A DFA-Based Functional Proxy Re-Encryption Scheme for Secure Public Cloud Data Sharing

Kaitai Liang, Man Ho Au, *Member, IEEE*, Joseph K. Liu, Willy Susilo, *Senior Member, IEEE*, Duncan S. Wong, *Member, IEEE*, Guomin Yang, *Member, IEEE*, Tran Viet Xuan Phuong, and Qi Xie

***Abstract*— In this paper, for the first time, we define a general notion for proxy re-encryption (PRE), which we call deterministic finite automata-based functional PRE (DFA-based FPRE). Meanwhile, we propose the first and concrete DFA-based FPRE system, which adapts to our new notion. In our scheme, a message is encrypted in a ciphertext associated with an arbitrary length index string, and a decryptor is legitimate if and only if a DFA associated with his/her secret key accepts the string. Furthermore, the above encryption is allowed to be transformed to another ciphertext associated with a new string by a semitrusted proxy to whom a re-encryption key is given. Nevertheless, the proxy cannot gain access to the underlying plaintext. This new primitive can increase the flexibility of users to delegate their decryption rights to others. We also prove it as fully chosen-ciphertext secure in the standard model.**

***Index Terms*— Functional encryption, functional proxy re-encryption, chosen-ciphertext security.**

I. Introduction

FUNCTIONAL Encryption (FE) is a useful cryptographic primitive that not only guarantees the confidentiality of data but also enhances the flexibility of data sharing. It is a general extension of Public Key Encryption (PKE). In traditional PKE, a data is encrypted to a particular receiver whose public key has registered to a trusted Certificate Authority. FE, however, provides more flexibility that the data can be encrypted under a description a, and the encryption can be decrypted if and only if there is a secret key whose description b matches a. As stated in [17] and [32], a classic example of FE is Attribute-Based Encryption (ABE) [11], [30] which comes to two flavors: Key-Policy ABE (KPABE) and Ciphertext-Policy ABE (CPABE). The former associates a secret key with an access policy such that the key can decrypt a ciphertext associated with attributes satisfying the policy. The latter, however, is complementary.

Although FE has many applications (e.g. audit-log [11]), it might not be flexible enough in some practical settings. For example, a social network user (e.g. LinkedIn[1]), say Alice, might choose to share her profile (e.g. educational details) with others under a policy, say P_1 = *("Region: United State" and "Occupation: student" and "Age: from 20 to 30")*. Suppose Alice's profile is encrypted under P_1 and stored in the cloud so that the users satisfying P_1 can access the profile. However, when trying to link herself with some companies for job applications, she might modify the access policy, e.g. P_2 = *("Region: all countries" and "Location: Local/overseas" and "Field: Finance")*. To guarantee the companies matching P_2 can access her profile, a new encryption under P_2 is required. A naive solution for Alice to generate the encryption is to first download the ciphertext under P_1 from the cloud, and next re-encrypt the profile under P_2 before uploading to the cloud. But the workload of Alice here is increased. If Alice is using some resource-limited devices which cannot afford the cost of encryption and decryption, she cannot share the profile unless some powerful computational devices (e.g. PC) are available. Besides, if the bandwidth is charged (by bit or megabit), the download and upload operations might yield a great amount of money.

Defined by Blaze, Bleumer and Strauss [5], Proxy Re-Encryption (PRE) is proposed to tackle the above problem. PRE is a useful extension of PKE, in which an *honest-but-curious* proxy is given a re-encryption key that allows it to transform ciphertexts intended for Alice into the ones intended for Bob without revealing either the plaintexts or the secret keys. PRE has many practical network applications, such as digital rights management [7] and secure email forwarding [5].

To achieve more flexibility on re-encryption, many variants of PRE have been proposed, such as Conditional PRE (CPRE) [33], Identity-Based PRE (IBPRE) [12] and Attribute-Based PRE (ABPRE) [23]. CPRE allows an encryption associated with a condition to be converted to a new ciphertext tagged with a new condition. The technologies of

Manuscript received February 22, 2014; revised May 4, 2014; accepted July 25, 2014. Date of publication August 7, 2014; date of current version September 11, 2014. This work was supported by the Australian Research Council Linkage Project under Grant ARC LP120200052. The work of K. Liang and D. S. Wong was supported by the Research Grants Council, Hong Kong, under Project CityU 121512. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Liqun Chen. *(Corresponding author: Joseph K. Liu.)*

K. Liang and D. S. Wong are with the Department of Computer Science, City University of Hong Kong, Hong Kong (e-mail: kliang4-c@my.cityu.edu.hk; duncan@cityu.edu.hk).

M. H. Au is with the Department of Computing, Hong Kong Polytechnic University, Hong Kong (e-mail: csallen@comp.polyu.edu.hk).

J. K. Liu is with the Department of Infocomm Security, Institute for Infocomm Research, Singapore 138632 (e-mail: ksliu@i2r.a-star.edu.sg).

W. Susilo, G. Yang, and T. V. X. Phuong are with the Centre for Computer and Information Security Research, School of Computer Science and Software Engineering, University of Wollongong, Wollongong, NSW 2522, Australia (e-mail: wsusilo@uow.edu.au; gyang@uow.edu.au; tvxp750@uowmail.edu.au).

Q. Xie is with the Hangzhou Key Laboratory of Cryptography and Network Security, Hangzhou Normal University, Hangzhou 310036, China (e-mail: qixie68@yahoo.com.cn).

Digital Object Identifier 10.1109/TIFS.2014.2346023

[1] http://www.linkedin.com/

IBPRE and ABPRE are somewhat similar, and a main difference between them is ABPRE enjoys more expressiveness in data sharing.

We might choose to employ ABPRE to solve the previous problem. Suppose Alice's profile is encrypted under P_1. When sharing her profile with some companies, she only needs to generate a re-encryption key from some descriptions and upload the key to the cloud. The cloud then will re-encrypt the encryption under P_1 to the one under P_2 such that the companies satisfying P_2 can access the profile. The cloud, nevertheless, cannot read the underlying plaintext.

A. Motivation

Although ABPRE can solve practical network problems, it leaves interesting open problems in terms of security and functionality. All existing ABPRE schemes [23], [26], [28] are only proved to be secure against *chosen-plaintext attacks* (CPA) in the selective model. Nonetheless, the selective CPA security is not sufficient enough in practice as it only achieves the secrecy against "passive" adversary (i.e. eavesdroppers). To guarantee a higher level of confidentiality for sensitive data, a stronger security notion is desirable, i.e. adaptive security against *chosen-ciphertext attacks* (CCA).

The functionality of an ABPRE system is another practical factor. Nonetheless, all existing ABPRE schemes only support access policy assembling with AND gates and fixed size inputs. Practically, an access policy might be required to assemble with AND, OR gates and NOT. Besides, in some particular applications, the access policy might be expressed by regular languages with arbitrary size. Thus it is desirable to propose an ABPRE system with expressive access policy supporting unlimited input size.

B. Our Contribution

1) This paper introduces a new notion, Deterministic Finite Automata based functional PRE (DFA-based FPRE).
2) A concrete scheme is proposed to adapt to the new notion. The scheme allows a data sender to encrypt a message in an encryption associated with an arbitrary length index string such that a secret key can be used to recover the underlying plaintext if and only if the DFA tagged with the key accepts the string. Furthermore, it permits a semi-trusted proxy to transform an encryption associated with an arbitrary length index string to another encryption associated with a new index string without leaking any useful message information to the proxy.
3) Our DFA-based FPRE can be seen as a type of Key-Policy ABPRE (KP-ABPRE). It is worth mentioning that our scheme is the first KP-ABPRE in the literature.
4) The present paper proves the scheme adaptively CCA secure in the standard model. To the best of our knowledge, it is the first of its type to achieve the *adaptive CCA* security in the standard model, but also to provide unlimited size input for access policy without degrading the functionality of proxy re-encryption.

C. Related Work

The concept of ABE is introduced by Sahai and Waters [30]. Goyal et al. [11] proposed the first KPABE system. The decryption is successful if the attributes tagged with ciphertext satisfy the access policy of the secret key. Reversely, Bethencourt, Sahai and Waters [4] defined Later on, Cheung and Newport [8] proposed a provably secure CPABE scheme supporting AND gates over attributes. Ostrovsky, Sahai and Waters [29] embedded negative attributes in access policy without increasing the size of ciphertext by employing the revocation technique in [11]. Goyal et al. [10] presented a construction in the standard model, but its large key size makes the scheme insufficient. More efficient and expressive CPABE systems were put forth by Waters [31]. Attrapadung et al. [2] proposed efficient ABE schemes with constant-size ciphertexts including a CPABE for threshold access policy, and two KPABE (with monotonic/non-monotonic access structures). Waters [32] proposed the first DFA-based FE system that supports the most expressive functionality for access policy.

The aforementioned schemes are proved selectively secure except that [4] is secure in the generic group model. To achieve CCA security, Yamada et al. [34] introduced a generic approach that works for both KPABE and CPABE. Using dual system encryption technology, Lewko et al. [16] converted [31] to achieve fully security. But the conversion leads to some loss of efficiency as it built on the composite order bilinear group. Lewko and Waters [17] then introduced a new proof method for converting a selective secure ABE to capture fully security by integrating the selective technique into the dual encryption system. Inspired by this, this paper proposes the first DFA-based FPRE with adaptive security in the standard model.

Following the introduction of decryption rights delegation [27], Ivan and Dodis [15] proposed a generic construction for proxy cryptography via sequential multi-encryption. Blaze, Bleumer and Strauss [5] defined PRE, and proposed a seminal PRE scheme. After that PRE comes to different flavors: unidirectional and bidirectional PRE, and single-hop and multi-hop PRE[2]. This work deals with the single-hop unidirectional PRE. Since its introduction there are many PRE systems (see [1], [7], [13], [14], [19]–[22], [24], [25]).

To implement PRE in the context of ABE, Liang et al. [23] defined CP-ABPRE, and proposed a construction on top of [8]. Mizuno and Doi [28] proposed a hybrid scheme where it can bridge ABE and IBE in the sense that ciphertexts generated in the context of ABE can be converted to the ones which can be decrypted in the IBE setting. Luo et al. [26] proposed a CP-ABPRE scheme supporting AND gates on multi-valued and negative attributes which can be viewed as a general extension of [23]. The schemes, however, are secure against selectively CPA, and their policies only operate over a fixed number of variables by AND gates only. Later on, Liang et al. [18] proposed a solution to tackle the above limitation. But their paper only considers limited input size for access policy. This paper deals with this issue without degrading security level.

[2]The definitions are defined in [1].

II. Definition and Security Model

By a DFA-based FPRE we mean a unidirectional single-hop DFA-based FPRE. Due to limited space we refer the reader to [32] for the definition of DFA and DFA-based FE.

Definition 1: A DFA-based functional proxy re-encryption (DFA-based FPRE) scheme includes the following algorithms:

1) $(PP, MSK) \leftarrow Setup(1^n, \sum)$: intakes a security parameter n and the description of a finite alphabet $\sum$, and outputs the public parameters PP and a master key MSK, where $n \in \mathbb{N}$. Note PP implicitly includes $\sum$.
2) $SK_M \leftarrow KeyGen(MSK, M = (Q, \mathcal{T}, q_0, F))$: intakes MSK and the description of a DFA M, and outputs a private key SK_M, where Q is a set of states, $\mathcal{T}$ is a set of transitions, q_0 is a start state, and F is a set of accept states.
3) $rk_{M\rightarrow w} \leftarrow ReKeyGen(SK_M, w)$: intakes SK_M for a DFA description M and an arbitrary length string $w \in \sum$, and outputs a re-encryption key $rk_{M\rightarrow w}$, where $REJECT(M, w)$. This re-encryption key is used to convert any ciphertext under a string w' (in which $ACCEPT(M, w')$) to be another ciphertext under w.
4) $C \leftarrow Encrypt(PP, w, m)$: intakes PP, a $w \in \sum$ and a message $m \in \mathbb{G}_T$, and outputs a ciphertext CT under w (which can be further re-encrypted).
5) $C^R \leftarrow ReEnc(rk_{M\rightarrow w}, CT)$: intakes $rk_{M\rightarrow w}$ and CT (under w'). If $ACCEPT(M, w')$, CT is converted to a re-encrypted ciphertext C^R under w (which cannot be further converted); otherwise, output an error symbol $\perp$.
6) $m/\perp \leftarrow Dec(SK_M, CT)$: intakes SK_M and CT (under w). If $ACCEPT(M, w)$, output a message m; otherwise, output an error symbol $\perp$.
7) $m/\perp \leftarrow Dec_R(SK_M, C^R)$: intakes SK_M and C^R (under w). If $ACCEPT(M, w)$, output a message m; otherwise, output an error symbol $\perp$.

Security: The IND-CCA security for DFA-based FPRE systems is as follows. Here we make the knowledge of secret key assumption where users will use their public keys when they know knowledge of the corresponding private keys.

Definition 2: A DFA-based FPRE scheme is IND-CCA secure at original ciphertext if no probabilistic polynomial time (PPT) adversary $\mathcal{A}$ can win the game below with non-negligible advantage. Let $\mathcal{B}$ be the game challenger.

Setup. $\mathcal{B}$ runs the algorithm $Setup$, and returns PP to $\mathcal{A}$.

Phase 1. $\mathcal{A}$ makes the following queries.

1) $\mathcal{O}_{SK}(M)$: on input a DFA description M, $\mathcal{B}$ runs $SK_M \leftarrow KeyGen(MSK, M)$ and returns SK_M to $\mathcal{A}$. Note the description M is based on $\sum$, i.e. each symbol used in M belongs to $\sum$.
2) $\mathcal{O}_{rk}(M, w)$: on input M and an arbitrary string w, $\mathcal{B}$ returns $rk_{M\rightarrow w} \leftarrow ReKeyGen(sk_M, w)$ to $\mathcal{A}$, where $SK_M \leftarrow KeyGen(MSK, M)$. Note w must be chosen from $\sum$, and $REJECT(M, w)$.
3) $\mathcal{O}_{re}(M, w', CT)$: on input M, a string w' and a CT (under w), $\mathcal{B}$ returns a re-encrypted ciphertext $C^R \leftarrow ReEnc(rk_{M\rightarrow w'}, CT)$ under w' to $\mathcal{A}$, where $rk_{M\rightarrow w'} \leftarrow ReKeyGen(SK_M, w')$, $SK_M \leftarrow KeyGen(MSK, M)$, $ACCEPT(M, w)$ and $REJECT(M, w')$.
4) $\mathcal{O}_{dec}(M, CT)$: on input M and CT (under w), $\mathcal{B}$ returns $m \leftarrow Dec(SK_M, CT)$, where $SK_M \leftarrow KeyGen(MSK, M)$ and $ACCEPT(M, w)$.
5) $\mathcal{O}_{dec_R}(M, C^R)$: on input M and C^R (under w), $\mathcal{B}$ returns $m \leftarrow Dec(SK_M, C^R)$, where $SK_M \leftarrow KeyGen(MSK, M)$ and $ACCEPT(M, w)$.

Note if the ciphertexts issued by $\mathcal{A}$ are ill-form, output $\perp$.

Challenge. $\mathcal{A}$ outputs two equal-length messages m_0, m_1 and a challenge string $w^* \in \sum$. If the queries: $\mathcal{O}_{SK}(M^*)$; $\mathcal{O}_{rk}(M^*, w')$ and $\mathcal{O}_{SK}(M')$ are never made, $\mathcal{B}$ returns the challenge original ciphertext $CT^* = Encrypt(PP, w^*, m_b)$ to $\mathcal{A}$, where $b \in_R \{0, 1\}$, $ACCEPT(M^*, w^*)$, $ACCEPT(M', w')$ and $REJECT(M^*, w')$.

Phase 2. The following queries are forbidden:

1) $\mathcal{O}_{SK}(M^*)$ for all M^* requested $ACCEPT(M^*, w^*)$;
2) $\mathcal{O}_{rk}(M^*, w')$ and $\mathcal{O}_{SK}(M')$ for all M^* and M' requested $ACCEPT(M^*, w^*)$, $ACCEPT(M', w')$ and $REJECT(M^*, w')$.
3) $\mathcal{O}_{dec}(M^*, CT^*)$ for all M^* requested $ACCEPT(M^*, w^*)$;
4) $\mathcal{O}_{re}(M^*, w', CT^*)$ and $\mathcal{O}_{SK}(M')$ for all M^* and M' requested $ACCEPT(M^*, w^*)$, $ACCEPT(M', w')$ and $REJECT(M^*, w')$; and
5) $\mathcal{O}_{dec_R}(M, C^R)$ for any M, C^R, where (w', C^R) is a derivative of (w^*, CT^*). As of [7], the derivative of (w^*, CT^*) is defined as follows.
   i. (w^*, CT^*) is a derivative of itself.
   ii. If $\mathcal{A}$ has issued (M^*, w') to $\mathcal{O}_{rk}$ to obtain $rk_{M^*\rightarrow w'}$ such that it can run $C^R \leftarrow ReEnc(rk_{M^*\rightarrow w'}, CT^*)$ under w', then (w', C^R) is a derivative of (w^*, CT^*) if $Dec_R(SK_{M'}, C^R) \in \{m_0, m_1\}$, where $ACCEPT(M', w')$, $ACCEPT(M^*, w^*)$ and $REJECT(M^*, w')$.
   iii. If $\mathcal{A}$ has issued (M^*, w', CT^*) to $\mathcal{O}_{re}$ to obtain C^R under w', then (w', C^R) is a derivative of (w^*, CT^*), where $ACCEPT(M^*, w^*)$ and $REJECT(M^*, w')$.

Guess. $\mathcal{A}$ outputs a guess bit $b' \in \{0, 1\}$.

The advantage of $\mathcal{A}$ is defined as $\epsilon_1 = |Pr[b' = b] - \frac{1}{2}|$.

Definition 3: A DFA-based FPRE scheme is IND-CCA secure at re-encrypted ciphertext if the advantage ϵ_2 is negligible for any PPT adversary $\mathcal{A}$ in the following experiment. Set $O = \{\mathcal{O}_{SK}, \mathcal{O}_{rk}, \mathcal{O}_{dec}, \mathcal{O}_{dec_R}\}$.

$$\epsilon_2 = \left| Pr\left[b' = b : (PP, MSK) \leftarrow Setup(1^n, \sum); \right.\right.$$
$$(m_0, m_1, w^*, w') \leftarrow \mathcal{A}^O(PP); b \in_R \{0, 1\};$$
$$\left.\left. C^{R*} \leftarrow ReEnc(rk_{M'\rightarrow w^*}, CT); b' \leftarrow A^O(C^{R*}) \right] - \frac{1}{2} \right|,$$

where w^* and w' are two "distinct" strings (chosen from $\sum$) so that if there is a SK_M in which $ACCEPT(M, w^*)$, then $REJECT(M, w')$ holds, $CT \leftarrow Encrypt(PP, w', m_b)$, $rk_{M'\rightarrow w^*} \leftarrow ReKeyGen(SK_{M'}, w^*)$, $SK_{M'} \leftarrow KeyGen(MSK, M')$. $\mathcal{O}_{SK}$, $\mathcal{O}_{rk}$, $\mathcal{O}_{dec}$, $\mathcal{O}_{dec_R}$ are the oracles defined in Definition 2 but limited to the following constraints. For $\mathcal{O}_{SK}$, $\mathcal{A}$ is forbidden to issue M^* where

$ACCEPT(M^*, w^*)$. If $\mathcal{A}$ queries to $\mathcal{O}_{dec_R}$ on (M^*, C^{R*}), the oracle outputs $\perp$. There is no restriction for $\mathcal{O}_{rk}$ and $\mathcal{O}_{dec}$.

III. Fully CCA-Secure DFA-Based FPRE

A. Preliminaries

1) Composite Order Bilinear Groups: Composite order bilinear groups were introduced in [6]. Let $\mathbb{G}$ and $\mathbb{G}_T$ be the two multiplicative cyclic groups of order $N = p_1p_2p_3$, where p_1, p_2, p_3 are distinct primes. We say that $\mathbb{G}_T$ has an admissible bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ if the following properties hold: (1) *Bilinearity*: $\forall g, h \in \mathbb{G}$ and $a, b \in_R \mathbb{Z}_N^*$, $e(g^a, h^b) = e(g, h)^{ab}$; (2) *Non-degeneracy*: $\exists g \in \mathbb{G}$ so that $e(g, g)$ has order N in $\mathbb{G}_T$. Assume that the group operations in $\mathbb{G}$ and $\mathbb{G}_T$ as well as the bilinear map e are computable in polynomial time with respect to a security parameter n, and that the group description of $\mathbb{G}$ and $\mathbb{G}_T$ include the generators of the respective cyclic groups. We denote by G_{p_1}, G_{p_2}, G_{p_3} the subgroups of order p_1, p_2, p_3 in $\mathbb{G}$ respectively.

2) Complexity Assumptions: Due to limited space, we refer the readers to [16] for the details of the 3 assumptions. Below two new assumptions are defined.

The Source Group l-Expanded Bilinear Diffie-Hellman Exponent (l-Expanded BDHE) Assumption in a Subgroup. It is closely relative to the Expanded l-BDHE assumption introduced in [32], but this requires the challenge term to lie in the source group.

3) The Source Group l-Expanded Bilinear Diffie-Hellman Exponent (l-Expanded BDHE) Assumption in a Subgroup: Given a group generator $\mathcal{G}$ and a positive integer l, we define

$$(N = p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, e) \leftarrow \mathcal{G}, g_1 \in_R \mathbb{G}_{p_1}, g_2 \in_R \mathbb{G}_{p_2},$$
$$g_3 \in_R \mathbb{G}_{p_3}, a, b, d, m, n, x, c_0, \ldots, c_{l+1} \in_R \mathbb{Z}_N,$$
$$D = \Big(N, \mathbb{G}, \mathbb{G}_T, e, g_1, g_3, g_2, g_2^a, g_2^b, g_2^{ab/dx}, g_2^{b/dx}, g_2^{ab/x}, g_2^n,$$
$$\forall i \in [0, 2l+1], i \neq l+1, j \in [0, l+1] \quad g_2^{a^i mn}, g_2^{a^i bmn/c_j x},$$
$$\forall i \in [0, l+1] \quad g_2^{c_i}, g_2^{a^i d}, g_2^{abc_i/dx}, g_2^{bc_i/dx},$$
$$\forall i \in [0, 2l+1], j \in [0, l+1] \quad g_2^{a^i bd/c_j x},$$
$$\forall i, j \in [0, l+1], i \neq j \quad g_2^{a^i bc_j/c_i x}\Big), T_0 = g_2^{a^{l+1}bm}, T_1 \in_R \mathbb{G}_{p_2}.$$

The advantage of an algorithm $\mathcal{A}$ in breaking the assumption is defined as $Adv_{\mathcal{A}}^{l\text{-}BDHE}(1^n) = |Pr[\mathcal{A}(D, T_0) = 1] - Pr[\mathcal{A}(D, T_1) = 1]|$. Using the same approach of proving q-based assumption [17], we can give the proof of the assumption in the generic group model. We hence omit the details.

Definition 4: The l-Expanded BDHE Assumption holds if $Adv_{\mathcal{A}}^{l\text{-}BDHE}(1^n)$ is negligible for any PPT algorithm $\mathcal{A}$.

The Source Group Modified q-Parallel Bilinear Diffie-Hellman Exponent (q-BDHE) Assumption in a Subgroup. It is a variant of the source group q-BDHE assumption [17].

4) The Source Group Modified q Bilinear Diffie-Hellman Exponent (q-BDHE) Assumption in a Subgroup: Given a group generator $\mathcal{G}$, we define the following distribution:

$$(N = p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, e) \leftarrow \mathcal{G}, g \in_R \mathbb{G}_{p_1}, g_2 \in_R \mathbb{G}_{p_2},$$
$$g_3 \in_R \mathbb{G}_{p_3}, c, a, e, f \in_R \mathbb{Z}_N,$$
$$D = (N, \mathbb{G}, \mathbb{G}_T, e, g, g_2, g_3, g_2^e, g_2^a, g_2^{aef}, g_2^{c+f/c}, g_2^{c^2}, \ldots,$$
$$g_2^{c^q}, g_2^{1/ac^q}), T_0 = g_2^{aec^{q+1}}, T_1 \in_R \mathbb{G}_{p_2}.$$

The advantage of an algorithm $\mathcal{A}$ in breaking the assumption is defined as $Adv_{\mathcal{A}}^{q\text{-}BDHE}(1^n) = |Pr[\mathcal{A}(D, T_0) = 1] - Pr[\mathcal{A}(D, T_1) = 1]|$.

Definition 5: The Source Group Modified q-BDHE Assumption holds if $Adv_{\mathcal{A}}^{q\text{-}BDHE}(1^n)$ is negligible for any PPT algorithm $\mathcal{A}$.

Note we can prove the source group modified q-BDHE assumption in the generic group model in the identical approach as that of the previous assumption, we hence omit the details.

B. Our Approach

It is challenging to propose a DFA-based FPRE system when considering adaptive CCA security in the standard model. The approach of achieving fully CCA security without jeopardizing the expressiveness of DFA is as follows.

Our system is built on top of Waters-FE system [32]. Accordingly, it is unavoidable that the system inherits the selective CPA security from Waters-FE scheme. To achieve fully security, we might choose to employ the dual encryption technology [16]. However, as stated in [17], the technique of [16] degrades the expressiveness of policy so that a single attribute can be used only once (in a policy) or a limited repetition with the cost of enlarging the size of system parameters and secret keys. This limitation for the policy (and efficiency) is incurred by information theoretic argument. Our system cannot get rid of this restriction by following the technology of [16]. In our system a symbol can be repeatedly used in DFA and index strings. Thus, the semi-functional parameters related to this symbol might leak information to adversary such that the nominality of secret key will not be hidden anymore.

To solve the problem, we leverage the proof idea of [17] by integrating the dual encryption technology with the selective proof technique. But we cannot trivially adapt the proof technique of [17] to our system as two systems are based on different primitives in which [17] is based on [31], and ours is built on [32]. Like [17], the most crucial part of our proof is to show that the nominality is hidden computationally from the view of adversary. This reflects on the indistinguishability from $Game_j^N$ to $Game_j^T$. Due to limited space, we refer the reader to Section III-D for the details. We let the challenger respectively simulate the queries of Phase 1 and Phase 2 (in the above indistinguishability simulation) as follows. In Phase 1, the challenger will receive the queries of secret keys associated with DFA before defining the delayed semi-functional parameters. Thus this phase is closely analogous to the context of selective security for a CPABE system. In Phase 2, the challenger will obtain a string first that is closely relative to the context of selective security for a KPABE system. We accordingly leverage the selective proof techniques of [31] and [32]. To adapt the techniques to our system, we need two new complexity assumptions (defined in Section III) which are closely relative to the l expanded bilinear Diffie-Hellman exponent assumption [32] and q-parallel bilinear Diffie-Hellman exponent assumption [31]. For the rest of the games defined in Section III-D, we prove their indistinguishability under the 3 assumptions [16].

We employ target collision resistant (TCR) hash function [9], strongly existential unforgeable one-time signature [3] and one-time symmetric encryption [9] to achieve CCA security. In the proof we offer decryption oracle to the adversary. This does not hinder the above framework as the challenger can construct any secret key. One might concern that in $Game_q$ (resp. $Game_{final}$) the challenger only generating semi-functional keys cannot respond decryption queries correctly. Actually, a semi-functional key can decrypt any normal ciphertext (issued by an adversary); and when the challenge ciphertext is issued for decryption query, the challenger will reject it.

To achieve adaptive security we let the elements of $\mathbb{G}_{p_1}$ represent all original components of our DFA-based FPRE scheme, and additionally use the elements of $\mathbb{G}_{p_3}$ to randomize the private key. The randomization will not hinder the functionality of the scheme due to the orthogonality property of subgroups $\mathbb{G}_{p_1}$, $\mathbb{G}_{p_2}$ and $\mathbb{G}_{p_3}$. Besides, the elements of $\mathbb{G}_{p_2}$ will not be used in the real scheme but in the security proof.

C. Construction

Our DFA-based FPRE scheme works as follows.

- **Setup**$(1^n, \sum)$**:** Choose $g, g_0, z, h_0 \in_R \mathbb{G}_{p_1}$, and $\alpha, k, a, b, \alpha_{end}, \alpha_{start} \in_R \mathbb{Z}_N^*$. Set $h_{start} = g^{\alpha_{start}}$, $h_{end} = g^{\alpha_{end}}$ and $h_k = g^k$. For each symbol $\sigma \in \sum$, choose a $\alpha_\sigma \in_R \mathbb{Z}_N^*$, and set $h_\sigma = g^{\alpha_\sigma}$. Choose a one-time signature scheme OTS, a one-time symmetric encryption scheme $SYM = (SYM.Enc, SYM.Dec)$, and two hash functions: $H_1 : \mathbb{G}_T \to \mathbb{Z}_N^*$ and $H_2 : \mathbb{G}_T \to \{0,1\}^{poly(n)}$. The PP is $\{e(g,g)^\alpha, g, g^{ab}, g_0, z, h_0, h_{start}, h_{end}, h_k, \forall_{\sigma\in\sum} h_\sigma, OTS, SYM, H_1, H_2\}$ along with the descriptions of $\mathbb{G}$ and the alphabet $\sum$. The MSK is $(g^{-\alpha}, X_3)$, where X_3 is a generator of $\mathbb{G}_{p_3}$.
- **KeyGen**$(MSK, M = (Q, \mathcal{T}, q_0, F))$: The description of M includes a set Q of states $q_0, \ldots, q_{|Q|-1}$ and a set of transitions $\mathcal{T}$ where each transition $t \in \mathcal{T}$ is a triple $(x, y, \sigma) \in Q \times Q \times \sum$. q_0 is designated as a unique start state and $F \subseteq Q$ is the set of accept states. The algorithm chooses $D_0, D_1, \ldots, D_{|Q|-1} \in_R \mathbb{G}_{p_1}$ (associating D_i with q_i), for each $t \in \mathcal{T}$ it chooses $r_t \in_R \mathbb{Z}_N^*$, $\forall q_x \in F$ it chooses $r_{end_x} \in_R \mathbb{Z}_N^*$, and chooses a $u \in_R \mathbb{Z}_N^*$. It also chooses $R_{start1}, R_{start2}, R_{start3}, R_{t,1}, R_{t,2}, R_{t,3}, R_{end_{x,1}}, R_{end_{x,2}} \in_R \mathbb{G}_{p_3}$ and a $r_{start} \in_R \mathbb{Z}_N^*$. The algorithm constructs the key as follows. First it sets:

$$K_{start1} = D_0 \cdot (h_{start})^{r_{start}} \cdot R_{start1},$$
$$K_{start2} = g^{r_{start}} \cdot R_{start2}, K_{start3} = g^u \cdot R_{start3}.$$

For each $t = (x, y, \sigma) \in \mathcal{T}$ the algorithm sets:

$$K_{t,1} = D_x^{-1} \cdot z^{r_t} \cdot R_{t,1}, \quad K_{t,2} = g^{r_t} \cdot R_{t,2},$$
$$K_{t,3} = D_y \cdot (h_\sigma)^{r_t} \cdot R_{t,3},$$

For each $q_x \in F$ it computes:

$$K_{end_{x,1}} = g^{-\alpha} \cdot D_x \cdot (h_{end} \cdot g^{ab})^{r_{end_x}} \cdot g^{ku} \cdot R_{end_{x,1}},$$
$$K_{end_{x,2}} = g^{r_{end_x}} \cdot R_{end_{x,2}}.$$

Finally, the key is

$$SK = \Big(M, K_{start1}, K_{start2}, K_{start3}, \forall t \in \mathcal{T}(K_{t,1}, K_{t,2}, K_{t,3}), \forall q_x \in F(K_{end_{x,1}}, K_{end_{x,2}})\Big).$$

- **ReKeyGen**(SK_M, w)**:**
  1) Choose a $y \in_R \mathbb{G}_T$ and $v_x \in_R \mathbb{Z}_N^*$ (for $\forall q_x \in F$), and set $rk_1 = K_{start1}^{H_1(y)}, rk_2 = K_{start2}^{H_1(y)}, rk_3 = K_{start3}^{H_1(y)}$, $\forall t \in \mathcal{T}(rk_{t,1} = K_{t,1}^{H_1(y)}, rk_{t,2} = K_{t,2}^{H_1(y)}, rk_{t,3} = K_{t,3}^{H_1(y)})$, $\forall q_x \in F(rk_{end_{x,1}} = K_{end_{x,1}}^{H_1(y)} \cdot h_{end}^{v_x}$, $rk_{end_{x,2}} = K_{end_{x,2}}^{H_1(y)} \cdot g^{v_x})$.
  2) Run $rk_4 \leftarrow Encrypt(PP, w, y)$, and finally output $rk_{M\to w} = (M, rk_1, rk_2, rk_3, rk_4, \forall t \in \mathcal{T}(rk_{t,1}, rk_{t,2}, rk_{t,3}), \forall q_x \in F(rk_{end_{x,1}}, rk_{end_{x,2}}))$.
- **Encrypt**(PP, w, m)**:** Choose $s_0, s_1, \ldots, s_l \in_R \mathbb{Z}_N^*$, run $(ssk, svk) \leftarrow KeyGen(1^n)$ and constructs CT as
  First set: $C_m = m \cdot e(g,g)^{\alpha \cdot s_l}, C_{start1} = C_{0,1} = g^{s_0}$, $C_{start2} = (h_{start})^{s_0}, C_{start3} = (g_0^{svk} h_0)^{s_0}$,
  for $i = 1$ to l, set: $C_{i,1} = g^{s_i}, C_{i,2} = (h_{w_i})^{s_i} \cdot z^{s_{i-1}}$,
  finally, set:

$$C_{end1} = C_{l,1} = g^{s_l}, C_{end2} = (h_{end} \cdot g^{ab})^{s_l}, C_{end3} = (h_k)^{s_l},$$
$$C_{end4} = Sign\big(ssk, (w, C_m, C_{start1}, C_{start2}, C_{start3}, (C_{1,1}, C_{1,2}), \ldots, (C_{end1}, C_{l,2}), C_{end2}, C_{end3})\big).$$

The original ciphertext is

$$CT = \big(svk, w, C_m, C_{start1}, C_{start2}, C_{start3}, (C_{1,1}, C_{1,2}), \ldots, (C_{l,1}, C_{l,2}), C_{end2}, C_{end3}, C_{end4}\big).$$

- **ReEnc**$(rk_{M\to w'}, CT)$**:**
  1) If $Verify(svk, (C_{end4}, (w, C_m, C_{start1}, C_{start2}, C_{start3}, (C_{1,1}, C_{1,2}), \ldots, (C_{end1}, C_{l,2}), C_{end2}, C_{end3}))) = 1$ and $e(C_{start1}, g_0^{svk} h_0) = e(g, C_{start3})$, proceed; otherwise, output $\perp$.
  2) CT is associated with a string $w = (w_1, \ldots, w_l)$ and the re-encryption key $rk_{M\to w'}$ is associated with a DFA $M = (Q, \mathcal{T}, q_0, F)$ where $ACCEPT(M, w)$. There must exist a sequence of $l+1$ states $u_0, u_1, \ldots, u_l$ and l transitions $t_1, \ldots, t_l$ where $u_0 = q_0$ and $u_l \in F$ and for $i = 1, \ldots, l$, we have $t_i = (u_{i-1}, u_i, w_i) \in \mathcal{T}$. The proxy re-encrypts CT as follows.
     a) It first computes: $A_0 = e(C_{start1}, rk_1) \cdot e(C_{start2}, rk_2)^{-1} = e(g, D_0)^{s_0 \cdot H_1(y)}$.
     b) For $i = 1$ to l, it computes:

$$A_i = A_{i-1} \cdot e(C_{(i-1),1}, rk_{t_i,1}) \cdot e(C_{i,2}, rk_{t_i,2})^{-1} \cdot e(C_{i,1}, rk_{t_i,3}) = e(g, D_{u_i})^{s_i \cdot H_1(y)}.$$

     Since M accepts w, we have that $u_l = q_x$ for some $q_x \in F$ and $A_l = e(g, D_x)^{s_l \cdot H_1(y)}$.
     c) It sets:

$$A_{end} = A_l \cdot e(C_{end_{x,1}}, rk_{end_{x,1}})^{-1} \cdot e(C_{end_{x,2}}, rk_{end_{x,2}}) \cdot e(C_{end_{x,3}}, rk_3) = e(g,g)^{\alpha \cdot s_l \cdot H_1(y)}.$$

d) The proxy sets $C_1 = SYM.Enc(H_2(\delta), \xi)$, $C_2 = Encrypt(PP, w', \delta)$, where $\delta \in_R \mathbb{G}_T$ and $\xi = (CT||A_{end}||rk_4)$. It finally outputs the re-encrypted ciphertext $C^R = (C_1, C_2)$.

- **Dec**(SK_M, CT): If $Verify(svk, (C_{end4}, (w, C_m, C_{start1}, C_{start2}, C_{start3}, (C_{1,1}, C_{1,2}), \ldots, (C_{end1}, C_{l,2}), C_{end2}, C_{end3}))) = 1$ and $e(C_{start1}, g_0^{svk} h_0) = e(g, C_{start3})$, proceed; otherwise, output $\perp$.
  First compute:

$$B_0 = e(C_{start1}, K_{start1}) \cdot e(C_{start2}, K_{start2})^{-1} = e(g, D_0)^{s_0}.$$

  For $i = 1$ to l, compute:

$$B_i = B_{i-1} \cdot e(C_{(i-1),1}, K_{t_i,1}) \cdot e(C_{i,2}, K_{t_i,2})^{-1} \cdot e(C_{i,1}, K_{t_i,3}) = e(g, D_{u_i})^{s_i}.$$

  Since M accepts w, we have that $u_l = q_x$ for some $q_x \in F$ and $B_l = e(g, D_x)^{s_l}$. Finally compute

$$B_{end} = B_l \cdot e(C_{end_{x,1}}, K_{end_{x,1}})^{-1} \cdot e(C_{end_{x,2}}, K_{end_{x,2}}) \cdot e(C_{end_{x,3}}, K_{start3}) = e(g, g)^{\alpha \cdot s_l},$$

  and output the message $m = C_m / B_{end}$.
- **Dec**$_R(SK_M, C^R)$:
  1) Run $\delta \leftarrow Decrypt(SK_M, C_2)$, compute $\xi \leftarrow SYM.Dec(H_2(\delta), C_1)$, where $\xi = (CT||A_{end}||rk_4)$.
  2) Run $y \leftarrow Decrypt(SK_M, rk_4)$, then compute $Key = A_{end}^{H_1(y)^{-1}}$.
  3) Verify

$$e(C_{start1}, g_0^{svk} h_0) \stackrel{?}{=} e(g, C_{start3}),$$
$$Verify\Big(svk, (C_{end4}, (w, C_m, C_{start1}, C_{start2}, C_{start3}, (C_{1,1}, C_{1,2}), \ldots, (C_{l,1}, C_{l,2}), C_{end2}, C_{end3}))\Big) \stackrel{?}{=} 1.$$

     If the equations hold, proceed; otherwise, output $\perp$.
  4) Output the message $m = C_m / Key$.

D. Security Analysis

Theorem 1: Suppose Assumption 1, 2 and 3, the source group modified q-BDHE assumption in a subgroup, and the source group l-BDHE assumption in a subgroup hold, SYM is a CCA-secure symmetric encryption, OTS is a strongly existential unforgeable one-time signature and H_1, H_2 are TCR hash functions, our DFA-based FPRE system is IND-CCA secure in the standard model.

Before proceeding, we define the semi-functional ciphertexts and the semi-functional keys as follows.

1) Semi-Functional Ciphertexts: We let g_2 be a generator of subgroup $\mathbb{G}_{p_2}$, choose $\gamma_0, \gamma_1, \ldots, \gamma_l \in_R \mathbb{Z}_N^*$, $\alpha'_\sigma \in_R \mathbb{Z}_N^*$ associated to each symbol σ belonging to $\sum$, and $\beta', \beta'_0, \beta'_1, \alpha'_{start}, \alpha'_{end}, k', a', b' \in_R \mathbb{Z}_N^*$. We run $(ssk, svk) \leftarrow KeyGen(1^n)$, and set the ciphertext as $(svk, w, C'_m, C'_{start1}, C'_{start2}, C'_{start3}, (C'_{1,1}, C'_{1,2}), \ldots, (C'_{end1}, C'_{l,2}), C'_{end2}, C'_{end3}, C'_{end4})$ in which

$$C'_{start1} = C'_{0,1} = g^{s_0} g_2^{\gamma_0}, C'_{start2} = (h_{start})^{s_0} g_2^{\alpha'_{start}\gamma_0},$$
$$C'_{start3} = (g_0^{svk} h_0)^{s_0} (g_2^{\beta'_0 svk} \cdot g_2^{\beta'_1})^{\gamma_0}, C'_{end1} = C'_{l,1} = g^{s_l} g_2^{\gamma_l},$$
$$C'_{end2} = (h_{end} g^{ab})^{s_l} g_2^{(\alpha'_{end} + a'b')\gamma_l}, C'_{end3} = g^{ks_l} g_2^{k'\gamma_l},$$

for $i = 1$ to l: $C'_{i,1} = g^{s_i} g_2^{\gamma_i}$, $C'_{i,2} = (h_{w_i})^{s_i} z^{s_{i-1}} g_2^{\alpha'_{w_i}\gamma_i + \beta'\gamma_{i-1}}$, C'_m and C'_{end4} are the normal ciphertext components generated by the encryption algorithm except that C'_{end4} is the signature for the above components. Note $k', \beta', \alpha'_{end}, \alpha'_{start}$ and (some of) α'_{w_i} will be shared in the nominal and temporary semi-functional keys.

We define three types of semi-functional keys as follows. Below we choose $R_{start1}, R_{start2}, R_{start3} \in_R \mathbb{G}_{p_3}$, an $R \in_R \mathbb{G}_{p_2}$, $d_0, \ldots, d_{|Q|-1} \in_R \mathbb{Z}_N^*$ associated to the states in Q, for each $t \in \mathcal{T}$ choose $\epsilon_t, r_t \in_R \mathbb{Z}_N^*$ and $R_{t,1}, R_{t,2}, R_{t,3} \in_R \mathbb{G}_{p_3}$, for each $q_x \in F$ choose $\epsilon_{end_x}, r_{end_x} \in_R \mathbb{Z}_N^*$ and $R_{end_{x,2}} \in_R \mathbb{G}_{p_3}$, and finally choose $\epsilon_{start}, r_{start}, u' \in_R \mathbb{Z}_N^*$.

2) Semi-functional Keys: We set the keys as

$$K'_{start1} = D_0 (h_{start})^{r_{start}} R_{start1},$$
$$K'_{start2} = g^{r_{start}} R_{start2}, K'_{start3} = g^u R_{start3},$$

for each $t = (x, y, \sigma) \in \mathcal{T}$:

$$K'_{t,1} = D_x^{-1} z^{r_t} R_{t,1}, K'_{t,2} = g^{r_t} R_{t,2}, K'_{t,3} = D_y (h_\sigma)^{r_t} R_{t,3},$$

for each $q_x \in F$:

$$K'_{end_{x,1}} = g^{-\alpha} D_x (h_{end} g^{ab})^{r_{end_x}} g^{ku} R_{end_{x,1}} R,$$
$$K'_{end_{x,2}} = g^{r_{end_x}} R_{end_{x,2}}.$$

3) Nominal Semi-Functional Keys: We set the keys as $(K'_{start1}, K'_{start2}, \forall t \in \mathcal{T}(K'_{t,1}, K'_{t,2}, K'_{t,3}), \forall q_x \in F(K'_{end_{x,1}}, K'_{end_{x,2}}))$ in which

$$K'_{start1} = D_0 (h_{start})^{r_{start}} R_{start1} (g_2^{\alpha'_{start}})^{\epsilon_{start}} g_2^{d_0},$$
$$K'_{start2} = g^{r_{start}} R_{start2} g_2^{\epsilon_{start}}, K'_{start3} = g^u R_{start3} g_2^{u'},$$

for each $t = (x, y, \sigma) \in \mathcal{T}$:

$$K'_{t,1} = D_x^{-1} z^{r_t} R_{t,1} (g_2^{\beta'})^{\epsilon_t} g_2^{-d_x},$$
$$K'_{t,2} = g^{r_t} R_{t,2} g_2^{\epsilon_t}, K'_{t,3} = D_y (h_\sigma)^{r_t} R_{t,3} (g_2^{\alpha'_\sigma})^{\epsilon_t} g_2^{d_y},$$

for each $q_x \in F$:

$$K'_{end_{x,1}} = g^{-\alpha} D_x (h_{end} g^{ab})^{r_{end_x}} g^{ku} \cdot R_{end_{x,1}} g_2^{d_x} g_2^{(\alpha'_{end} + a'b')\epsilon_{end_x}} g_2^{k'u'},$$
$$K'_{end_{x,2}} = g^{r_{end_x}} R_{end_{x,2}} g_2^{\epsilon_{end_x}}.$$

4) Temporary Semi-Functional Keys: We set the keys as

$$K'_{start1} = D_0 (h_{start})^{r_{start}} R_{start1} (g_2^{\alpha'_{start}})^{\epsilon_{start}} g_2^{d_0},$$
$$K'_{start2} = g^{r_{start}} R_{start2} g_2^{\epsilon_{start}}, K'_{start3} = g^u R_{start3} g_2^{u'},$$

for each $t = (x, y, \sigma) \in \mathcal{T}$:

$$K'_{t,1} = D_x^{-1} z^{r_t} R_{t,1} (g_2^{\beta'})^{\epsilon_t} g_2^{-d_x},$$
$$K'_{t,2} = g^{r_t} R_{t,2} g_2^{\epsilon_t}, K'_{t,3} = D_y (h_\sigma)^{r_t} R_{t,3} (g_2^{\alpha'_\sigma})^{\epsilon_t} g_2^{d_y},$$

for each $q_x \in F$:

$$K'_{end_{x,1}} = g^{-\alpha} D_x (h_{end} g^{ab})^{r_{end_x}} g^{ku} R_{end_{x,1}} R,$$
$$K'_{end_{x,2}} = g^{r_{end_x}} R_{end_{x,2}} g_2^{\epsilon_{end_x}}.$$

We will prove Theorem 1 in a hybrid argument over a sequence of games. The total number of queries is $q = q_{sk} + q_{rk} + q_{re} + q_{dec}$, where $q_{sk}, q_{rk}, q_{re}, q_{dec}$ denote the number of the secret key extraction, re-encryption key extraction, re-encryption and decryption queries, respectively. We define $Game_{real}$ to be the first game. It is the IND-CCA security game for DFA-based FPRE systems in which the challenge ciphertext (for original ciphertext security and re-encrypted ciphertext security) is normal. In this game, $\mathcal{B}$ will use normal secret keys as knowledge to respond secret key extraction, re-encryption key extraction, re-encryption and decryption queries. We define $Game_0$ to be the second game which is identical to $Game_{real}$ except that the challenge ciphertext is semi-functional. Hereafter by "keys" (resp. "key") we mean the secret key(s) (constructed by $\mathcal{B}$) used to respond the secret key extraction, re-encryption key extraction, re-encryption and decryption queries. In the following games, we will convert the "keys" to be semi-functional one by one. But for clarity we first turn the "keys" for the secret key extraction queries, and then convert the "keys" for the re-encryption key extraction queries, the re-encryption queries and the decryption queries in sequence. Besides, $\mathcal{A}$ is only allowed to issue one corresponding query in each of the following games. We further define $Game_i$ as follows, where $i \in [1, q]$. We let $j_\iota \in [1, q_\iota]$, where $\iota \in \{sk, rk, re, dec\}$. For each game $Game_{j_\iota}$ we define two sub-games $Game^N_{j_\iota}$ and $Game^T_{j_\iota}$ in which the challenge ciphertext is semi-functional. In $Game^N_{j_\iota}$ the first $(j-1)_\iota$ "keys" are semi-functional, the j_ι-th "key" is nominal semi-functional, and the rest of "keys" are normal. In $Game^T_{j_\iota}$ the first $(j-1)_\iota$ "keys" are semi-functional, the j_ι-th "key" is temporary semi-functional, and the remaining "keys" are normal. To transform $Game_{(j-1)_\iota}$ (where j_ι-th "key" is normal) to $Game_{j_\iota}$ (where j_ι-th "key" is semi-functional), we start from converting $Game_{(j-1)_\iota}$ to $Game^N_{j_\iota}$, then to $Game^T_{j_\iota}$, and finally to $Game_{j_\iota}$. Note to get from $Game^N_{j_\iota}$ to $Game^T_{j_\iota}$, we deal with the simulations for the queries of Phase 1 and that of Phase 2 differently: the former is based on the source group modified q-BDHE assumption in a subgroup, and the latter is based on the source group l-expanded BDHE assumption in a Subgroup. In $Game_q = Game_{q_{dec}}$ all "keys" are semi-functional, and the challenge ciphertext is semi-functional for one of the given messages. We define $Game_{final}$ to be the final game in which all "keys" are semi-functional and the challenge ciphertext is semi-functional for a random message, independent of the two message given by $\mathcal{A}$. We will prove the above games to be indistinguishable by the following lemmas. Below we assume SYM is a CCA-secure, OTS is a strongly existential unforgeable and H_1, H_2 are TCR hash functions, and it is hard to find a non-trivial factor of N.

Lemma 1: If there is an algorithm $\mathcal{A}$ such that $Game_{real} Adv^{DFA\text{-}FPRE}_{\mathcal{A}} - Game_0 Adv^{DFA\text{-}FPRE}_{\mathcal{A}} = \delta$, we can build an algorithm $\mathcal{B}$ breaking Assumption 1 with advantage δ.

Proof: For simplicity, we combine the security proof of original and re-encrypted ciphertexts into one simulation. Below by original/re-encrypted game we mean the security game for original/re-encrypted ciphertext.

Setup. $\mathcal{B}$ is given an instance (D, T) of Assumption 1, and simulates either $Game_{real}$ or $Game_0$ with $\mathcal{A}$. $\mathcal{B}$ chooses $a, b, \alpha, \beta, \beta_0, \beta_1, \alpha_{start}, \alpha_{end}, k \in_R \mathbb{Z}^*_N$, $\alpha_\sigma \in_R \mathbb{Z}^*_N$ for all symbols in $\sum$, two TCR hush functions H_1, H_2, a one-time signature system OTS and a one-time symmetric encryption scheme SYM, and outputs PP:

$$e(g,g)^\alpha, g, g^{ab}, g_0 = g^{\beta_0}, z = g^\beta, h_0 = g^{\beta_1}, h_{start} = g^{\alpha_{start}},$$
$$h_{end} = g^{\alpha_{end}}, h_k = g^k, \forall_{\sigma \in \sum} h_\sigma = g^{\alpha_\sigma}, H_1, H_2, OTS, SYM.$$

$\mathcal{B}$ keeps α and X_3 secretly.

Phase 1. $\mathcal{A}$ makes the following queries:

1) $\mathcal{O}_{SK}(M)$: If $ACCEPT(M, w^*)$, $\mathcal{B}$ output $\perp$. Otherwise, $\mathcal{B}$ returns SK_M to $\mathcal{A}$ by running the algorithm $KeyGen$ as it has knowledge of MSK.
2) $\mathcal{O}_{rk}(M, w)$:
   - For original game: if $ACCEPT(M, w^*)$ and $SK_{M'}$ (for any DFA M' so that $ACCEPT(M', w)$) is obtained by $\mathcal{A}$, $\mathcal{B}$ outputs $\perp$. Otherwise, $\mathcal{B}$ constructs SK_M as in $\mathcal{O}_{SK}$, and next generates $rk_{M \to w}$ for $\mathcal{A}$ by running the algorithm $ReKeyGen$.
   - For re-encrypted game: $\mathcal{B}$ can construct generates any re-encryption key $rk_{M \to w}$ with knowledge of MSK.
3) $\mathcal{O}_{re}(M, w', CT)$:
   - For original game: if $ACCEPT(M, w^*)$, CT is the challenge ciphertext, and $SK_{M'}$ (for any DFA M' so that $ACCEPT(M', w')$) is obtained by $\mathcal{A}$, $\mathcal{B}$ outputs $\perp$. Otherwise, $\mathcal{B}$ constructs $rk_{M \to w'}$ as in $\mathcal{O}_{rk}$, and next generates the re-encrypted ciphertext C^R by running the algorithm $ReEnc$.
   - For re-encrypted game: $\mathcal{O}_{re}$ is not offered to $\mathcal{A}$.
4) $\mathcal{O}_{dec}(M, CT)$:
   - For original game: if $ACCEPT(M, w^*)$, and CT is the challenge ciphertext, $\mathcal{B}$ outputs $\perp$. Otherwise, $\mathcal{B}$ constructs SK_M with knowledge of MSK, and next recovers m by running the algorithm Dec.
   - For re-encrypted game: $\mathcal{B}$ recovers the private key with knowledge of MSK and recovers m.
5) $\mathcal{O}_{dec_R}(M, C^R)$:
   - For original game: $\mathcal{B}$ constructs SK_M with knowledge of MSK, and next recovers m by running Dec_R. If (w', C^R) is a derivative, $\mathcal{B}$ outputs $\perp$. To distinguish the derivatives from the submitted ciphertexts, $\mathcal{B}$ can use the following approaches. If the re-encrypted ciphertext is output by $\mathcal{O}_{re}(M, w', CT)$, then the ciphertext is indeed a derivative, where CT is the challenge ciphertext and $SK_{M'}$ (for any DFA M' so that $ACCEPT(M', w')$) is not obtained by $\mathcal{A}$. Otherwise, it indicates that the re-encrypted ciphertext is constructed by $\mathcal{A}$ with

a re-encryption key given by $\mathcal{B}$. $\mathcal{B}$ then recovers the underlying CT from the re-encrypted ciphertext (by using the corresponding private key), and re-constructs A_{end} as in the real scheme. If the value (of A_{end}) is equal to the one hidden in the symmetric encryption, and CT is the challenge ciphertext, it knows that the re-encrypted ciphertext is a derivative.

- For re-encrypted game: $\mathcal{B}$ uses SK_M to decrypt C^R as in the real scheme. If C^R is the challenge ciphertext, $\mathcal{B}$ outputs $\perp$.

Challenge. $\mathcal{B}$ implicitly sets g^{s_0} to be the $\mathbb{G}_{p_1}$ part of T, runs $(ssk, svk) \leftarrow KeyGen(1^n)$, chooses a random $b \in \{0, 1\}$ and generates the challenge ciphertext as follows.

- For original game: $\mathcal{A}$ outputs m_0, m_1, and w^* (with length l). $\mathcal{B}$ sets the challenge original ciphertext as

$$svk, w^*,\ C_m = m_b \cdot e(g^{\alpha}, T)^{s'_l},\ C_{start1} = T,$$
$$C_{start2} = T^{\alpha_{start}},\ C_{start3} = T^{\beta_0 \cdot svk} \cdot T^{\beta_1},$$
$$C_{end1} = T^{s'_l},\ C_{end2} = T^{\alpha_{end} \cdot s'_l},\ C_{end3} = T^{k \cdot s'_l},$$

for $i = 1$ to l: $C_{i,1} = T^{s'_i}, C_{i,2} = T^{s'_i \cdot \alpha_{w_i}} \cdot T^{s'_{i-1} \cdot \beta}$, and $C_{end4} = Sign\Big(ssk, (w^*, C_m, C_{start1}, C_{start2}, C_{start3}, (C_{1,1}, C_{1,2}), \ldots, (C_{l,1}, C_{l,2}), C_{end2}, C_{end3})\Big)$, where $s'_1, \ldots, s'_l \in_R \mathbb{Z}_N^*$. $\mathcal{B}$ outputs $CT = (svk, w^*, C_m, C_{start1}, C_{start2}, C_{start3}, (C_{1,1}, C_{1,2}), \ldots, (C_{l,1}, C_{l,2}), C_{end2}, C_{end3}, C_{end4})$ to $\mathcal{A}$.

- For re-encrypted game: $\mathcal{A}$ outputs m_0, m_1, a string w' and a challenge string w^* (both with length l). $\mathcal{B}$ runs $CT = Encrypt(PP, w', m_b)$, generates the re-encryption key $rk_{M \to w^*}$ and constructs A_{end} as in the real scheme. It further sets C_2 in the identical approach described above. $\mathcal{B}$ finally sets $C_1 = SYM.Enc(H_2(\delta), \xi)$, and outputs the challenge re-encrypted ciphertext $C^R = (C_1, C_2)$ to $\mathcal{A}$, where $\xi = (CT || A_{end} || rk_4)$.

Phase 2. Same as Phase 1.

Guess. $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs.

If $T \in \mathbb{G}_{p_1}$, the challenge ciphertext is a properly distributed normal ciphertext so that this is in $Game_{real}$. If $T \in \mathbb{G}_{p_1 p_2}$, we let g^{s_0} be the $\mathbb{G}_{p_1}$ part of T and $g_2^{\gamma_0}$ be the $\mathbb{G}_{p_2}$ part of T, i.e. $T = g^{s_0} g_2^{\gamma_0}$. We will have the semi-functional ciphertext with $\gamma_i = \gamma_0 s'_i$, $s_i = s_0 s'_i$. In addition, the values of $a, b, \alpha_{start}, \alpha_{end}, \alpha_{\sigma}, \beta, k, s'_1, \ldots, s'_l$ modulo p_2 are uncorrelated from their values modulo p_1 by the Chinese Remainder Theorem (assume finding a nontrivial factor of N is hard). Thus the challenge ciphertext is a properly distributed semi-functional ciphertext so that this is in $Game_0$. Note it can be easily seen that all private keys and re-encryption keys generated in the simulation are normal. Therefore $\mathcal{B}$ can use the output of $\mathcal{A}$ to break Assumption 1 with advantage δ. ■

Lemma 2: If there is an algorithm $\mathcal{A}$ such that $Game_{(j-1)_t} Adv_{\mathcal{A}}^{DFA\text{-}FPRE} - Game_{j_t}^N Adv_{\mathcal{A}}^{DFA\text{-}FPRE} = \delta$, we can construct an algorithm $\mathcal{B}$ breaking Assumption 2 with advantage δ.

Proof: **Setup.** $\mathcal{B}$ is given an instance (D, T) of Assumption 2, and simulates either $Game_{(j-1)_t}$ or $Game_{j_t}^N$ with $\mathcal{A}$. $\mathcal{B}$ generates PP and MSK as in the proof of Lemma 1.

Phase 1. $\mathcal{A}$ makes the following queries:

1) $\mathcal{O}_{SK}(M)$: $\mathcal{B}$ constructs the keys for $\mathcal{A}$ as follows.

- For the first $(j-1)_{sk}$ key queries, $\mathcal{B}$ generates the semi-functional keys for $\mathcal{A}$. $\mathcal{B}$ chooses $R_{start1}, R_{start2}, R_{start3}, (\forall t \in \mathcal{T})\ R_{t,1}, R_{t,2}, R_{t,3}, (\forall q_x \in F)\ R_{end_x,1}, R_{end_x,2} \in_R \mathbb{G}_{p_3}$. For each $t \in \mathcal{T}$ it chooses $r_t \in_R \mathbb{Z}_N^*$, and $\forall q_x \in F$ it chooses $r_{end_x}, \tau_x \in_R \mathbb{Z}_N^*$. It also chooses $r_{start}, u, k \in_R \mathbb{Z}_N^*$, $D_0, D_1, \ldots, D_{|Q|-1} \in_R \mathbb{G}_{p_1}$, where D_i is associated with q_i. It sets

$$K'_{start1} = D_0 (h_{start})^{r_{start}} R_{start1},$$
$$K'_{start2} = g^{r_{start}} R_{start2}, K'_{start3} = g^u R_{start3},$$

for each $t = (x, y, \sigma) \in \mathcal{T}$:

$$K'_{t,1} = D_x^{-1} z^{r_t} R_{t,1},$$
$$K'_{t,2} = g^{r_t} R_{t,2}, K'_{t,3} = D_y (h_{\sigma})^{r_t} R_{t,3},$$

for each $q_x \in F$:

$$K'_{end_x,1} = g^{-\alpha} D_x (h_{end} g^{ab})^{r_{end_x}} g^{ku} R_{end_x,1} (Y_2 Y_3)^{\tau_x},$$
$$K'_{end_x,2} = g^{r_{end_x}} R_{end_x,2}.$$

The value of τ_x modulo p_2 is uncorrelated from its values modulo p_3 by the Chinese Remainder Theorem. Thus the above key is properly distributed.

- For the $> j_{sk}$ key queries, $\mathcal{B}$ runs the algorithm $KeyGen$ to generate keys.
- For the j_{sk}-th key query, $\mathcal{B}$ implicitly lets $g^{r_{start}}$ be the $\mathbb{G}_{p_1}$ part of T. $\mathcal{B}$ chooses $d'_0, d'_1, \ldots, d'_{|Q|-1} \in_R \mathbb{Z}_N^*$, for each $t \in \mathcal{T}$ chooses $r'_t \in_R \mathbb{Z}_N^*$, $\forall q_x \in F$ chooses $r'_{end_x} \in_R \mathbb{Z}_N^*$, a $u' \in_R \mathbb{Z}_N^*$, $R_{start1}, R_{start2}, R_{start3}, R_{t,1}, R_{t,2}, R_{t,3}, R_{end_x,1}, R_{end_x,2} \in_R \mathbb{G}_{p_3}$ (here $\mathcal{B}$ can simply set $R_{start1} = X_3^{\varphi_{start1}}$, $R_{start2} = X_3^{\varphi_{start2}}$, $R_{start3} = X_3^{\varphi_{start3}}$, $R_{t,1} = X_3^{\varphi_{t,1}}$, $R_{t,2} = X_3^{\varphi_{t,2}}$, $R_{t,3} = X_3^{\varphi_{t,3}}$, $R_{end_x,1} = X_3^{\varphi_{end_x,1}}$, $R_{end_x,2} = X_3^{\varphi_{end_x,2}}$, where $\varphi_{start1}, \varphi_{start2}, \varphi_{start3}, \varphi_{t,1}, \varphi_{t,2}, \varphi_{t,3}, \varphi_{end_x,1}, \varphi_{end_x,2} \in_R \mathbb{Z}_N^*$). It sets the semi-functional key as

$$K_{start1} = R_{start1} T^{d'_0 + \alpha_{start}},$$
$$K_{start2} = R_{start2} T_{start3} = R_{start3} T^{u'},$$

for each $t = (x, y, \sigma) \in \mathcal{T}$:

$$K_{t,1} = R_{t,1} T^{-d'_x + \beta r'_t},$$
$$K_{t,2} = R_{t,2} T^{r'_t}, K_{t,3} = R_{t,3} T^{d'_y + \alpha_{\sigma} r'_t},$$

for each $q_x \in F$:

$$K_{end_x,1} = g^{-\alpha} R_{end_x,1} T^{d'_x + (\alpha_{end} + ab) r'_{end_x} + ku'},$$
$$K_{end_x,2} = R_{end_x,2} T^{r'_{end_x}}.$$

Note this implicitly sets $r_t = r_{start} r'_t$, $D_x = g^{r_{start} d'_x}$ and $r_{end_x} = r_{start} r'_{end_x}$. If $T \in \mathbb{G}_{p_1 p_3}$, the key is a properly distributed normal key so that $\mathcal{B}$ has properly simulated $Game_{(j-1)_t}$. Otherwise, $\mathcal{B}$ has properly simulated $Game_{j_t}^N$. We implicitly let $g_2^{\epsilon_{start}}$ be the $\mathbb{G}_{p_2}$ part of T, set $\epsilon_t = \epsilon_{start} r'_t$,$\epsilon_{end_x} =$

$\epsilon_{start} r'_{end_x}$ and $d_x = \epsilon_{start} d'_x$. Besides, $u' r_{start}$ and $u' \epsilon_{start}$ are the exponents of the $\mathbb{G}_{p1}$ part and $\mathbb{G}_{p2}$ part (of K_{start3}), and the $\mathbb{G}_{p_2}$ parts of $K_{start1}, K_{start2}, K_{start3}, K_{t,1}, K_{t,2}, K_{t,3}, K_{end_{x,1}}$ and $K_{end_{x,2}}$ are $g_2^{\alpha_{start}\epsilon_{start}+d_0}, g_2^{\epsilon_{start}}, g_2^{u}, g_2^{\beta\epsilon_t - d_x}$, $g_2^{\epsilon_t}$, $g_2^{\alpha_\sigma \epsilon_t + d_y}$, $g_2^{(\alpha_{end}+ab)\epsilon_{end_x}+d_x+ku}$ and $g_2^{\epsilon_{end_x}}$, respectively.

2) $\mathcal{O}_{rk}(M, w)$:
   - For original game: since $\mathcal{B}$ can construct normal private keys, it first constructs SK_M with knowledge of MSK and next generates the re-encryption key $rk_{M\to w}$ by running the algorithm $ReKeyGen$. If $ACCEPT(M, w^*)$ and $SK_{M'}$ (for any DFA M' so that $ACCEPT(M', w)$) is given to $\mathcal{A}$, $\mathcal{B}$ outputs $\perp$.
   - For re-encrypted game: $\mathcal{B}$ generates any re-encryption key for $\mathcal{A}$.
3) $\mathcal{O}_{re}(M, w', CT)$:
   - For original game: $\mathcal{B}$ constructs the re-encryption key $rk_{M\to w'}$ as in $\mathcal{O}_{rk}$, next generates C^R via the algorithm $ReEnc$. If $ACCEPT(M, w^*)$, CT is the challenge ciphertext, and $SK_{M'}$ is obtained by $\mathcal{A}$, $\mathcal{B}$ outputs $\perp$, where $ACCEPT(M', w')$.
   - For re-encrypted game: no need to issue $\mathcal{O}_{re}$.
4) $\mathcal{O}_{dec}(M, CT)$:
   - For original game: if $ACCEPT(M, w^*)$, and CT is the challenge ciphertext, $\mathcal{B}$ outputs $\perp$. Otherwise $\mathcal{B}$ constructs SK_M to recover m.
   - For re-encrypted game: $\mathcal{B}$ constructs the private key to decrypt the ciphertext as in the real scheme.
5) $\mathcal{O}_{dec_R}(M, C^R)$:
   - For original game: if (M, C^R) is a derivative, $\mathcal{B}$ outputs $\perp$. Otherwise $\mathcal{B}$ constructs the private key to recover the message m via the algorithm Dec_R.
   - For re-encrypted game: if C^R is the challenge ciphertext, $\mathcal{B}$ outputs $\perp$. Otherwise $\mathcal{B}$ constructs SK_M as in $\mathcal{O}_{SK}$ to decrypt the ciphertext.

Challenge. $\mathcal{B}$ implicitly sets $g^{s_0} = X_1$ and $g_2^{\gamma_0} = X_2$, runs $(ssk, svk) \leftarrow KeyGen(1^n)$, chooses a random $b \in \{0, 1\}$ and constructs the challenge ciphertext as follows.

- For original game: $\mathcal{A}$ outputs m_0, m_1, and w^*. $\mathcal{B}$ then sets the challenge original ciphertext CT as

$$svk, w^*,$$
$$C_m = m_b e(g^\alpha, X_1X_2)^{s'_l}, C_{start1} = X_1X_2,$$
$$C_{start2} = (X_1X_2)^{\alpha_{start}}, C_{start3} = (X_1X_2)^{\beta_0 svk}(X_1X_2)^{\beta_1},$$
$$C_{end1} = (X_1X_2)^{s'_l}, C_{end2} = (X_1X_2)^{(\alpha_{end}+ab)s'_l},$$
$$C_{end3} = (X_1X_2)^{ks'_l},$$

  for $i = 1$ to l: $C_{i,1} = (X_1X_2)^{s'_i}, C_{i,2} = (X_1X_2)^{s'_i \alpha_{w_i}} (X_1X_2)^{s'_{i-1}\beta}$, and $C_{end4} = Sign(ssk, (w^*, C_m, C_{start1}, C_{start2}, C_{start3}, (C_{1,1}, C_{1,2}), \ldots, (C_{l,1}, C_{l,2}), C_{end2}, C_{end3}))$, where $s'_1, \ldots, s'_l \in_R \mathbb{Z}_N^*$. $\mathcal{B}$ outputs $CT = (svk, w^*, C_m, C_{0,1}, C_{start2}, C_{start3}, (C_{1,1}, C_{1,2}), \ldots, (C_{end1}, C_{l,2}), C_{end2}, C_{end3}, C_{end4})$ to $\mathcal{A}$. Note we have the semi-functional ciphertext with $\gamma_i = \gamma_0 \cdot s'_i$, and $s_i = s_0 \cdot s'_i$, where $i \in \{1, \ldots, l\}$, and the values of the exponents of X_1X_2 modulo p_1 are uncorrelated from their values modulo p_2.
- For re-encrypted game: $\mathcal{A}$ outputs m_0, m_1, w' and w^*. $\mathcal{B}$ runs $CT = Encrypt(PP, w', m_b)$, generates the re-encryption key $rk_{M\to w^*}$ and constructs A_{end} as in the real scheme. It further sets C_2 in the identical method described above. $\mathcal{B}$ finally sets $C_1 = SYM.Enc(H_2(\delta), \xi)$, and outputs the challenge re-encrypted ciphertext $C^R = (C_1, C_2)$ to $\mathcal{A}$, where $\xi = (CT||A_{end}||rk_4)$.

Phase 2. Same as Phase 1.

Guess. $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs.

Therefore if $T \in \mathbb{G}_{p_1p_3}$, the simulation is in $Game_{(j-1)_t}$. Otherwise, the simulation is in $Game^N_{j_t}$. $\mathcal{B}$ can use the output of $\mathcal{A}$ to break Assumption 2 with advantage δ. ■

Lemma 3: If there is an algorithm $\mathcal{A}$ such that $Game^N_{j_t} Adv^{DFA\text{-}FPRE}_{\mathcal{A}} - Game^T_{j_t} Adv^{DFA\text{-}FPRE}_{\mathcal{A}} = \delta$ for a j from Phase 1, we can build an algorithm $\mathcal{B}$ breaking the source group modified q-BDHE assumption in a subgroup with advantage δ.

Proof: **Setup.** $\mathcal{B}$ is given an instance (D, T) of the source group modified q-BDHE assumption in a subgroup, and simulates either $Game^N_{j_t}$ or $Game^T_{j_t}$ with $\mathcal{A}$. $\mathcal{B}$ generates PP and MSK as in the proof of the previous lemma.

Phase 1. $\mathcal{A}$ makes the following queries:

1) $\mathcal{O}_{SK}(M)$: $\mathcal{B}$ constructs the private keys for $\mathcal{A}$ as follows.
   - For the first $(j-1)_{sk}$ and $> j_{sk}$ key queries, $\mathcal{B}$ generates the semi-functional keys and the normal keys for $\mathcal{A}$ as in the previous lemma.
   - For the j_{sk}-th key query, $\mathcal{B}$ runs the algorithm $KeyGen$ to generate a normal key K_{start1}, K_{start2}, $\forall t \in \mathcal{T}(K_{t,1}, K_{t,2}, K_{t,3}), \forall q_x \in F(K_{end_{x,1}}, K_{end_{x,2}})$, and next sets

$$K_{start1} g_2^{d'_0} g_2^{\alpha'_{start}\epsilon'_{start}}, K_{start2} g_2^{\epsilon'_{start}}, K_{start3} g_2^{a},$$

   for each $t = (x, y, \sigma) \in \mathcal{T}$:

$$K_{t,1} g_2^{-d'_x} g_2^{\beta'\epsilon'_t}, K_{t,2} g_2^{\epsilon'_t}, K_{t,3} g_2^{d'_y} g_2^{\alpha'_\sigma \epsilon'_t},$$

   for each $q_x \in F$:

$$K_{end_{x,1}} g_2^{d'_x} T^{r'_{end_x}} g_2^{afer'_{end_x}}, K_{end_{x,2}} g_2^{er'_{end_x}},$$

   where $d'_0, \forall\ t = (x, y, \sigma) \in \mathcal{T}\ \epsilon'_t, d_x, \forall\ x \in F\ r'_{end_x}, \forall\ \sigma \in \sum\ \alpha'_\sigma, \alpha'_{start}, \epsilon'_{start}, \beta' \in_R \mathbb{Z}_N^*$. This implicitly sets $(ab + \alpha_{end}) \cdot \epsilon_{end_x} = (aec^{q+1} + afe) \cdot r'_{end_x}$, $b = -c^q - c^{q-1} - \cdots - c^{q-n+2} + f$, $\alpha_{end} = ac \cdot (c^q + c^{q-1} + \cdots + c^{q-n+1})$ and $\epsilon_{end_x} = e \cdot r'_{end_x}$, where q is the maximum allowable number of distinct symbols in $\sum$, and n is the total number of the distinct symbols used in the DFA. Note we here give a limitation to n such that $n \le q - 1$. If $T = g_2^{aec^{q+1}}$, the above key is a properly distributed nominal semi-functional key. If $T \in_R \mathbb{G}_{p_2}$, $\mathcal{B}$ has properly simulated $Game^T_{j_{sk}}$.
2) The responses of the queries to $\mathcal{O}_{rk}, \mathcal{O}_{re}, \mathcal{O}_{dec}, \mathcal{O}_{dec_R}$ are the same as that of previous lemma.

Challenge. $\mathcal{B}$ chooses random elements $\gamma_0', \ldots, \gamma_l' \in_R \mathbb{Z}_N^*$. It then runs $(ssk, svk) \leftarrow KeyGen(1^n)$, chooses a random $b \in \{0, 1\}$ and constructs the challenge ciphertext as follows.

- For original game: $\mathcal{A}$ outputs m_0, m_1, and w^*. $\mathcal{B}$ then runs the algorithm Enc to generate a normal ciphertext consisting of

$$C_m, C_{start1}, C_{start2}, C_{start3}, (C_{1,1}, C_{1,2}), \ldots, (C_{l,1}, C_{l,2}), C_{end2}, C_{end3},$$

and sets the challenge semi-functional ciphertext CT as

$$svk, w^*, C_m, C_{start1} = C_{start1} g_2^{1/ac^q \gamma_0'},$$
$$C_{start2} = C_{start2} g_2^{1/ac^q \gamma_0' \alpha_{start}'},$$
$$C_{start3} = C_{start3} (g_2^{1/ac^q})^{\gamma_0' \beta_0' svk} (g_2^{1/ac^q})^{\gamma_0' \beta_1'},$$
$$C_{end1} = C_{end1} (g_2^{1/ac^q})^{\gamma_l'}, C_{end2} = C_{end2} (g_2^{1/ac^q})^{\gamma_l' \alpha_{end}'},$$
$$C_{end3} = C_{end3} (g_2^{1/ac^q})^{\gamma_l' k'},$$

for $i = 1$ to l: $C_{i,1} = C_{i,1} (g_2^{1/ac^q})^{\gamma_i'}, C_{i,2} = C_{i,2} g_2^{1/ac^q \alpha_{w_i}' \gamma_i' + 1/ac^q \gamma_{i-1}' \beta'}$, and $C_{end4} = Sign(ssk, (w, C_m, C_{start1}, C_{start2}, C_{start3}, (C_{1,1}, C_{1,2}), \ldots, (C_{l,1}, C_{l,2}), C_{end2}, C_{end3}))$, where $\beta_0', \beta_1' \in_R \mathbb{Z}_N^*$. $\mathcal{B}$ outputs $CT = (svk, w, C_m, C_{start1}, C_{start2}, C_{start3}, (C_{1,1}, C_{1,2}), \ldots, (C_{end1}, C_{l,2}), C_{end2}, C_{end3}, C_{end4})$ to $\mathcal{A}$.
- For re-encrypted game: $\mathcal{A}$ outputs m_0, m_1, a w' and a w^*. $\mathcal{B}$ runs $CT = Encrypt(PP, w', m_b)$, generates $rk_{M \to w^*}$ and constructs A_{end} as in the real scheme. It sets $C_2 = (ssk, (w, C_\delta, C_{start1}, C_{start2}, C_{start3}, (C_{1,1}, C_{1,2}), \ldots, (C_{end1}, C_{l,2}), C_{end2}, C_{end3}, C_{end4}))$ as above. $\mathcal{B}$ finally sets $C_1 = SYM.Enc(H_2(\delta), \xi)$, and outputs $C^{R*} = (C_1, C_2)$ to $\mathcal{A}$, where $\xi = (CT || A_{end} || rk_4)$.

Guess. $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs.

Therefore if $T \in_R \mathbb{G}_{p_2}$, the simulation is in $Game_{j_t}^T$. Otherwise, the simulation is in $Game_{j_t}^N$. Thus $\mathcal{B}$ can use the output of $\mathcal{A}$ to break the source group q-BDHE assumption in a subgroup with advantage δ. ∎

Lemma 4: If there is an algorithm $\mathcal{A}$ such that $Game_{j_t}^N Adv_{\mathcal{A}}^{DFA\text{-}FPRE} - Game_{j_t}^T Adv_{\mathcal{A}}^{DFA\text{-}FPRE} = \delta$ for a j from Phase 2, we can build an algorithm $\mathcal{B}$ breaking the source group l-Expanded BDHE assumption in a subgroup with advantage δ.

Proof: **Setup.** $\mathcal{B}$ is given an instance (D, T) of the source group l-Expanded BDHE assumption, and simulates either $Game_{j_t}^N$ or $Game_{j_t}^T$ for some j from Phase 2 with $\mathcal{A}$. $\mathcal{B}$ generates PP and MSK as in the proof of Lemma 1.

Challenge. $\mathcal{B}$ chooses a random $b \in \{0, 1\}$, runs $(ssk, svk) \leftarrow KeyGen(1^n)$ and generates the challenge ciphertext.

- For original game: $\mathcal{A}$ outputs m_0, m_1, and w^*. $\mathcal{B}$ first generates the normal components of the challenge ciphertext as in $Encrypt$, and obtains the normal components consisting of $(w, C_m, C_{start1}, C_{start2}, C_{start3}, (C_{1,1}, C_{1,2}), \ldots, (C_{l,1}, C_{l,2}), C_{end1}, C_{end2})$. $\mathcal{B}$ chooses $v_z, v_{start}, v_{end}, k' \in_R \mathbb{Z}_N^*$ and $\forall \sigma \in \sum$, $v_\sigma \in_R \mathbb{Z}_N^*$. It implicitly sets $\beta' = v_z + ab/dx, \alpha_{start}' = v_{start} - \sum_{j \in [1,l^*]} a^j b/c_j x, \alpha_{end}' = v_{end} - \sum_{j \in [2,l^*+1]} a^j b/c_j x, \forall \sigma \in \sum, \alpha_{w_i}' = v_\sigma - b/dx - \sum_{j \in [0,l^*+1] s.t. w_j^* \neq \sigma} a^{(l^*+1-j)} b/c_{(l^*+1-j)} x$, and $\gamma_i = mna^i$, and next constructs the challenge ciphertext by adding the parts in $\mathbb{G}_{p_2}$ to the normal components as follows.

$$C_{start1} = C_{start1} g_2^{mna^0} = C_{start1} g_2^{\gamma_0},$$
$$C_{start2} = C_{start2} (g_2^{mna^0})^{v_{start}} \prod_{j \in [1,l^*]} g_2^{-a^j bmna^0/c_j x} = C_{start2} g_2^{\gamma_0 v_{start}} \prod_{j \in [1,l^*]} g_2^{-a^j b \gamma_0 / c_j x},$$
$$C_{start3} = C_{start3} (g_2^{mna^0})^{\beta_0' svk} (g_2^{mna^0})^{\beta_1'} = C_{start3} g_2^{\gamma_0 \beta_0' svk} g_2^{\gamma_0 \beta_1'},$$
$$C_{end1} = C_{end1} g_2^{mna^{l^*}} = C_{end1} g_2^{\gamma_{l^*}},$$
$$C_{end2} = C_{end2} (g_2^{mna^{l^*}})^{v_{end} + \hat{a}\hat{b}} \prod_{j \in [2,l^*+1]} g_2^{-a^{l^*+j} bmn/c_j x} = C_{end2} g_2^{\gamma_{l^*}(v_{end} + \hat{a}\hat{b})} \prod_{j \in [2,l^*+1]} g_2^{-a^j b \gamma_{l^*}/c_j x},$$
$$C_{end3} = C_{end3} (g_2^{mna^{l^*}})^{k'} = C_{end3} g_2^{k' \gamma_{l^*}},$$

for $i = 1$ to l

$$C_{i,1} = C_{i,1} g_2^{mna^i} = C_{i,1} g_2^{\gamma_i},$$
$$C_{i,2} = C_{i,2} (g_2^{mna^i})^{v_{w_i^*}} (g_2^{mna^{i-1}})^{v_z} \cdot \prod_{j \in [0,l^*+1] s.t. w_j^* \neq w_i^*} g_2^{-a^{l^*+1-j+i} bmn/c_j x} = C_{i,2} g_2^{\gamma_i v_{w_i^*}} g_2^{\gamma_{i-1} v_z} \cdot \prod_{j \in [0,l^*+1] s.t. w_j^* \neq w_i^*} g_2^{-a^{l^*+1-j} b \gamma_i / c_j},$$

where $v_z, v_{w_i^*}, \beta_1', \beta_0', \hat{a}, \hat{b} \in_R \mathbb{Z}_N^*$ chosen by $\mathcal{B}$. Finally, $\mathcal{B}$ sets $C_{end4} = Sign(ssk, (w, C_m, C_{start1}, C_{start2}, C_{start3}, (C_{1,1}, C_{1,2}), \ldots, (C_{l,1}, C_{l,2}), C_{end2}, C_{end3}))$, and outputs the challenge original ciphertext $CT^* = (svk, w, C_m, C_{start1}, C_{start2}, C_{start3}, (C_{1,1}, C_{1,2}), \ldots, (C_{end1}, C_{l,2}), C_{end2}, C_{end3}, C_{end4})$ to $\mathcal{A}$. It is not difficult to see that $\mathcal{B}$ can construct the challenge ciphertext using the terms given in the problem instance.
- For re-encrypted game: $\mathcal{A}$ outputs m_0, m_1, a w' and a w^*. $\mathcal{B}$ then runs $CT = Encrypt(PP, w', m_b)$, generates $rk_{M \to w^*}$ (using the normal private key SK_M) and constructs A_{end} as in the real scheme. It sets C_2 as above. $\mathcal{B}$ sets $C_1 = SYM.Enc(H_2(\delta), \xi)$, and outputs $C^R = (C_1, C_2)$ to $\mathcal{A}$, where $\xi = (CT || A_{end} || rk_4)$.

Phase 2. $\mathcal{A}$ makes the following queries:

1) $\mathcal{O}_{SK}(M)$: $\mathcal{A}$ submits a DFA M to $\mathcal{B}$ where for any M such that $REJECT(M, w^*)$. For the first $(j-1)_{sk}$ and $> j_{sk}$ key queries, $\mathcal{B}$ generates the semi-functional keys and the normal keys for $\mathcal{A}$ as in the previous lemma. Otherwise, $\mathcal{B}$ constructs the private key for $\mathcal{A}$ as follows. Note we use $w^{*(i)}$ denote the last i

symbols of w^*, M_k denote a DFA $M_k = (Q, \mathcal{T}, q_k, F)$, where q_k is the start state and $k \in \{0, \ldots, |Q| - 1\}$. For each $q_k \in Q$ we defined a set S_k including indices in $\{0, 1, \ldots, l^*\}$, we say $i \in \{0, 1 \ldots, l^*\}$ is in S_k if and only if $ACCEPT(M_k, w^{*(i)})$. We set $D_k = (\prod_{i \in S_k} g_2^{a^{i+1} \cdot b/x}) \cdot g_2^{a^{l^*+1} bm} \cdot g_2^n$. Actually, $\mathcal{B}$ cannot directly compute D_k from the problem instance. Fortunately the uncomputable components will be canceled out so that the key components to be consistent with the values.

a) $\mathcal{B}$ implicitly sets $\epsilon_{start} = \sum_{i \in S_0} c_{i+1}$ and $d_0 = \sum_{i \in S_0} a^{i+1} \cdot b/x + a^{l^*+1} bm$. Thus we have

$$\begin{aligned}&\alpha_{start} \cdot \epsilon_{start} + d_0 \\ &= (v_{start} - \sum_{j \in [1, l^*]} a^j \cdot b/c_j x) \cdot \sum_{i \in S_0} c_{i+1} + \\ &\sum_{i \in S_0} a^{i+1} \cdot b/x + a^{l^*+1} bm + n.\end{aligned}$$

Thus $\mathcal{B}$ sets the $\mathbb{G}_{p_2}$ parts for K_{start1} and K_{start2} as $g_2^{\sum_{i \in S_0} c_{i+1}}$ and $g_2^{v_{start} \cdot \sum_{i \in S_0} c_{i+1}} \cdot g_2^{-\sum_{j \in [1, l^*], i \in S_0, j \neq i+1} a^j \cdot b \cdot c_{i+1}/c_j x} \cdot T \cdot g_2^n$ respectively.

b) Similarly, $\mathcal{B}$ sets $\epsilon_{end_x} = \sum_{i \in S_x, i \neq 0} c_{i+1}$ and $d_x = \sum_{i \in S_x} a^{i+1} \cdot b/x + a^{l^*+1} bm$ such that

$$\begin{aligned}&(\alpha_{end} + \hat{a}\hat{b}) \cdot \epsilon_{end_x} + d_x + k'u' \\ &= \Big(v_{end} - \sum_{j \in [2, l^*+1]} a^j \cdot b/c_j x + \hat{a}\hat{b}\Big) \cdot \sum_{i \in S_x, i \neq 0} c_{i+1} \\ &\quad + \sum_{i \in S_x} a^{i+1} \cdot b/x + a^{l^*+1} bm + n + k'u',\end{aligned}$$

where $\hat{a}, \hat{b}, u' \in_R \mathbb{Z}_N^*$. Here $\mathcal{B}$ can construct the $\mathbb{G}_{p_2}$ parts for $K_{end_{x,1}}$ and $K_{end_{x,2}}$ using the terms given in the problem instance as $g_2^{\sum_{i \in S_x, i \neq 0} c_{i+1}}, g_2^{(v_{end} + \hat{a}\hat{b}) \cdot \sum_{i \in S_x, i \neq 0} c_{i+1}} \cdot g_2^{-\sum_{j \in [2, l^*+1], i \in S_x, i \neq 0, j \neq i+1} a^j \cdot b \cdot c_{i+1}/c_j x} T g_2^{ab/x} g_2^n g_2^{k'u'}$.

c) $\mathcal{B}$ constructs the key components $K_{t,1}, K_{t,2}, K_{t,3}$ for each transition $t = (x, y, \sigma) \in \mathcal{T}$. Like [32] for $i = 0$ to $l^* + 1$ we define $(K_{t,1,i}, K_{t,2,i}, K_{t,3,i})$ such that $K_{t,1} = \prod_{i \in [0, l^*+1]} K_{t,1,i}$, $K_{t,2} = \prod_{i \in [0, l^*+1]} K_{t,2,i}$ and $K_{t,3} = \prod_{i \in [0, l^*+1]} K_{t,3,i}$. $\mathcal{B}$ will generate these components through four possible cases.

- **Case 1:** $i \notin S_x \wedge (i-1) \notin S_y$, $\mathcal{B}$ sets $K_{t,1,i}, K_{t,2,i}, K_{t,3,i}$ to be 1.
- **Case 2:** $i \in S_x \wedge (i-1) \in S_y$, $\mathcal{B}$ sets $K_{t,2,i} = g_2^{a^i d}$ so that $K_{t,1,i} = g_2^{(v_z + ab/dx) \cdot a^i d - a^{i+1} b/x + a^{l^*+1} bm + n}$ and $K_{t,3,i} = g_2^{(v_\sigma - b/dx - a^{l^*+1-j} b/c_{l^*+1-j} x) \cdot a^i d + a^i b/x + a^{l^*+1} bm + n}$.
  $\mathcal{B}$ then sets $K_{t,1,i} = g_2^{a^i d v_z} \cdot T \cdot g_2^n = K_{t,2,i}^{v_z} \cdot T \cdot g_2^n$, and $K_{t,3,i} = K_{t,2,i}^{v_\sigma} \cdot \prod_{j \in [0, l^*+1] s.t. w_j^* \neq \sigma} g_2^{-a^{(l^*+1-j+i)} bd/c_{(l^*+1-j)} x} \cdot T \cdot g_2^n$.
- **Case 3:** $i \notin S_x \wedge (i-1) \in S_y \wedge w_{l^*+1-i}^* \neq \sigma$, $\mathcal{B}$ sets $K_{t,2,i} = g_2^{c_i}$ so that $K_{t,1,i} = g_2^{(v_z + ab/dx) \cdot c_i}$ and $K_{t,3,i} = g_2^{(v_\sigma - b/dx - a^{l^*+1-j} b/c_{l^*+1-j} x) \cdot c_i + a^i b/x + a^{l^*+1} bm + n}$.
  It then sets $K_{t,1,i} = g_2^{v_z \cdot c_i + abc_i/dx} = K_{t,2,i}^{v_z} \cdot g_2^{abc_i/dx}$, and $K_{t,3,i} = K_{t,2,i}^{v_\sigma} \cdot g_2^{-bc_i/dx} \cdot \prod_{j \in [0, l^*+1] s.t. j \neq l^*+1-i \wedge w_j^* \neq \sigma} g_2^{-a^{(l^*+1-j)} bc_i/c_{(l^*+1-j)} x} \cdot T \cdot g_2^n$.
- **Case 4:** $i \in S_x \wedge (i-1) \notin S_y \wedge w_{l^*+1-i}^* \neq \sigma$, $\mathcal{B}$ sets $K_{t,2,i} = g_2^{a^i d - c_i}$ so that $K_{t,1,i} = g_2^{(v_z + ab/dx) \cdot (a^i d - c_i) - a^{i+1} b/x - a^{l^*+1} bm - n}$ and $K_{t,3,i} = g_2^{(v_\sigma - b/dx - a^{l^*+1-j} b/c_{l^*+1-j} x) \cdot (-c_i + a^i d)}$.
  It then sets $K_{t,1,i} = K_{t,2,i}^{v_z} \cdot g_2^{-abc_i/dx} \cdot T^{-1} \cdot g_2^{-n}$, and $K_{t,3,i} = K_{t,2,i}^{v_\sigma} \cdot g_2^{bc_i/dx} \cdot \prod_{j \in [0, l^*+1] s.t. w_j^* \neq \sigma} g_2^{-a^{(l^*+1-j+i)} bd/c_{(l^*+1-j)} x} \cdot \prod_{j \in [0, l^*+1] s.t. j \neq l^*+1-i \wedge w_j^* \neq \sigma} g_2^{-a^{(l^*+1-j)} bc_i/c_{(l^*+1-j)} x}$.
  $\mathcal{B}$ can compute all the above components using the terms given in the problem instance.

2) The responses of the queries to $\mathcal{O}_{rk}, \mathcal{O}_{re}, \mathcal{O}_{dec}, \mathcal{O}_{dec_R}$ are the same as that of previous lemma.

Guess. $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs.

If $T \in_R \mathbb{G}_{p_2}$, the j_t-th private key constructed above is a properly distributed temporary semi-functional key. If $T = g_2^{a^{l^*+1} bm}$, we have the properly distributed nominal semi-functional key. Thus $\mathcal{B}$ can use $\mathcal{A}$ to break the source group l-BDHE assumption in a subgroup with advantage δ. ■

Lemma 5: If there is an algorithm $\mathcal{A}$ such that $Game_{j_t}^T Adv_{\mathcal{A}}^{DFA\text{-}FPRE} - Game_{j_t} Adv_{\mathcal{A}}^{DFA\text{-}FPRE} = \delta$, we can construct an algorithm $\mathcal{B}$ breaking Assumption 2 with advantage δ.

Proof: This proof is identical to that of Lemma 2 except that $\mathcal{B}$ will use $Y_2 Y_3$ to construct random elements of $\mathbb{G}_{p_2}$ ($\forall q_x \in F$) such that all $K_{end_{x,1}}$ parts of the i-th key will be randomly masked, and the rest of key components will not have $\mathbb{G}_{p_2}$ parts. Namely the j_t-th key is semi-functional. ■

Lemma 6: If there is an algorithm $\mathcal{A}$ such that $Game_q Adv_{\mathcal{A}}^{DFA\text{-}FPRE} - Game_{final} Adv_{\mathcal{A}}^{DFA\text{-}FPRE} = \delta$, we can build an algorithm $\mathcal{B}$ breaking Assumption 3 with advantage δ.

Proof: **Setup.** $\mathcal{B}$ is given an instance (D, T) of Assumption 3, and simulates either $Game_q$ or $Game_{final}$ with $\mathcal{A}$. $\mathcal{B}$ chooses $\beta, \beta_0, \beta_1, \alpha_{start}, \alpha_{end}, a, b, k \in_R \mathbb{Z}_N^*$, and $\alpha_\sigma \in_R \mathbb{Z}_N^*$ for all symbols in $\sum$. It then chooses H_1, H_2, an OTS and an SYM as in the real scheme, and outputs PP:

$$\begin{aligned}&g, g^{ab}, g_0 = g^{\beta_0}, z = g^\beta, h_0 = g^{\beta_1}, h_k = g^k, h_{start} = g^{\alpha_{start}}, \\ &h_{end} = g^{\alpha_{end}}, \forall_{\sigma \in \sum} h_\sigma = g^{\alpha_\sigma}, e(g, g^\alpha X_2), \\ &H_1, H_2, OTS, SYM.\end{aligned}$$

Note here α is unknown to $\mathcal{B}$.

Phase 1. $\mathcal{A}$ makes the following queries:

1) $\mathcal{O}_{SK}(M)$: $\mathcal{B}$ chooses $D_0, D_1, \ldots, D_{|Q|-1} \in_R \mathbb{G}_{p_1}$. For each $t \in \mathcal{T}$ it chooses $r_t, \delta_{t,1}, \delta_{t,2}, \delta_{t,3} \in_R \mathbb{Z}_N^*$, and $\forall q_x \in F$ it chooses $r_{end_x}, \delta_{end_{x,1}}, \delta_{end_{x,2}}, k_x \in_R \mathbb{Z}_N^*$. It also chooses $r_{start}, \delta_{start1}, \delta_{start2}, u' \in_R \mathbb{Z}_N^*$. It then sets

$$K_{start1} = D_0(h_{start})^{r_{start}} X_3^{\delta_{start1}},$$
$$K_{start2} = g^{r_{start}} X_3^{\delta_{start2}}, K_{start3} = g^u X_3^{u'},$$

for each $t = (x, y, \sigma) \in \mathcal{T}$: $K_{t,1} = D_x^{-1} z^{r_t} X_3^{\delta_{t,1}}$, $K_{t,2} = g^{r_t} X_3^{\delta_{t,2}}$, $K_{t,3} = D_y(h_\sigma)^{r_t} X_3^{\delta_{t,3}}$, for each $q_x \in F$: $K_{end_{x,1}} = (g^\alpha X_2)^{-1} D_x(h_{end} g^{ab})^{r_{end_x}} g^{ku} X_3^{\delta_{end_{x,1}}} Z_2^{k_x}$, $K_{end_{x,2}} = g^{r_{end_x}} X_3^{\delta_{end_{x,2}}}$.

2) $\mathcal{O}_{rk}(M, w)$: $\mathcal{B}$ can construct any re-encryption key as it knows any semi-functional private key for a DFA M.
   - For original game: if $ACCEPT(M, w^*)$ and $SK_{M'}$ (for any DFA M' so that $ACCEPT(M', w)$) is obtained by $\mathcal{A}$, $\mathcal{B}$ outputs $\perp$. Else, $\mathcal{B}$ constructs SK_M as in $\mathcal{O}_{SK}$, and next constructs $rk_{M \to w}$ via $ReKeyGen$.
   - For re-encrypted game: $\mathcal{B}$ generates any re-encryption key for $\mathcal{A}$.
3) $\mathcal{O}_{re}(M, w', CT)$:
   - For original game: if $ACCEPT(M, w^*)$, CT is the challenge ciphertext, and $SK_{M'}$ (for any DFA M' so that $ACCEPT(M', w')$) is obtained by $\mathcal{A}$, $\mathcal{B}$ outputs $\perp$. Otherwise, $\mathcal{B}$ constructs $rk_{M \to w'}$ as in $\mathcal{O}_{rk}$, next generates C^R via $ReEnc$.
   - For re-encrypted game: no need to issue $\mathcal{O}_{re}$.
4) $\mathcal{O}_{dec}(M, CT)$:
   - For original game: $\mathcal{B}$ constructs the semi-functional private key SK_M as in $\mathcal{O}_{SK}$, and next recovers m via Dec. If $ACCEPT(M, w^*)$, and CT is the challenge ciphertext, $\mathcal{B}$ outputs $\perp$.
   - For re-encrypted game: $\mathcal{B}$ decrypts the ciphertext by using the corresponding semi-functional key.
5) $\mathcal{O}_{dec_R}(M, C^R)$:
   - For original game: $\mathcal{B}$ constructs the semi-functional private key SK_M, and next recovers m via Dec_R. If (M, C^R) is a derivative, $\mathcal{B}$ outputs $\perp$.
   - For re-encrypted game: $\mathcal{B}$ recovers m as above except that $\mathcal{B}$ outputs $\perp$ if C^R is the challenge ciphertext.

Challenge. $\mathcal{B}$ chooses a random $b \in \{0, 1\}$, runs $(ssk, svk) \leftarrow KeyGen(1^n)$ and generates the challenge ciphertext.

- For original game: $\mathcal{A}$ commits to two equal-length messages m_0, m_1, and a challenge string w^*. $\mathcal{B}$ sets svk, w^*,

$$C_m = m_b \cdot T^{s_l'}, C_{start1} = g^s Y_2, C_{start2} = (g^s Y_2)^{\alpha_{start}},$$
$$C_{start3} = (g^s Y_2)^{\beta_0 \cdot svk} \cdot (g^s Y_2)^{\beta_1}, C_{end1} = (g^s Y_2)^{s_l'},$$
$$C_{end2} = (g^s Y_2)^{\alpha_{end} \cdot s_l'}, C_{end3} = (g^s Y_2)^{k \cdot s_l'},$$

for $i = 1$ to l: $C_{i,1} = (g^s Y_2)^{s_i'}$, $C_{i,2} = (g^s Y_2)^{s_i' \cdot \alpha_{w_i}} \cdot (g^s Y_2)^{s_{i-1}' \cdot \beta}$, where $s_1', \ldots, s_l' \in_R \mathbb{Z}_N^*$. Finally, $\mathcal{B}$ sets $C_{end4} = Sign(ssk, (w, C_m, C_{start1}, C_{start2}, C_{start3}, (C_{1,1}, C_{1,2}), \ldots, (C_{end1}, C_{l,2}), C_{end2}, C_{end3}))$, and outputs the challenge original ciphertext $CT = (svk, w, C_m, C_{start1}, C_{start2}, C_{start3}, (C_{1,1}, C_{1,2}), \ldots, (C_{end1}, C_{l,2}), C_{end2}, C_{end3}, C_{end4})$ to $\mathcal{A}$.
- For re-encrypted game: $\mathcal{A}$ outputs m_0, m_1, a w' and a w^*. $\mathcal{B}$ runs $CT = Encrypt(PP, w', m_b)$, generates $rk_{M \to w^*}$ (using the semi-functional private key SK_M) and constructs A_{end} as in the real scheme. It sets C_2 to be an encryption of a random element $\delta \in_R \mathbb{Z}_N^*$ as above, sets $C_1 = SYM.Enc(H_2(\delta), \xi)$, and outputs $C^R = (C_1, C_2)$, where $\xi = (CT || A_{end} || rk_4)$.

Phase 2. Same as Phase 1.

Guess. $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs.

This implicitly sets $Y_2 = g_2^{\gamma_0}$, $s = s_0, s_i = s \cdot s_i'$ for each $i \in \{1, \ldots, l\}$. If $T \in \mathbb{G}_T$, the above ciphertext is a properly distributed semi-functional ciphertext of a random message in $\mathbb{G}_T$. If $T = e(g, g)^{\alpha \cdot s}$, we have the semi-functional ciphertext with $\gamma_i = \gamma_0 \cdot s_i$. This is a properly distributed semi-functional encryption of m_b. ■

IV. Conclusion

In this paper for the first time we defined the notion of DFA-based FPRE, and meanwhile proposed a concrete scheme satisfying the new notion. Furthermore we proved the scheme, which is the first of its type, to be adaptively CCA secure in the standard model by employing Lewko et al.'s dual encryption technology. This work motivates some interesting open problems. One of them is how to convert our DFA-based FPRE in the prime order bilinear group.

References

[1] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Trans. Inf. Syst. Security*, vol. 9, no. 1, pp. 1–30, Feb. 2006.

[2] N. Attrapadung, J. Herranz, F. Laguillaumie, B. Libert, E. de Panafieu, and C. Rafols, "Attribute-based encryption schemes with constant-size ciphertexts," *Theoretical Comput. Sci.*, vol. 422, pp. 15–38, Mar. 2012.

[3] M. Bellare and S. Shoup, "Two-tier signatures, strongly unforgeable signatures, and Fiat-Shamir without random oracles," in *Public Key Cryptography* (Lecture Notes in Computer Science), vol. 4450. Berlin, Germany: Springer-Verlag, 2007, pp. 201–216.

[4] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Security Privacy*, May 2007, pp. 321–334.

[5] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *Advances in Cryptology*. Berlin, Germany: Springer-Verlag, 1998, pp. 127–144.

[6] D. Boneh, E.-J. Goh, and K. Nissim, "Evaluating 2-DNF formulas on ciphertexts," in *Theory of Cryptography* (Lecture Notes in Computer Science), vol. 3378. Berlin, Germany: Springer-Verlag, 2005, pp. 325–341.

[7] R. Canetti and S. Hohenberger, "Chosen-ciphertext secure proxy re-encryption," in *Proc. 14th ACM CCS*, 2007, pp. 185–194.

[8] L. Cheung and C. Newport, "Provably secure ciphertext policy ABE," in *Proc. 14th ACM CCS*, 2007, pp. 456–465.

[9] R. Cramer and V. Shoup, "Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack," *SIAM J. Comput.*, vol. 33, no. 1, pp. 167–226, Jan. 2004.

[10] V. Goyal, A. Jain, O. Pandey, and A. Sahai, "Bounded ciphertext policy attribute based encryption," in *Proc. 35th Int. Colloq. ICALP*, Jul. 2008, pp. 579–591.

[11] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. 13th ACM CCS*, 2006, pp. 89–98.

[12] M. Green and G. Ateniese, "Identity-based proxy re-encryption," in *Proc. 5th Int. Conf. ACNS*, vol. 4512. Jun. 2007, pp. 288–306.

[13] G. Hanaoka *et al.*, "Generic construction of chosen ciphertext secure proxy re-encryption," in *Topics in Cryptology* (Lecture Notes in Computer Science), vol. 7178. Berlin, Germany: Springer-Verlag, 2012, pp. 349–364.

[14] T. Isshiki, M. H. Nguyen, and K. Tanaka, "Proxy re-encryption in a stronger security model extended from CT-RSA2012," in *Topics in Cryptology* (Lecture Notes in Computer Science), vol. 7779. Berlin, Germany: Springer-Verlag, pp. 277–292.

[15] A. A. Ivan and Y. Dodis, "Proxy cryptography revisited," in *Proc. NDSS*, Feb. 2003.

[16] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, "Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 6110. Berlin, Germany: Springer-Verlag, 2010, pp. 62–91.

[17] A. 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, 2012, pp. 180–198.

[18] K. Liang, M. H. Au, W. Susilo, D. S. Wong, G. Yang, and Y. Yu, "An adaptively CCA-secure ciphertext-policy attribute-based proxy re-encryption for cloud data sharing," in *Proc. 10th Int. Conf. ISPEC*, vol. 8434. May 2014, pp. 448–461.

[19] K. Liang, C.-K. Chu, X. Tan, D. S. Wong, C. Tang, and J. Zhou, "Chosen-ciphertext secure multi-hop identity-based conditional proxy re-encryption with constant-size ciphertexts," *Theoretical Comput. Sci.*, vol. 539, pp. 87–105, Jun. 2014.

[20] K. Liang, L. Fang, W. Susilo, and D. S. Wong, "A ciphertext-policy attribute-based proxy re-encryption with chosen-ciphertext security," in *Proc. 5th Int. Conf. INCoS*, Sep. 2013, pp. 552–559.

[21] K. Liang, Q. Huang, R. Schlegel, D. S. Wong, and C. Tang, "A conditional proxy broadcast re-encryption scheme supporting timed-release," in *Information Security Practice and Experience* (Lecture Notes in Computer Science), vol. 7863, R. H. Deng and T. Feng, Eds. Berlin, Germany: Springer-Verlag, 2013, pp. 132–146.

[22] K. Liang, Z. Liu, X. Tan, D. S. Wong, and C. Tang, "A CCA-secure identity-based conditional proxy re-encryption without random oracles," in *Information Security Practice and Experience* (Lecture Notes in Computer Science), vol. 7839. Berlin, Germany: Springer-Verlag, 2013, pp. 231–246.

[23] X. Liang, Z. Cao, H. Lin, and J. Shao, "Attribute based proxy re-encryption with delegating capabilities," in *Proc. 4th Int. Symp. ASIACCS*, 2009, pp. 276–286.

[24] B. Libert and D. Vergnaud, "Tracing Malicious proxies in proxy re-encryption," in *Pairing-Based Cryptography* (Lecture Notes in Computer Science), vol. 5209. Berlin, Germany: Springer-Verlag, 2008, pp. 332–353.

[25] B. Libert and D. Vergnaud, "Unidirectional chosen-ciphertext secure proxy re-encryption," in *Public Key Cryptography* (Lecture Notes in Computer Science), vol. 4939. Berlin, Germany: Springer-Verlag, 2008, pp. 360–379.

[26] S. Luo, J. Hu, and Z. Chen, "Ciphertext policy attribute-based proxy re-encryption," in *Information and Communications Security* (Lecture Notes in Computer Science), vol. 6476, M. Soriano, S. Qing, and J. López, Eds. Berlin, Germany: Springer-Verlag, 2010, pp. 401–415.

[27] M. Mambo and E. Okamoto, "Proxy cryptosystems: Delegation of the power to decrypt ciphertexts," *IEICE Trans. Fund. Electron., Commun. Comput. Sci.*, vol. E80-A, no. 1, pp. 54–63, 1997.

[28] T. Mizuno and H. Doi, "Hybrid proxy re-encryption scheme for attribute-based encryption," in *Information Security and Cryptology* (Lecture Notes in Computer Science), vol. 6151. Berlin, Germany: Springer-Verlag, 2011, pp. 288–302.

[29] R. Ostrovsky, A. Sahai, and B. Waters, "Attribute-based encryption with non-monotonic access structures," in *Proc. 14th ACM CCS*, 2007, pp. 195–203.

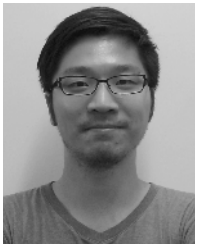
[30] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 3494. Berlin, Germany: Springer-Verlag, 2005, pp. 457–473.

[31] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Public Key Cryptography* (Lecture Notes in Computer Science), vol. 6571. Berlin, Germany: Springer-Verlag, 2011, pp. 53–70.

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

[33] J. Weng, R. H. Deng, X. Ding, C.-K. Chu, and J. Lai, "Conditional proxy re-encryption secure against chosen-ciphertext attack," in *Proc. 4th Int. Symp. ASIACCS*, 2009, pp. 322–332.

[34] S. Yamada, N. Attrapadung, G. Hanaoka, and N. Kunihiro, "Generic constructions for chosen-ciphertext secure attribute based encryption," in *Public Key Cryptography* (Lecture Notes in Computer Science), vol. 6571. Berlin, Germany: Springer-Verlag, 2011, pp. 71–89.



Kaitai Liang received the B.Eng. degree in software engineering and the M.S. degree in computer applied technology from South China Agricultural University, Guangzhou, China, in 2008 and 2011, respectively. He is currently pursuing the Ph.D. degree with the Department of Computer Science, City University of Hong Kong, Hong Kong. His research interests are applied cryptography (encryption/signature and RFID) and information security, such as network security, wireless security, database security, and security in cloud computing.

Man Ho Au (M'12) is an Assistant Professor with the Department of Computing, Hong Kong Polytechnic University, Hong Kong. His research interests include information security and privacy. He has authored over 60 refereed journal and conference papers, including two papers in the ACM CCS Conference that received Runners-Up for the 2009 Pet Award for Outstanding Research in Privacy Enhancing Technologies.

Joseph K. Liu received the Ph.D. degree in information engineering from the Chinese University of Hong Kong, Hong Kong, in 2004, specializing in cryptographic protocols for securing wireless networks, privacy, authentication, and provable security. He is currently a Research Scientist with the Department of Infocomm Security, Institute for Infocomm Research, Singapore. His current technical focus is, in particular, lightweight cryptography, wireless security, security in smart grid system, and cloud computing environment.

Willy Susilo (SM'01) received the Ph.D. degree in computer science from the University of Wollongong, Wollongong, NSW, Australia. He is currently a Professor with the School of Computer Science and Software Engineering and the Director of Centre for Computer and Information Security Research, University of Wollongong. His main research interests include cryptography and information security. He has authored numerous publications in the areas of digital signature schemes and encryption schemes.

Duncan S. Wong (M'13) is an Associate Professor with the Department of Computer Science, City University of Hong Kong, Hong Kong. His primary research interest is cryptography, in particular, cryptographic protocols, encryption and signature schemes, and anonymous systems. He is also interested in information security, such as network security, wireless security, database security, and security in cloud computing.

Guomin Yang (M'13) received the Ph.D. degree from the Department of Computer Science, City University of Hong Kong, Hong Kong, in 2009. From 2009 to 2012, he was a Research Scientist with the Temasek Laboratories, National University of Singapore, Singapore. He is currently a Lecturer with the School of Computer Science and Software Engineering, University of Wollongong, Wollongong, NSW, Australia.

Tran Viet Xuan Phuong received the bachelor's degree from the University of Science, Ho Chi Minh City, Vietnam, in 2010, and the M.S. degree from the Japan Advanced Institute Science and Technology, Nomi, Japan, in 2012. She is currently pursuing the Ph.D. degree in information security from the University of Wollongong, Wollongong, NSW, Australia.

Qi Xie received the Ph.D. degree in applied mathematics from Zhejiang University, Hangzhou, China, in 2005. He has been an Academic Visitor with the School of Computer Science, University of Birmingham, Birmingham, U.K., and the City University of Hong Kong, Hong Kong. He is currently a Professor with the Institute of Service Engineering, Hangzhou Normal University, Hangzhou, China. His research interests are network security and applied cryptography.