Efficient Functional Encryption and Proxy Re-cryptography for Secure Public Cloud Data Sharing

Hisham Abdalla1[1, 2], Xiong Hu[1], Abubaker Wahaballa[1], Nabeil Eltayieb [1, 2], Mohammed Ramadan[1, 2], Qin Zhiguang[1]

[1] School of Computer Science and Software Engineering
University of Electronic Science and Technology of China
Chengdu, China

[2] Electrical and computer department, College of Engineering
Karary University, Khartoum, 12304, Sudan
Email: hisham_awaw@hotmail.com

***Abstract*—with the rapid development in cloud computing, public cloud data sharing concept has been extending and expanding by many scholars. In 2014, Liang et al. proposed a general notion for proxy re-encryption (PRE), which is called deterministic finite automata-based functional PRE (DFA-based FPRE). Motivated by the issues of efficiency and flexibility associated with this new primitive, we propose outsourcing decryption scheme to increase the flexibility of users by delegating their decryption rights to the semi trusted proxy. Through analysis we showed that our scheme is provably secure and efficient.**

Keywords-Deterministic Finite Automata (DFA), Proxy re-encryption (PRE), Outsourcing Decryption, Cloud Computing.

I. INTRODUCTION

With the rapid development in availability of cloud services, the techniques for securely outsourcing the prohibitively expensive computation to untrust servers are brings more attention in the scientific community. In the outsourcing computation paradigm, the resource-constrained devices can enjoy the unlimited computation resources in a pay-per-use manner, which allows users to concentrate on their core business issues rather than incurring substantial hardware, software, or personal costs. However, owners still have to remain cautious to protect their data from being pirated [1]. The cloud service provider is semi-trusted [2]. In this sense, the semi-trusted cloud service provider follows the normal flow of the protocol in the system. For instance, during the interaction with the users, a CSP may collect user's personal information and consumption profiles, which inspires a serious security concern for cloud user. Proxy re-encryption (PRE) technique [3] is devised to prevent the CSP from accessing the data in semi-trusted cloud environment.

Functional Encryption (FE) is a useful cryptographic primitive that guarantees the confidentiality of data and also enhances the flexibility of data sharing as stated in [4] and [5]. Recently (Kaitai Liang et al., 2014 [6]) introduced a general notion for proxy re-encryption (PRE), which is called deterministic finite automata-based functional PRE (DFA-based FPRE). In their scheme, a message is encrypted in a ciphertext associated with an arbitrary length index string w, and a decryptor is legitimate if and only if a DFA associated with his/her secret key accepts the string w. Furthermore, the above encryption is allowed to be transformed to another ciphertext associated with a new string by a semi trusted proxy to whom a re-encryption key is given. However, the proxy cannot gain access to the underlying plaintext. In addition, their scheme having proved to be secure against CCA in the standard model. In this paper, based on the new primitive introduced by Liang et al., we propose outsourcing decryption scheme to increase the flexibility of users by delegating their decryption rights to the semi trusted proxy. Our scheme can be proved as fully chosen-ciphertext secure in the standard model.

The rest of this paper is organized as follows. In next Section the preliminaries required in this paper are presented. Our scheme based on DFA-based FPRE scheme is presented in Section [3]. Analysis of our scheme is discussed in Section [4]. Finally, the conclusion and future work are introduced in Section [5].

II. PRELIMINARIES

Our scheme relies on a DFA-based functional proxy re-encryption scheme. We will briefly introduce the DFA-based functional proxy re-encryption scheme and the groups underlying our encryption scheme.

A. Composite Order Bilinear Groups

Let $\mathbb{G}$ and $\mathbb{G}_T$ be multiplicative cyclic groups of same order $N = P_1P_2P_3$ (where P_1, P_2, P_3 are distinct primes). We call a map $e: \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ bilinear if it should satisfy the following properties [7]:

Bilinear. $e(g^a, h^b) = e(g, h)^{ab}$, $\forall\, g, h \in \mathbb{G}$

And $\forall\, a, b \in \mathbb{Z}_N^*$; and Non-degenerate. There exists $g \in \mathbb{G}$ such that $e(g, g)$ is a generator of $\mathbb{G}_T$.

We denote by G_{P_1}, G_{P_2} and G_{P_3} the subgroups of $\mathbb{G}$ of respective orders P_1, P_2 and P_3.

B. Complexity Assumptions

Definition 1 (The Source Group l-Expanded Bilinear Diffie-Hellman Exponent (l-Expanded BDHE) Assumption in a Subgroup [6]).

978-1-4673-7755-3/16/$31.00 ©2016 IEEE

Given a group generator $\mathscr{g}$ and a positive integer l, we define the following distribution:

$N = (P_1P_2P_3, \mathbb{G}, \mathbb{G}_T, \mathrm{e}) \leftarrow \mathscr{g}, g_1 \overset{R}{\leftarrow} \mathbb{G}_{P_1}$,

$g_2 \overset{R}{\leftarrow} \mathbb{G}_{P_2}, g_3 \overset{R}{\leftarrow} \mathbb{G}_{P_3}, a, b, d, m, n, c_0, \dots, c_{l+1} \overset{R}{\leftarrow} \mathbb{Z}_N$,

$D = (N, \mathbb{G}, \mathbb{G}_T, \mathrm{e}, \mathrm{g}_1, g_2, g_3, {g_2}^{a}, {g_2}^{b}, {g_2}^{ab/dx}, {g_2}^{b/dx}, {g_2}^{ab/x}, {g_2}^{n})$,

$\forall\, i \in [0, 2l+1], i \neq l+1, j \in [0, l1] {g_2}^{a^i mn}, {g_2}^{a^i bmn/c_j x}$,

$\forall\, i \in [0, l+1] {g_2}^{c_i}, {g_2}^{a^i d}, {g_2}^{abc_i/dx}, {g_2}^{bc_i/dx}$,

$\forall\, i \in [0, 2l+1], i \neq l+1, j \in [0, l+1] {g_2}^{a^i bd/c_j x}$,

$\forall\, i, j \in [0, l+1], i \neq j {g_2}^{a^i bc_j/c_i x}),\ T_0 = {g_2}^{a^{l+1}bm}, T_1 \overset{R}{\leftarrow} \mathbb{G}_{P_2}$,

The advantage of an algorithm $\mathcal{A}$ in breaking this assumption is ${Adv_{\mathcal{A}}}^{l-BDHE}(1^n) = |\Pr[\mathcal{A}(D, T_0) = 1] - \Pr[\mathcal{A}(D, T_1) = 1]|$. We say that $\mathscr{g}$ satisfies the l-Expanded BDHE Assumption if ${Adv_{\mathcal{A}}}^{l-BDHE}(1^n)$ is negligible for any PPT algorithm $\mathcal{A}$.

Definition 2 (The Source Group Modified q Bilinear Diffie-Hellman Exponent (q-BDHE) Assumption in a Subgroup [6]).

Given a group generator $\mathscr{g}$, we define the following distribution:

$N = (P_1P_2P_3, \mathbb{G}, \mathbb{G}_T, \mathrm{e}) \leftarrow \mathscr{g},\ g_1 \overset{R}{\leftarrow} \mathbb{G}_{P_1},\ g_2 \overset{R}{\leftarrow} \mathbb{G}_{P_2},\ g_3 \overset{R}{\leftarrow} \mathbb{G}_{P_3},\ c, a, e, f \overset{R}{\leftarrow} \mathbb{Z}_N$,

$D = (N, \mathbb{G}, \mathbb{G}_T, e, g, g_2, g_3, {g_2}^{e}, {g_2}^{a}, {g_2}^{eaf}, {g_2}^{c+f/c}, {g_2}^{c^2}, \dots, {g_2}^{c^q}, {g_2}^{1/ac^q},\ T_0 = {g_2}^{aec^{q+1}}, T_1 \overset{R}{\leftarrow} \mathbb{G}_{P_2}$.

The advantage of an algorithm $\mathcal{A}$ in breaking this assumption is ${Adv_{\mathcal{A}}}^{q-BDHE}(1^n) = |\Pr[\mathcal{A}(D, T_0) = 1] - \Pr[\mathcal{A}(D, T_1) = 1]|$. We say that $\mathscr{g}$ satisfies the Source Group Modified q-BDHE Assumption if ${Adv_{\mathcal{A}}}^{q-BDHE}(1^n)$ is negligible for any PPT algorithm $\mathcal{A}$.

A DFA-Based Functional Proxy Re-Encryption Scheme

For more details we refer the reader to [5] for the definition of DFA and DFA-based FE. The DFA-based functional proxy re-encryption scheme consists of the following seven algorithms [6]:

$(PP, MSK) \leftarrow Setup(1^n, \Sigma)$: The system setup algorithm takes a security parameter n and the description of a finite alphabet Σ as input. It outputs the public parameters PP and a master key MSK, where $n \in N$. Here, we note that PP implicitly includes Σ . $SK_M \leftarrow k.Gen(MSK, M = (Q, \tau, g_0, F))$: The key generation algorithm takes the master key MSK and a DFA description M as input. It outputs a private key SK_M, where Q is a set of states, τ is a set of transitions, $g_0 \in Q$ is a start state and $F \subseteq Q$ is a set of accept states.

$RK_{M\to w} \leftarrow ReKeyGen(SK_M, w)$: This algorithm takes SK_M for a DFA description M and an arbitrary length string $w \in \sigma$ as input. It outputs a re-encryption key $RK_{M\to w}$. Using the re-encryption key any ciphertext under a string w' (in which $ACCEPT(M, w')$), it can converted to another ciphertext under w.

$CT \leftarrow DFA.E(PP, w, m)$: The encryption algorithm takes the public parameters PP, a message m and a $w \in \sigma$ as input. It outputs the ciphertext CT under w.

$CT^R \leftarrow DFA.E(PP, w, m)$: The encryption algorithm takes $RK_{M\to w}$ and CT (under w'). If $ACCEPT(M, w')$, it outputs the ciphertext CT^R under w.

${}^{m}/_{\perp} \leftarrow DFA.D(SK_M, CT)$: The decryption algorithm takes a secret key SK_M and ciphertext CT (under w) as input. The decryption can be done if $ACCEPT(M, w)$, then it outputs a message m; otherwise, outputs an error symbol $\perp$.

${}^{m}/_{\perp} \leftarrow DFA.D_R(SK_M, CT^R)$: The decryption algorithm takes a secret key SK_M and ciphertext CT^R (under w) as input. The decryption can be done if $ACCEPT(M, w)$, then it outputs a message m; otherwise, outputs an error symbol $\perp$.

III. Proposed scheme

A. Intuition

- To tackle the focusing issue on efficiency, our proposed construction operates as follows: In the key generation phase, the secret key SK_M is divided in to two parts denoted as $SK_M = (\{M, \varepsilon, \Omega\}, DK)$ for each user.
- In transformation phase, the user first sends his partial decryption key DK to the semi trusted proxy, which is part of his private key. Then semi trusted proxy transforms the ciphertext CT_m to $CT_{m'}$ and finally sends $CT_{m'}$ back to the user.
- In decryption phase, upon receiving the $CT_{m'}$ value the user can retrieve the plain massage m with only one exponential operation instead of 12 pairing operations. Since the computation of bilinear pairings has been considered the prohibitively expensive operation [8] [9], therefore our scheme is considered as more efficient.

B. Concrete Construction

In this section, we will explain a detailed construction for the proposed scheme as follows:

1) System setup

$Setup(1^n, \Sigma)$ The setup algorithm selects random group elements $g, g_0, z, h_0 \in \mathbb{G}_{P_1}$ and randomly chooses an exponents $\alpha, k, a, b, \alpha_{End}, \alpha_{Start} \in \mathbb{Z}_N^*$. Then set $H_{Start} = g^{\alpha_{Start}}$, $H_{End} = g^{\alpha_{End}}$ and $H_k = g^k$. In addition, $\forall \sigma \in \Sigma$ it chooses random $\alpha_\sigma \in \mathbb{Z}_N^*$ and set $H_\sigma = g^{\alpha_\sigma}$. After that it choose a one-time symmetric encryption scheme $Sym = (sym.Enc, sym.Dec)$, a one-time signature scheme Ots and two target collision resistant (TCR) hash functions namely H_1 and H_2, where $H_1: \mathbb{G}_T \to \mathbb{Z}_N^*$ and $H_2: \mathbb{G}_T \to \{0,1\}^{poly(n)}$. Finally, the public authority publishes the public parameters $PP = \{e(g, g)^\alpha, g, g^{ab}, z, h_0, H_{Start}, H_{End}, H_k, \forall \sigma \in \Sigma\ H_\sigma, Sym, Ots, H_1, H_2\}$ along with the description of the group $\mathbb{G}$ and the alphabet Σ, while the $MSK = (g^{-\alpha}, X_3)$, is kept secretly. Here, X_3 is a generator of $\mathbb{G}_{P_3}$.

2) Key Generation

$k.Gen(MSK, M = (Q, \tau, g_0, F)$

The key generation algorithm takes the master key MSK and a DFA description M as input. It outputs a private key SK_M, where Q, is a set of states $q_0, \ldots, q_{|Q|-1}, \tau$ is a set of transitions, for each transition $t \in T$ is a triple $(x, y, \sigma) \in Q \times Q \times \Sigma$. $q_0 \in Q$, is a start state and $F \subseteq Q$ is a set of accept states. The algorithm chooses random group elements $D_0, D_1, \ldots, D_{|Q|-1} \in \mathbb{G}_{P_1}$, where D_i is associated with state q_i, for each transition $t \in T$ it randomly selects $r_t \in \mathbb{Z}_N^*$, for all $q_x \in F$ it randomly selects $r_{End_x} \in \mathbb{Z}_N^*$, and selects $u \in \mathbb{Z}_N^*$. It also randomly selects.
$R_{Start_1}, R_{Start_2}, R_{Start_3}, R_{t,1}, R_{t,2}, R_{t,3}, R_{End_{x,1}}, R_{End_{x,2}} \in \mathbb{G}_{P_3}$, it also selects randoms $\varepsilon, \Omega \in \mathbb{G}_T$ and random $r_{Start} \in \mathbb{Z}_N^*$. The algorithm computes the private key as follows.

Firstly it computes:

$K_{Start_1} = D_0 \cdot (H_{Start})^{r_{Start}} \cdot R_{Start_1}$,

$K_2 = g^{r_{Start}} \cdot R_{Start_2}$,

$K_2 = g^u \cdot R_{Start_3}$.

Secondly, for each transition $t = (x, y, \sigma) \in \tau$ it computes:

$K_{t,1} = {D_x}^{-1} \cdot z^{r_t} \cdot R_{t,1}$,

$K_{t,2} = g^{r_t} \cdot R_{t,2}$,

$K_{t,3} = D_y \cdot (H_\sigma)^{r_t} \cdot R_{t,3}$.

Thirdly, for all $q_x \in F$ the algorithm sets:

$K_{End_{x,1}} = g^{-\alpha} \cdot D_x \cdot (H_{End} \cdot g^{ab})^{r_{End_x}} \cdot g^{ku} \cdot R_{End_{x,1}}$,

$K_{End_{x,2}} = g^{r_{End_x}} \cdot R_{End_{x,2}}$.

Finally, generates the SK_M, and sends it to the authorized user in secure channel. SK_M, is denoted as:

$SK_M = (\{M, \varepsilon, \Omega\}, DK)$, where $DK = (DK_1 = {K_{Start_1}}^{H_1(\varepsilon)},\ DK_2 = {K_{Start_2}}^{H_1(\varepsilon)}, DK_3 = {K_{Start_3}}^{H_1(\varepsilon)}$, $t \in \tau(DK_{t,1} = {K_{t,1}}^{H_1(\varepsilon)},\ DK_{t,2} = {K_{t,2}}^{H_1(\varepsilon)}, DK_{t,3} = {K_{t,3}}^{H_1(\varepsilon)})$, $q_x \in F(DK_{End_{x,1}} = {K_{End_{x,1}}}^{H_1(\varepsilon)}, DK_{End_{x,2}} = {K_{End_{x,2}}}^{H_1(\varepsilon)}))$

Re-encryption Key Generation $ReKeyGen(SK_M, w)$

To generate the re-encryption key $RK_{M\to w}$ for authorized user this algorithm works as follows:

Firstly, selects random $\beta^r \in \mathbb{Z}_N^*$ for all $q_x \in F$. Then compute $RK_1 = {K_{Start_1}}^{H_1(\Omega)},\ RK_2 = {K_{Start_2}}^{H_1(\Omega)},\ RK_3 = {K_{Start_3}}^{H_1(\Omega)}$, for all $t \in \tau(RK_{t,1} = {K_{t,1}}^{H_1(\Omega)}, RK_{t,2} = {K_{t,2}}^{H_1(\Omega)}, RK_{t,3} = {K_{t,3}}^{H_1(\Omega)}$,

For all $q_x \in F(RK_{End_{x,1}} = {K_{End_{x,1}}}^{H_1(\Omega)} \cdot {H_{End}}^{\beta^r}, RK_{End_{x,2}} = {K_{End_{x,2}}}^{H_1(\Omega)} \cdot g^{\beta^r})$.

Finally, the $RK_{M\to w} = (M, RK_1, RK_2, RK_3,\ t \in \tau(RK_{t,1}, RK_{t,2}, RK_{t,3}, q_x \in F(RK_{End_{x,1}}, RK_{End_{x,2}}))$

Encryption $DFA.E(PP, w, m)$ The encryption algorithm randomly selects $\lambda_0, \lambda_1, \ldots, \lambda_l \in \mathbb{Z}_N^*$, run $(ssk, svk) \leftarrow key.Gen(1^n)$ and computes CT_m as:

First set:

$C_m = m \cdot e(g, g)^{\alpha \cdot \lambda_l}, C_{Start_1} = C_{0,1} = g^{\lambda_0},\ C_{Start_2} = (H_{Start})^{\lambda_0},\ C_{Start_3} = ({g_0}^{svk} h_0)^{\lambda_0}$, for $i = 1$ to l, set: $C_{i,1} = g^{\lambda_i}, C_{i,2} = (h_{wi})^{\lambda_i} \cdot z^{\lambda_{i-1}}$

Finally, set:

$C_{End_1} = C_{l,1} = g^{\lambda_l},\ C_{End_2} = (H_{End} \cdot g^{ab})^{\lambda_l},\ C_{End_3} = (H_k)^{\lambda_l},\ C_{End_4} = Sign(ssk, (w, C_m, C_{Start_1}, C_{Start_2}, C_{Start_3}, (C_{1,1}, C_{1,2}), \ldots, (C_{End_1}, C_{l,2}),\ C_{End_2},\ C_{End_3}))$.

The ciphertext CT_m is:

$CT_m = (svk, w, C_m, C_{Start_1}, C_{Start_2}, C_{Start_3}, (C_{1,1}, C_{1,2}), \ldots, (C_{End_1}, C_{l,2}),\ C_{End_2},\ C_{End_3},\ C_{End_4})$

Re-encryption $ReEec(RK_{M\to w'}, CT_m)$

The process in this phase represented as follows:

Check

$verify(svk, (w, C_m, C_{Start_1}, C_{Start_2}, C_{Start_3}, (C_{1,1}, C_{1,2}), \ldots, (C_{End_1}, C_{l,2}),\ C_{End_2},\ C_{End_3}))$ =1 and $e(C_{Start_1}, {g_0}^{svk} h_0) = e(g, C_{Start_3})$, outputs "True" if valid and "False" otherwise. If the verification is hold then proceed.

The string $w = w_1, \ldots, w_l$ is associated with the CT_m, and the DFA $M = (Q, \tau, g_0, F)$ is associated with the user's re-encryption key $RK_{M\to w'}$ where $ACCEPT(M, w)$. There must exist a sequence of $l+1$ states $\mu_0, \ldots, \mu_l$ and l transitions $t_0, \ldots, t_l$ where $\mu_0 = q_0$ and $\mu_l \in F$, we have $t_i = (\mu_{i-1}, \mu_i, w_i) \in \tau$. The key server re-encrypts CT_m as follows.

First computes $\varphi_0 = e(C_{Start_1}, RK_1) \cdot e(C_{Start_2}, RK_2)^{-1} = e(g, D_0)^{\lambda_0 \cdot H_1(\Omega)}$.

For $i = 1$ to l, computes: $\varphi_i = \varphi_{i-1} \cdot e\left(C_{(i-1)}, RK_{t_{i,1}}\right) \cdot e(C_{i,2}, RK_{t_{i,2}})^{-1} \cdot e\left(C_{i,1}, RK_{t_{i,3}}\right) = e(g, D_{\mu_i})^{\lambda_i \cdot H_1(\Omega)}$

Whenever M accepts w, we have that $\mu_l = q_x$ for some $q_x \in F$, and $\varphi_l = e(g, D_x)^{\lambda_l \cdot H_1(\Omega)}$.

Then sets $\varphi_{End} = \varphi_l \cdot e(C_{End_{x,1}}, RK_{End_{x,1}}) \cdot e(C_{End_{x,2}}, RK_{End_{x,2}})^{-1} \cdot e(C_{End_{x,3}}, RK_3) = e(g, D_{\mu_l})^{\alpha \cdot \lambda_l \cdot H_1(\Omega)}$.

Selects random $\gamma \in \mathbb{G}_T$ and sets $\pi_1 = sym.Enc(H_2(\gamma), A)$, and $\pi_2 = DFA.E(PP, w', \gamma)$, where $A = (CT_m)||(\varphi_{End})$. Finally, the re-encrypted ciphertext $CT_m^R = (\pi_1, \pi_2)$.

Transform (DK, CT_m)

The user sends his partial decryption key DK to the semi trusted proxy for partial decryption, the semi trusted proxy works the following:

$If\ verify(svk, (w, C_m, C_{Start_1}, C_{Start_2}, C_{Start_3}, (C_{1,1}, C_{1,2}), \ldots, (C_{End_1}, C_{l,2}),\ C_{End_2},\ C_{End_3})) = 1$

And $e(C_{Start_1}, {g_0}^{svk} h_0) = e(g, C_{Start_3})$, outputs "True" if valid and "False" otherwise. If the verification is hold then proceed.

Then the semi trusted proxy computes $CT_{m'}$ as follows:

$\theta_0 = e(C_{Start_1}, DK_1) \cdot e(C_{Start_2}, DK_2)^{-1} = e(g, D_0)^{\lambda_0 \cdot H_1(\varepsilon)}$.

For $i = 1$ to l, computes: $\theta_i = \theta_{i-1} \cdot e\left(C_{(i-1)}, DK_{t_{i,1}}\right) \cdot$

$e(C_{i,2}, DK_{t_{i,2}})^{-1} \cdot e\left(C_{i,1}, DK_{t_{i,3}}\right) = e(g, D_{\mu_i})^{\lambda_i \cdot H_1(\epsilon)}$

Whenever M accepts w, we have that $\mu_l = q_x$ for some $q_x \in F$ and $\theta_l = e(g, D_x)^{\lambda_l \cdot H_1(\epsilon)}$.

Finally compute: $\theta_{End} = \theta_l \cdot e\left(C_{End_{x,1}}, DK_{End_{x,1}}\right) \cdot e(C_{End_{x,2}}, DK_{End_{x,2}})^{-1} \cdot e\left(C_{End_{x,3}}, DK_3\right) = e(g, D_{\mu_i})^{\alpha \cdot \lambda_l \cdot H_1(\epsilon)}$ and sends the message $CT_{m'} = \theta_{End}$ to the user.

Dec (SK_m, CT_m)

The user can retrieve m by simply computing:

$$m = {C_m}/{\{CT_{m'} = \theta_{End}\}^{H_1(\epsilon)^{-1}}}$$

$Dec_R \ \ SK_m, CT_{m'}$

Firstly, the user can retrieve γ as follows:

$$\gamma = {C_\gamma}/{\{CT_{m'}\}^{H_1(\epsilon)^{-1}}}$$

We note that $C_\gamma = \gamma \cdot e(g,g)^{\alpha \cdot \lambda_l}$, since it encrypted using the $DFA.E(PP, w, m)$ encryption algorithm.

Secondly, the user computes A as follows:

$A \leftarrow sem.Dec(H_2(\gamma), \pi_1)$, where $A = (CT_m)||\varphi_{End}$.

Thirdly, computes Key as follows: $Key = {\varphi_{End}}^{H_1(\Omega)^{-1}}$

Fourthly, verifies:

$verify(svk, (w, C_m, C_{Start_1}, C_{Start_2}, C_{Start_3}, \left(C_{1,1}, C_{1,2}\right), \ldots, \left(C_{End_1}, C_{l,2}\right), C_{End_2}, C_{End_3}))$=1

And $e(C_{Start_1}, {g_0}^{svk} h_0) = e(g, C_{Start_3})$, outputs "True" if valid and "False" otherwise. If the verification is hold then proceed.

Finally, outputs the \$m\$ by simply computing:

$$m = {C_m}/{Key}$$

IV. Analysis

The proposed scheme ensure data confidentiality of the owner's personal data against unauthorized users and the curious semi trusted proxy. As we presented, the authorized user who's a DFA associated with his/her secret key accepts the string w will recover the massage m. For both unauthorized users and the curious semi trusted proxy cannot retrieve the desired value $e(g,g)^{\alpha \cdot \lambda}$, which is required for the decryption. For the reason that the two values ϵ and Ω are random exponents and only given to authorize user, which renders the description is impossible for unauthorized users. Hence, our scheme ensures confidentiality of the data. Besides, even if the semi trusted proxy transform the ciphertext CT_m to $CT_{m'}$ and obtain $CT_{m'}$ using the user's partial decryption key DK, he still cannot recover m, because he does not know the secret values ϵ and Ω. Therefore, both a curious semi trusted proxy and unauthorized users can read the massage.

V. Conclusion

Based on DFA-based FPRE scheme, we proposed a secure and efficient outsourcing decryption scheme which increases the flexibility of users to delegate their decryption rights to the semi trusted proxy. In our scheme, the user who has a DFA associated with his/her secret key accepts the string associated with the ciphertext can quite efficiently access the encrypted data with the help of the semi trusted proxy. Furthermore, it can also prove it as fully chosen-ciphertext secure in the standard model. Moreover, it is safe to draw the conclusion that our present work could be considered a secure and high efficient. Finally, our future research will concentrate on proving the security of our scheme in the standard model.

Acknowledgment

The author would like to acknowledge National Natural Science Foundation of China under Grant Nos.61003230, 61370026 and 61202445, the Fundamental Research Funds for the Central Universities under Grant No.ZYGX2013J073 and ZYGX2012J067.

References

[1] D. Zissis and D. Lekkas, "Addressing cloud computing security issues", Future Generation computer systems, vol. 28.3, pp. 583-592, 2012.

[2] S. Subashini and V. Kavitha, "A survey on security issues in service delivery models of cloud computing", Journal of network and computer applications, vol. 34.1, pp. 1-11, 2011.

[3] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage", in Network and Distributed System Security. NDSS'05, 2005.

[4] Lewko and B. Waters, "New proof methods for attribute-based encryption: Achieving full security through selective techniques", in Advances in Cryptology (Lecture Notes in Computer Science), vol. 7417. Berlin, Germany: Springer-Verlag, pp. 180–198, 2012.

[5] Waters, "Functional encryption for regular languages", in Advances in Cryptology (Lecture Notes in Computer Science), vol. 7417. Berlin, Germany: Springer-Verlag, pp. 218–235, 2012.

[6] K. Liang, M. Au, J. Liu, W. Susilo, D. Wong, G. Yang, P. Tran, and Qi Xie, "A DFA-Based Functional Proxy Re-Encryption Scheme for Secure Public Cloud Data Sharing", IEEE Transactions on Information Forensics and Security, vol. 9, No. 10, pp. 1667-1680, OCTOBER 2014.

[7] Lewko, "Tools for simulating features of composite order bilinear groups in the prime order setting", Advances in Cryptology-EUROCRYPT 2012, Springer Berlin Heidelberg, pp. 318-335, 2012.

[8] H. Debiao, J. Chen and R. Zhang, "An efficient identity-based blind signature scheme without bilinear pairings", Comput. Electr. Eng. 37, 4, pp. 444-450, July 2011.

[9] H. Debiao, C. Jianhua, "An efficient certificateless designated verifier signature scheme", Int. Arab J. Inf. Technol. 10 no. 4, pp.389-396, 2013.