An Efficient Public Key Trace and Revoke Scheme Secure against Adaptive Chosen Ciphertext Attack†

Chong Hee KIM, Yong Ho HWANG, and Pil Joong LEE

IS Lab, Dept. of Electronic and Electrical Eng., POSTECH, KOREA
{chhkim, yhhwang}@oberon.postech.ac.kr, pjl@postech.ac.kr
http://islab.postech.ac.kr

Abstract. We propose a new public key trace and revoke scheme secure against adaptive chosen ciphertext attack. Our scheme is more efficient than the DF scheme suggested by Y. Dodis and N. Fazio[9]. Our scheme reduces the length of enabling block of the DF scheme by (about) half. Additionally, the computational overhead of the user is lower than that of the DF scheme; instead, the computational overhead of the server is increased. The total computational overhead of the user and the server is the same as that of the DF scheme, and therefore, our scheme is more practical, since the computing power of the user is weaker than that of the server in many applications. In addition, our scheme is secure against adaptive chosen ciphertext attack under only the decision Diffie-Hellman (DDH) assumption and the collision-resistant hash function H assumption, whereas the DF scheme also needs the *one-time* MAC (message authentication code) assumption.

1 Introduction

A broadcast encryption scheme enables a center to send encrypted data to a large group of users over an insecure channel, where only legitimate users can decrypt the data. The set of legitimate users is dynamically changing, so it should be possible to prevent some revoked users from decrypting the data. The broadcast encryption scheme has numerous applications, such as pay-TV systems, the distribution of copyrighted materials, internet multicasting of video, music, magazines, and so on.

A. Fiat and M. Naor first formalized the basic definitions and paradigms of the broadcast encryption scheme [11]. Afterwards, many variants have been investigated. One example is the scheme of tracing traitors [6]. In this setting, the center can trace the traitors after a pirate decoder is confiscated. There are two types of approaches to the traitor-tracing scheme. One is a scheme that uses a secret key and coding approach [4, 6, 12, 16–19] and the other uses a public key [3, 14]. In the secret key scheme, the keys in the pirate decoder can be identified

† This research was supported by University IT Research Center Project, the Brain Korea 21 Project, and Com2MaC-KOSEF.

by combinatorial methods. In the public key approach, the size of the enabling block is independent of the number of subscribers. In addition, the public key traitor tracing schemes enable the center to prepare a public key that allows any entity to broadcast data to users. There is another variant of broadcast encryption, the revoke system, which concentrates on the problem of excluding a certain subset of users from receiving the data in a dynamically changing set of users. There are many revoke systems that use the secret key setting. These schemes are also divided into two categories. One is for stateless receivers [15, 13, 2] and the other is for non-stateless receivers [21, 22].

Recently, a public key traitor-tracing scheme with the revocation capability was introduced by W. Tzeng and Z. Tzeng [20]. They also proposed a variant of their basic scheme to be secure against *adaptive chosen ciphertext attack* (CCA2). However, Dodis and Fazio noted that W. Tzeng and Z. Tzeng's scheme was *not* secure against CCA2 even if a single user is corrupted [9]. Dodis and Fazio also proposed their own scheme secure against CCA2 under the decision Diffie-Hellman (DDH) assimption, the collision-resistant hash function H assumption, and the *one-time* MAC assumption [9].

Our results We propose a new public key trace and revoke scheme secure against CCA2. Our scheme does not use the additional *one-time* MAC, so its security does not depend on the *one-time* MAC assumption. The length of the enabling block of our scheme is about half that of the DF scheme. Additionally, the computational overhead of the user is lower than that of the DF scheme instead the computational overhead of the server is increased. The total computational overhead of the user and the server is the same as that of the DF scheme. (We only consider the computation of exponentiation computed by the server and the user. If we did the analysis more precisely, our scheme is more efficient than the DF scheme because it does not require computational overhead for the MAC). Our scheme is more practical, since the computing power of the user is weaker than that of the server in many applications.

By slightly modifying standard tracing algorithms from previous schemes (e.g. [20]), our scheme can be a fully functional trace and revoke scheme. However, due to space limitations we will omit the tracing part and focus only on the revoke scheme, which is the original contribution of this paper.

2 Preliminaries

In this section, we review the Lagrange interpolation in the exponent, the decision Diffie-Hellman (DDH) assumption, and public key encryption schemes secure against CCA2.

The Lagrange Interpolation in the Exponent Let q be a prime and $f(x)=\sum_{t=0}^{z} a_t x_t$ a polynomial of degree z over Z_q. Let $x_0, \ldots, x_z$ be distinct elements in Z_q. Then using the Lagrange interpolation, we can express $f(x)$

as $\sum_{t=0}^{z}(f(x_t) \cdot \lambda_t(x))$, where $\lambda_t(x) = \prod_{0 \leq j \neq t \leq z} \frac{x_j - x}{x_j - x_t}$, $0 \leq t \leq z$. We define the Lagrange interpolation operator as: $LI(x_0, \ldots, x_z; f(x_0), \ldots, f(x_z))(x) = \sum_{t=0}^{z}(f(x_t) \cdot \lambda_t(x))$.

Next, we consider a cyclic group G of order q and a generator g of G. Let $v_t = g^{f(x_t)}$, $0 \leq t \leq z$, where $x_t \in Z_q$ and $v_t \in G$. Then we define the Lagrange interpolation operator in the exponent as: $EXP-LI(x_0, \ldots, x_z; v_0, \ldots, v_z)(x) = g^{LI(x_0,\ldots,x_z;f(x_0),\ldots,f(x_z))} = \prod_{t=0}^{z} g^{(f(x_t)\cdot\lambda_t(x))} = \prod_{t=0}^{z} v_t^{\lambda_t(x)}$. We also remark that $EXP - LI(x_0, \ldots, x_z; v_0^r, \ldots, v_z^r)(x) = [EXP - LI(x_0, \ldots, x_z; v_0, \ldots, v_z)(x)]^r$. In what follows, we will refer to a function of the form $g^{f(x)}$, where $f(x)$ is polynomial, as an EXP- polynomial.

The Decision Diffie-Hellman Assumption Let G be a group of large prime order q, and consider the following two distributions:

- the distribution **R** of random quadruples $(g_1, g_2, u_1, u_2) \in G^4$,
- the distribution **D** of quadruples $(g_1, g_2, u_1, u_2) \in G^4$, where g_1, g_2 are random, and $u_1 = g_1^r$ and $u_2 = g_2^r$ for random $r \in Z_q$.

The decision Diffie-Hellman (DDH) assumption is that it is computationally hard to distinguish these two distributions. That is, we consider an algorithm that should output 0 or 1, given a quadruple coming from one of the two distributions. Then the difference between the probability that it outputs a 1 given an input from **R**, and the probability that it outputs a 1 given an input from **D** is negligible.

Our scheme is based on the modified Cramer-Shoup (M-CS) scheme [5] and the DF scheme is based on the Cramer-Shoup (CS) scheme [7]. The M-CS scheme is a variant of the CS scheme. We briefly review these schemes.

The Cramer-Shoup scheme Given a security parameter 1^λ, the secret key is (x_1, x_2, y_1, y_2, z) and the public key is $(p, q, g_1, g_2, c, d, h, H)$, where p is a λ-bit prime, g_1, g_2 are generators of G(a subgroup of Z_p^* of a large prime order q), function H is a hash function chosen from a collision-resistant hash function family, $x_1, x_2, y_1, y_2, z \overset{R}{\leftarrow} Z_q$, $c = g_1^{x_1} g_2^{x_2}$, $d = g_1^{y_1} g_2^{y_2}$, and $h = g_1^z$.

Given a message $m \in G$, the encryption algorithm runs as follows. First, it chooses $r \overset{R}{\leftarrow} Z_q$ and computes $u_1 = g_1^r, u_2 = g_2^r, e = h^r m, \alpha = H(u_1, u_2, e), v = c^r d^{r\alpha}$. The ciphertext is (u_1, u_2, e, v). Given a ciphertext, the decryption algorithm runs as follows. First, it computes $v' = u_1^{x_1+y_1\alpha} \cdot u_2^{x_2+y_2\alpha}$. Next, it performs a validity check. If $v \neq v'$, then it outputs an error message, denoted '$\perp$'; otherwise, it outputs $m = \frac{e}{u_1^z}$. The security of this scheme against CCA2 is proven, based on DDH assumption, in [7].

The Modified Cramer-Shoup scheme R. Canetti and S. Goldwasser slightly modified the above CS scheme as follows, without losing in security [5]. If the decryption algorithm finds $v \neq v'$, instead of outputting '$\perp$' it outputs a random value in G. In a sense, the modified scheme is even "more secure" since the adversary is not notified by the decryption algorithm whether a ciphertext is valid.

Now that the decryption algorithm does not explicitly check validity, given (u_1, u_2, e, v) it outputs $(\frac{e}{u_1^z}) \cdot (\frac{v'}{v})^s$ instead, where $v'$is computed as in the CS scheme and $s \xleftarrow{R} Z_q$. Note that the decryption algorithm is now randomized. To see the validity of this modification, notice that if $v = v'$ then $(\frac{v}{v'})^s$=1 for all s, and the correct value is outputted. If $v \neq v'$, then the decryption algorithm outputs a uniformly distributed value in G, independent of m. The security of M-CS scheme against CCA2 is proven, based on the DDH assumption, in [5].

3 Public key broadcast encryption scheme

We use the definition in [9]. In a *public key broadcast encryption scheme* **BE**, a session key s is encrypted and broadcasted with the symmetric encryption of the "actual" message. Generally, the encryption of s is called the *enabling block*.

3.1 Public key broadcast encryption scheme

A *public key broadcast encryption scheme* **BE** consists of a 4-tuple of poly-time algorithms (**KeyGen**, **Reg**, **Enc**, **Dec**):

- KeyGen, *the key generation algorithm, is a probabilistic algorithm used by the center to set up all the parameters of the scheme.* **KeyGen** takes as input a security parameter 1^λ and a revocation threshold z (i.e. the maximum number of users that can be revoked) and generates the public key PK and the master secret key SK_{BE}.
- Reg, *the registration algorithm, is a probabilistic algorithm used by the center to compute the secret initialization data needed to construct a new decoder each time a new user subscribes to the system.* **Reg** receives as input the master secret key SK_{BE} and a (new) index i associated with the user; it returns the user's secret key SK_i.
- Enc, *the encryption algorithm, is a probabilistic algorithm used to encapsulate a given session key s within an enabling block T.* **Enc** takes as input the public key PK, the session key s, and a set R of revoked users (with $|R| \leq z$) and returns the enabling block T.
- Dec, *the decryption algorithm, is a deterministic algorithm that takes as input the secret key SK_i of user i and the enabling block T and returns the session key s that was encapsulated within T if i was a legitimate user when T was constructed, or the special symbol "$\perp$".*

3.2 Security against adaptive chosen ciphertext attack

An adversary $\mathcal{A}$ in an *adaptive chosen ciphertext attack* (CCA2) is a probabilistic, poly-time *oracle query* machine. The attack game is defined in terms of an interactive computation between the adversary and its environment. We describe the attack game used to define the security against CCA2; that is, we define the environment in which $\mathcal{A}$ runs. We assume that the input to $\mathcal{A}$ is 1^λ for some λ.

Stage 1: The adversary queries a key generation oracle. The key generation oracle computes $(PK, SK_{BE})\leftarrow$ **BE.KeyGen**$(1^\lambda,\ z)$ and responds with PK.

Stage 2: The adversary enters the *user corruption stage*, where she is given oracle access to the *User Corruption Oracle* $Cor_{SK_{BE}}(\cdot)$. This oracle receives as input the index i of the user to be corrupted, computes $SK_i \leftarrow$ **BE.Reg**(SK_{BE}, i) and returns the user's secret key SK_i. This oracle can be called adaptively for at most z times. Let us say that at the end of this stage the set R of at most z users is corrupted.

Stage 3: The adversary submits two session keys s_0, s_1 to an encryption oracle. On input s_0, s_1, the encryption oracle computes: $\sigma \overset{R}{\leftarrow} \{0,1\}$; $T^* \leftarrow$ **BE.Enc**(PK, s_σ, R) and responds with the "target" enabling block T^*.

Stage 4: The adversary continues to make calls to the decryption oracle, subject only to the restriction that a submitted enabling block T is not identical to T^*.

Stage 5: The adversary outputs $\sigma^* \in \{0,1\}$.

We define the advantage of $\mathcal{A}$ as $Adv^{CCA2}_{BE,\mathcal{A}}(\lambda)= |Pr(\sigma^* = \sigma) - \frac{1}{2}|$

We consider a variant of the CCA2, *generalized chosen ciphertext attack* (gCCA2) [1, 9]. The attack game of gCCA2 is the same as that of CCA2 except **Stage 4**. In the attack game of gCCA2, the adversary cannot ask about enabling blocks *closely related* to the "target" enabling block. That is, in **Stage 4**, the decryption oracle first checks whether *equivalence relation* $R_i(T, T^*)$ holds. If so, it outputs "$\perp$".

Definition 1 (z-resilience of a public key broadcast encryption scheme) *We say that a public key broadcast encryption scheme* ***BE*** *is z-resilient against CCA2 attack if for all probabilistic, poly-time oracle query machines* $\mathcal{A}$*, the function* $Adv^{CCA2}_{BE,\mathcal{A}}(\lambda)$ *grows negligibly in* λ .

4 The DF schemes

Y. Dodis and N. Fazio proposed three broadcast encryption schemes (we call them DF-CPA, DF-gCCA, DF-CCA2) that achieve z-resilience in an adaptive setting for the case of CPA (chosen plaintext attack), gCCA2, and CCA2, respectively. Subsequent schemes build on the previous one in an incremental manner. Therefore, the DF-CPA scheme is more efficient than the DF-gCCA2 scheme and DF-gCCA2 scheme is more efficient than the DF-CCA2 scheme in the length of the enabling block and the computational overhead. In the next section, we define DF-gCCA2 and DF-CCA2. For a more detailed description, see [9].

4.1 DF-gCCA2

Key generation algorithm: KeyGen selects two random generators $g_1, g_2 \in G$, where G is a group of order q, in which q is a large prime such that $2q = p - 1$, and p is a large prime. **KeyGen** selects six z-degree polynomials $X_1(\xi)$, $X_2(\xi)$, $Y_1(\xi)$, $Y_2(\xi)$, $Z_1(\xi)$, $Z_2(\xi)$ over Z_q, and computes $c_t = g_1^{X_1(t)} \cdot g_2^{X_2(t)}$, $d_t = g_1^{Y_1(t)} \cdot g_2^{Y_2(t)}$, $h_t = g_1^{Z_1(t)} \cdot g_2^{Z_2(t)}$, for $0 \leq t \leq z$. Finally, **KeyGen** chooses a hash function H from a family of $\mathcal{F}$ of collision resistant hash functions, and outputs (PK, SK_{BE}), where $PK = (p, q, g_1, g_2, c_0, \cdots, c_z, d_0, \cdots, d_z, h_0, \cdots, h_z, H)$ and $SK_{BE} = (X_1, X_2, Y_1, Y_2, Z_1, Z_2)$.

Registration algorithm: Each time a new user $i > z$ decides to subscribe to the system, the center provides him with a decoder box containing the secret key $SK_i = (i, X_1(i), X_2(i), Y_1(i), Y_2(i), Z_1(i), Z_2(i))$.

Encryption algorithm: Using the ides of the CS scheme [7, 8], in order to obtain a non-malleable ciphertext, they "tag" each encrypted message so that it can be verified before proceeding with the actual decryption. In the broadcast encryption scenario, where each user has a different decryption key, the tag cannot be a single point - they need to distribute an entire EXP-polynomial $V(x)$. This is accomplished by appending $z+1$ tags, $v_0, \ldots, v_z$, to the ciphertext.

The encryption algorithm receives as input the public key PK, the session key s, and a set $R = \{j_1, \ldots, j_z\}$ of revoked users. It proceeds as described in Fig. 1, and finally it outputs T.

Encryption algorithm $\mathbf{Enc}(PK, s, R)$	Decryption algorithm $\mathbf{Dec}(i, T)$
E_1. $r_1 \leftarrow_r Z_q$	D_1. $\alpha \leftarrow H(S, u_1, u_2, (j_1, H_{j_1}),.., (j_z, H_{j_z}))$
E_2. $u_1 \leftarrow g_1^{r_1}$	D_2. $\bar{v_i} \leftarrow u_1^{X_1(i)+Y_1(i)\alpha} \cdot u_2^{X_2(i)+Y_2(i)\alpha}$
E_3. $u_2 \leftarrow g_2^{r_1}$	D_3. $v_i \leftarrow EXP\text{-}LI(0,.., z; v_0,.., v_z)(i)$
E_4. $H_t \leftarrow h_t^{r_1}, (t = 0,.., z)$	D_4. if $v_i = \bar{v_i}$
E_5. $H_{j_t} \leftarrow EXP\text{-}LI(0,.., z; H_0,.., H_z)(j_t)$ $(t = 1,.., z)$	D_5. then $H_t \leftarrow u_1^{Z_1(i)} \cdot u_2^{Z_2(i)}$ D_6. $s \leftarrow \frac{S}{EXP-LI(j_1,..,j_z,i;H_{j_1},..,H_{j_z},H_i)(0)}$
E_6. $S \leftarrow s \cdot H_0$	D_7. return s
E_7. $\alpha \leftarrow H(S, u_1, u_2, (j_1, H_{j_1}),.., (j_z, H_{j_z}))$	D_8. else return $\perp$
E_8. $v_t \leftarrow c_t^{r_1} d_t^{r_1 \alpha}, (t = 0,.., z)$	
E_9. $T \leftarrow < S, u_1, u_2, (j_1, H_{j_1}),.., (j_z, H_{j_z}), v_0,.., v_z >$	

Figure 1. DF-gCCA2

Decryption algorithm: To recover the session key, a legitimate user i can proceed as in Fig. 1. He computes the tag $\bar{v}_i$ using his private key and then verifies the validity of the ciphertext by checking the interpolation of the $z+1$ values in point i against its $\bar{v}_i$ (Step D_2, D_3, and D_4). If i is a revoked user, the algorithm fails in Step D_6, since the interpolation points $j_1, \cdots, j_z, i$ are not pairwise distinct.

Security: The adversary can make the ciphertext malleable because of the use of an EXP-polynomial $V(x)$. Since each user i can verify the value of $V(x)$ in only one point, the adversary can modify $v_0, \cdots, v_z$ and construct a different EXP-polynomial $V'(x)$ such that $V'(x = x_i)$=$V(x_i)$, thus fooling user i to accept as valid a corrupted ciphertext. To prevent this, a family of *equivalence relations*$\{R_i\}$ is introduced. Two ciphertext T and T' are *equivalent* for user i if they have the same "data" components, and the tag "relevant to user i" is correctly verified, i.e. $v_i = v'_i$ (even though other "irrelevant" tags could be different)[9]. By using this equivalent relation, DF-gCCA2 is secure against gCCA2. In **Stage 4** of the attack game, the adversary cannot ask T which is *equivalently related* to the "target" T^*.

4.2 DF-CCA2

In Section 4.1, we saw that the DF-gCCA2 scheme does not provide a complete solution to the CCA2 problem, but only suffices for gCCA2 security. Indeed, given a challenge T^* with tag sequence $v_0, \cdots, v_z$, it is trivial to make a different sequence $v'_0, \cdots, v'_z$ such that $v_i = v'_i$, resulting in a "different" enabling block $T \neq T^*$: however, $\mathbf{Dec}(i, T^*)$=$\mathbf{Dec}(i, T)$, allowing the adversary to "break" CCA2 security.

To achieve CCA2 security Dodis and Fazio used a trick to make the tag sequence $v_0, \cdots, v_z$ non-malleable. To this end, they used a *message authentication code* (MAC). The key generation algorithm and the registration algorithm are the same as those of DF-gCCA2. The encryption and decryption algorithm are shown in Fig. 2. The encryption algorithm operates similarly to the gCCA2 encryption algorithm, but the main difference is that now a MAC key k is used to MAC the tag sequence $v_0, \cdots, v_z$, and is encapsulated within T along with the session key s.

If the DDH problem is hard in G, H is chosen from a collision-resistant hash function family $\mathcal{F}$, and MAC is a *one-time* message authentication code, then the DF-CCA2 scheme is z-resilient against CCA2[9].

5 Proposed scheme

In this section, we propose a new public key trace and revoke scheme secure against CCA2. Our scheme does not use the additional *one-time* MAC, so its security does not depend on the *one-time* MAC. The length of the enabling block of

Encryption algorithm $\mathbf{Enc}(PK, s, R)$	Decryption algorithm $\mathbf{Dec}(i, T)$
$E_1.r_1 \leftarrow_r Z_q$	$D_1.\ \alpha \leftarrow H(S, u_1, u_2, (j_1, H_{j_1}),.., (j_z, H_{j_z}))$
$E_2.u_1 \leftarrow g_1^{r_1}$	$D_2.\ \bar{v_i} \leftarrow u_1^{X_1(i)+Y_1(i)\alpha} \cdot u_2^{X_2(i)+Y_2(i)\alpha}$
$E_3.u_2 \leftarrow g_2^{r_1}$	$D_3.\ v_i \leftarrow EXP\text{-}LI(0,.., z; v_0,.., v_z)(i)$
$E_4.H_t \leftarrow h_t^{r_1}, (t = 0,.., z)$	$D_4.$ if $v_i = \bar{v_i}$
$E_5.H_{j_t} \leftarrow EXP\text{-}LI(0,.., z; H_0,.., H_z)(j_t)$ $(t = 1,.., z)$	$D_5.$ then $H_t \leftarrow u_1^{Z_1(i)} \cdot u_2^{Z_2(i)}$
	$D_6.\ s\vert\vert k \leftarrow \frac{S}{EXP-LI(j_1,..,j_z,i;H_{j_1},..,H_{j_z},H_i)(0)}$
$E_6.k \leftarrow_r K$	$D_7.$ extract s and k from $(s\vert\vert k)$
$E_7.S \leftarrow (s\vert\vert k) \cdot H_0$	$D_8.$ if $\tau \neq MAC_k(v_0,..v_z)$
$E_8.\alpha \leftarrow H(S, u_1, u_2, (j_1, H_{j_1}),.., (j_z, H_{j_z}))$	$D_9.$ then return $\perp$
$E_9.v_t \leftarrow c_t^{r_1} d_t^{r_1\alpha}, (t = 0,.., z)$	$D_{10}.$ else return s
$E_{10}.\tau \leftarrow MAC_k(v_0,..v_z)$	$D_{11}.$else return $\perp$
$E_{11}.T \leftarrow < S, u_1, u_2, (j_1, H_{j_1}),.., (j_z, H_{j_z}),$ $v_0,.., v_z, \tau >$	

Figure 2. DF-CCA2

our scheme is about half that of the DF-CCA2 (DF-gCCA2) scheme. Additionally, the computational overhead of the user is lower than that of the DF-CCA2 (DF-gCCA2) scheme. Instead, the computational overhead of the server is increased, but the total computational overhead of the user and the server is the same as that of the DF-CCA2 (DF-gCCA2) scheme. We only consider the computation of exponentiation computed by the server and user. Our scheme is more efficient precisely because it does not require the computational overhead for the MAC but the DF-CCA2 scheme does. Our scheme is more practical, since the computing power of the user is weaker than the server in many applications.

Main Idea: In the DF-CCA2 scheme, given the enabling block $T \leftarrow< S, u_1, u_2, (j_1, H_{j_1}), \cdots, (j_z, H_{j_z}), v_0, \ldots, v_z, \tau) >$, to check the validity of T user i constructs $V(x)$ using $v_0, \ldots, v_z$ and checks whether $V(x = i) = v_i$. He also checks the validity of $v_0, \cdots, v_z$ by use of the MAC value τ. Our idea starts from the problem of the DF-gCCA2 scheme. In the DF-gCCA2 scheme, the decryption oracle cannot distinguish $V'(x)$ such that $V'(i) = V(i)$, but $v'_0, \cdots, v'_z \neq v_0, \cdots, v_z$. The DF-CCA2 scheme solves this problem by the use of the MAC.

We make the enabling block $T \leftarrow< S, u_1, u_2, c^r d^{r\alpha}, v_1, \ldots, v_z >$. Given T, user i computes $V(x)$ using $v_1, \cdots, v_z$ and his secret share v_i. Then he checks the validity of T using $c^r d^{r\alpha}$ and $V(x = 0)$. The adversary cannot compute $V(x = 0)$, since he knows only z shares of the degree-z polynomial $V(x)$. Therefore, the adversary cannot cheat the decryption oracle.

Key generation algorithm: KeyGen selects two random generators $g_1, g_2 \in G$, where G is a group of order q in which, q is a large prime such that $2q = p-1$, and p is a large prime. It selects $x_1, x_2, y_1, y_2 \in Z_q$ and z-degree polynomials $X_1(\xi)$, $X_2(\xi)$, $Y_1(\xi)$, $Y_2(\xi)$ over Z_q such that $X_1(0) = x_1$, $X_2(0) = x_2$, $Y_1(0) = y_1$,

$Y_2(0) = y_2$. It also selects z-degree polynomials $Z_1(\xi)$, $Z_2(\xi)$ over ξ and computes $c = g_1^{x_1} g_2^{x_2}$, $d = g_1^{y_1} g_2^{y_2}$. Then, it computes $h_t = g_1^{Z_1(t)} g_2^{Z_2(t)}$, $0 \le t \le z$ and $x_{1,t} = g_1^{X_1(t)}$, $x_{2,t} = g_2^{X_2(t)}$, $y_{1,t} = g_1^{Y_1(t)}$, $y_{2,t} = g_2^{Y_2(t)}$, $0 \le t \le z$.

Finally, **KeyGen** chooses a hash function H from a family $\mathcal{F}$ of collision resistant hash functions, and outputs (PK, SK_{BE}), where $PK = (p,\ q,\ g_1, g_2,\ c,\ d,\ x_{1,0},\ \ldots,\ x_{1,z},\ x_{2,0},\ \ldots,\ x_{2,z},\ y_{1,0},\ldots,\ y_{1,z},\ y_{2,0},\ldots,\ y_{2,z},\ h_0,\ \cdots,\ h_z, H)$ and $SK_{BE} = (X_1, X_2, Y_1, Y_2, Z_1, Z_2)$.

Registration algorithm: Each time a new user $i > z$ decides to subscribe to the system, the center provides him with a decoder box containing the secret key $SK_i = (i, X_1(i), X_2(i), Y_1(i), Y_2(i), Z_1(i), Z_2(i))$.

Encryption algorithm: Our scheme is based on the idea of M-CS [5]. The encryption algorithm receives as input the public key PK, the session key s, and a set $R = \{j_1, \cdots, j_z\}$ of revoked users. It proceeds as described in Fig. 3, and finally it outputs T. **Enc** computes and distributes F_{j_t}, $1 \le t \le z$. We can think that $F_{j_t} = g_1^{Q(j_t)}$ where $Q(\xi)$ is z-degree polynomial in Z_q. Therefore, the adversary who only knows z shares of F_{j_t} cannot cheat the decryption oracle.

Encryption algorithm **Enc**(PK, s, R)	Decryption algorithm **Dec**(i, T)
$E_1.\ r_1 \leftarrow_r Z_q$	$D_1.\ \alpha \leftarrow H(S, u_1, u_2)$
$E_2.\ u_1 \leftarrow g_1^{r_1}$	$D_2.\ C_i \leftarrow u_1^{X_1(i)+Y_1(i)\alpha} \cdot u_2^{X_2(i)+Y_2(i)\alpha}$
$E_3.\ u_2 \leftarrow g_2^{r_1}$	$D_3.\ H_i \leftarrow u_1^{Z_1(i)} \cdot u_2^{Z_2(i)}$
$E_4.\ H_t \leftarrow h_t^{r_1}, (t = 0,.., z)$	$D_4.\ F_i \leftarrow H_i \frac{C}{C_i}$
$E_5.\ H_{j_t} \leftarrow EXP\text{-}LI(0,.., z; H_0,.., H_z)(j_t)$ $(t = 1,.., z)$	$D_5.\ s \leftarrow \frac{S}{EXP-LI(j_1,..,j_z,i;F_{j_1},..,F_{j_z},F_i)(0)}$
$E_6.\ S \leftarrow s \cdot H_0$	
$E_7.\ \alpha \leftarrow H(S, u_1, u_2)$	
$E_8.\ C_t \leftarrow (x_{1,t} x_{2,t})^{r_1} (y_{1,t} y_{2,t})^{r_1 \alpha}$ $(t = 0,.., z)$	
$E_9.\ C_{j_t} \leftarrow EXP\text{-}LI(0,.., z; C_0,.., C_z)(j_t)$, $(t = 1,.., z)$	
$E_{10}.C \leftarrow c^{r_1} d^{r_1 \alpha}$	
$E_{11}.F_{j_t} = H_{j_t} \frac{C}{C_{j_t}}, (t = 1,.., z)$	
$E_{12}.T \leftarrow < S, u_1, u_2, c^{r_1} d^{r_1 \alpha}$, $(j_1, F_{j_1}),.., (j_z, F_{j_z}) >$	

Figure 3. Our Proposed scheme

Decryption algorithm: To recover the session key, a legitimate user i can proceed as in Fig. 3. A legitimate user can compute s in Step D_5, but the revoked user fails, since the interpolation of $j_1, \cdots, j_z, i$ are not pairwise distinct.

We here verify that the output of the decryption algorithm is identical to the session key s if the user i is a legitimate user. We can rewrite F_i computed from Step D_4 as follows (let $g_2 = g_1^w$):

$$
\begin{aligned}
F_i &= H_i \cdot (\tfrac{C}{C_i}) \\
&= (u_1^{Z_1(i)} u_2^{Z_2(i)})(c^{r_1} d^{r_1\alpha})(u_1^{-X_1(i)-Y_1(i)\alpha} \cdot u_2^{-X_2(i)-Y_2(i)\alpha}) \\
&= g_1^{r_1 Z_1(i)+wr_1 Z_2(i)-r_1 X_1(i)-r_1 Y_1(i)\alpha-wr_1 X_2(i)-wr_1 Y_2(i)\alpha} c^{r_1} d^{r_1\alpha} \\
&= g_1^{r_1 Z_1(i)+wr_1 Z_2(i)-r_1 X_1(i)-r_1 Y_1(i)\alpha-wr_1 X_2(i)-wr_1 Y_2(i)\alpha+(r_1 x_1+wr_1 x_2+r_1 y_1\alpha+wr_1 y_2\alpha)} \\
&= g_1^{Q(i)}
\end{aligned}
$$

Consequently, $F_i{=}g_1^{Q(i)}$ where $Q(\xi)$ is z-degree polynomial in Z_q. If we compute F_0 using the *Lagrange interpolation in the exponent* as in Step D_5, we can obtain the following value:

$$
\begin{aligned}
F_0 =& EXP - LI(j_1, \cdots, j_z, i; F_{j_1}, \ldots, F_{j_z}, F_i)(0) \\
=& g_1^{(r_1 z_1+wr_1 z_2)-r_1 x_1-r_1 y_1\alpha-wr_1 x_2-wr_1 y_2\alpha+(r_1 x_1+wr_1 x_2+r_1 y_1\alpha+wr_1 y_2\alpha)} \\
=& H_0 \frac{c^{r_1} d^{r_1\alpha}}{C} \\
=& H_0
\end{aligned}
$$

Therefore, $\frac{S}{F_0} = \frac{(s \cdot H_0)}{H_0} = s$.

Security:

Theorem 1 *If the DDH problem is hard in G and H is chosen from a collision-resistant hash function family $\mathcal{F}$, then our scheme is z-resilient against the adaptive chosen ciphertext attack.*

Proof. Our overall strategy for the proof follows the structural approach in [8]. We shall define a sequence $\mathbf{G_0}, \mathbf{G_1}, \ldots, \mathbf{G_l}$ of modified attack games. Each of the games $\mathbf{G_0}, \mathbf{G_1}, \ldots, \mathbf{G_l}$ operates on the same underlying probability space. In particular, the public key cryptosystem, the coin tosses **Coins** of $\mathcal{A}$, and the hidden bit σ take on identical values across all games, while some of the rules that define how the environment responds to oracle queries may differ from game to game. For any $1 \le i \le l$, we let T_i be the event that $\sigma = \sigma^*$ in the game $\mathbf{G_i}$. Our strategy is to show that for $1 \le i \le l$, the quantity $|Pr[T_{i-1}] - Pr[T_i]|$ is negligible. In addition, it will be evident from the definition of game $\mathbf{G_l}$ that $Pr[T_l] = \frac{1}{2}$, which will imply that $|Pr[T_0] - \frac{1}{2}|$ is negligible.

Before continuing, we state the following simple but useful lemma in [8].

Lemma 1 *Let U_1, U_2, and F be the events defined on some probability space. Suppose that the event $U_1 \wedge \neg F$ occurs if and only if $U_2 \wedge \neg F$ occurs. Then $|Pr[U_1] - Pr[U_2]| \le Pr[F]$.*

Game $\mathbf{G_0}$: Let $\mathbf{G_0}$ be the original attack game, let $\sigma^* \in \{0,1\}$ denote the output of $\mathcal{A}$, and let T_0 be the event that $\sigma = \sigma^*$ in $\mathbf{G_0}$, so that $Adv^{CCA2}_{Ourscheme,\mathcal{A}}(\lambda) = |Pr[T_0] - \frac{1}{2}|$.

Game $\mathbf{G_1}$: $\mathbf{G_1}$ is identical to $\mathbf{G_0}$, except that in $\mathbf{G_1}$, steps E_4 and E_8 are replaced with the following:

$$E'_4.\ H_t \leftarrow u_1^{Z_1(t)} \cdot u_2^{Z_2(t)},\ t = 0, \ldots, z$$
$$E'_8.\ C_t \leftarrow u_1^{X_1(t)+Y_1(t)\alpha} \cdot u_2^{X_2(t)+Y_2(t)\alpha},\ t = 0, \ldots, z$$

The change we have made is purely conceptual, it is just to make explicit any functional dependency of the above quantities on u_1 and u_2. Cleary, it holds that $Pr[T_0] = Pr[T_1]$.

Game $\mathbf{G_2}$: We again modify the encryption oracle, replacing steps E_1 and E_3 by

$$E'_1.\ r_1 \leftarrow_r Z_q, r_2 \leftarrow_r Z_q \backslash \{r_1\}$$
$$E'_3.\ u_2 \leftarrow g_2^{r_2}$$

Notice that while in $\mathbf{G_1}$ the values u_1 and u_2 are obtained using the same value r_1, in $\mathbf{G_2}$ they are independent subject to $r_1 \neq r_2$. Therefore, any difference in behavior between $\mathbf{G_1}$ and $\mathbf{G_2}$ immediately yields a PPT algorithm $\mathcal{A}_1$ that is able to distinguish DH tuples from totally random tuples with a non negligible advantage. That is, $|Pr[T_2] - Pr[T_1]| \leq \epsilon_1$ for some negligible ϵ_1.

Game $\mathbf{G_3}$: In this game, we modify the decryption oracle in $\mathbf{G_2}$ to obtain $\mathbf{G_3}$ as follows:

$$\begin{array}{ll}
D_1. & \alpha \leftarrow H(S, u_1, u_2) \\
D'_2. & C_i \leftarrow u_1^{X_1(i)+Y_1(i)\alpha+(X_2(i)+Y_2(i)\alpha)w} \\
D_{2-1}. & \text{if } (u_2 = u_1^w) \\
D'_3. & \text{then } H_i \leftarrow u_1^{Z_1(i)+Z_2(i)w} \\
D'_4. & \qquad F_i \leftarrow H_i \frac{C}{C_i} \\
D'_5. & \qquad s \leftarrow \frac{S}{EXP-LI(j_1,\ldots,j_z,i,F_{j_1},\ldots,F_{j_z},F_i)(0)} \\
D'_6. & \text{else return } \perp
\end{array}$$

At this point, let R_3 be the event that the adversary $\mathcal{A}$ submits some decryption queries that are rejected in Step D_{2-1} in $\mathbf{G_3}$, but passed in $\mathbf{G_2}$. Note that if a query passes in D_{2-1} in $\mathbf{G_3}$, it would have also passed in $\mathbf{G_2}$. It is clear that $\mathbf{G_2}$ and $\mathbf{G_3}$ proceed identically until the event R_3 occurs. In particular, the event $T_2 \wedge \neg R_3$ and $T_3 \wedge \neg R_3$ are identical. Therefore, by Lemma 1, we have

$$|Pr[T_3] - Pr[T_2]| \leq Pr[R_3]$$

and so it suffices to bound $Pr[R_3]$. To do this we consider two more games, $\mathbf{G_4}$ and $\mathbf{G_5}$

Game $\mathbf{G_4}$: This game is identical to $\mathbf{G_3}$, except for a change in Step E_6 as follows:

$$E_6'.e \leftarrow_r Z_q, S \leftarrow g_1^e$$

It is clear by construction that $Pr[T_4] = \frac{1}{2}$, since in $\mathbf{G_4}$, the variable σ is never used at all, and so the adversary's output is independent of σ.

Let R_4 be the event that some decryption queries that would have passed in $\mathbf{G_2}$, fail to pass in Step D_{2-1} in $\mathbf{G_4}$. Then we have the following facts.

Lemma 2 $Pr[T_4] = Pr[T_3]$ *and* $Pr[R_4] = Pr[R_3]$.

The proof of Lemma 2 is shown in the Appendix

Game $\mathbf{G_5}$: This game is identical to $\mathbf{G_4}$, except for the following modification. In the decryption algorithm, we add the following *special rejection rule*, to prevent $\mathcal{A}$ from submitting an illegal enabling block to the decryption oracle once she has received her challenge T^*.

Special rejection rule: After the adversary $\mathcal{A}$ receives the challenge $T^* = (S^*, u_1^*, u_2^*, (c^r d^{r\alpha})^*, (j_1^*, F_{j_1}^*), \ldots, (j_z^*, F_{j_z}^*))$, the decryption oracle rejects any query $< i, T >$, with $T = (S, u_1, u_2, (c^r d^{r\alpha}), (j_1, F_{j_1}), \ldots, (j_z, F_{j_z}))$, such that $(S^*, u_1^*, u_2^*) \neq (S, u_1, u_2)$, but $\alpha = \alpha^*$, and it does so before executing the test in Step D_{2-1}.

To analyze this game, we define two events. Let C_5 be the event that the adversary $\mathcal{A}$ submits a decryption query that is rejected using the above special rejection rule, and R_5 the event that the adversary $\mathcal{A}$ submits some decryption query that would have passed in $\mathbf{G_2}$, but fails to pass in Step D_{2-1} in $\mathbf{G_5}$. Now it is clear that $\mathbf{G_4}$ and $\mathbf{G_5}$ proceed identically until event C_5 occurs. In particular, the event $R_4 \wedge \neg C_5$ and $R_5 \wedge \neg C_5$ are identical. Therefore, by Lemma 1, we have

$$|Pr[R_5] - Pr[R_4]| \leq Pr[C_5]$$

Now, if event C_5 occurs with non-negligible probability, we can construct a PPT algorithm $\mathcal{A}_2$ that breaks the collision resistance assumption with non-negligible probability. So, $|Pr[C_5]| \leq \epsilon_2$ for some negligible ϵ_2.

Finally, we show that event R_5 occurs with negligible probability.

Lemma 3 $Pr[R_5] \leq \frac{Q_{\mathcal{A}}(\lambda)}{q}$.

Where, $Q_{\mathcal{A}}(\lambda)$ is an upper bound on the number of decryption queries made by the adversary $\mathcal{A}$. The proof of Lemma 3 is shown in the Appendix.

Finally, combining the intermediate results, we conclude that the adversary $\mathcal{A}$'s advantage is negligible:

$$Adv^{CCA2}_{Ourscheme,\mathcal{A}}(\lambda) \leq \epsilon_1 + \epsilon_2 + \frac{Q_{\mathcal{A}}(\lambda)}{q}$$

□

References

1. J.H. An, Y. Dodis, T. Rabin, On the security of joint signature and encryption, *EUROCRYPT'02, LNCS V.2332*, pp.83-107, 2002.
2. T. Asano, A revocation scheme with minimal storage at receivers, *ASIACRYPT'02, LNCS V.2501*, pp.433-450, 2002.
3. D. Boneh, M. Franklin, An efficient public key traitor tracing scheme, *CRYTO'99, LNCS V.1666*, pp.338-353, 1999.
4. D. Boneh, J. Shaw, Collusion-secure fingerprinting for digital data, *IEEE Transaction on Information Theory 44(5)*, pp.1897-1905, 1998.
5. R. Canetti, S. Goldwasser, An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack, *EUROCRYPT'99, LNCS V.1592*, pp.90-106, 1999.
6. B. Chor, A. Fiat, M. Naor, Tracing traitor, *CRYPTO'94, LNCS V.839*, pp.257-270, 1994.
7. R. Cramer, V. Shoup, A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack, *CRYPTO'98, LNCS V.1462*, pp.13-25, 1998.
8. R. Cramer, V. Shoup, Design and analysis of practical public key encryption scheme secure against adaptive chosen ciphertext attack, *Manuscript*, 2001.
9. Y. Dodis, N. Fazio, Public key trace and revoke scheme secure against adaptive chosen ciphertext attack, *PKC'03*, pp.100-115, 2003.
10. Y. Dodis, N. Fazio, Public key trace and revoke scheme secure against adaptive chosen ciphertext attack, *Full version of* [9]
11. A. Fiat, M. Naor, Broadcast encryption, *CRYPTO'93, LNCS V.773* pp.480-491, 1993.
12. E. Gafni, J. Staddon, Y.L. Yin, Efficient methods for integrating traceability and broadcast encryption, *CRYPTO'99, LNCS V.1666*, pp.372-287, 1999.
13. D. Halevy, A. Shamir, The LSD broadcast encryption scheme, *CRYPTO'02, LNCS, V.2442*, pp.47-60, 2002.
14. K. Kurosawa, Y. Desmedt, Optimum traitor tracing and asymmetric schemes, *EUROCRYPT'98, LNCS V.1403*, pp.145-157, 1998.
15. D. Naor, M. Naor, J. Lostpiech, Revocation and tracing schemes for stateless receivers, *CRYPTO'01, LNCS V.2139*, pp.41-62, 2001.
16. M. Naor, B. Pinkas, Threshold traitor tracing, *CRYPTO'98, LNCS V.1462*, pp.502-517, 1998.
17. B. Pfitzmann, Trials of traced traitors, *Workshop on Information Hiding, LNCS V.1174*, pp.49-64, 1996.
18. B. Pfitzmann, M. Waidner, Asymmetric fingerprinting for large collusions, *ACM conference on Computer and Communication Security*, pp.151-160, 1997.
19. D.R. Stinson, R. Wei, Combinatorial properties and constructions of traceability schemes and frameproof codes, *SIAM Journal on Discrete Math 11(1)*, pp.41-53, 1998.
20. W.G. Tzeng, Z.J. Tzeng, A public-key tracing scheme with revocation using dynamic shares, *PKC'01, LNCS 1992*, pp.207-224, 2001.

21. D.M. Wallner, E.J. Harder, R.C. Agee, Key management for multicast: Issues and Architectures, *IETF Network Working Group, RFC 2627*, 1999.
22. C.K. Wong, M. Gouda, S. Lam, Secure group communications using key graphs, *ACM SIGCOMM'98*, pp.68-79, 1998.

Appendix

To prove Lemma 2 and Lemma 3, the following lemma is useful. The proof of Lemma 4 is shown in [8]. Our proofs follow the structural approach in [8, 10]. Therefore, they are similar to that of [10] except for some variables and notations.

Lemma 4 *Let k, n be integers with $1 \leq k \leq n$, and let K be a finite field. Consider a probability space with random variables $\boldsymbol{\alpha} \in K^{n\times 1}$, $\boldsymbol{\beta}=(\beta_1,\ldots,\beta_k)^T \in K^{k\times 1}$, $\boldsymbol{\gamma} \in K^{k\times 1}$, and $M \in K^{k\times n}$, such that $\boldsymbol{\alpha}$ is uniformly distributed over $K^{n\times 1}$, $\boldsymbol{\beta}=M\boldsymbol{\alpha}+\boldsymbol{\gamma}$, and for $1 \leq i \leq k$, the i th rows of M and $\boldsymbol{\gamma}$ are determined by $\beta_1,\ldots,\beta_{i-1}$.*

Then conditioning on any fixed values of $\beta_1,\ldots,\beta_{k-1}$ such that the resulting matrix M has rank k, the value of β_k is uniformly distributed over K in the resulting conditional probability space.

In what follows, we define:

Coins: the coin tosses of $\mathcal{A}$;
$X_t = X_1(t) + wX_2(t), Y_t = Y_1(t) + wY_2(t), Z_t = Z_1(t) + wZ_2(t), t = 0,\ldots,z;$
$w = log_{g_1} g_2$

Proof of Lemma 2

Lemma 2. $Pr[T_4] = Pr[T_3]$ and $Pr[R_4] = Pr[R_3]$.

Proof. Consider the quantity $X := (Coins, H,\ w,\ X_1(0),\ \ldots,\ X_1(z),\ X_2(0),\ \ldots,\ X_2(z),\ Y_1(0),\ \ldots,\ Y_1(z),\ Y_2(0),\ \ldots,\ Y_2(z),\ Z_1,\ \ldots,\ Z_z,\ \sigma,\ r_1^*,\ r_2^*)$ and the quantity Z_0. Note that X and Z_0 take on the same values in $\mathbf{G_3}$ and $\mathbf{G_4}$. Consider also the quantity $e^*=log_{g_1}S^*$. This quantity takes on different values in $\mathbf{G_3}$ and $\mathbf{G_4}$. For clarity, let us denote these values as $[e^*]_3$ and $[e^*]_4$, respectively.

It is clear by inspection that the events R_3 and T_3 are determined as functions of X, Z_0, and $[e^*]_3$. Also, the events R_4 and T_4 are determined as functions of X, Z_0 and $[e^*]_4$. Therefore to prove Lemma 2, it suffices to show that the distributions of (X, Z_0, $[e^*]_3$) and (X, Z_0, $[e^*]_4$) are identical. Observe that by the construction, conditioning on any fixed values of X and Z_0, the distribution of $[e^*]_4$ is uniform over Z_q. Therefore, it will suffice to show that conditioning on any fixed values of X and Z_0, the distribution of $[e^*]_3$ is uniform over Z_q .

We have the following equation:

$$\begin{pmatrix} Z_0 \\ [e^*]_3 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & w \\ r_1^* & wr_2^* \end{pmatrix}}_{M} \cdot \begin{pmatrix} Z_1(0) \\ Z_2(0) \end{pmatrix} + \begin{pmatrix} 0 \\ log_{g_1} s_\sigma \end{pmatrix}$$

where det(M)=$w(r_2^* - r_1^*) \neq 0$ since $r_2^* \neq r_1^*$.

Conditioning only on a fixed value of X, the matrix M is fixed, but the values $Z_1(0)$ and $Z_2(0)$ are still uniformly and independently distributed over Z_q. If we further condition on a fixed value of Z_0, the value of s_σ is fixed; hence, by Lemma 4, the distribution of $[e^*]_3$ is uniform over Z_q. □

Proof of Lemma 3

Lemma 3. $Pr[R_5] \leq \frac{Q_{\mathcal{A}}(\lambda)}{q}$.

Proof. For $1 \leq j \leq Q_{\mathcal{A}}(\lambda)$, we define the following events;

- $R_5^{(j)}$: the event that the jth ciphertext $< i, T >$, submitted to the decryption oracle in $\mathbf{G_5}$, fails to pass D_{2-1}, but would have passed in $\mathbf{G_2}$,

- $B_5^{(j)}$: the event that the jth ciphertext $< i, T >$, submitted to the decryption oracle $before$ $\mathcal{A}$ received her challenge,

- $\hat{B}_5^{(j)}$: the event that the jth ciphertext $< i, T >$, submitted to the decryption oracle $after$ $\mathcal{A}$ received her challenge.

If we show that $Pr[R_5^{(j)}|B_5^{(j)}] \leq \frac{1}{q}$ and $Pr[R_5^{(j)}|\hat{B}_5^{(j)}] \leq \frac{1}{q}$, then Lemma 3 is proved. □

Lemma 5 *For all* $1 \leq j \leq Q_{\mathcal{A}}(\lambda)$, *we have* $Pr[R_5^{(j)}|B_5^{(j)}] \leq \frac{1}{q}$.

Lemma 6 *For all* $1 \leq j \leq Q_{\mathcal{A}}(\lambda)$, *we have* $Pr[R_5^{(j)}|\hat{B}_5^{(j)}] \leq \frac{1}{q}$.

Proof of the Lemma 5. Fix $1 \leq j \leq Q_{\mathcal{A}}(\lambda)$ and consider the quantities:

$$X := (Coins, H, w, Z_0, \ldots, Z_z),\ X' := (X_0, \ldots, X_z, Y_0, \ldots, Y_z)$$

These two quantities completely determine the behavior of the adversary up to the moment that $\mathcal{A}$ performs the encryption query, and in particular, they completely determine the event $B_5^{(j)}$. Let us call X and X' *relevant* if the event $B_5^{(j)}$ occurs. Hence to prove Lemma 5, it suffice to prove that the probability of event $R_5^{(j)}$, conditioned on any *relevant* values of X and X', is less than $\frac{1}{q}$.

The test D_{2-1} fails if and only if $u_2 \neq u_1^w$. Thus if the test in D_{2-1} fails but would have passed in G_2, it must be the case that $u_2 \neq u_1^w$ and $c^{r_1} d^{r_1 \alpha}$=*EXP-LI*($j_1, \ldots, j_z, i$: $C_{j_1}, \ldots, C_{j_z}, C_i$)(0). Taking the logs (base g_1), the condition

$u_2 \neq u_1^w$ is equivalent to $r_2 \neq r_1$. If we let $\beta{=}log_{g_1} c^{r_1} d^{r_1 \alpha}$ and $\hat{\beta}{=}log_{g_1} EXP\text{-}LI(j_1, \ldots, j_z, i{:}\ C_{j_1}, \ldots, C_{j_z}, C_i)(0)$, then $\hat{\beta}{=}r_1 X_1(0){+}wr_2 X_2(0){+}\alpha r_1 Y_1(0){+}\alpha w\, r_2 Y_2(0)$. Notice that $\hat{\beta}$ can be expressed in terms of $(X_1(0), X_2(0), \ldots, X_1(z), X_2(z), Y_1(0), Y_2(0), \ldots, Y_1(z), Y_2(z))^T$. Therefore, we can make the following equation (for details, see [10]):

$$\begin{pmatrix} X_0 \\ \vdots \\ X_z \\ Y_0 \\ \vdots \\ Y_z \\ \hat{\beta} \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & w & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & & \ddots & & & & & & & \vdots \\ 0 & 0 & \cdots & 1 & w & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 1 & w & \cdots & 0 & 0 \\ \vdots & & & & & & & \ddots & & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 1 & w \\ \sigma_0 & \sigma_1 & \cdots & \sigma_{2z} & \sigma_{2z+1} & \sigma_{2z+2} & \sigma_{2z+3} & \cdots & \sigma_{4z+2} & \sigma_{4z+3} \end{pmatrix}}_{M} \cdot \begin{pmatrix} X_1(0) \\ X_2(0) \\ \vdots \\ X_1(z) \\ X_2(z) \\ Y_1(0) \\ Y_2(0) \\ \vdots \\ Y_1(z) \\ Y_2(z) \end{pmatrix}$$

Let us first fix X, which fixes the first $2z{+}2$ rows of the matrix M, but the values $(X_1(0), X_2(0), \ldots, Y_1(z), Y_2(z))$ are still uniformly distributed over Z_q. Next fix X' such that X and X' are *relevant* and $r_1 \neq r_2$. Then the last row of the matrix M is fixed. From this, it follows by Lemma 4 that $\hat{\beta}$ is uniformly distributed over Z_q, but β is fixed, we have $Pr[\beta = \hat{\beta}] = \frac{1}{q}$. □

Proof of the Lemma 6. Fix $1 \leq j \leq Q_{\mathcal{A}}(\lambda)$ and consider the quantities:

$$X := (Coins, H, w, Z_0, \ldots, Z_z, r_1^*, r_2^*, e^*),\ X' := (X_0, \ldots, X_z, Y_0, \ldots, Y_z, \beta^*).$$

where $\beta^* = log_{g_1}(c^{r_1} d^{r_1 \alpha})^*$ and $i > z$. The values of X and X' completely determine the adversary's entire behavior in Game G_5, in particular, they completely determine the event $\hat{B}_5^{(j)}$. Let us call X and X' *relevant* if the event $\hat{B}_5^{(j)}$ occurs. It will suffice to prove that conditioned on any fixed, *relevant* values of X and X', the probability that $R_5^{(j)}$ occurs is bounded by $\frac{1}{q}$. As in the proof of Lemma 5, we have the following equation (for the detail, see [10]):

$$\begin{pmatrix} X_0 \\ \vdots \\ X_z \\ Y_0 \\ \vdots \\ Y_z \\ \beta^* \\ \hat{\beta} \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & w & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & & \ddots & & & & & & & \vdots \\ 0 & 0 & \cdots & 1 & w & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 1 & w & \cdots & 0 & 0 \\ \vdots & & & & & & & \ddots & & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 1 & w \\ \sigma_0^* & \sigma_1^* & \cdots & \sigma_{2z}^* & \sigma_{2z+1}^* & \sigma_{2z+2}^* & \sigma_{2z+3}^* & \cdots & \sigma_{4z+2}^* & \sigma_{4z+3}^* \\ \sigma_0 & \sigma_1 & \cdots & \sigma_{2z} & \sigma_{2z+1} & \sigma_{2z+2} & \sigma_{2z+3} & \cdots & \sigma_{4z+2} & \sigma_{4z+3} \end{pmatrix}}_{M} \cdot \begin{pmatrix} X_1(0) \\ X_2(0) \\ \vdots \\ X_1(z) \\ X_2(z) \\ Y_1(0) \\ Y_2(0) \\ \vdots \\ Y_1(z) \\ Y_2(z) \end{pmatrix}$$

Again conditioning on a fixed value of X and X', we have that $\hat{\beta}$ is uniformly distributed over Z_q, but β^* is fixed. Therefore, we have $Pr[\beta^* = \hat{\beta}]{=}\frac{1}{q}$ $\square$