem. In 1973, Lotifi Zadeh's [8] came up with new fuzzy algorithms to analyse complex systems and decision processes. Later, Ebrahim Mamdani [9] 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. However, Mamdani-style inference is not computationally efficient, Michio Sugeno [10] proposed a new fuzzy inference using a single spike (a singleton) as the rule consequent. Recently, Fuzzy sets theory has been applied successfully in many areas. Singh et al.[11] pointed out fuzzy systems could applied to classification, modelling control problems. Lermontov et al. [12] analysed water quality using fuzzy set. Meanwhile, Marchini et al. [13] proposed a framework for fuzzy indices of environmental conditions.

This paper formally defines a new PEKS scheme named *Public Key Encryption with Multi-keywords Search using Mamdani System (m-PEMKS)* and then presents a concrete construction of it. Besides, m-PEMKS is proved to be semantic secure under random oracle models so that it could resist OKGA. In addition, the proposed scheme incorporates with Mamdani System to solve fuzzy keyword search problem, which is the first paper combining Fuzzy Logic and PEKS.

2 Methodology

2.1 Bilinear pairings

Let G_1 be an additive cyclic group and G_T be a multiplicative cyclic group. g is a generator of G_1 and a prime number p is the order of G_1. Suppose a and b are the elements in Z_p. A bilinear pairing can be regarded as a map $e : G_1 \times G_1 \to G_T$, which has the following properties:
i. Bilinear: $e(aU, bV) = e(U, V)^{ab}$ for all $U, V \in G_1$ and $a, b \in Z_p$.
ii. Computable: $e(U, V) \in G_T$ is computable in a polynomial time algorithm, for

any $U, V \in G_1$.
iii. Non-degenerate: $e(U, V) \neq 1$.

2.2 The Bilinear Diffie-Hellman (BDH) assumption

Given g, xg, yg, zg as input (where $x, y, z \in Z_p$), compute $e(g, g)^{xyz} \in G_T$. An algorithm A has an advantage ε in solving BDH assumption in G_1, if $Pr[A(g, xg, yg, zg) = e(g, g)^{xyz}] \geq \varepsilon$. It is shown that BDH assumption holds in G_1 if no t time algorithm has an advantage at least ε in solving BDH assumption in G_1.

2.3 The 1-Bilinear Diffie-Hellman Inversion (1-BDHI) assumption

Given g, xg as input (where $x \in Z_p$), compute $e(g, g)^{\frac{1}{x}}$. An algorithm A has an advantage in solving 1-BDHI assumption in G_1, if $Pr[A(g, xg) = e(g, g)^{\frac{1}{x}}] \geq \varepsilon$. It is shown that 1-BDHI assumption holds in G_1 if no t time algorithm has an advantage at least ε in solving 1-BDHI assumption in G_1.

2.4 Fuzzy Rule Based Model

The fuzzy rule based model has 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.
2. *Rules evaluation*: The fuzzified inputs are applied to the antecedents of the fuzzy rules and then apply "AND" operation 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*: The defuizzification method, center of gravity (COG), is utilized to transform fuzzy outputs into crisp outputs.

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

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, it will run Test algorithm to estimate whether two ciphertexts contain the same keyword or not, and replies to the receiver in the end.

3.1 Formal Definition of m-PEMKS

The proposed scheme has eight Probabilistic Polynomial Time algorithms:
1. $KeyGen_{Param-PEMKS}(1^{\varsigma})$: Input 1^{ς} for generating a common parameter *cp*.

2. $KeyGen_{Param-RSA}(k)$: Input k for generating a global parameter gp.
3. $KeyGen_{Server-PEMKS}(cp)$: Input cp and then produce a public and private PEMKS key pair ($pk_{Ser-PEMKS}$, $sk_{Ser-PEMKS}$) of the server.
4. $KeyGen_{Server-RSA}(gp)$: Input gp and then produce a public and private RSA key pair ($pk_{Ser-RSA}$, $sk_{Ser-RSA}$) of the server.
5. $KeyGen_{Receiver-PEMKS}(cp)$: Input cp and then produce a public and private PEMKS key pair ($pk_{Rec-PEMKS}$, $sk_{Rec-PEMKS}$) of the receiver.
6. $Encryption(pk_{Ser-PEMKS}, pk_{Rec-PEMKS}, pk_{Ser-RSA}, W)$: A searchable encryption E=(E_1,E_2)=SCF-PEMKS($pk_{Ser-PEMKS}$,$pk_{Rec-PEMKS}$,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. $Request(pk_{Ser-PEMKS}, sk_{Rec-PEMKS}, pk_{Ser-RSA}, W)$: A trapdoor request R=(R_1,R_2)=Trapdoor($pk_{Ser-PEMKS}$,$sk_{Rec-PEMKS}$,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. $Test(E, R, sk_{Ser-PEMKS}, sk_{Ser-RSA})$: Test algorithm contains two parts: Exact Match and Fuzzy Match.
For Exact Match: Input the server's PEMKS private key $sk_{Ser-PEMKS}$, an encryption E_1=SCF-PEMKS($pk_{Ser-PEMKS}$,$pk_{Rec-PEMKS}$,W_{part1}) and a request R_1=Trapdoor($pk_{Ser-PEMKS}$,$sk_{Rec-PEMKS}$,W^*_{part1}). If $W^*_{part1} \in W_{part1}$, the system will go to Fuzzy Match. Otherwise, the system will terminate.
For Fuzzy Match: Input the server's RSA private key $sk_{Ser-RSA}$, an encryption E_2=RSA($pk_{Ser-RSA}$,W_{part2}) and a request R_2=RSA($pk_{Ser-RSA}$,W^*_{part2}). Then, the server decrypts E_2 and R_2 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. Next, the encrypted file is filtered by Mamdani system and the server will reply to the receiver in the end.

3.2 The Concrete Construction of m-PEMKS

The details of m-PEMKS are listed in the following (Fig. 1):
1. $KeyGen_{Param-PEMKS}(1^{\varsigma})$: Let G_1 be an additive cyclic group and G_T be a multiplicative cyclic group. g is a random generator of G_1 whose order is a prime number p. A bilinear pairing is a map $e : G_1 \times G_1 \rightarrow G_T$. Let $H : \{0,1\}^{\circ} \rightarrow G_1$ and $H^* : G_T \rightarrow \{0,1\}^{\bullet}$ be two specific hash functions. This algorithm returns the common parameter $cp = \{g, p, G_1, G_T, e, H, H^*\}$.
2. $KeyGen_{Param-RSA}(K)$: Randomly select prime numbers u and v (where $u \neq v$) and calculate $L = u \times v$ and $\phi(L) = (u-1) \times (v-1)$.
3. $KeyGen_{Server-PEMKS}(cp)$: The server randomly chooses $a \in Z_p$ and then computes $A = aP$. Besides, the server chooses $B \in G_1$ uniformly at random. Therefore, the server's PEMKS public key is $pk_{Ser-PEMKS} = (cp, A, B)$ and the PEMKS private key is $sk_{Ser-PEMKS} = (cp, a)$.
4.$KeyGen_{Server-RSA}(gp)$: The server randomly selects $x \in Z_q$, where $gcd(\phi(L), x) = 1$ and $1 < x < \phi(L)$. Next, the server calculates y by $y \equiv x^{-1}(mod \phi(L))$. Therefore, the server's RSA public key is $pk_{Ser-RSA} = (x, L)$ and the private key is $sk_{Ser-RSA} = (y, L)$.
5. $KeyGen_{Receiver-PEMKS}(cp)$: The receiver randomly chooses $c \in Z_p$ and then computes $C = cP$. Therefore, the receiver's PEMKS public key is $pk_{Rec-PEMKS} =$

(cp, C) and the PEMKS private key is $sk_{Rec} = (cp, c)$.
6. $Encryption(pk_{Ser-PEMKS}, pk_{Rec-PEMKS}, pk_{Ser-RSA}, W)$: The sender randomly chooses $t \in Z_p$ and a keyword-vector $W{=}(W_{part1}, W_{part2}){=}[(w_1,w_2,...,w_n); w_{n+1}]$. The sender then computes a searchable encryption $E = (E_1, E_2) = [(M, N_1, N_2, ..., N_n); N_{n+1}] = [(tA, H^*(D_1), H^*(D_2), ..., H^*(D_n)); (w_{n+1})^x mod\, L]$, where $D_1 = e(H(w_1), C)^t$,$D_2 = e(H(w_2), C)^t$,...,$D_n = e(H(w_n), C)^t$.
7. $Request(pk_{Ser-PEMKS}, sk_{Rec-PEMKS}, pk_{Ser-RSA}, W)$: The receiver randomly chooses $t^* \in Z_p$ and a keyword-vector $W{=}(W_{part1}, W_{part2}){=}[(w_1,w_2,...,w_m); w_{fuzzy}]$. The receiver then computes $R = (R_1, R_2) = [(Z, T_1, T_2, ..., T_m), T_{fuzzy}] = [(e(A, t^*B), cH(w_1){\oplus}e(A, B)^{t^*+c}, cH(w_2){\oplus}e(A, B)^{t^*+c}, ..., cH(w_{m-1}){\oplus}e(A, B)^{t^*+c})\, ; (w_{fuzzy})^x mod\, L]$.
8. $Test(E, R, sk_{Ser-PEMKS}, sk_{Ser-RSA})$: For $i \in \{1, 2, ..., n\}$ and $j \in \{1, 2, ..., m\}$, where $j \leq i$.
(i) *For Exact Match*: Firstly, the server calculates
$T_{w_1} = T_1 \oplus Z \bullet e(aB, C) = cH(w_1^*), ...,$
$T_{w_j} = T_j \oplus Z \bullet e(aB, C) = cH(w_j^*), ...,$
$T_{w_m} = T_m \oplus Z \bullet e(aB, C) = cH(w_m^*)$
Then, the server checks whether $H^*[e(T_{w_j}, \frac{M}{a})] = N_i$ or not. If "yes", the system will go to Fuzzy Match. Otherwise, the system will terminate.
(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.

Without loss of generality, suppose w_{fuzzy} is the keyword "latest" while w_{n+1} stands for a set of "DATE". Therefore, three rules can be defined in the following:
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.

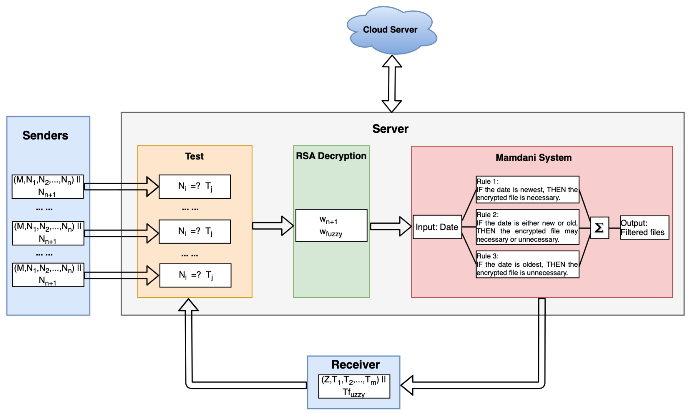
Cloud Server
Server
Senders
(M,N1,N2,...,Nn) II Nn+1
... ...
Test
Ni =? Tj
RSA Decryption
wn+1
wfuzzy
Mamdani System
Input: Date
Rule 1: IF the date is newest, THEN the encrypted file is necessary.
Rule 2: IF the date is either new or old, THEN the encrypted file may necessary or unnecessary.
Rule 3: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

Fig. 1. The concrete construction of m-PEMKS

3.3 The Correctness of m-PEMKS

For Exact Match: for $i \in \{1, 2, ..., n\}$ and $j \in \{1, 2, ..., m\}$, the proposed scheme is completely correct in the following:
Firstly,
$T_{w_j} = T_j \oplus Z \bullet e(aB, C) = cH(w_j^*) \oplus e(A, B)^{t^*+c} \oplus e(A, t^*B) \bullet e(aB, cP) = cH(w_j^*) \oplus e(A, B)^{t^*+c} \oplus e(A, B)^{t^*} \bullet e(A, B)^c = cH(w_j^*) \oplus e(A, B)^{t^*+c} \oplus e(A, B)^{t^*+c} = cH(w_j^*)$
Then,
$H^*[e(T_{w_j}, \frac{M}{a})] = H^*[e(cH(w_j^*), \frac{tA}{a})] = H^*[e(cH(w_j^*), tP)] = H^*[e(H(w_j^*), C)^t] = N_i$

For Fuzzy Match: this algorithm is still correct due to the properties of Mamdani Fuzzy Inference System.

3.4 The Security Analysis of m-PEMKS

The proposed scheme contains two cryptographic algorithms: PEKS and RSA. The security of proposed scheme mainly relies on Ciphertext Indistinguishability of Chosen Plaintext Attack (IND-CPA) and Trapdoor Indistinguishability of Chosen Plaintext Attack (Trapdoor-IND-CPA).

IND-CPA security is that a malicious server (**Game1**) could not decide which PEMKS ciphertext contains which encrypted keyword, if it has not received the Trapdoor containing the given keyword. Besides, if a malicious receiver (**Game2**) that has not obtained the server's PEMKS private key cannot check whether PEMKS ciphertext and Trapdoor have the same keyword, even if he/she intercepts all Trapdoors for any specific keyword. For **Trapdoor-IND-CPA** security, it is an outside attacker excluding the server and the receiver (**Game3**) cannot differentiate any difference between Trapdoors containing the same keyword. To conclude, the proposed scheme satisfies Ciphertext Indistinguishability and Trapdoor Indistinguishability against a Chosen Plaintext Attack (CPA).

Theorem 1. The m-PEMKS above is IND-CPA secure against CPA in Game1 under the random oracle model assuming that BDH assumption is intractable.

Game1: A is supposed to be a malicious server.
Proof: Suppose that **E** has $(g, p, G_1, G_T, e, xg, yg, zg)$ as an input of BDH assumption whose running time is bounded by T. **E**'s aim is to calculate a BDH key $e(g, g)^{xyz}$ of xg, yg and zg using **A**'s IND-CPA. Besides, **A** asks for at most h and h^* times for the queries of H and H^* hash functions.

Setup Simulation
E firstly sets $C = xg$ and randomly selects $a \in Z_p$ and then calculates $A = ag$. **E** also picks up $B \in G_1$ uniformly at random. After that, **A** obtains the common parameter $(g, p, G_1, G_T, e, H, H^*)$, the server's PEMKS public key (cp,A,B) and

PEMKS private key (cp,a) and the receiver's PEMKS public key (cp,C). Besides, **E** chooses two hash functions H and H^* in the following:
- **A** can query a keyword w_i to H function at any time. To respond, **E** searches H_List for a tuple $(w_i, F_i, f_i, \theta_i)$ and the H_List is empty in original. If the tuple exists, **A** will receive $H(w_i) = F_i$ as a response. Otherwise, **E** does the following steps:
i. **E** picks up a coin θ_i uniformly at random and then calculates $Pr[\theta_i = 0] = \frac{1}{h+1}$.
ii. **E** selects $f_i \in Z_p$ uniformly at random. If $\theta_i = 0$, **E** will calculate $F_i = yg + f_i g$. If $\theta_i = 1$, **E** will calculate $F_i = f_i g$.
iii. **E** returns F_i as an answer to **A** and adds $(w_i, F_i, f_i, \theta_i)$ into H_List.
- **A** queries D_i to H^* function at any time. Then, **E** searches H^*_List for a tuple (D_i, N_i). If the tuple exists, **A** will receive N_i as a response. Otherwise, **E** selects $N_i \in \{0,1\}^\bullet$ uniformly at random and then returns it to **A** and also adds (D_i, N_i) into H^*_List.

Phase 1-1 Simulation (Trapdoor queries)
A issues a query for the trapdoor corresponding to the keyword-vector $W_l = (w^*_{l1}, w^*_{l2}, ..., w^*_{lm})$. To respond, **E** executes the following steps:
- **E** randomly selects $i' \in \{1, 2, ..., m\}$
- **E** runs the above algorithms for simulating H function to create a tuple $(w_{li'}, F_{li'}, f_{li'}, \theta_{li'})$. If $\theta_{li'} = 0$, **E** will output "Suspend" and terminate the system. Otherwise, **E** conducts the following:
• **E** selects $t^* \in Z_p$ and then computes $T_1 = f_{l1}C \oplus e(A,B)^{t^*+x} = f_{l1}xg \oplus e(A,B)^{t^*+x} = xF_{l1} \oplus e(A,B)^{t^*+x} = xH(w_{l1}) \oplus e(A,B)^{t^*+x}$, $T_2 = xH(w_{l2}) \oplus e(A,B)^{t^*+x}$,...,$T_m = xH(w_{lm})$ and $Z = e(A, t^*B)$. Therefore, $T_W = (Z, T_1, T_2, ..., T_m)$.

Challenge Simulation
A sends W_0=(w_{01},w_{02},...,w_{0n}) and W_1=(w_{11},w_{12},...,w_{1n}) to **E**. Once **E** receives the target keyword-vector pair, he/she does the following:
- **E** randomly selects $i \in \{1, 2, ..., n\}$.
- **E** runs the above algorithms for simulating H function to obtain two vectors of tuples $(W^*_{0i}, F^*_{0i}, f^*_{0i}, \theta^*_{0i})$ and $(W^*_{1i}, F^*_{1i}, f^*_{1i}, \theta^*_{1i})$. If θ^*_{0i} and θ^*_{1i} are equal to 1, **E** will output "Suspend" and terminate the system. Otherwise, **E** runs the above algorithms for simulating H function at $2(n-1)$ times to obtain two vectors of tuples $((w^*_{01}, F^*_{01}, f^*_{01}, \theta^*_{01}), ..., (w^*_{0i-1}, F^*_{0i-1}, f^*_{0i-1}, \theta^*_{0i-1}), (w^*_{0i+1}, F^*_{0i+1}, f^*_{0i+1}, \theta^*_{0i+1}), ..., (w^*_{0n}, F^*_{0n}, f^*_{0n}, \theta^*_{0n}))$ 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 θ^*_{0i} and θ^*_{1i} are equal to 0 for all $i = 0, ..., i-1, i+1, ..., n$, **E** will output "Suspend" and terminate the system. Otherwise, **E** does the following:
– **E** chooses $\beta \in \{0,1\}$ uniformly at random.
– **E** chooses $N_i \in \{0,1\}^\bullet$ uniformly at random and creates a target SCF-PEMKS Ciphertext $S^* = (M^*, N^*_1, N^*_2, ..., N^*_n) = (zA, H^*[J_1], H^*[J_2], ..., H^*[J_n])$
So, $S^* = (M^*, N^*_1, ..., N^*_{i-1}, N^*_{i+1}, ..., N^*_n) = (zA, H^*[e(H(w_{\beta_1}), C)^z], ..., H^*[e(H(w_{\beta_{i-1}}), C)^z], H^*[e(H(w_{\beta_1+1}), C)^z], ..., H^*[e(H(w_{\beta_n}), C)^z])$
Note that $J_i = e(H(w_{\beta_i}), C)^z = e(yg + f_{\beta_i}g, xg)^z = e(yg, xg)^z \bullet e(f_{\beta_i}g, xg)^z$

$= e(g,g)^{xyz} \bullet e(zg,xg)^{f_{\beta i}}$
Note also that $e(f_{\gamma_i}g,xg)^z = e(f_{\gamma_i}g,C)^z = e(H(w_{\gamma_i}),C)^z$
Phase 1-2 Simulation (Trapdoor queries)
A can continue to ask **E** for Trapdoor queries for the keyword-vector W_i. **E** answers to **A** as in Phase 1-1, as long as $w_i \notin W_0, W_1$.
Guess
A outputs the guess $\beta^* \in \{0,1\}$. Then, **E** selects d in the list for H^* function and returns $\frac{d_{\beta_i^*}}{e(zg,xg)^{f_{\beta_i^*}}}$ as its guess for BDH key.

Analysis of Game1
Let *Event1* and *Event2* be events that **E** does not suspend during Phase 1-1 and Phase 1-2 (Trapdoor queries) and **E** does not suspend during Challenge Simulation respectively. Therefore, the probability of *Event1* happening is at least $[(1-\frac{1}{h+1})^m]^h \geq \frac{1}{e^m}$. Besides, the probability of *Event2* happening is at least $(1-\frac{1}{h+1})^{2(n-1)}\{1-(1-\frac{1}{h+1})^2\} \geq (\frac{1}{h+1}) \bullet (\frac{h}{h+1})^{2(n-1)}$. In addition, let $Hybrid_r$ for $r \in \{1,2,...,n\}$ be an *event* that the attacker **A** can successfully guess the keyword of the left part of a "hybrid" PEMKS Ciphertext formed with r, coordinates from W_β followed by $(n-r)$ coordinates from $W_{1-\beta}$. Consequently, $Pr[Event3] = 2\Sigma_{k=1}^{n}(Pr[Hybrid_r] - Pr[Hybrid_{r-1}]) = 2(Pr[Hybrid_r] - Pr[Hybrid_0]) = 2\varepsilon$. However, the probability that **A** requests a query for either $H^*(e(H(W_{0i}^*),C)^z)$ or $H^*(e(H(W_{1i}^*),C)^z)$ is at least 2ε, so the probability that **A** issues a query for $H^*(e(H(W_i^*),C)^z)$ is at least ε. In total, **E**'s success probability ε^* is $(\frac{h}{h+1})^{2(n-1)} \bullet \frac{\varepsilon}{e^m(h+1)h^*}$, which is negligible.

Theorem 2. The m-PEMKS above is IND-CPA secure against CPA in Game2 under the random oracle model assuming that 1-BDHI assumption is intractable.

Game2: **A** is supposed to be a malicious receiver.
Proof: Suppose that **E** has (g,p,G_1,G_T,e,xg) as an input of 1-BDHI assumption whose running time is bounded by T. **E**'s aim is to calculate a 1-BDHI key $e(g,g)^{\frac{1}{x}}$ of xg using **A**'s IND-CPA. Besides, **A** asks for at most h and h^* times for the queries of H and H^* hash functions.

Setup Simulation
E firstly sets $A = xg$ and $B \in G_1$. **E** also selects $c \in Z_p$ uniformly at random and calculates $C = cP$. Then, **A** obtains the common parameter (g,p,G_1,G_T,e,H,H^*), the server's PEMKS public key (cp,A,B), the receiver's PEMKS public key (cp,C) and PEMKS private key (cp,c). Besides, **E** chooses two hash functions H and H^* in the following:
– **A** can query a keyword w_i to H function at any time. To respond, **E** selects $f_i \in Z_p$ uniformly at random and then calculates $F_i = f_i g$ and finally returns F_i as a response to **A**.
– **A** can query D_i to H^* function at any time. Then, **E** searches H^*_List for a tuple (D_i, N_i). If the tuple exists, **A** will receive N_i as a response. Otherwise, **E** selects $N_i \in \{0,1\}^\bullet$ uniformly at random and then sends it to **A**. **E** also adds

(D_i, N_i) into H^*_List.

Challenge Simulation

A sends $(W^*_{0i}, F^*_{0i}, f^*_{0i}, \theta^*_{0i})$ and $(W^*_{1i}, F^*_{1i}, f^*_{1i}, \theta^*_{1i})$ to **E**, where $W^*_0 = (w_{01}, w_{02}, ...,$ $w_{0n})$ and $W^*_1 = (w_{11}, w_{12}, ..., w_{1n})$. **E** randomly chooses $\beta \in \{0, 1\}$ and $N_i \in \{0,1\}^\bullet$. Then, **E** creates a target PEMKS Ciphertext $S^* = (M^*, N^*_1, N^*_2, ..., N^*_n)$ $= (\psi xg, H^*[J_1], H^*[J_2], ..., H^*[J_n])$

So, $S^* = (M^*, N^*_1, N^*_2, ..., N^*_n) = (\psi xg, H^*(e(H(w_{\beta_1}), C)^\psi), H^*(e(H(w_{\beta_2}), C)^\psi)$ $, ..., H^*(e(H(w_{\beta_n}), C)^\psi))$

Notice that $e(H(w_{\beta^*_i}), C)^\psi) = e(f_i g, cg)^\psi = e(g, g)^{\psi \cdot f_i c}$.

Guess

A outputs the guess $\beta^* \in \{0, 1\}$. Then, **E** returns $\psi = \frac{1}{x \cdot f_i c}$ as the guess for 1-BDHI key.

Analysis of Game2

Let *Event4* and *Event5* be events that **E** does not suspend during Challenge Simulation and **A** does not issue a query for either one of $H^*(e(H(W^*_{0i}), C)^\psi)$ or $H^*(e(H(W^*_{1i}), C)^\psi)$ respectively. So, the probability of *Event4* happening is equal to 1. Besides, according to Bayes's rule and the definition above, the probability of *Event5* happening is at least 2ε and therefore, the probability that **A** issues a query for $H^*(e(H(W^*_i), C)^\psi)$ is at least ε. Therefore, $e(H(W^*_j), C)^\psi = e(g, g)^{\psi \cdot f_i c}$ will appear in H*_List. Due to **A** asks for at most h^* times H^* hash function queries, the probability that **E** selects the correct answer is at least $\frac{1}{h^*}$. In total, E's success probability ε^* is $\frac{\varepsilon}{h^*}$, which is negligible.

Theorem 3. The m-PEMKS above is Trapdoor-IND-CPA secure against CPA in Game3 under the random oracle model assuming that BDH assumption is intractable.

Game3: **A** is supposed to be an outside attacker excluding the server and the receiver.

Proof: Suppose that **E** has $(g, p, G_1, G_T, e, xg, yg, zg)$ as an input of BDH assumption whose running time is bounded by T. **E**'s aim is to calculate a BDH key $e(g, g)^{xyz}$ of xg, yg and zg using **A**'s Trapdoor-IND-CPA. Besides, **A** asks for at most h and h^* times for the queries of H and H^* hash functions.

Setup Simulation

E firstly sets $A = xg, B = yg$, $C = zg$ and returns (cp, A, B) as the server's PEMKS public key and (cp, C) as the receiver's PEMKS public key. **E** also chooses two H and H^* hash functions at random.

Phase 3-1 Simulation (Trapdoor queries)

A issues a query for the trapdoor corresponding the keyword-vector W_i, where $i \in \{1, 2, ..., m\}$. To respond, **E** chooses $t^* \in Z_p$ uniformly at random. Then, **E** computes $T_1 = zH(w_{i1}) \oplus e(yg, xg)^{t^*+z}, T_2 = zH(w_{i2}) \oplus e(yg, xg)^{t^*+z}, ..., T_m = zH(w_{im}) \oplus e(yg, xg)^{t^*+z}$ and $Z = e(t^* yg, xg)$. So $T_W = (Z, T_1, T_2, ..., T_m)$. Finally, **E** returns T_W to **A**.

Challenge Simulation
A sends (W_0^*, W_1^*) to **E**, where $W_0^* = (w_{01}, w_{02}, ..., w_{0m})$, $W_1^* = (w_{11}, w_{12}, ..., w_{1m})$.
E creates the challenge Trapdoor request as follows:
E randomly selects a bit $\beta \in \{0,1\}$. Therefore, $T_1 = zH(w_{\beta_1^*}) \oplus e(yg, xg)^{t^*+z} = zH(w_{\beta_1^*}) \oplus e(g,g)^{xyz} \bullet e(g,g)^{xyt^*}$, $T_2 = zH(w_{\beta_2^*}) \oplus e(g,g)^{xyz} \bullet e(g,g)^{xyt^*}$, ..., $T_m = zH(w_{\beta_m^*}) \oplus e(g,g)^{xyz} \bullet e(g,g)^{xyt^*}$, $R = e(t^*yg, xg)$.
Phase 3-2 Simulation (Trapdoor queries)
A can continue to ask Trapdoor queries for the keyword-vector W_i. While, **E** answers to **A** as in Phase 3-1, as long as $W_i \neq W_0, W_1$.
Guess
A outputs the guess $\beta^* \in \{0,1\}$. If $\beta = \beta^*$, **E** outputs "yes", otherwise, **E** outputs "no".

Analysis of Game3
Due to **A** is a malicious outside attacker, he/she cannot distinguish any difference between two Trapdoors even though these two Trapdoors have the same keyword. The reason is that **E** randomly chooses $t^* \in Z_p$ and t^* changes every time leading to $T_i = cH(w_{\beta i}) \oplus e(A,B)^{t^*+c}$ changes every time. Even if two Trapdoors have the same keyword, the results are still different because of t^*. Therefore, the key part of Trapdoor Indistinguishability in this proposed scheme is the confidentiality of $e(A,B)^{t^*+c}$.

Suppose the attacker **A** obtains the value of $e(A,B)^{t^*+c}$, he/she can distinguish whether two Trapdoors have the same keyword. The reason is that the attacker **A** only calculates one extra XOR operation as $T_i = cH(w_{\beta i}) \oplus e(A,B)^{t^*+c} \oplus e(A,B)^{t^*+c} = cH(w_{\beta i})$. Therefore, the attack **A** can distinguish that $T_{w_{0i}} = cH(w_{0i})$ and $T_{w_{1i}} = cH(w_{1i})$ are equal as long as $w_{0i} = w_{1i}$. Consequently, the attack A could distinguish two Trapdoors T_{W_0} and T_{W_1}. However, according to Challenge Simulation in Game3, it is easy to acquire $e(A,B)^{t^*+c} = e(g,g)^{xyz} \bullet e(g,g)^{xyt^*}$, which satisfies BDH assumption. Hence, the attacker **A** cannot calculate the value of $e(A,B)^{t^*+c}$ and therefore, it cannot compute $T_i = cH(w_{\beta i}) \oplus e(A,B)^{t^*+c}$.

4 The Performance of m-PEMKS

The proposed system is implemented by JAVA, which requires two libraries in the following: JPBC library [14] and jFuzzyLogic library [15, 16].

The proposed scheme applies the Single Input Single Output (SISO) Mamdani Fuzzy Inference System. The is because of the properties of Artificial Intelligence and Cryptography. Artificial Intelligence explores and analyses the data for discovering the relationships between the different data sets. On the contrary, the purpose of cryptography is hiding as much as possible information. Besides, 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 data will be exposed to the public networks and therefore, crackers are able to launch attacks to recover more information. Fig. 2 shows the membership functions for an example of searching "latest" financial reports and Fig.3 shows the assessed value for each input. More specially, three senders upload the financial reports with different dates to the server by m-PEMKS system. Once the server receives them, it will run *Test* algorithm incorporating with SISO Mamdani Fuzzy Inference System to filter the "latest" reports. By Fig.3, it can be seen that the first report partly belongs to the old and acceptable financial report while the second report belongs to the acceptable financial report. However, the third report completely belongs to the "latest" financial report.

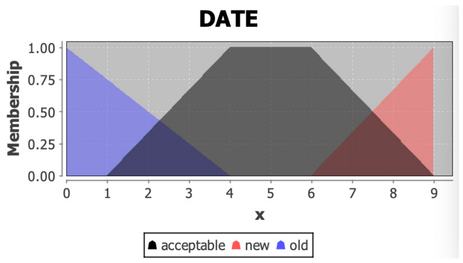

Fig. 2. Membership functions for an example of searching "latest" financial reports

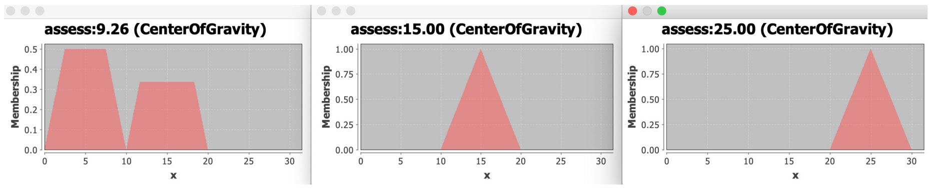

Fig. 3. Assessed values for an example of searching "latest" financial reports

5 Conclusion

In this paper, a novel and a robust *Public Key Encryption with Multiple Keywords Search using Mamdani System(m-PEMKS)* scheme is presented. The new scheme is proved to be semantic secure in the random oracle models under BDH and 1-BDHI assumptions and also satisfies the properties of Ciphertext Indistinguishability and Trapdoor Indistinguishability and therefore, it is able to resist Off-line Keyword Guessing Attack. Furthermore, Single Input Single Output Mamdani technique is applied to m-PEMKS so that it has the ability to solve fuzzy and imprecise keywords, such as "latest" and "tallest", etc., as described in the paper.

References

1. 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.
2. 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.
3. 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.
4. 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.
5. Tang, Q. and Chen, L. (2010). Public-Key Encryption with Registered Keyword Search. Public Key Infrastructures, Services and Applications, pp.163-178.
6. 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.
7. 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).
8. 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.
9. 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.
10. Takagi, T.; Sugeno, M. Fuzzy identification of systems and its applications to modeling and control. IEEE Trans. Sys. Man. Cybern. 1985, 15, 116?132.
11. 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.
12. 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.
13. Marchini, A.; Facchinetti, T.; Mistri, M. F-IND: A framework to design fuzzy indices of environmental conditions. Eco. Indic. 2009, 9, 485?496.
14. De Caro, A. and Iovino, V. (2011). jPBC: Java pairing based cryptography. 2011 IEEE Symposium on Computers and Communications (ISCC).
15. Cingolani, Pablo, and Jesús Alcalá-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.
16. 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.