한국군사과학기술학회지 제15권 제1호, pp. 46~56, 2012년 2월

학술논문 정보기술 부문

개인키 업데이트가 가능한 공개키 기반 공모자 추적 암호 알고리즘

A Public Key Traitor Tracing Scheme with Key-update Method

이 문 식*
MoonShik Lee

Abstract

Traitor Tracing schemes are broadcast encryption systems where at least one of the traitors who were implicated in the construction of a pirate decoder can be traced. This traceability is required in various contents delivery system like satellite broadcast, DMB, pay-TV, DVD and so on. In this paper, we propose a public key traitor tracing scheme with key-update method. If the system manager can update a secret key which is stored in an authorized decode, it makes a pirate decoder useless by updating a secret key A pirate decoder which cannot update a secret key does not decrypt contents in next session or during tracing a traitor, this scheme has merits which will make a pirate decoder useless, therefore this scheme raises the security to a higher level.

Keywords : Cryptography, Broadcast Encryption System, Traitor Tracing, Traitor Trace and Revoke, Key-update

1. 서 론

네트워크 상에서는 디지털 콘텐츠 제공 센터는 정당한 사용자만이 콘텐츠를 볼 수 있도록 전송하고 정당하지 않은 사용자는 콘텐츠에 대한 접근을 방지하기 위하여 콘텐츠를 암호화하여 전송한다. 하지만 악의적인 사용자가 자신의 개인키를 이용하여 콘텐츠를 복호화 할 수 있는 다른 키를 만들거나, 정당한 사용자들이 서로 공모하여 복호화 할 수 있는 키가 저장된 불법 디코더를 만들 수 있다. 공모자 추적 암호 알고리즘(Traitor Tracing Scheme : 이하 TT 알고리즘)의 목적은 이러한 불법 디코더가 발견되었을 때, 이를 만들기 위해서 공모한 사용자(Traitor)를 찾아내는 것이다. TT 알고리즘이 "l-collusion resilient"라는 것은 l명 이하의 공모자가 불법 디코더를 만들었을 때 적어도 한 명의 공모자를 추적할 수 있다는 것을 의미한다.

지금까지 제안된 다수의 TT 알고리즘은 ElGamal 시스템을 응용한 알고리즘 이라 할 수 있고 ElGamal 응용 알고리즘은 선형공격(linear attack)에 취약한 단점을 가지고 있다. 즉, 개인키들의 선형 결합(linear combination)을 하게 되면 새로운 개인키를 만들 수 있는 단점을 가지고 있다.

또한 몇몇 제안된 알고리즘들은 센터가 적어도 한 명의 공모자를 추적하는 것이 아니라 공모자로 예상되

† 2011년 11월 1일 접수~2012년 1월 27일 게재승인
* 공군사관학교 수학과(Korea Air Force Academy)
책임저자 : 이문식(kafa0443@gmail.com)

는 집합에 반드시 공모자가 포함되어 있다는 것을 확인(black box confirmation[6])하거나 공모자 예상 집합을 리스트화(black box list[2])하는 것으로 추적 연산량이 비현실적으로 크다. 이러한 알고리즘들은 본질적인 TT 알고리즘과는 다소 거리가 있다고 할 수 있다. 이에 반해 공모자중 적어도 한명을 추적할 수 있는 공모자 추적 알고리즘이 [10]에서 제안되었다. 하지만 추적 알고리즘(black box tracing)을 사용하기 위한 전제 조건은 센터가 불법 디코더를 습득했다는 가정을 전제로 한다. 즉, 센터가 불법 디코더를 습득하지 못했다면 추적 알고리즘을 사용하는 것이 불가능하다.

이러한 이유로 [7] 논문에서는 traitor tracing 개념에 공모자들 중 적어도 한명을 추적하는 것은 물론 현재 사용되는 불법 디코더를 더 이상 사용하지 못하는 개념까지 포함을 시켰다. 하지만 [7]의 논문은 불법 디코더를 영구히 시스템에서 제외시키기 위해서는 전송량이 상대적으로 많아지는 단점이 있다. 본 논문에서는 불법 디코더를 습득한다면 공모자를 추적하는 것은 물론 습득되지 않은 불법 디코더를 쓸모없게 만드는 방법으로 전송량도 효율적인 개인키의 업데이트(update) 기능이 있는 TT 알고리즘을 제안하고자 한다. 즉, 복호화에 필요한 개인키를 새롭게 업데이트(update)한다면 불법 디코더는 저장되어있는 키를 업데이트(update)하지 못하므로 다음 세션에서 사용되는 콘텐츠를 복호화 할 수 없다. 이를 통해 추적 알고리즘으로 공모자를 추적하는 시간 동안 해당 키를 저장하고 있는 불법 디코더를 시스템에서 쓸모없게 만드는 장점이 있으며 일정 세션이 지난 후에 개인키를 업데이트(update)함으로써 시스템의 안전성을 보다 높일 수 있는 장점이 있다.

2. T. Matsushita, H. Imai의 알고리즘[10]

대표적인 ElGamal 시스템 응용 TT 알고리즘은 K. Kurosawa, Y. Desmedt가 1998년 Eurocrypt에서 발표한 "Optimum Traitor Tracing and Asymmetric Schemes"[8]라고 할 수 있다. 이를 시작으로 D. Boneh, M. Franklin이 1999년 Crypto에 발표한 "An Efficient Public Key Traitor Tracing Scheme"[4], A. Kiayias, M. Yung이 2002년 Eurocrypt에 발표한 "Traitor Tracing with Constant Transmission Rate"[2]과 T. Matsushita, H. Imai가 2004년 Asiacrypt에 발표한 "A Public Key Black Box Traitor Tracing Scheme with Sublinear Ciphertext Size Against Self-Defensive Pirates"[10] 등이 있다. 위의 알고리즘 중 본 논문에서 응용되는 [10] 알고리즘을 간단히 살펴보면 다음과 같다. 먼저 논문에서 사용되는 용어를 정의하면 다음과 같다.

- N : 총 사용자의 수
- l : 최대 공모자의 수
- p, q : 소수이며 $q|(p-1)$, $q > n+2l+1$을 만족한다.
- g : q-th root of unity over Z_p^*
- G_q : Z_p^*의 부분군이고 차수는 q
- U : 사용자의 전체 집합($U \subseteq Z_q$)

가. 키 생성(Key-Generation)

센터는 U를 $U_0,...,U_{m-1}$ $s.t$ $|U_i|=2l$, $m=N/2l$, $0 \le i \le m-1$로 분해한다.

임의의 $a_0,...,a_{2l-1},b_0,...,b_{m-1} \in Z_q$를 선택한다. 공개키 $e=(g,g^{a_0},...,g^{a_{2l-1}},g^{b_0},...,g^{b_{m-1}})$를 계산한다. 만약 사용자 $u \in U_i$라면 u의 개인키는 $(u,i,f_i(u))$이다. 여기서

$$f_i(u)=\sum_{j=0}^{2l-1} a_{i,j}u^j \mod q,$$

$$a_{i,j}=\begin{cases} a_j & (j \neq i \mod 2l) \\ b_i & (j=i \mod 2l) \end{cases}$$

나. 암호화(Encryption)

디지털 콘텐츠 제공 센터는 세션키 $s \in Z_q$와 메시지 M을 선택하여 다음 암호문을 계산하고 네트워크 상에서 사용자들에게 전송한다.

$$C=(C_0,...,C_{m-1},F_s(M)),\ 0 \le i \le m-1$$
$$C_i=(h_i,h_{i,0},...,h_{i,2l-1})$$

$$h_i=g^{r_i},\quad h_{i,j}=\begin{cases} g^{a_j} & (j \neq i \mod 2l) \\ sg^{r_i b_i} & (j=i \mod 2l) \end{cases},$$

$$r_i \in_R \{0,1\}$$

여기서 F는 블록 암호이다.

다. 복호화(Decryption)

만약 $u \in U_i$라면 u는 자신의 개인키 $(u, i, f_i(u))$를 가지고 암호문 C_i에서 세션키 s를 다음과 같이 올바르게 찾을 수 있고 M을 복호화 할 수 있다.

$$s = \left\{ (h_{i,0} \times h_{i,1}^{u} \times \cdots \times h_{i,2l-1}^{u^{2l-1}}) / h_i^{f_i(u)} \right\}^{1/u^{i \bmod 2l}}$$

$$= \left\{ s^{u^{i \bmod 2l}} \; g^{\sum_{j=0}^{2l-1} r_i a_{i,j} u^j} / \; g^{r_i f_i(u)} \right\}^{1/u^{i \bmod 2l}}$$

3. 개인키 업데이트가 가능한 공개키 기반 공모자 추적 암호 알고리즘

[10] 알고리즘의 기본적인 아이디어는 사용자 전체 n명을 $m(=n/2l)$개의 집합으로 나누고 각 집합마다 서로 다른 다항식을 대응시킨다는 것이다. 그리고 일방향 함수 트리(One-Way Function Tree[4] : 이하 OFT) 알고리즘을 사용하여 사용자의 개인키를 업데이트(update)할 수 있는 알고리즘을 제안한다. 본 논문에서는 디코더의 능력을 history-recording과 내부적으로 추적 알고리즘을 무력화할 수 있는 지능적인 디코더가 아닌 일반적인 능력을 가진 디코더로 제한하였다.

가. 예비지식(Preliminary)

1) Discrete Logarithm Problem

소수(prime) q($\approx$ 1024비트)에 대해서 순환군(cyclic group) $Z_q^* = \{1, ..., q-1\}$의 생성원을 $g \in Z_q^*$라고 하자. 만약 $b = g^a \pmod q$가 주어졌을 때, a를 구하는 것이 상당히 어렵다는 문제이다.

2) Decision Diffie-Hellman assumption(DDH)

순환군 G는 소수 q의 위수를 가지고, g를 G의 생성원이라고 하자. *DDH* 가정은 다음 두개의 ditstributions (g^a, g^b, g^{ab})와 (g^a, g^b, g^c), $a, b, c \in Z_q$을 다항식 시간 안에 구분할 수 있는 확률적인 알고리즘이 존재하지 않는다는 가정이다.

3) Lagrange interpolation method

차수가 z인 다항식 $f(x)$은 z+1개의 $(j_0, f(j_0)), ..., (j_z, f(j_z))$ 점이 주어진다면 만들 수 있다는 방법으로서 많은 암호학적인 도구로 사용된다. 즉, 먼저 q를 소수라고 하고 차수가 z인 다항식 $f(z)$를 Z_q위에서 정의된 다항식이라 하자. 그러면, 다항식 $f(x)$는 다음과 같이 구해진다.

$$f(x) = \sum_{t=0}^{z} (f(j_t) \cdot \lambda_t(x))$$

여기서 $\lambda_t(x) = \prod_{\substack{j \neq t \\ j=0}}^{z} \frac{j_i - x}{j_i - j_t}$, $t = 0, ..., z$이고, Lagrange coefficient이다. 또한 위의 방법을 순환군에서 지수연산으로 자연스럽게 확장할 수 있다. 즉, G를 소수 q의 위수를 갖는 순환군이라 하고, g를 생성원이라 하자.

z+1개의 점 $(j_0, g^{f(j_0)}), ..., (j_z, g^{f(j_z)})$이 주어진다면 Lagrange interpolation method을 사용하여 $g^{f(x)}$를 다음과 같이 계산할 수 있다.

$$g^{f(x)} = \prod_{t=0}^{z} g^{f(j_t) \cdot \lambda_t(x)}$$

여기서 $\lambda_t(x)$는 Lagrange coefficient이다.

나. 구조(Structure)

센터는 총 사용자 n명의 집합 $U = \{u_0, u_1, ..., u_{N-1}\}$를 $U_0, ..., U_{m-1}$ $s.t$ $|U_i| = l$, $m = N/l$, $0 \le i \le m-1$로 분해하고 $m = 2^d$인 이진트리를 만든다. 여기서 l, d는 자연수이다. 노드에 다음과 같이 label을 부여한다. 트리구조의 루트 노드는 $\varnothing \in \{0,1\}^0$의 label을 부여하고 label $s_{j,i} \in \{0,1\}^j$, $1 \le j \le d-1$에 대해서 왼쪽 child 노드는 label은 $(s_{j,i}) \| 0 \in \{0,1\}^{j+1}$, 오른쪽 child 노드의 label은 $(s_{j,i}) \| 1 \in \{0,1\}^{j+1}$을 부여한다. 예를 들면 m = 2^3인 이진 트리의 label은 Fig. 1과 같다.

이진 트리에 OFT 알고리즘을 적용하여 모든 내부 노드에 다음과 같이 키를 대응시킨다.

$$k_s = g(h(k_{(s\|0)}), h(k_{(s\|1)}), \; s \in \{0,1\}^j,$$

$$0 \le j \le d-1$$

여기서 g 함수는 혼합(mixing) 함수, h 함수는 일방향(one-way) 함수이다. 본 논문에서는 Fig. 2에서와

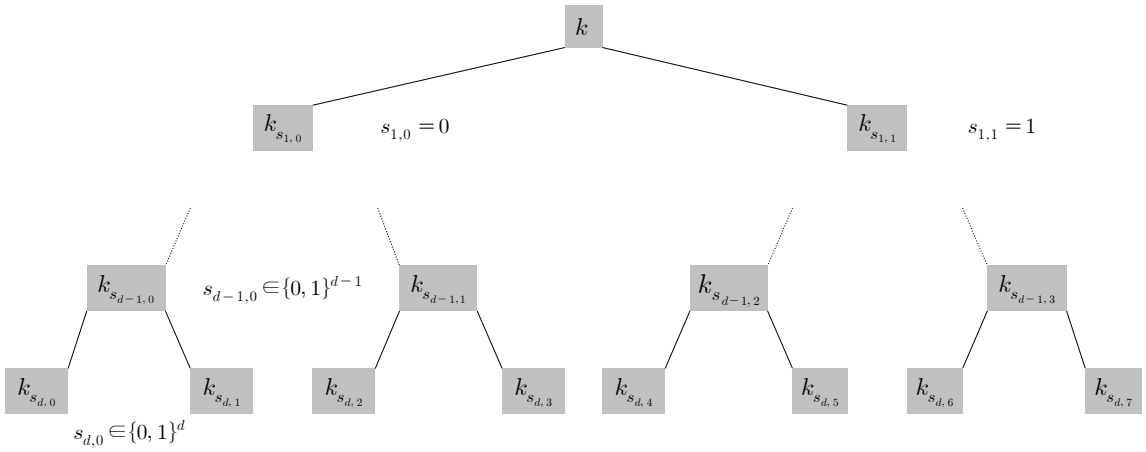

Fig. 1.

같이 내부 노드에 대응되는 키를 다음과 같이 계산한다.

$$g(h(k_{(s\|0)}), h(k_{(s\|1)}) = h(k_{(s\|0)}) \oplus h(k_{(s\|1)})$$
$$= H(k_{(s\|0)}) \oplus H(k_{(s\|1)})$$

여기서 g 함수를 xor 함수로, h 함수는 안전한 해쉬 함수 $H : \{0,1\}^d \rightarrow \{0,1\}^d$를 사용한다.

그리고 이진 트리의 leaf 노드에 각각 l명의 사용자 노드를 갖도록 트리를 구성한다.

예를 들면, $N = 3 * 2^2$ 이라면 Fig. 2와 같다.

다. 키 생성(Key Generation)

센터는 각각의 사용자들이 속해있는 이진트리의 leaf 노드 키에서 루트 키까지 상위(ancestor)노드 키들과 sibling 노드 키의 해쉬값을 계산하여 사용자에게 안전하게 전송한다.

사용자 개인키의 저장량을 줄이기 위해 leaf 노드 키와 상위(ancestor) 노드 키의 sibling 노드 키의 해쉬값만 저장한다. 즉, Fig. 2에서 사용자 $u \in U_0$는 $\{k_{00},, H(k_{01}), H(k_1), k\}$만을 저장한다.

루트 키의 안전성은 기본적으로 혼합(mixing) 함수 g로부터 얻어지며, 일방향 함수 h는 내부 노드 키를 숨기는 중요한 역할을 한다. 따라서 각각의 내부 노드는 노드 밑의 모든 사용자 집합을 위한 부분 그룹 키로서의 역할을 할 수 있다.

또한 센터는 임의의 $a_0, ..., a_{2l-1}, b_0, ..., b_{m-1} \in Z_q$를 선택하고 다음을 만족하는 다항식을 계산한다.

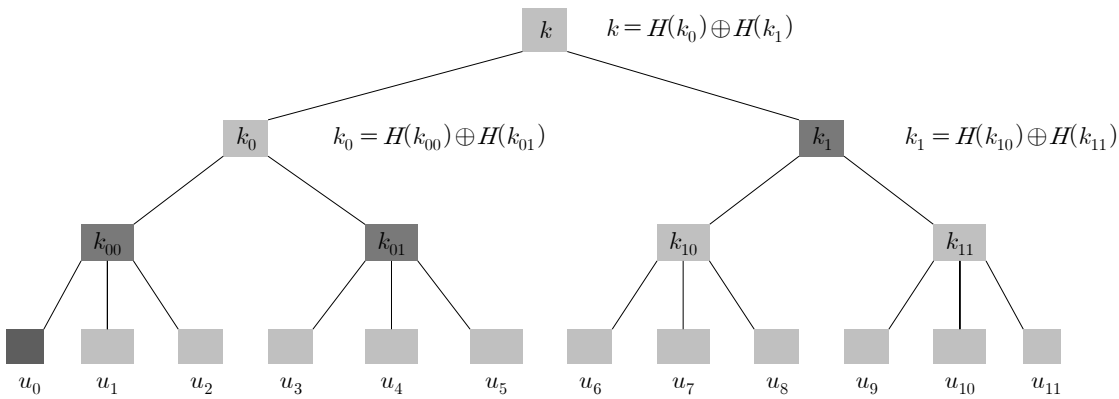

Fig. 2.

$$f_i(x)=\sum_{j=0}^{2l-1}a_{i,j}x^j \mod q,$$

$$a_{i,j}=\begin{cases}a_j & (j\neq i \mod 2l)\\ b_i & (j=i \mod 2l)\end{cases},\ 0\le i\le m-1$$

공개키 $e=(g, g^{a_0},..., g^{a_{2l-1}}, g^{b_0},..., g^{b_{m-1}})$를 계산하고 집합 $u\in U_i$, $0\le i\le m-1$의 사용자에게 개인키 $\{u, i, f_i(u)\}$를 안전하게 전송한다. 그러면 사용자 $u\in U_i$의 개인키는 $\{u,\ i,\ f_i(u),\ k_{s_{d,i}},\ H(\overline{k_{s_{d,i}}}),\ ...,\ H(\overline{k_{s_{1,i}}}),\ k\}$이다.

여기서 $\overline{s_{h,i}}$는 $s_{h,i}$의 sibling 노드의 label이다.

라. 암호화(Encryption)

루트 키 k와 세션키 S, 메시지 M을 선택하여 다음과 같이 암호문 $C=(C_0,..., C_{m-1}, F_{S\oplus k}(M))$를 계산하고 사용자들에게 전송한다.

$$C_i=(h_i, h_{i,0},..., h_{i,2l-1}, E_{s_i\oplus k_{s_{d,i}}}(S)),\quad r_i, s_i\in_R Z_q,$$

$$0\le i\le m-1$$

$$h_i=g^{r_i},\ h_{i,j}=\begin{cases}s_i\cdot g^{r_i a_0} & (j=0 \mod 2l)\ \text{if } i=0,\ s_0\cdot g^{r_0 b_0}\\ g^{r_i a_j} & (j\neq i \mod 2l)\\ g^{r_i b_i} & (j=i \mod 2l)\end{cases}$$

여기서 E, F는 블록암호이다.

마. 복호화(Decryption)

만약 $u\in U_i$라면 u는 자신의 개인키 $\{u, i, f_i(u), k_{s_{d,i}}, H(\overline{k_{s_{d,i}}}),..., H(\overline{k_{s_{1,i}}}), k\}$를 가지고 다음과 같이 암호문 C_i에서 s_i를 올바르게 복호화 할 수 있다.

$$s_i = \left\{(h_{i,0}\times h_{i,1}^{u}\times \cdots \times h_{i,2l-1}^{u^{2l-1}})\ /\ h_i^{f_i(u)}\right\}$$

$$= \left\{(s_i)\ g^{r_i\sum_{j=0}^{2l-1}a_{i,j}u^j}\ /\ g^{r_i f_i(u)}\right\}$$

leaf 노드 키 $k_{s_{d,i}}$를 가지고 $s_i\oplus k_{s_{d,i}}$를 구하여 $E_{s_i\oplus k_{s_{d,i}}}(S)$에서 S를 구하고 $F_{S\oplus k}(M)$을 통해 M을 복구한다.

바. 제외(Revocation)

센터가 집합 U_i의 모든 사용자를 제외시키고자 한다면 암호문 $C=(C_0,..., C_{m-1}, F_{S\oplus k}(M))$ 중에서 C_i 부분만 임의의 $z_i\in Z_q$를 선택하여 다음과 같이 암호문을 계산하여 전송하면 U_i의 모든 사용자는 s_i를 구할 수 없다.

$$C_i=(h_i, h_{i,0},..., h_{i,2l-1}, E_{s_i\oplus k_{s_{d,i}}}(S)),\quad r_i, s_i\in_R Z_q,$$

$$0\le i\le m-1$$

$$h_i=g^{r_i},\ h_{i,j}=\begin{cases}s_i\cdot g^{r_i a_0} & (j=0 \mod 2l)\ \text{if } i=0,\ s_0\cdot g^{r_0 b_0}\\ g^{r_i a_j} & (j\neq i \mod 2l)\\ g^{z_i} & (j=i \mod 2l)\end{cases}$$

또한 센터가 집합 U_i 중에서 사용자 u_α만을 제외시키고자 한다면 $U_i\setminus\{u_\alpha\}=\{u_{i_1},..., u_{i_{l-1}}\}$에 대해서 $\sum_{i=0}^{2l-1}c_i u_\alpha^i\neq 0 \mod q$과 $\sum_{i=0}^{2l-1}c_i u_j^i=0 \mod q$, $1\le j\le l-2$을 만족하는 임의의 $c_0, c_1,..., c_{2l-1}\in Z_q$를 선택하여 다음과 같이 계산하여 전송하면 된다.

$$C_i=(h_i, h_{i,0},..., h_{i,2l-1}, E_{s_i\oplus k_{s_{d,i}}}(S)),\quad r_i, s_i\in_R Z_q,$$

$$0\le i\le m-1$$

$$h_i=g^{r_i},\ h_{i,j}=\begin{cases}s_i\cdot g^{c_0+r_i a_0} & (j=0 \mod 2l)\\ g^{c_j+r_i a_j} & (j\neq i \mod 2l)\\ g^{c_i+r_i b_i} & (j=i \mod 2l)\end{cases}$$

예를 들어, u_α의 경우

$$\left\{(h_{i,0}\times h_{i,1}^{u_\alpha}\times \cdots \times h_{i,2l-1}^{u_\alpha^{2l-1}})\ /\ h_i^{f_i(u_\alpha)}\right\}$$

$$= \left\{(s_i)\ g^{\sum_{j=0}^{2l-1}c_j u_\alpha^j}\ g^{r_i\sum_{j=0}^{2l-1}a_{i,j}u_\alpha^j}\ /\ g^{r_i f_i(u_\alpha)}\right\}$$

에서 $\sum_{i=0}^{2l-1}c_i u_\alpha^i\neq 0 \mod q$이므로 s_i를 구할 수 없다.

하지만 제외되지 않은 다른 사용자 u_j, $1 \le j \le l-2$의 경우(즉, $\sum_{i=0}^{2l-1} c_i u_j^i = 0 \mod q$)는 s_i를 복구할 수 있으며 따라서 M을 복호화 할 수 있다.

사. 키 업데이트(Key-update)

센터는 불법 디코더를 발견했을 때, 또는 일정 시간이 지난 뒤에 시스템의 안전성을 높이기 위해 사용자의 개인키를 새롭게 업데이트(update)할 수 있다.

1) 집합 U_i의 모든 사용자 개인키 업데이트

사용자 $u \in U_i$의 개인키 $\{f_i(u),\ k_{s_{d,i}},\ H(k_{\overline{s_{d,i}}}),\ ...,\ H(k_{\overline{s_{1,i}}}), k\}$를 새로운 개인키 $\{s_i \cdot f_i(u),\ k'_{s_{d,i}}, H(k'_{\overline{s_{d,i}}}),\ ...,\ H(k'_{\overline{s_{1,i}}}), k'\}$로 업데이트(update)하기 위하여 다음과 같이 키 업데이트(update) 암호문을 전송한다.

$$KU = (C_0, ..., C_{m-1})$$

$$C_i = (h_i, h_{i,0}, ..., h_{i,2l-1}, E_{s_i \oplus k_{S_{d,i}}}(k'_{s_{d,i}})),\quad r_i, s_i \in_R Z_q,$$

$$0 \le i \le m-1$$

$$h_i = g^{r_i},\quad h_{i,j} = \begin{cases} s_i \cdot g^{r_i a_0} & (j = 0 \mod 2l) \\ g^{r_i a_j} & (j \ne i \mod 2l) \\ g^{r_i b_i} & (j = i \mod 2l) \end{cases}$$

여기서 구한 s_i를 $f_i(u)$와 곱한 $s_i \cdot f_i(u)$로 부분키 $f_i(u)$를 업데이트(update)한다.

개인키 중 $k_{\overline{s_{d,i}}} \in U_j$에 대해서는 암호문 $C_j = (h_j,\ h_{j,0},\ ...,\ h_{j,2l-1},\ E_{s_j \oplus k_{\overline{S_{d,i}}}}(H(k'_{s_{d,i}})))$로, 개인키 중 $k_{\overline{s_{d-1,i}}} \in U_\alpha$, U_β에 대해서는 $C_\alpha = (h_\alpha,\ h_{\alpha,0},\ ...,\ h_{\alpha,2l-1},\ E_{s_\alpha \oplus k_{\overline{S_{d-1,i}}}}(H(k'_{s_{d-1,i}})))$, $C_\beta = (h_\beta,\ h_{\beta,0},\ ...,\ h_{\beta,2l-1},\ E_{s_\beta \oplus k_{\overline{S_{d-1,i}}}}(H(k'_{s_{d-1,i}})))$를 계산한다. 위의 과정을 노드 키 $k_{\overline{s_{1,i}}}$을 가지는 집합 U_γ들까지 암호문 C_γ들을 계산하여 전송한다면 모든 사용자들은 이진 트리의 해당되는 내부 노드 키를 복구할 수 있다.

예를 들어, Fig. 2에서 모든 사용자 $u \in U_0$의 개인키 $\{f_i(u), k_{00}, k_0, k\}$를 $\{s_i \cdot f_i(u), k'_{00}, k'_0, k'\}$로 업데이트(update)하고자 한다면 다음을 전송하면 된다.

$$KU = (C_0, ..., C_3)$$

$$C_0 = (h_0, h_{0,0}, ..., h_{0,2l-1}, E_{s_0 \oplus k_{00}}(k'_{00}))$$

$$C_1 = (h_1, h_{1,0}, ..., h_{1,2l-1}, E_{s_1 \oplus k_{01}}(H(k'_{00})))$$

$$C_2 = (h_2, h_{2,0}, ..., h_{2,2l-1}, E_{s_2 \oplus k_1}(H(k'_0)))$$

$$C_3 = (h_3, h_{3,0}, ..., h_{3,2l-1}, E_{s_3 \oplus k_1}(H(k'_0)))$$

$$h_i = g^{r_i},\quad h_{i,j} = \begin{cases} s_i \cdot g^{r_i a_0} & (j = 0 \mod 2l) \\ g^{r_i a_j} & (j \ne i \mod 2l) \\ g^{r_i b_i} & (j = i \mod 2l) \end{cases}$$

2) 모든 사용자의 개인키 업데이트(update)

집합 $U_0, ..., U_{m-1}$에 해당되는 이진 트리의 노드 키 $k_{s_{d,0}}, ..., k_{s_{d,m-1}}$를 새로운 노드 키 $k'_{s_{d,0}}, ..., k'_{s_{d,m-1}}$로 업데이트(update)하고, 루트 노드 키까지의 모든 내부 노드 키 $k_{s_{d-1,0}}, ...,\ k_{s_{1,0}},\ k_{s_{d-1,1}}, ...,\ k_{s_{1,1}}, ...,\ k_{s_{d-1,m-1}}, ...,\ k_{s_{1,m-1}}$를 $k'_{s_{d-1,0}}, ...,\ k'_{s_{1,0}},\ k'_{s_{d-1,1}}, ...,\ k'_{s_{1,1}}, ...,\ k'_{s_{d-1,m-1}}, ...,\ k'_{s_{1,m-1}}$로 업데이트(update)하기 위해 다음과 같이 키 업데이트(update) 암호문을 전송한다.

$$KU = (C_0, ..., C_{m-1})$$

$$C_i = (h_i, h_{i,0}, ..., h_{i,2l-1},$$

$$E_{s_i \oplus k_{S_{d,i}}}(k'_{s_{d,i}}), E_{H(k_{\overline{S_{d,i}}})}(H(k'_{\overline{s_{d,i}}})), ..., E_{k_{S_{1,i}}}(H(k'_{\overline{s_{1,i}}})))$$

$$h_i = g^{r_i},\quad h_{i,j} = \begin{cases} s_i \cdot g^{r_i a_0} & (j = 0 \mod 2l) \\ g^{r_i a_j} & (j \ne i \mod 2l) \\ g^{r_i b_i} & (j = i \mod 2l) \end{cases}$$

위의 암호문을 통해 $u \in U_i$는 C_i의 암호문을 통해서 s_i를 복구하고 $f_i(u)$를 $s_i \cdot f_i(u)$로 업데이트(update)한다. 그리고 $k'_{s_{d,i}}$를 복호화 하고 $E_{k'_{S_{d,i}}}(H(k'_{\overline{s_{d,i}}}))$에서 $H(k'_{\overline{s_{d,i}}})$를 복호화 한다. 해쉬 함수를 이용하여 $H(k'_{s_{d,i}}) \oplus H(k'_{\overline{s_{d,i}}}) = k'_{s_{d-1,i}}$를 계산하고 이 과정을 반복하여 최종적으로는 $H(k'_{s_{1,i}}) \oplus H(k'_{\overline{s_{1,i}}}) = k'$을 계산한다.

따라서 u의 개인키 $f_i(u_i), k_{s_{d,i}}, H(k_{\overline{s_{d,i}}}), ..., H(k_{\overline{s_{1,i}}}),$ k를 $s_i \cdot f_i(u_i), k'_{s_{d,i}}, H(k'_{\overline{s_{d,i}}}), ..., H(k'_{\overline{s_{1,i}}}), k'$로 업데이트(update) 할 수 있다.

예를 들어, Fig. 2에서 $u \in U_0$의 경우, u는 C_0에서 s_0를 복구하고 $s_0 \cdot f_i(u)$ 부분키를 얻고, k'_{00}을 구한다. $E_{k'_{00}}(H(k'_{01}))$에서 $H(k'_{01})$을 구한다. 해쉬 함수를 이용하여 $H(k'_{00}) \oplus H(k'_{01}) = k'_0$을 구하고 $E_{k'_0}(H(k'_1))$에서 $H(k'_1)$을 구하여 최종적으로 $H(k'_0) \oplus H(k'_1) = k'$를 구한다. 따라서 U_0에 있는 모든 사용자들의 이진 트리의 노드 키 $\{f_0(u_0), k_{00}, H(k_{01}), H(k_1), k\}$를 $\{s_0 \cdot f_0(u_0), k'_{00}, H(k'_{01}), H(k'_1), k'\}$로 업데이트(update)를 할 수 있다. 따라서 키 업데이트(update) 과정은 정당한 사용자의 개인키 없이는 새로운 개인키로 업데이트(update)할 수 없다는 것을 알 수 있다. 업데이트(update) 이후의 암호화 과정은

$$C = (C_0, ..., C_{m-1}, F_{S \oplus k'}(M))$$

$$C_i = (h_i, h_{i,0}, ..., h_{i,2l-1}, E_{s'_i \oplus k'_{S_{d,i}}}(S)), \quad r_i, s'_i \in_R Z_q,$$

$$0 \le i \le m-1$$

$$h_i = g^{r_i}, \quad h_{i,j} = \begin{cases} s'_i \cdot g^{s_i r_i a_0} & (j = 0 \bmod 2l) \\ g^{s_i r_i a_j} & (j \ne i \bmod 2l) \\ g^{s_i r_i b_i} & (j = i \bmod 2l) \end{cases}$$

이고 복호화 과정은 동일하다.

아. 추적 알고리즘(Tracing algorithm)

추적 알고리즘은 불법 디코더가 발견되었을 때, 디코더를 분해하지 않고 저장되어있는 키를 확인하여 공모한 사용자들 중 적어도 한 명을 추적하는 방법이다. 먼저 $|U_0| = ... = |U_{m-1}| = l, m = n/l$로 구분한다. s = 0으로 설정하고 다음 알고리즘을 수행한다.

Step 1. 센터는 k_s의 child 노드 $k_{(s\|0)}$, $k_{(s\|1)}$를 선택하여 암호문 $C = (C_0, ..., C_{m-1}, F_{S \oplus k_{(S\|0)}}(M))$, $C' = (C_0, ..., C_{m-1}, F_{S \oplus k_{(S\|1)}}(M))$를 계산한다.

Step 2. 암호문 $C = (C_0, ..., C_{m-1}, F_{S \oplus k_{(S\|0)}}(M))$을 불법 디코더에 입력한다.

- 디코더가 올바르게 복호화 한다면 센터는 $k_{(s\|0)}$에 대하여 Step 1 과정으로 되돌아간다. 만약 $s \in \{0,1\}^{d-1}$이면 **Step 3**로 이동한다.

- 디코더가 복호화하지 못한다면 $C' = (C_0, ..., C_{m-1}, F_{S \oplus k_{(S\|1)}}(M))$를 디코더에 입력한다.

- 디코더가 올바르게 복호화 한다면 센터는 $k_{(s\|1)}$에 대하여 **Step 1** 과정으로 되돌아간다. 만약 $s \in \{0,1\}^{d-1}$이면 **Step 3**로 이동한다.

- 디코더가 복호화하지 못한다면 $k_{(s\|0)}$, $k_{(s\|1)}$에 대하여 각각 **Step 1** 과정으로 되돌아간다.

Step 3. 불법 디코더가 암호문 $C = (C_0, ..., C_{m-1}, F_{S \oplus k_{S_{d,i}}}(M))$, $s_{d,i} \in \{0,1\}^d$를 올바르게 복호화 했다면 U_i의 사용자들 중에는 반드시 공모자가 존재한다. 따라서 다음 알고리즘을 수행한다.

$1 \le j \le l$에 대하여 다음 알고리즘을 m번 반복한다.

- 먼저 ctr_j = 0으로 설정하고 반복할 때 마다, r_i, $s_i \in Z_q$를 임의로 선택한다.
  제외 집합을 $A = \{u_{i_1}, ..., u_{i_j}\}$라 하고 $U_i \setminus A = \{x_1, ..., x_w\}$라고 가정하자.
  센터는 $2l-1-w$개의 서로 다른 임의의 $\{x_{w+1}, ..., x_{2l-1}\} \setminus U_i$를 선택한다.
  또한 $\sum_{j=0}^{2l-1} c_j x_\alpha^j = 0 \mod q, 0 \le \alpha \le 2l-1$를 만족하는 $c_0, ..., c_{2l-1}$를 찾고 다음과 같이 암호문을 전송한다.

$$C_i = (h_i, h_{i,0}, ..., h_{i,2l-1}, E_{s_i \oplus k_{S_{d,i}}}(S))$$

$$h_i = g^{r_i}, \quad h_{i,j} = \begin{cases} s_i \cdot g^{c_0 + r_i a_0} & (j = 0 \bmod 2l) \\ g^{c_i + r_i a_j} & (j \ne i \bmod 2l) \\ g^{c_i + r_i b_i} & (j = i \bmod 2l) \end{cases},$$

$$r_i, s_i \in_R Z_q$$

만약 $t \neq i$이고 $A \cap U_t = \oslash$을 만족하는 집합 U_t에 대해서는 암호문

$$C_t = (h_t, h_{t,0}, ..., h_{t,2l-1}, E_{s_t \oplus k_{S_{d,t}}}(S))$$

$$h_t = g^{r_t},\ h_{t,j} = \begin{cases} s_t \cdot g^{r_t a_0} & (j = 0 \bmod 2l) \\ g^{r_t a_j} & (j \neq t \bmod 2l) \\ g^{r_t b_t} & (j = t \bmod 2l) \end{cases}$$

$$r_t, s_t \in_R Z_q,\quad 0 \leq t \leq m-1$$

을 전송한다.

- 암호문 $C = (C_0, ..., C_{m-1}, F_{S \oplus k}(M))$을 불법 디코더에 입력한다.

- 불법 디코더가 올바르게 복호화 한다면 ctr_j를 하나씩 증가시킨다.

Step 4. 최종적으로 $ctr_{j-1} - ctr_j$가 가장 큰 값을 나타내는 $j \in \{1, ..., n\}$를 찾는다. 그러면 불법 디코더를 만들 때 공모한 사용자는 u_j라는 것을 확인할 수 있다.

여기서 **Step 3**의 알고리즘을 반복하는 m은 $O(l^2 \log^2 l)$이다. 따라서 Step 3의 알고리즘의 총 연산량은 $O(l^3 \log^2 l)$이라는 것을 알 수 있다.

4. 제안 알고리즘의 안전성 및 효율성 분석

가. 안전성 분석

제안하는 시스템은 다음 security를 반드시 만족해야 한다.

1) Semantic security

시스템에 가입하지 않는 사용자라면 암호문을 통해서 세션키 s를 복구할 수 없어야 하고, 따라서 M을 구할 수 없어야 한다.

2) Computational secrecy

l명 이하의 악의적인 사용자가 자신의 개인키를 유출 또는 공모를 하더라도 시스템 매니저가 생성하지 않는 새로운 개인키를 만들 수 없어야 한다.

또한 올바른 암호문과 틀린 암호문을 구분할 수 없어야하고, OFT 알고리즘의 안전성을 증명할 수 있어야 한다.

[정리 1] DDH 가정하에, 제안한 알고리즘의 암호화 과정은 정당하지 않은 사용자에 대하여 안전(semantic security)하다.

[증명] 암호화 과정이 안전하지 않다고 가정하자. 그렇다면 알고리즘의 공개키 $e = \{g, g^{a_0}, g^{a_1}, ..., g^{a_{2l-1}}, g^{b_0}, ..., g^{b_{m-1}}\}$에 대하여 정당하지 않은 사용자가 두개의 세션키 $s^{(0)}, s^{(1)} \in G_q$를 생성하여 DDH 문제를 푸는 알고리즘에 전송한다. 알고리즘은 두 개의 세션키 중 하나로 암호문을 계산하여 정당하지 않은 사용자에게 전송한다. 암호문을 전송받은 사용자는 어떤 세션키로 암호문을 만들었는지 구분할 수 있으므로 DDH 문제를 풀 수 있다는 것을 증명한다. 이러한 과정의 증명 단계는 다음과 같다. 먼저 $< g_1, g_2, u_1, u_2 >$이 주어졌다고 하자.

Step 1. α_i, β, $a_i \in_R Z_q$를 선택하여 공개키 $e = \{g_1, g_1^{a_0}, g_1^{a_1}, ..., g_1^{a_{2l-1}}, g_1^{b_0}, ..., g_1^{b_{m-1}}\}$, $g_1^{b_i} = g_1^{\alpha_i} g_2^{\beta}$를 계산한다.

Step 2. $e = \{g_1, g_1^{a_0}, g_1^{a_1}, ..., g_1^{a_{2l-1}}, g_1^{b_0}, ..., g_1^{b_{m-1}}\}$를 정당하지 않은 사용자에게 전송하고 사용자는 $s^{(0)}, s^{(1)} \in G_q$를 다시 전송한다.

Step 3. $b \in_R \{0,1\}$를 선택하여 다음과 같이 암호문을 계산하여 사용자에게 전송한다.

$$C = (C_0, ..., C_{m-1}, F_{S \oplus k}(M))$$

$$C_i = (h_i, h_{i,0}, ..., h_{i,2l-1}, E_{s^{(b)} \oplus k_{S_{d,i}}}(S)),$$

$$r_i \in_R Z_q,\quad 0 \leq i \leq m-1$$

$$h_i = g_1^{r_i} u_1,\quad h_{i,j} = \begin{cases} s^{(b)} \cdot (g_1^{r_i} u_1)^{a_0} & (j = 0 \bmod 2l) \\ (g_1^{r_i} u_1)^{a_j} & (j \neq i \bmod 2l) \\ (g_1^{r_i} u_1)^{\alpha_i} (g_2^{r_i} u_2)^{\beta} & (j = i \bmod 2l) \end{cases}$$

$$\text{if}\quad i = 0,\ s^{(b)} \cdot (g_1^{r_0} u_1)^{\alpha_0} (g_2^{r_0} u_2)^{\beta}$$

Step 4. 사용자는 $b' \in \{0,1\}$을 전송한다.

Step 5. 만약 $b = b'$이면 "D"를 출력하고, 그렇지 않으면 "R"을 출력한다.

만약 $< g_1, g_2, u_1, u_2 >$이 D라면 암호문 $C_i = (h_i, h_{i,0}, ..., h_{i,2l-1}, E_{s^{(b)} \oplus k_{S_{d,i}}}(S))$는 세션키 $s^{(b)}$를 암호화한 것이고 $< g_1, g_2, u_1, u_2 >$이 R라면 암호문 C_i는 임의의 값을 암호화한 것을 알 수 있다. 따라서 정당하지 않은 사용자는 DDH 문제를 무시하지 못할 확률로 풀 수 있다는 것을 의미한다.

두 번째 정리로 공모자들은 암호문과 틀린 암호문을 구분할 수 없다는 것으로 다음과 같다.

[정리 2] DDH 가정하에 l명의 제외되지 않은 공모자들은 정상적인 암호문과 추적 알고리즘에 사용되는 틀린 암호문을 구별할 수 없다.

[증명] 암호문을 구분할 수 있는 알고리즘이 있다면 DDH 문제를 풀 수 있다는 것을 보인다. 먼저 $< g_1, g_2, u_1, u_2 >$가 주어졌다고 가정하자. DDH 문제를 푸는 알고리즘을 다음과 같이 만든다. 추적 알고리즘의 Step 2에서 얻은 공모자가 포함된 집합 U_i, 제외하고자 하는 집합을 A라고 하자. 제외되지 않은 l명의 공모자 집합 $B = \{u_1, ..., u_l\}$를 $A \cap B = \oslash$이 되도록 선택한다. 그리고 $u_{l+1}, ..., u_{2l-1} \in Z_q \setminus B$와 임의의 $\beta_1, ..., \beta_l, \lambda, \mu, \psi_t, \omega_t \in Z_q$, $l+1 \le t \le 2l-1$를 선택하여 다음 다항식

$$\alpha(x) = \sum_{m=0}^{2l-1} \alpha_m x^m \mod q \;\; s.t \; g_1^{\alpha_0} = g_1^{\lambda} g_2^{\mu} \text{ 과}$$

$$(\alpha(u_1), ..., \alpha(u_{2l-1}))^T = (\beta_1, ..., \beta_{2l-1})^T = (\alpha_0, ..., \alpha_0)^T + V(\alpha_1, ..., \alpha_{2l-1})^T,$$

$$g_1^{\beta_t} = g_1^{\psi_t} g_2^{\omega_t} \;\; (l+1 \le t \le 2l-1)$$

을 계산한다.

여기서 $V = \begin{pmatrix} x_1 & \cdots & x_1^{2l-1} \\ \vdots & \ddots & \vdots \\ x_{2l-1} & \cdots & x_{2l-1}^{2l-1} \end{pmatrix} \mod q$이고, V는 Vandermonde 행렬이므로 역행렬이 존재하여 다음을 얻는다.

$$(\alpha_1, ..., \alpha_{2l-1})^T = V^{-1}(\beta_1 - \alpha_0, ..., \beta_{2l-1} - \alpha_0)^T$$

$(v_{m,1}, ..., v_{m,2l-1})$을 V^{-1}의 m번째 행이라 하면,

$1 \le m \le 2l-1$에 대하여 $\alpha_m = v_{m,1}\beta_1 + ... + v_{m,2l-1}\beta_{2l-1} - \alpha_0(v_{m,1} + ... + v_{m,2l-1})$이므로 $g_1^{\alpha_m} = g_1^{v_{m,1}\beta_1 + ... + v_{m,2l-1}\beta_{2l-1}} / (g_1^{\lambda} g_2^{\mu})^{v_{m,1} + ... + v_{m,2l-1}}$을 얻는다. $B = \{u_1, ..., u_l\} \subset U_i$에 대하여 다음을 만족하는 $\delta_i, \gamma_i \; s.t \; \delta_i = b_i + \gamma_i - \alpha_i$, $g_1^{b_i} = g_1^{\lambda_i} g_2^{\mu_i}$를 선택하여 (x_j, i, d_j), $1 \le j \le l$의 개인키 d_j를 다음과 같이 계산한다.

$$d_j = \alpha(x_j) + \delta_i x_j^i = \alpha_0 + ... + b_i x_j^i + ... + \alpha_{2l-1} x_j^{2l-1} + \gamma_i x_j^i$$

$d_j = f_i(x_j)$와 같도록 f_i의 계수 $a_0, ..., a_{2l-1}$를 표현하기 위하여 l개의 원소 $\{\theta_1, ..., \theta_l\}$에 대하여

$$g_1^{\alpha'_{\theta_1}}, ..., g_1^{\alpha'_{\theta_l}} \; s.t \; g_1^{\sum_{\tau \in \{\theta_1, ..., \theta_l\}} \alpha'_\tau x_j^\tau} = g_1^{\gamma_i x_j^i} = (g_1^{\delta_i} g_2^{\alpha_i} / g_1^{b_i})^{x_j^i} \quad (1 \le j \le l)$$

를 계산하여 다음 공개키 e를 만든다.

$$g_1^{a_m} = \begin{cases} g_1^{\alpha_m} & m \notin \{\theta_1, ..., \theta_l\} \\ g_1^{\alpha_m + \alpha'_m} & m \in \{\theta_1, ..., \theta_l\} \end{cases}$$

$$e = \left\{ g_1, g_1^{a_0}, ..., g_1^{a_{2l-1}}, g_1^{b_0}, ..., g_1^{b_{l-1}} \right\}$$

그리고 다음과 같이 암호문을 만든다.

$$C_i = (h_i, h_{i,0}, .., h_{i,2l-1}, E_{s \oplus k_{S_{d,i}}}(S)), \; r_i, s \in_R Z_q$$

$$h_i = u_1, \quad h_{i,j} = \begin{cases} s_i . u_1^{a_0} & (j = 0 \mod 2l) \\ u_1^{a_j} & (j \ne i \mod 2l) \\ u_1^{\lambda_i} u_2^{\mu_i} & (j = i \mod 2l) \end{cases}$$

$$\text{if } i = 0, \;\; s_0 . u_1^{\lambda_0} u_2^{\mu_0}$$

$$u_1^{a_j} = \begin{cases} u_1^{\alpha_j} & j \notin \{\theta_1, ..., \theta_l\} \\ u_1^{\alpha_j + \alpha'_j} & j \in \{\theta_1, ..., \theta_l\} \end{cases},$$

여기서 $u_1^{\alpha_j}, u_1^{\alpha'_{\theta_1}}, ..., u_1^{\alpha'_{\theta_l}}$은 위의 과정과 비슷하게 얻어진다. 만약 $< g_1, g_2, u_1, u_2 >$이 "D"라면 위의 암호문 C_i는 올바른 암호문이고, $< g_1, g_2, u_1, u_2 >$이 "R"이라면 위의 암호문 C_i는 틀린 암호문이다. 따라서 위의 $C_i, e, (x_1, i, d_1), ..., (x_l, i, d_l)$을 구분할 수 있는 알고리즘에 입력하여 출력이 "D"라면 올바른 암호문이고, 출력이 "R"이라면 틀린 암호문이라는 것을 알 수 있다. 따라서 알고리즘은 DDH 문제를 무시하지 못할 확률로 풀 수 있다는 것을 의미한다.

본 논문에서 이진 트리의 키 부여(key assignment)로 사용되는 OFT 알고리즘의 안전성에 관한 내용으로 제안 알고리즘에서 가장 중요한 성질로 다음과 같은 성질을 가지고 있다.

[성질 1] 함수 h가 일방향 함수이고, 정당하지 않은 사용자에게 $h(x)$가 주어졌을 때 무시하지 못할 확률로 pre-image x를 계산할 수 없다면 OFT 알고리즘은 안전하다.

만약 정당하지 않는 사용자가 주어진 $h(x)$에서 x를 계산할 수 있다면, x노드 아래의 모든 사용자 집합과 트리 노드의 개인키가 같아진다. 그렇게 된다면 추적 알고리즘에서 공모한 사용자가 있는 집합을 찾는 과정인 Step 1, Step 2에서 어느 사용자 집합에 포함되어 있는지 구별할 수 없게 된다. 따라서 제안한 추적 알고리즘으로는 공모자를 추적할 수 없게 되므로 제안 알고리즘이 안전하지 않게 된다.

네 번째로는 추적 알고리즘으로 적어도 한명의 공모자를 높은 확률로 찾을 수 있다는 것을 다음 정리로 증명한다.

[정리 3] 제안한 알고리즘에서 l명 이하의 사용자가 공모하여 불법 디코더를 만들더라도 적어도 한명의 공모자를 높은 확률로 추적할 수 있다.

[증명] 제외하고자 하는 집합 $B = \{u_1, ..., u_j\}$라고 하자. ctr_j는 불법 디코더가 집합 B에 해당되는 암호문에 대해서 올바르게 복호화한다면 ctr_j를 1을 증가시키고, 그렇지 않다면 증가시키지 않는 것을 확인하는 변수이다. 따라서 집합 $B = \oslash$이면 $j = 0$이고, $ctr_0 = m$이라는 것과 집합 $B = \{u_1, ..., u_N\}$이면 $ctr_N = 0$이다.

삼각 부등식에 의해서 $ctr_{j-1} - ctr_j \geq m/N$을 만족하는 $j \in \{1, ..., N\}$가 반드시 존재한다. 결국 $ctr_{j-1} - ctr_j$이 큰 값을 가진다면 사용자 u_j가 높은 확률로 공모자임을 확인할 수 있다.

나. 효율성 분석

센터가 모든 사용자에게 전송하기 위한 암호문 $C = (C_0, ..., C_{m-1}, F_{S \oplus k}(M))$을 생성하기 위해서는 $2l + N/l$개의 지수승 연산이 필요하므로 전송량은 $O(l + N/l)$이다. 사용자 $u \in U_i$의 개인키는 u, i, $f_i(u)$, $k_{s_{d,i}}$, $H(k_{s_{d,i}})$, ..., $H(k_{s_{1,i}})$, k이므로 $O(\log(N/l))$의 저장량이 필요하며, 사용자의 복호화에 필요한 연산량은 $O(l)$의 곱셈 연산과 기껏해야 $O(\log N/l)$번의 xor 비트연산이 필요하다. 또한 센터가 사용자의 개인키 모두를 업데이트(update)하기 위해서는 $O(l + N/l)$의 전송량과 함께 $O(\log N/l)$개의 대칭키 암호시스템의 암호문이 필요하고, 사용자는 $O(l)$의 곱셈 연산과 $O(\log N/l)$번의 xor 비트연산으로 개인키를 업데이트(update)할 수 있다.

또한 추적 알고리즘의 연산량은 먼저 공모자가 속해있는 U_i 집합을 추적하는데 필요한 연산량 $O(\log N/l)$과 집합 U_i의 사용자들 중에서 공모자 한 명을 추적하는데 필요한 연산량 $O(l^3 \log^2 l)$의 합이므로 $O(\log N/l + l^3 \log^2 l)$ 이다.

기존에 제시된 대표적인 알고리즘 [5], [7]과 본 논문과의 효율성 비교를 하면 다음과 같다.

[7] 논문의 CS(Complete Subtree), SD(Subset Difference) 알고리즘은 제외자가 r 일때의 전송량, 키의 저장량, 연산량을 의미하며, 안전성은 수학적 가정에 기반한 것이 아닌 블록암호의 안전성에 기반을 하고 있어 알고리즘의 척도로는 가장 효율적인 알고리즘이다. 하지만 [3]에서 불법 디코더의 진화에 따라 추적 알고리즘을 적용하는 것이 현실적으로 어렵다는 것을 증명하였다. [5]논문은 겹선형 사상을 이용한 알고리즘으로 안전성은 겹선형 사상의 디피-헬만(Bilinear Diffie-Hellman assumption) 문제에 기반하고 있고 전송량은 본 논문과 비슷하지만 개인키의 저장량이 상대적으로 매우 크다는 단점이 있다. 또한 암호문을 만들기 위해서 $O(\sqrt{n})$의 겹선형 사상 연산이 필요하므로 현실적으로 응용되기엔 어려운 알고리즘이라 할 수 있다. 트리 구조를 사용한 효율적인 TT 알고리즘으로 CS, SD 알고리즘은 사용자의 stateless 상황을 가정하여 설계하

였으며, 본 논문에서도 같은 전제조건으로 알고리즘을 설계하였다. 사용자의 stateful 환경을 가정한다면, 사용자의 변화에 따라 매번 키 업데이트 암호문을 전송해야 하는 번거로움이 있으므로 본 논문에서와 같이 일정세션이 지난 후에 또는 불법 디코더가 발견되었을 시에 키 업데이트를 하는 것이 더욱 효과적이다.

Table 1. 효율성 비교(exp : 순환군에서의 지수승 연산, pair : 겹선형 사상 연산을 의미, dec : 블록암호 복호화 개수, r : 제외자 수)

알고리즘	암호문 전송량	개인키 저장량	복호화 연산량	키 업데이트
CS[7]	$O(r\log\frac{n}{r})$	$O(\log\sqrt{n})$	$O(1)$dec	×
SD[7]	$O(2r-1)$	$O(\log^2\sqrt{n})$	$O(1)$dec	×
[5]	$O(\sqrt{n})$	$O(\sqrt{n})$	3 pair	×
제안 시스템	$O(\sqrt{n})$	$O(\log\sqrt{n})$	$O(\sqrt{n})$exp	○

5. 결 론

이 논문에서는 사용자의 개인키를 업데이트(update)할 수 있는 공모자 추적 암호 알고리즘을 제안하였다. 논문의 기본적인 프레임은[10] 알고리즘과 $n=l\,2^d$ 트리 구조에 OFT 알고리즘을 응용하여 사용자의 개인키를 업데이트(update)하는 알고리즘을 제안하였다. 제안하는 알고리즘은 $O(l+n/l)$의 전송량과 사용자 개인키의 $O(\log(n/l))$ 저장량을 가지며 적은 연산량 $O(n\sqrt{n}\log^2\sqrt{n})$을 필요로 하는 tracing 알고리즘을 제안하였다. 만약 센터가 최대 공모자의 수 $l\approx\sqrt{n}$으로 설정한다면, 센터의 전송량은 $O(\sqrt{n})$이고, 사용자의 저장량은 $O(\log\sqrt{n})$이다. 또한 복호화에 필요한 연산량은 $O(\sqrt{n})$이고, 추적 알고리즘에 필요한 연산량은 $O(n\sqrt{n}\log^2\sqrt{n})$이므로 일반적인 추적 알고리즘의 연산량 $O(n^3\log^2 n)$ 보다 줄일 수 있다.

사용자 개인키를 업데이트(update)하는 이유는 불법 디코더가 발견되었을 때, 추적 알고리즘의 연산량이 크기 때문에 추적 알고리즘을 수행해야 할 때, 공모자들 중 적어도 한명을 추적하기 위한 시간 동안 또 다른 디코더를 쓸모없게 하기 위함이다. 즉, 업데이트(update) 이후의 전송되는 콘텐츠에 대해서 더 이상 복호화를 하지 못하게 되므로 발견되지 않은 불법 디코더를 쓸모없게 만드는 알고리즘이라 할 수 있다.

후 기

이 논문은 국방과학연구소의 지원을 받아 수행된 연구임(UD110076ED).

References

[1] A. Kiayias, M. Yung, “On Crafty Pirates and Foxy Tracers”, DRM 2001, LNCS 2320.

[2] A. Kiayias, M. Yung, “Traitor Tracing with Constant Transmission Rate”, Eurocrypt 2002, LNCS 2332.

[3] A. Kiayias, S. Pehlivanoglu, “Pirate Evolution : How to Make the Most of Your Traitor Keys”, Crypto 2007.

[4] D. A. McGrew, A. T. Sherman, “Key Establishment in Large Dynamic Groups Using One-Way Function Trees”, IEEE Transactions on Software Engineering 2003, Vol. 29, No. 5.

[5] D. Boneh, B. Waters, “A Fully Collusion Resistant Broadcast Trace and Revoke System”, ACM CCS 2006.

[6] D. Boneh, M. Franklin, “An Efficient Public Key Traitor Tracing Scheme”, Crypto 1999, LNCS 1666.

[7] D. Naor, M. Naor, J. B. Lotspiech, “Revocation and Tracing Schemes for Stateless Receivers, Crypto 2001, LNCS 2139.

[8] K. Kurosawa, Y. Desmedt, “Optimum Traitor Tracing and Asymmetric Schemes”, Eurocrypt 1998, LNCS 1403.

[9] K. Kurosawa, T. Yoshida, “Linear Code Implies Public Key Traitor Tracing”, PKC 2002, LNCS 2274.

[10] T. Matsushita, H. Imai, “A Public Key Black Box Traitor Tracing Scheme with Sublinear Ciphertext Size Against Self Defensive Pirates”, Asiacrypt 2004, LNCS 3329.