정확성을 보장하는 결정적 Private Matching
(Deterministic Private Matching with Perfect Correctness)

홍 정 대 † 김 진 일 †† 천 정 희 ††† 박 근 수 ††††
(Jeongdae Hong) (Jinil Kim) (Jung Hee Cheon) (Kunsoo Park)

요 약 Private Matching은 각기 다른 두 참여자가 가진 데이타의 교집합을 구하는 문제이다. Private matching은 보험사기 방지시스템, 항공기 탑승 금지자 목록 검색, 의료 정보 검색 등에 이용될 수 있으며 다자간의 계산으로 확장하면 전자투표, 온라인 게임 등에도 이용될 수 있다. 2004년 Freedman 등 [1]은 이 문제를 확률적으로 해결하는 프로토콜을 제안하고 악의적인 공격자 모델과 다자간 계산으로 확장하였다. 이 논문에서는 기존의 프로토콜을 결정적(deterministic) 방법으로 개선하여 Semi-Honest 모델에서 결과의 정확성을 보장하는 한편, 이를 악의적인 공격자 모델에 확장하여 신뢰도와 연산속도를 향상시키는 새로운 프로토콜을 제안한다.

키워드 : Private Matching, 준동형 암호시스템, Semi-Honest 모델, 악의적인 클라이언트/서버 모델

Abstract Private Matching is a problem of computing the intersection of private datasets of two parties. One could envision the usage of private matching for Insurance fraud detection system, Do-not-fly list, medical databases, and many other applications. In 2004, Freedman et al. [1] introduced a probabilistic solution for this problem, and they extended it to malicious adversary model and multi-party computation. In this paper, we propose a new deterministic protocol for private matching with perfect correctness. We apply this technique to adversary models, achieving more reliable and higher speed computation.

Key words : Private Matching, Homomorphic Cryptosystem, Semi-Honest, Malicious-Client/Server Model

1. 서 론

특정 자료를 저장하고 있는 데이타 베이스나 기관 사이에 프라이버시를 유지(privacy preserving)한 가운데 공통의 자료를 추출하는 일은 유용한 정보를 생성하기 위한 효과적인 수단이다. 예를 들어 두 보험회사가 보험사기를 적발하기 위하여 두 보험사에 공통으로 가입한 고객을 찾고 싶을 때, 혹은 항공회사와 경찰청이 항공기 탑승자 명단에서 탑승 금지자를 찾고 싶을 때 서로의 고객 정보를 노출시키지 않을 수 있다면 서로 협력하기가 훨씬 쉬워질 것이다.

Private Matching은 두 참여자(two-party)가 자신의 집합을 가지고 있을 때, 서로의 원소를 노출시키지 않으면서 교집합을 찾는 문제로 위의 예와 같이 많은 분야에서 응용할 수 있는 중요한 문제이다.

Private Matching을 찾기 위한 연구는 Freedman 등에 의해 준동형 암호시스템(Homomorphic cryptosystem)을 사용하여 참여자의 데이타를 다항식으로 표현하여 전달하는 방법이 제안되었고[1], Kissner 등에 의해 일반적인 집합 연산과 다자간 모델로 확장되었다[2]. Kissner 등은 Freedman 등의 결과를 확장하기 위해 참여자의 데이타를 표현한 다항식에 차수가 같거나 큰 랜덤 다항식을 곱하여 이들을 더하는 방법을 사용하였다.

하지만 이들의 연구는 낮은 확률로 부정확한 결과를 낼 수 있는 단점이 있었으며, 정확도를 높이기 위해 실제 데이타의 개수에 비해 큰 정의역을 사용해야 했기 때문에 전처리와 같은 연산을 필요로 하였다.

우리는 이들의 방법을 개선하여 Semi-Honest 공격자 모델에서 항상 정확한 결과를 보장하는 결정적(deterministic)인 프로토콜을 제안하고 이를 응용하여 악의적인 공격자 모델에서 연산의 효율성을 높이는 방법을 제

† 학생회원 : 서울대학교 전기컴퓨터공학부
jdhong@theory.snu.ac.kr
†† 비 회 원 : 서울대학교 전기컴퓨터공학부
jikim@theory.snu.ac.kr
††† 비 회 원 : 서울대학교 수리과학부 교수
jhcheon@snu.ac.kr
†††† 종신회원 : 서울대학교 전기컴퓨터공학부 교수
kpark@theory.snu.ac.kr
논문접수 : 2006년 10월 31일
심사완료 : 2007년 6월 1일

안한다.

이 논문은 총 6개의 절로 구성되어 있는데 2절에서는 배경지식을 설명하고, 3절에서는 Freedman 등이 제안한 Private Matching을 구하는 방법을 간단하게 살펴본다. 그리고 4절에서는 Freedman 등의 방법을 개선한 우리의 프로토콜을 설명하고, 5절에서는 제안된 프로토콜의 정확도 및 효율성을 분석한 다음 마지막으로 6절에서는 결론을 맺는다.

2. 배경지식(Preliminary)

2.1 Private matching(PM)

Private matching(이하 PM)은 그림 1과 같이 두 참여자인 클라이언트(client) C와 서버(server) S가 각각 자신의 데이타로 구성된 집합 $X = \{x_1,...,x_{k_c}\}$와 $Y = \{y_1,...,y_{k_s}\}$를 가지고 있을 때, 프로토콜(protocol)을 수행한 후 어느 한쪽 집합에만 존재하는 원소는 노출시키지 않으면서 $X \cap Y$을 알아내는 것을 말한다[3].

우리의 프로토콜은 Freedman 등이 제안했던 방법과 동일하게 프로토콜 수행 후 C만 $X \cap Y$을 얻고, S는 C의 어떤 원소도 알지 못한다.

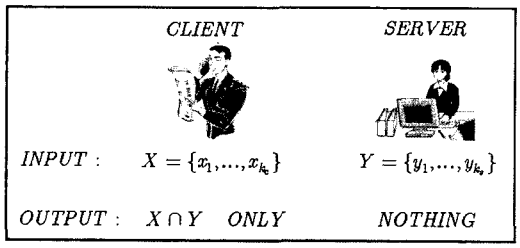

그림 1 Private Matching

2.2 준동형 암호시스템(Homomorphic Cryptosystem)

준동형 암호시스템은 개인키(private key)를 모르는 상태에서도 다음과 같은 연산이 가능한 공개키(public key) 암호시스템을 말한다.

- 공개키 pk에 의해 암호화된 2개의 암호문 $Enc_{pk}(m_1)$과 $Enc_{pk}(m_2)$가 주어졌을 때 이를 더한 새로운 암호문 $Enc_{pk}(m_1+m_2)$을 계산할 수 있다
- 임의의 상수 c와 암호문 $Enc_{pk}(m)$이 주어졌을 때 이를 곱한 암호문 $Enc_{pk}(cm)$을 계산할 수 있다.

위의 두 가지 성질을 이용하면 차수가 k인 다항식 $P(y) = a_0 + a_1y + \cdots + a_ky^k$의 계수들의 암호문 $\{Enc_{pk}(a_0),...,Enc_{pk}(a_k)\}$이 주어졌을 때, 다항식에 임의의 값 y를 대입한 $Enc_{pk}(P(y))$의 계산을 유도할 수 있다.

대표적인 준동형 암호시스템으로는 Paillier 암호시스템[4,5]이 있다.

2.3 공격자 모델(Adversary Model)

우리는 프로토콜을 C와 S가 모두 Semi-Honest한 Semi-Honest 모델, C는 악의적이고 S는 Semi-Honest한 악의적인 클라이언트(Malicious-Client) 모델, S는 악의적이고 C는 Semi-Honest한 악의적인 서버(Malicious-Server) 모델, C와 S가 모두 악의적인 모델의 4가지 경우로 구분하여 분석한다[6].

각각의 모델의 의미는 다음과 같다. 먼저 Semi-Honest 모델은 프로토콜에 참여하는 S와 C가 프로토콜의 모든 규칙을 준수하되 각자 상대방이 보내는 메시지와 프로토콜의 중간결과를 저장하여 두었다가 가능하다면 그 정보를 이용하여 상대방의 데이타를 알아내려는 시도를 할 수 있는 상황을 의미한다. 두 번째로 악의적인 클라이언트 모델은 C가 프로토콜을 벗어난 임의의 정보를 S에게 보내서 PM이 아닌 S의 데이타까지 알려고 하는 상황을 의미한다. (이 때 C가 단순히 S의 잘못된 계산을 유도하는 것은 의미가 없다.) 세 번째로 악의적인 서버 모델에서는 S가 의도적으로 S가 잘못된 결과를 얻게 하는 상황을 의미한다. 여기서 S의 공격은 이상적인 TTP(Trusted Third Party)[7]가 존재하여 S와 C의 집합을 입력으로 받아 C에게 교집합을 알려주는 TTP모델과 비교하여 TTP모델에서는 존재할 수 없는 공격으로 제한한다. 마지막으로 모두 악의적인 모델은 악의적인 클라이언트 모델과 악의적인 서버 모델을 결합하여 양쪽이 악의적인 상황을 의미한다.

2.4 보안성(Security)

위의 각 공격자 모델에서 보안성은 [6]의 기준을 따라 클라이언트와 서버의 관점에서 구분할 수 있다. 먼저 클라이언트의 보안성(client's security)은 C가 데이타를 변경해가면서 프로토콜을 수행할 때 서버가 데이타의 변경여부를 알지 못하는(indistinguishable) 것을 의미한다.

서버의 보안성(server's security)은 C가 정상적인 S와 프로토콜을 수행하여 얻은 PM이 가상의 신뢰할 만한 제3자(Trusted Third Party: TTP)로부터 받은 PM과 구분할 수 없이 같다는 것을 의미한다.

3. FNP 방법

Freedman 등은 Semi-Honest 및 악의적인(malicious) 공격자 모델에서 PM을 계산할 수 있는 프로토콜(이하 FNP)을 제안하였다[1]. 이 절에서는 FNP 방법을 간단하게 소개한다.

3.1 Semi-Honest 모델에서 PM 계산

Semi-Honest 모델에서는 다음과 같은 방법으로 PM을 계산한다. 먼저 클라이언트 C는 자신의 집합

$X = \{x_1, ..., x_{k_c}\}$의 원소들을 근(root)으로 가지는 다항식 $P(y)$를 만든다.

$$P(y) = (x_1 - y)(x_2 - y)\cdots(x_{k_c} - y) = \sum_{u=0}^{kc} \alpha_u y^u$$

그리고 $P(y)$의 모든 계수들을 자신의 공개키(public key)로 암호화한 값 $\{Enc_{pk}(\alpha_0), ..., Enc_{pk}(\alpha_{k_c})\}$을 서버 S에게 보낸다. S는 자신의 집합 $Y = \{y_1, ..., y_{k_s}\}$의 원소를 다항식에 대입한 값 $Enc_{pk}(P(y_i))$을 암호시스템의 준동형(homomorphic) 성질을 이용하여 계산하고 여기에 랜덤값(random value) r_i을 곱한 후, 다시 대입했던 원소를 더한 값 $Enc_{pk}(r_i \cdot P(y_i) + y_i)$을 C에게 보낸다.

C는 S로부터 받은 k_s개의 암호문을 자신의 개인키로 복호화(decryption)하여 $d_i := r_i \cdot P(y_i) + y_i$를 얻는다. $P(y_i) = 0$인 경우 $d_i = y_i \in X$인 반면 $P(y_i) \neq 0$인 경우 d_i는 랜덤 값이므로 높은 확률로 $d_i \notin X$이다. 그러므로 $d_i \in X$이면 d_i를 $X \cap Y$의 원소로 인식하고, 아니면 랜덤 값으로 인식한다. 계산된 집합이 $X \cap Y$ 값일 확률을 충분히 크게 하기 위해 X와 Y의 원소들이 충분히 큰 정의역에서 표현되어야 한다.

X와 Y의 원소의 개수가 많은 경우 효과적인 다항식 계산을 위해 균등 분배 함수(balanced allocation hash function), Horner's rule 등을 이용할 수 있다. 이 경우 C가 균등 분배 함수를 이용하여 X를 최대 M개의 원소를 갖는 B개의 부분집합 $X_1, X_2, ..., X_B$로 나눈 다음 각각의 부분집합에 대해 별도로 위의 프로토콜을 수행한다. 그러면 사용하는 다항식의 차수가 M으로 줄어들어 다항식 전개 및 대입에 필요한 연산량을 줄일 수 있다.

3.2 악의적인 클라이언트 모델에서 PM 계산

악의적인 클라이언트 C는 서버 S의 집합 Y의 원소 중 교집합에 속하지 않는 원소를 알아내거나 S가 잘못된 계산을 수행하도록 할 수 있다. 그런데 S는 C로부터 얻고자 하는 정보가 없으므로 C가 S로 하여금 잘못된 계산을 수행하도록 하더라도 얻을 수 있는 이익이 없다. 그러므로 여기서 악의적인 C는 Y의 원소를 알아내려고 하는 것으로 제한한다.

악의적인 C가 Y의 원소를 알아내기 위해서는 알아내고자 하는 Y의 원소가 C의 다항식 $P(y)$의 근이어야 한다. 그런데 정의역이 충분히 큰 경우 C가 미리 Y의 원소를 예상하여 다항식의 근으로 만드는 것은 현실적으로 어렵다. 그러므로 이 경우 C가 할 수 있는 공격은 무한개의 근을 가지는 다항식을 만들어 S에게 보내는 것이다. 이러한 다항식은 모든 계수가 0인 다항식으로 유일하다. C가 모든 계수가 0인 다항식을 보내면 S가 계산한 $Enc_{pk}(r_i \cdot P(y_i) + y_i)$는 항상 $Enc_{pk}(r \cdot 0 + y_i)$가 되므로 C가 복호화를 수행하면 y_i를 얻을 수 있다. (여기서 사용되는 암호시스템이 semantically secure하므로 S는 다항식의 계수들이 같은지 다른지 구분할 수 없다.)

FNP에서는 악의적인 C의 공격을 막기 위해서 다항식의 상수항을 1로 고정하는 방법을 사용하였다. 즉, C가 $P(y)$에 적절한 상수를 곱하여 다항식의 상수항을 1로 만들고 상수항을 제외한 계수를 전송한다. S는 미리 상수항을 1로 가정하여 받은 계수에 1을 C의 공개키로 암호화한 값을 상수항의 암호화한 값으로 사용한다. 이 방법을 사용하면 C는 $P(y)$로 모든 계수가 0인 다항식을 보낼 수 없게 되므로 악의적인 C의 공격을 막을 수 있게 된다.

그런데 이 방법은 균등 분배 함수를 이용한 방법에는 확장할 수 없기 때문에 FNP는 Cut-and-Choose 방법을 사용하여 L개의 다항식 집합을 만들어서 $L/2$개로부터 C의 근의 개수를 확인하고 나머지 $L/2$개로는 PM을 확인하는 방법을 사용하였다.

3.3 악의적인 서버 모델에서 PM 계산

악의적인 서버 S의 공격은 S가 의도적으로 C가 $X \cap Y$를 잘못 계산하도록 하는 것이다. 그런데 S가 자신의 집합 Y를 임의의 집합으로 대체해버리는 것과 같이 이상적인 TTP가 존재하는 경우에도 가능한 공격은 현실적으로 막기 어렵다. 그러므로 여기서는 S의 공격을 이상적인 TTP를 통하는 경우 존재할 수 없는 공격으로 제한하였다. 예를 들어 TTP는 $Enc_{pk}(r \cdot (P(y')) + y'')$와 같이 S가 보내지 않은 데이타를 PM으로 계산하지 않을 것이며, $Enc_{pk}(r \cdot (P(y) + (P(y')) + y'')$와 같이 C의 원소 하나에 S의 원소 2개를 비교 하지도 않을 것이다. (만약 y, y'이 X의 원소이면, C는 y''을 $X \cap Y$으로 인식하게 된다.)

FNP는 랜덤 오라클(random oracle)을 사용하여 위와 같은 형태의 공격을 막는 방법을 제안하였다. 즉 S와 C 모두 접근이 가능한 랜덤 오라클 H_1, H_2가 있어서 S가 임의의 랜덤값 s를 선택하여 $r = H_1(s)$를 얻고 $(e,h) \leftarrow (Enc_{pk}(r' \cdot P(y)+s), H_2(r'',y))$를 계산해서 C에게 보낸다. (이때 암호시스템에 사용되는 랜덤값은 H_1으로부터 얻은 r을 사용하며 r', r''은 r로부터 추출한다.) C는 e를 복호화하여 $\tilde{s}$을 얻고 $\tilde{r} = H_1(\tilde{s})$를 계산하여 $\tilde{r}', \tilde{r}''$를 추출한다. 그리고 S의 연산을 재구성하여 $(\tilde{e}, \tilde{h})$를 구하여 그 값이 S로부터 받은 (e, h)와 일치하는 자신의 데이타 x_j가 있는지를 확인하여 PM을 구한다.

3.4 상호 악의적인 모델

S와 C가 모두 악의적인 모델은 악의적인 클라이언트 모델과 악의적인 서버 모델을 결합하였다.

4. 결정적(Deterministic) Private Matching 프로토콜

4.1 DPM-Semi-Honest 프로토콜

FNP는 Semi-Honest 모델에서 X와 Y의 원소들의 정의역이 큰 경우 높은 확률로 정확한 $X \cap Y$를 계산할 수 있는 프로토콜을 제안하였다[1]. 하지만 FNP 방법은 확률적이라는 한계를 지니고 있기 때문에 정의역이 충분히 큰 경우에도 계산된 $X \cap Y$ 값은 부정확할(incorrect) 가능성을 내포하고 있다. 우리는 이들의 방법을 개선하여 프로토콜 수행 결과 얻은 $X \cap Y$ 값의 정확성을 보장하는 Deterministic PM-Semi-Honest(이하 DPM) 프로토콜을 제안한다.

DPM 프로토콜은 기본적으로 FNP 방법을 따르되 다항식으로 FNP에서 약간 변형된 다음과 같은 $P(y)$를 사용한다. (이 때, 최고차항은 항상 1이다.)

$$P(y) = (y - x_1)(y - x_2)\cdots(y - x_{k_C}) = \sum_{u=0}^{kc} \alpha_u y^u$$

그리고 프로토콜 수행에서는 S가 $P(y)$의 암호화된 계수들을 받은 후 Y의 원소 y_i에 대하여 하나의 랜덤값 r_i를 선택하여 $Enc_{pk}(r_i \cdot P(y_i)+y_i)$를 계산하는 대신, 두 개의 서로 다른 랜덤값 $r_{i,1}$, $r_{i,2}$ $(r_{i,1} \neq r_{i,2})$를 선택하여 계산한 $Enc_{pk}(r_{i,1} \cdot P(y_i) + y_i)$, $Enc_{pk}(r_{i,2} \cdot P(y_i) + y_i)$를 C에게 보낸다. C는 이를 복호화하여 두 값이 서로 같으면 $P(y_i)$=0이 되어 y_i가 $X \cap Y$의 원소임을 알 수 있다. 두 값이 서로 다르면 두 랜덤 $r_{i,1}$, $r_{i,2}$가 서로 다르다는 가정에 의해 y_i는 $X \cap Y$의 원소가 될 수 없다. (증명은 Section 5.1의 Theorem 1 참조) 프로토콜의 자세한 동작은 그림 2와 같다.

DPM 프로토콜은 FNP와 같이 균등 분배 함수를 사용한 경우로도 확장 가능하다. 이 경우 서버와 클라이언트가 서로 공유하는 균등 분배 함수 F를 이용하여 각각 자신의 집합을 최대 M개의 원소를 가지는 부분집합으로 분할한 다음 분할된 부분집합에 대하여 별도로 DPM-Semi-Honest를 수행한다. 이 때, 사용되는 다항식의 차수를 모두 M으로 고정하기 위해 차수가 M보다 작은 l-차 다항식에는 y^{M-l}을 곱한다. 균등 분배 함수를 사용하면 다항식의 차수를 모두 M으로 제한할 수 있으므로 다항식의 전개 시간을 줄일 수 있다. 프로토콜은 그림 3과 같다.

4.2 DPM-Malicious Client 프로토콜

FNP와 유사하게 DPM의 경우도 악의적인 클라이언트 모델로 확장할 수 있다. DPM에서는 최고차항을 1로 고정함으로써 다항식이 무한 개의 근을 가지지 못하게 한다. 이 방법을 사용하면 FNP에서 다항식의 상수항을

Protocol DPM-Semi-Honest

Input : C 의 집합 $X = \{x_1, ..., x_{k_c}\}$
S 의 집합 $Y = \{y_1, ..., y_{k_s}\}$

Output : $Result = X \cap Y$

1. 클라이언트 C
   (a) X 의 원소를 근으로 가지는 다항식 계산
   $$P(y) = (y - x_1)(y - x_2)\cdots(y - x_{k_c}) = \sum_{u=0}^{k_c} \alpha_u y^u$$
   (b) $P(y)$의 계수를 암호화하여
   $\{Enc_{pk}(\alpha_0), ..., Enc_{pk}(\alpha_{k_c})\}$ S 에게 전송.

2. 서버 S
   (a) 집합 $Y = \{y_1, ..., y_{k_s}\}$ 의 모든 원소 y_i 에
   2개의 다른 랜덤값 $r_{i,1}$과 $r_{i,2}$를 선택하여
   $$\begin{pmatrix} e_{i,1} \\ e_{i,2} \end{pmatrix} := \begin{pmatrix} Enc_{pk}(r_{i,1} \cdot P(y_i) + y_i) \\ Enc_{pk}(r_{i,2} \cdot P(y_i) + y_i) \end{pmatrix}$$
   암호문 쌍 (pair) 계산
   (b) $\{(e_{i,1}, e_{i,2}) \mid 1 \le i \le k_s\}$를 permute하여
   C 에게 전송

3. 클라이언트 C
   (a) $Result := \phi$
   (b) k_s 개의 암호문쌍을 개인키 (private key)로 복호화 (decryption)
   $(d_{i,1}, d_{i,2}) := (Dec_{sk}(e_{i,1}), Dec_{sk}(e_{i,2}))$
   (c) $Candidate := \{(d_{i,1}, d_{i,2}) \mid 1 \le i \le k_s\}$ 중에서
   $d_{i,1} = d_{i,2}$이면 $Result := Result \cup \{d_{i,1}\}$

그림 2 DPM-Semi-Honest 프로토콜

Protocol DPM-Semi-Honest (균등분배함수)

Input : C 의 집합 $X = \{x_1, ..., x_{k_c}\}$
S 의 집합 $Y = \{y_1, ..., y_{k_s}\}$

Output : $Result = X \cap Y$

1. 클라이언트 C
   (a) 균등 분배 함수 F를 선택하여 S 에게 전송
   (b) F를 이용하여 $X = \{x_1, ..., x_{k_c}\}$ 분배(partition)
   최대 M 개의 원소를 갖는 B 개의 부분집합
   $X_1, X_2, ..., X_B$ 생성
   (c) 집합 X_h의 모든 원소를 근으로 갖는 다항식 계산
   $$P_h(y) = (y - x_{h,1})(y - x_{h,2})\cdots(y - x_{h,l})y^{M-l}$$
   $$= \sum_{u=0}^{M} \alpha_{h,u} y^u = y^M + \sum_{u=0}^{M-1} \alpha_{h,u} y^u$$
   (d) 다항식 $P_1, ..., P_B$의 계수들을 암호화하여
   S 에게 전송
   $\{Enc_{pk}(\alpha_{h,0}), ..., Enc_{pk}(\alpha_{h,M-1}) \mid 1 \le h \le B\}$

2. 서버 S

(a) $Y = \{y_1, ..., y_{k_s}\}$에 균등 분배 함수 F를 적용하여 B 개의 부분집합 $Y_1, Y_2, ..., Y_B$ 생성

(b) Y의 각각의 원소 $y_j \in Y_h$에 대하여 서로 다른 랜덤값 $r_{j,1}$ 과 $r_{j,2}$ 선택

$$\begin{pmatrix} e_{j,1} \\ e_{j,2} \end{pmatrix} := \begin{pmatrix} Enc_{pk}(r_{j,1} \cdot P_h(y_j) + y_j) \\ Enc_{pk}(s_{j,2} \cdot P_h(y_j) + y_j) \end{pmatrix}$$

암호문 쌍 (pair) 계산

(b) $\{(e_{j,1}, e_{j,2}) \mid 1 \le j \le k_s\}$ 를 permute하여 C에게 보냄

3. 클라이언트 C

(a) $Result := \phi$

(b) k_s 개의 암호문 쌍을 개인키 (private key)로 복호화 (decryption)

$(d_{j,1}, d_{j,2}) := (Dec_{sk}(e_{j,1}), Dec_{sk}(e_{j,2}))$

(c) $Candidate := \{(d_{i,1}, d_{i,2}) \mid 1 \le i \le k_s\}$ 중에서 $d_{j,1} = d_{j,2}$ 이면 $Result = Result \cup \{d_{j,1}\}$

그림 3 DPM-Semi-Honest 프로토콜(균등분배함수 사용)

1로 바꾸기 위해 각항에 상수를 곱하는 연산을 없앨 수 있을 뿐 아니라 균등 분배 함수를 사용하여 각각의 다항식의 차수를 M차로 제한하는 경우에도 확장 가능하다.

악의적인 클라이언트 모델에서 다항식이 무한개의 근을 가지지 못하도록 하는 프로토콜은 DPM-Semi-Honest 프로토콜(그림 2)의 step 1에서 클라이언트가 서버에게 다항식의 계수를 보낼 때 최고차항의 계수를 제외하는 점과 step 2에서 서버가 받은 다항식의 최고차항의 계수를 미리 1로 가정하고 계산하는 점을 제외하고는 DPM-Semi-Honest와 동일하다. (DPM-Semi-Honest와 유사하므로 여기서는 생략한다.)

악의적인 클라이언트 모델에서 균등 분배 함수를 사용한 프로토콜도 균등 분배 함수를 사용한 DPM-Semi-Honest의 경우(그림 3)와 최고차항을 처리하는 부분을 제외하고 동일하다. 특히 이 경우 다항식의 크기를 M으로 고정하기 위해 차수가 M보다 작은 l-차 다항식에는 y^{M-l}을 곱하여도 최고차항의 계수는 1로 고정할 수 있으므로 다항식 하나를 사용한 방법을 직접적으로 확장할 수 있다. (FNP의 경우 상수항을 1로 고정하였는데 이 경우 y^{M-l}를 곱하면 상수항이 없어지므로 근의 개수를 확인하기 위해 별도로 Cut-and-Choose 방법을 사용하였다.) 악의적인 클라이언트 모델에서 균등 분배 함수를 사용한 프로토콜 DPM-Malicious-Client는 그림 4와 같다.

악의적인 클라이언트 모델에서 DPM 방법은 최고차항의 계수만 1로 고정하였으므로 DPM-Semi-Honest와 마찬가지로 결과의 정확성을 보장할 수 있다.

4.2 DPM-Malicious-Server 프로토콜

Protocol DPM-Malicious-Client

Input : C의 집합 $X = \{x_1, ..., x_{k_c}\}$

S의 집합 $Y = \{y_1, ..., y_{k_s}\}$

Output : $Result = X \cap Y$

1. 클라이언트 C

(a) 균등 분배 함수 F를 선택하여 S에게 전송

(b) F를 이용하여 $X = \{x_1, ..., x_{k_c}\}$ 분배(partition) 최대 M개의 원소를 갖는 B개의 부분집합 $X_1, X_2, ..., X_B$ 생성

(c) 집합 X_h의 모든 원소를 근으로 갖는 다항식 계산

$$P_h(y) = (y - x_{h,1})(y - x_{h,2}) \cdots (y - x_{h,l}) y^{M-l}$$

$$= \sum_{u=0}^{M} \alpha_{h,u} y^u = y^M + \sum_{u=0}^{M-1} \alpha_{h,u} y^u$$

(d) 다항식 $P_1, ..., P_B$의 최고차항을 제외한 나머지 계수들을 암호화하여 S에게 전송

$\{Enc_{pk}(\alpha_{h,0}), ..., Enc_{pk}(\alpha_{h,M-1}) \mid 1 \le h \le B\}$

2. 서버 S

(a) $Y = \{y_1, ..., y_{k_s}\}$에 균등 분배 함수 F를 적용하여 B 개의 부분집합 $Y_1, Y_2, ..., Y_B$ 생성

(b) Y의 각각의 원소 $y_j \in Y_h$에 대하여 서로 다른 랜덤값 $r_{j,1}$ 과 $r_{j,2}$ 선택

$$\begin{pmatrix} e_{j,1} \\ e_{j,2} \end{pmatrix} := \begin{pmatrix} Enc_{pk}(r_{j,1} \cdot P_h(y_j) + y_j) \\ Enc_{pk}(s_{j,2} \cdot P_h(y_j) + y_j) \end{pmatrix}$$

(c) $\{(e_{j,1}, e_{j,2}) \mid 1 \le j \le k_s\}$ 를 C에게 보냄

3. 클라이언트 C

(a) $Result := \phi$

(b) k_s 개의 암호문 쌍을 개인키 (private key)로 복호화 (decryption)

$(d_{j,1}, d_{j,2}) := (Dec_{sk}(e_{j,1}), Dec_{sk}(e_{j,2}))$

(c) $d_{j,1} = d_{j,2}$ 이면 $Result = Result \cup \{d_{j,1}\}$

그림 4 DPM-Malicious-Client 프로토콜

악의적인 서버 모델에서 DPM 방법은 FNP 방법을 개선하여 계산량을 줄일 수 있는 방법을 제안한다. 기존의 FNP 방법은 랜덤 오라클을 이용하여 이상적인 TTP 모델에서 불가능한 공격을 차단할 수 있는 방법을 제안하고 이를 증명하였다. 그러나 FNP 방법은 클라이언트가 서버로부터 받은 각각의 값에 대해 자신의 집합의 모든 원소를 대입하여 서버의 연산을 재구성하여야 하므로 비효율적이었다. DPM 방법은 클라이언트가 서버로부터 받은 값에서 선택적으로 서버의 연산을 재구성하고, 재구성된 값에서 직접 일치하는 값을 얻을 수 있으므로 FNP보다 효율적이다.

악의적인 서버 모델에서 DPM 프로토콜은 다음과 같

이 동작한다. 먼저 클라이언트는 $P(y)$의 계수를 암호화하여 서버에게 보낸다. 서버는 Y의 원소 y_i에 대하여 2개의 랜덤 값 $s_{i,1}, s_{i,2}$를 생성하여 $r_{i,1} := H_1(s_{i,1})$, $r_{i,2} := H_1(s_{i,2})$를 구한다. 그리고 $r_{i,1}, r_{i,2}$에서 $r_{i,1} = u_{i,1} \| v_{i,1}$, $r_{i,2} = u_{i,2} \| v_{i,2}$를 추출하여

$$\begin{pmatrix} e_{i,1} \\ h_{i,1} \end{pmatrix} := \begin{pmatrix} Enc_{pk}(u_{i,1} \cdot P(y_i) + s_{i,1} \| y_i) \\ H_2(v_{i,1}, y_i) \end{pmatrix}$$

$$\begin{pmatrix} e_{i,2} \\ h_{i,2} \end{pmatrix} := \begin{pmatrix} Enc_{pk}(u_{i,2} \cdot P(y_i) + s_{i,2} \| y_i) \\ H_2(v_{i,2}, y_i) \end{pmatrix}$$

를 계산한다. 이때 암호화 시스템에 필요한 랜덤 값은 $r_{i,1}, r_{i,2}$을 사용한다. (여기서 $e_{i,1}, e_{i,2}$는 $P(y_i) = 0$일 때 $s_{i,1}, s_{i,1}$과 y_i가 함께 전송된다.)

클라이언트는 서버로부터 각각의 y_i에 대하여 $(e_{i,1}, h_{i,1}), (e_{i,2}, h_{i,2})$쌍을 받아 $e_{i,1}, e_{i,2}$값을 복호화하여 $\tilde{s}_{i,1} \| \tilde{y}_{i,1} := Dec_{sk}(e_{i,1})$, $\tilde{s}_{i,2} \| \tilde{y}_{i,2} := Dec_{sk}(e_{i,2})$를 얻는다. $y_i \in X \cap Y$ ($P(y_i) = 0$)이면 $\tilde{y}_{i,1} = \tilde{y}_{i,2}$되므로 대우관계에 의하여 $\tilde{y}_{i,1} \neq \tilde{y}_{i,2}$이면 $y_i \notin X \cap Y$이 된다. $\tilde{y}_{i,1} = \tilde{y}_{i,2}$인 경우 $\tilde{s}_{i,1}, \tilde{s}_{i,2}$값을 이용하여 서버가 한 연산을 똑같이 계산한다(이하 재구성). 서버가 암호화 연산에 사용한 랜덤 값도 재구성된다.

재구성을 통해 얻은 $(\tilde{e}_{i,1}, \tilde{h}_{i,1}), (\tilde{e}_{i,2}, \tilde{h}_{i,2})$가 서버로부터 받은 $(e_{i,1}, h_{i,1}), (e_{i,2}, h_{i,2})$값과 같으면 ($(\tilde{e}_{i,1}, \tilde{e}_{i,2})$과 $(e_{i,1}, e_{i,2})$는 암호화된 상태에서 비교) $\tilde{y}_{i,1}$을 PM에 포함한다.

위와 같은 연산을 통하여 PM을 계산하면 악의적인 서버는 클라이언트를 속이기 어렵다. 왜냐하면 Semantically secure한 준동형 암호화 시스템에서는 연산과정에 서로 다른 랜덤 값이 적용되므로 같은 입력값도 암호화하면 다른 출력값으로 표현되기 때문이다. 즉 악의적인 서버가 어떤 값 y'를 PM으로 판단하게 하기 위하여 자신의 원소를 프로토콜에 따라 그대로 계산하지 않고 다른 형태의 식으로 계산하면 연산 과정에 의해 암호화한 상태에서의 값이 달라지게 된다. 프로토콜에 따라 y'을 직접 Y의 원소로 대입하는 것은 이상적인 TTP 모델에서도 가능하므로 공격의 형태에서 제외한다. FNP와 유사하게 이 프로토콜에서 가능한 공격은 이상적인 TTP 모델에서도 가능함을 증명할 수 있다. (증명은 Section 5를 참조) 자세한 프로토콜은 아래와 같다. 균등분배 함수를 사용하는 경우는 쉽게 확장 가능하므로 여기서는 생략한다.

Protocol DPM-Malicious-Server
Input : C의 집합 $X = \{x_1, ..., x_{k_c}\}$
S의 집합 $Y = \{y_1, ..., y_{k_s}\}$
Output : $Result = X \cap Y$
Random Oracle : H_1, H_2

1. 클라이언트 C
   (a) X의 원소를 근으로 가지는 다항식 계산
   $$P(y) = (y - x_1)(y - x_2)\cdots(y - x_{k_c}) = \sum_{u=0}^{k_c} \alpha_u y^u$$
   (c) $P(y)$의 계수를 암호화하여 S에게 전송

2. 서버 S
   (a) 데이터 $Y = \{y_1, ..., y_{k_s}\}$의 모든 원소 y_i에
      (a-1) 2개의 다른 랜덤값 $s_{i,1}, s_{i,2}$를 선택
      $r_{i,1} := H_1(s_{i,1}), (r_{i,1} = u_{i,1} \| v_{i,1})$
      $r_{i,2} := H_1(s_{i,2}), (r_{i,2} = u_{i,2} \| v_{i,2})$ 계산
      (이하 S의 암호시스템에 사용되는 랜덤값은 $r_{i,1}, r_{i,2}$로 고정)
      (a-2) 암호시스템의 준동형 성질을 이용하여 다음을 계산
      $$\begin{pmatrix} e_{i,1} \\ h_{i,1} \end{pmatrix} := \begin{pmatrix} Enc_{pk}(u_{i,1} \cdot P(y_i) + s_{i,1} \| y_i) \\ H_2(v_{i,1}, y_i) \end{pmatrix}$$
      $$\begin{pmatrix} e_{i,2} \\ h_{i,2} \end{pmatrix} := \begin{pmatrix} Enc_{pk}(u_{i,2} \cdot P(y_i) + s_{i,2} \| y_i) \\ H_2(v_{i,2}, y_i) \end{pmatrix}$$
   (b) $\{(e_{i,1}, h_{i,1}), (e_{i,2}, h_{i,2}) \mid 1 \le i \le k_s\}$를 C에게 전송

3. 클라이언트 C
   (a) $Result := \phi$
   (b) S로부터 받은 암호문 $(e_{i,1}, h_{i,1}), (e_{i,2}, h_{i,2})$에서 $e_{i,1}, e_{i,2}$를 복호화하여 다음을 계산
   $\tilde{s}_{i,1} \| \tilde{y}_{i,1} := Dec_{sk}(e_{i,1})$, $\tilde{s}_{i,2} \| \tilde{y}_{i,2} := Dec_{sk}(e_{i,2})$
   (c) $(\tilde{y}_{i,1} = \tilde{y}_{i,2}) \wedge (\tilde{y}_{i,1} \in X)$이면
      (c-1) $\tilde{s}_{i,1}, \tilde{s}_{i,2}$를 H_1에 대입하여
      $\tilde{r}_{i,1} := H_1(\tilde{s}_{i,1}), (\tilde{r}_{i,1} = \tilde{u}_{i,1} \| \tilde{v}_{i,1})$,
      $\tilde{r}_{i,2} := H_1(\tilde{s}_{i,2}), (\tilde{r}_{i,2} = \tilde{u}_{i,2} \| \tilde{v}_{i,2})$ 계산
      (c-2) $\tilde{h}_{i,1} = H_2(\tilde{v}_{i,1}, \tilde{y}_{i,1})$ $\tilde{h}_{i,2} = H_2(\tilde{v}_{i,2}, \tilde{y}_{i,1})$
      (c-3) S의 암호화 연산 재구성
      (랜덤값은 $\tilde{r}_{i,1}, \tilde{r}_{i,2}$로 고정)
      $\tilde{e}_{i,1} = Enc_{pk}(\tilde{u}_{i,1} \cdot P(\tilde{y}_{i,1}) + \tilde{s}_{i,1} \| \tilde{y}_{i,1})$
      $\tilde{e}_{i,2} = Enc_{pk}(\tilde{u}_{i,2} \cdot P(\tilde{y}_{i,1}) + \tilde{s}_{i,2} \| \tilde{y}_{i,1})$
      (c-4) $\begin{pmatrix} \tilde{e}_{i,1} \\ \tilde{h}_{i,1} \end{pmatrix} = \begin{pmatrix} e_{i,1} \\ h_{i,1} \end{pmatrix} \wedge \begin{pmatrix} \tilde{e}_{i,2} \\ \tilde{h}_{i,2} \end{pmatrix} = \begin{pmatrix} e_{i,2} \\ h_{i,2} \end{pmatrix}$ 이면
      $Result := Result \cup \{\tilde{y}_{i,1}\}$

그림 5 DPM-Malicious-Server 프로토콜

4.4 DPM-Malicious-Client/Server 프로토콜

S와 S가 모두 악의적인 모델의 경우는 DPM-Malicious-Client 프로토콜과 DPM-Malicious-Server 프로토콜을 결합하면 확장 가능하다. 다항식의 최고차항을 1로 고정하여 클라이언트의 악의적인 공격을 제한하면서 악의적인 서버 모델에서와 같이 랜덤 오라클을 사용하면 서로 악의적인 모델에서 PM을 계산할 수 있다. 자세한 프로토콜은 아래와 같다. 균등 분배 함수를 사용하지 않는 방법은 DPM-Malicious-Server 프로토콜(그림 5)의 step 1에서 클라이언트가 서버에게 다항식의 계수를 보낼 때 최고차항의 계수를 제외하는 점과 step 2에서 서버가 받은 다항식의 최고차항의 계수를 미리 1로 가정하고 계산하는 점을 제외하고는 동일하므로 여기서는 생략한다.

Protocol DPM-Malicious-Client/Server

Input : C 의 집합 $X = \{x_1, ..., x_{k_c}\}$

S 의 집합 $Y = \{y_1, ..., y_{k_s}\}$

Output : $Result = X \cap Y$

Random Oracle : H_1, H_2

1. 클라이언트 C

(a) 균등 분배 함수 F를 선택하여 S에게 전송

(b) F를 이용하여 $X = \{x_1, ..., x_{k_c}\}$ 분배(partition) 최대 M개의 원소를 갖는 B개의 부분집합 $X_1, X_2, ..., X_B$ 생성

(c) 집합 X_h의 모든 원소를 근으로 갖는 다항식 계산

$$P_h(y) = (y - x_{h,1})(y - x_{h,2}) \cdots (y - x_{h,l}) y^{M-l} = \sum_{u=0}^{M} \alpha_{h,u} y^u = y^M + \sum_{u=0}^{M-1} \alpha_{h,u} y^u$$

(d) 다항식 $P_1, ..., P_B$의 최고차항을 제외한 나머지 계수들을 암호화하여 S에게 전송

$\{Enc_{pk}(\alpha_{h,0}), ..., Enc_{pk}(\alpha_{h,M-1}) \mid 1 \le h \le B\}$

2. 서버 S

(a) $Y = \{y_1, ..., y_{k_s}\}$에 균등 분배 함수 F를 적용하여 B개의 부분집합 $Y_1, Y_2, ..., Y_B$ 생성

(b) Y의 각각의 원소 $y_j \in Y_h$에 대하여

(b-1) 2개의 다른 랜덤값 $s_{j,1}, s_{j,2}$를 선택

$r_{j,1} := H_1(s_{j,1}), (r_{j,1} = u_{j,1} \parallel v_{j,1})$

$r_{j,2} := H_1(s_{j,2}), (r_{j,2} = u_{j,2} \parallel v_{j,2})$ 계산

(이하 S의 암호시스템에 사용되는 랜덤값은 $r_{j,1}, r_{j,2}$로 고정)

(b-2) 암호시스템의 준동형 성질을 이용하여 다음을 계산

$$\begin{pmatrix} e_{j,1} \\ h_{j,1} \end{pmatrix} := \begin{pmatrix} Enc_{pk}(u_{j,1} \cdot P_h(y_j) + s_{j,1} \parallel y_j) \\ H_2(v_{j,1}, y_j) \end{pmatrix}$$

$$\begin{pmatrix} e_{j,2} \\ h_{j,2} \end{pmatrix} := \begin{pmatrix} Enc_{pk}(u_{j,2} \cdot P_h(y_j) + s_{j,2} \parallel y_j) \\ H_2(v_{j,2}, y_j) \end{pmatrix}$$

(c) $\{(e_{j,1}, h_{j,1}), (e_{j,2}, h_{j,2}) \mid 1 \le j \le k_s\}$를 C에게 전송

3. 클라이언트 C

(a) $Result := \phi$

(b) S 로부터 받은 암호문 $(e_{j,1}, h_{j,1}), (e_{j,2}, h_{j,2})$ 에서 $e_{j,1}, e_{j,2}$를 복호화하여 다음을 계산

$\tilde{s}_{j,1} \parallel \tilde{y}_{j,1} := Dec_{sk}(e_{j,1})$, $\tilde{s}_{j,2} \parallel \tilde{y}_{j,2} := Dec_{sk}(e_{j,2})$

(c) $(\tilde{y}_{j,1} = \tilde{y}_{j,2}) \wedge (\tilde{y}_{j,1} \in X)$ 이면

(c-1) $\tilde{s}_{j,1}, \tilde{s}_{j,2}$를 H_1에 대입하여

$\tilde{r}_{j,1} := H_1(\tilde{s}_{j,1}), (\tilde{r}_{j,1} = \tilde{u}_{j,1} \parallel \tilde{v}_{j,1})$,

$\tilde{r}_{j,2} := H_1(\tilde{s}_{j,2}), (\tilde{r}_{j,2} = \tilde{u}_{j,2} \parallel \tilde{v}_{j,2})$ 계산

(c-2) $\tilde{h}_{j,1} = H_2(\tilde{v}_{j,1}, \tilde{y}_{j,1})$ $\tilde{h}_{j,2} = H_2(\tilde{v}_{j,2}, \tilde{y}_{j,1})$

(c-3) S의 암호화 연산 재구성 (랜덤값은 $\tilde{r}_{j,1}, \tilde{r}_{j,2}$로 고정)

$\tilde{e}_{j,1} = Enc_{pk}(\tilde{u}_{j,1} \cdot P_h(\tilde{y}_{j,1}) + \tilde{s}_{j,1} \parallel \tilde{y}_{j,1})$

$\tilde{e}_{j,2} = Enc_{pk}(\tilde{u}_{j,2} \cdot P_h(\tilde{y}_{j,1}) + \tilde{s}_{j,2} \parallel \tilde{y}_{j,1})$

(c-4) $\begin{pmatrix} \tilde{e}_{j,1} \\ \tilde{h}_{j,1} \end{pmatrix} = \begin{pmatrix} e_{j,1} \\ h_{j,1} \end{pmatrix} \wedge \begin{pmatrix} \tilde{e}_{j,2} \\ \tilde{h}_{j,2} \end{pmatrix} = \begin{pmatrix} e_{j,2} \\ h_{j,2} \end{pmatrix}$ 이면

$Result := Result \cup \{\tilde{y}_{j,1}\}$

그림 6 DPM-Malicious-Client/Server 프로토콜

5. 분 석

5.1 정확성(Correctness)

FNP의 방법과 Kissner 등의 방법은 확률적으로 (probablistic) PM을 계산하였는데, DPM은 S와 C가 Semi-Honest 한 경우 결정적 (deterministic)으로 PM을 계산할 수 있다.

Theorem 1. S와 C가 모두 Semi-Honest 한 경우 프로토콜 DPM-Semi-Honest의 결과는 항상 $X \cap Y$이다.

증명. $Result = \{d_{i,1} \mid d_{i,1} = d_{i,2}, 1 \le i \le k_S\}$ 이므로 $d_{i,1} = d_{i,2}$이면 $d_{i,1} \in X \cap Y$이고 그 역도 성립함을 보이면 $Result = X \cap Y$이다.

$$\begin{aligned} & d_{i,1} = d_{i,2} \\ \Leftrightarrow\ & r_{i,1} \cdot P(y_i) + y_i = r_{i,2} \cdot P(y_i) + y_i \\ \Leftrightarrow\ & (r_{i,1} - r_{i,2}) P(y_i) = 0 \\ \Leftrightarrow\ & P(y_i) = 0 (\because r_{i,1} \neq r_{i,2}) \\ \Leftrightarrow\ & d_{i,1} = y_i \in X \\ \Leftrightarrow\ & d_{i,1} \in X \cap Y (\because y_i \in Y) \end{aligned}$$

또한 DPM-Semi-Honest 프로토콜은 C가 프로토콜의 결과값 $X \cap Y$이 자신의 데이타에 포함되는지 확인

할 필요가 없다. 즉 $d_{i,1} = d_{i,2}$ 이면 $d_{i,1} \in X$ 임이 보장되기 때문에 별도로 X에 포함되는지 여부를 확인할 필요가 없다.

Malicious-Client 및 Malicious-Server의 경우도 DPM-Semi-Honest 프로토콜을 기본 스킴으로 사용하기 때문에 동일한 정확성을 가진다.

5.2 보안성(Security)

Section 2.4에서 정의한 보안성을 고려할 때 PM-Semi-Honest 및 PM-Malicious-Client 환경에서 보안성은 위의 각 프로토콜에서 사용되는 암호시스템이 Semantically Secure한 특성으로 인해 쉽게 설명된다. 여기에서는 Malicious-Server 환경에서 클라이언트와 서버의 보안성을 살펴본다. Malicious-Server 환경에서 서버의 보안성은 $P(y) \neq 0$ 일 때 클라이언트가 (e,h)쌍과 랜덤값을 구분할 수 없기 때문에 유지된다. 클라이언트의 보안성은 랜덤 오라클 모델에서 증명할 수 있다. 클라이언트를 속일 수 있는 이상적인 모델에서 Malicious-Server의 존재 문제를 정상모델에서 Malicious-Server의 존재 문제로 리덕션(reduction) 한다. 즉 우리는 만약 정상모델에서 Malicious-Server가 존재한다고 가정한다면 그런 서버와 동일하게 클라이언트를 속일 수 있는 이상적인 모델에서의 서버를 만들 수 있음을 보인다. 그러나 가정으로부터 이상적인 모델에서 Malicious-Server가 존재하지 않기 때문에 정상모델에서 Malicious-Server도 존재하지 않게 된다.

Theorem 2. Malicious-Server 환경에서 동작하는 모든 서버 S와 동일한(indistinguishable) 결과를 내는 이상적인 모델(TTP)에서의 서버 S^*가 존재한다.

증명.

가. 이상적인 모델에서의 서버 S^*가 차수가 k_c인 임의의 다항식 $P(y)$을 만들고 다항식의 계수를 암호화하여 정상모델의 서버 S에게 보낸다. 이때 서버 S는 암호화된 형태로 다항식을 받게 되므로 임의의 다항식과 C의 다항식을 구분할 수 없다.

나. 서버 S^*는 프로토콜 수행간 S가 랜덤오라클 H_1, H_2에 요청한 값들을 모두 저장한다. $\tilde{S}$을 H_1에 대한 입력값의 집합, $\tilde{Y}$을 H_2에 대한 입력값의 집합이라 한다.

다. 이상적인 모델에서의 서버 S^*가 실제 서버 S의 모든 출력값 $\{(e_{i,1},h_{i,1}),(e_{i,2},h_{i,2}) \mid 1 \le i \le k_s\}$가 $\tilde{s}_{i,1},\tilde{s}_{i,2} \in \tilde{S}$ 이고 $\tilde{y}_{i,1},\tilde{y}_{i,2} \in \tilde{Y}$인지 여부를 확인한다. 출력값이 $\tilde{s}_{i,1},\tilde{s}_{i,2} \in \tilde{S}$와 $\tilde{y}_{i,1},\tilde{y}_{i,2} \in \tilde{Y}$를 만족하면 $y := \tilde{y}$, 만족하지 않으면 $y := \perp$이라 한다.

라. S^*가 다)에서 계산된 y를 TTP에게 보내면, 결국 S^*는 서버 S와 동일한 결과를 내게 된다.

DPM은 FNP 프로토콜과 동일한 수준의 안전도를 가진다. DPM에서 결정적(deterministic) 특성을 충족하기 위해 하나의 데이타에 2개의 암호문이 생성되지만 암호시스템의 Semantically Secure한 특성으로 인해 도청자가 중간에 암호문 쌍을 가로채더라도 복호화 했을 때 같은 값인지 여부를 알 수 없다.

5.3 복잡도(Complexity)

5.3.1 공간(Space) 복잡도

각각의 원소가 n비트로 표현될 때 FNP에서 높은 확률(에러률 $\varepsilon = 1-(1-k_c/2^n)^{k_s}$)로 정확한 PM을 계산하기 위해서는 n이 충분히 커야 한다. 이들의 프로토콜에서는 각각의 원소를 저장하는데 공간이 많이 필요하거나 프로토콜 수행 전 별도의 전처리 과정이 필요하다.

DPM은 결정적(deterministic)으로 PM을 계산하기 때문에 원소를 $n = \log(\max(k_C,k_S)) + 1$ 비트 만을 이용하여 표현할 수 있다. 예를 들어 $\max(k_s,k_c) = 1024$ 일 때 DPM은 원소를 $n = 11$비트로 표현하여 정확한 PM을 얻을 수 있지만, FNP에서는 $n = 20$비트일 때 $\varepsilon = 0.63$, $n = 30$비트일 때 $\varepsilon = 9.76\times10^{-4}$의 에러확률로 PM을 얻게 된다. 즉, FNP는 n을 크게 하여 에러 확률을 줄일 수 있지만 확률적이라는 한계를 지닌다.

5.3.2 시간(Time) 복잡도

프로토콜을 수행하는데 소요되는 시간은 연산시간과 통신량으로 구분하여 분석할 수 있다. 먼저 연산 시간으로 C가 데이타를 다항식으로 표현하여 전개하는 시간 $O(k_c^2)$과 S가 데이타를 다항식에 대입하는데 걸리는 시간은 $O(k_s k_c)$로 FNP와 동일하다. (균등분배함수, Horner's rule 등을 적용하는 경우 다항식에 데이타를 대입하는 시간은 $O(k_c + k_s \ln\ln k_c)$가 된다.)

통신량을 살펴보면 DPM 프로토콜은 모두 S가 C에게 데이타 별로 2개의 암호문을 보내기 때문에 FNP의 방법에 비해 2배의 통신량이 필요하다. 그러나 DPM의 결정적인 성질로 인하여 DPM-Malicious-Client 는 균등 분배함수를 한번만 사용하여 통신 횟수를 FNP의 $1/L$로 줄였으며, DPM-Malicious-Server는 C가 일치하는 (e,h)쌍을 찾기 위해 서버의 연산을 재구성하는 연산을 한번만 수행하기 때문에 실제 연산량이 크게 줄어든다. 따라서 DPM은 S가 C에게 보내는 단위 원소당 통신량이 증가하지만 단위 데이타의 크기를 줄일 수 있을 뿐 아니라, 결정적 특성을 갖기 때문에 악의적인 공격자 모델에서의 프로토콜과 같이 연산 횟수가 줄어들어 전체적인 프로토콜 수행시간은 빨라질 수 있다.

6. 결 론

우리는 FNP의 PM 계산방법을 결정적(deterministic)인 방법으로 개선하고 이를 악의적인 공격자 상황으로 확대한 DPM 프로토콜을 제안하였다. DPM 프로토콜은 FNP에 비해 단위 데이타의 크기를 줄일 수 있었으며, 결정적인 결과를 산출하는 특성을 가지고 있다. 현재 DPM 프로토콜은 참여자가 둘인 경우를 가정하였는데 이를 효율적인 다자간의 계산으로 확장하는 것은 이후 좋은 연구 주제가 될 것이다.

참 고 문 헌

[1] Michael Freedman, Kobi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In Proc. Of Eurocrypt, pages 1-19, 2004.

[2] Lea Kissner and Dawn Song. Privacy-preserving set operations. In Proc. Of Crypto, pages 241-257, 2005.

[3] Yaping Li, J. D. Tygar and Joseph M. Hellerstein. Private Matching. In Computer Security in the 21st Century, eds. pages 25-50, 2005.

[4] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Advances in Cyptology-EUROCYTPT '99, pages 223-238, 1999.

[5] Ivan Damgard and Mads Jurik. A generalization, a simplification and some applications of Paillier's probabilistic public-key system. In 4-th International Workshop on practice and Theory in Public Key Cryptosystems(PKC 2001), pages 13-15, 2001.

[6] O. Goldreich. Foundations of Cryptography, volume 2, chapter 7. Cambridge university press, 2004.

[7] S. Ajmani, R. Morris, and B.Liskov. A trusted third-party computation service. Technical Report MIT-LCS-TR-847, MIT, 2001.

홍 정 대

1993년 육군사관학교 졸업. 2005년 서울대학교 컴퓨터 공학석사. 2006년~현재 서울대학교 컴퓨터공학부 박사과정. 관심분야는 암호학, 컴퓨터 이론

김 진 일

2005년 서울대학교 전기공학부 학사. 2007년 서울대학교 전기컴퓨터공학부 석사 2007년~현재 서울대학교 전기컴퓨터공학부 박사과정. 관심분야는 암호학, 컴퓨터 이론

천 정 희

1997년 2월 한국과학기술원 수학과 박사 1997년 3월~2000년 1월 한국전자통신연구원 선임연구원. 2000년 1월~2000년 12월 Brown 대학 박사후 연구원. 2000년 12월~2003년 2월 한국정보통신대학교 공학부 조교수. 2003년 3월~현재 서울대학교 수리과학부 조교수, 부교수. 관심분야는 계산 수론, 암호론, 응용 암호

박 근 수

정보과학회논문지 : 시스템 및 이론 제 34 권 제 5 호 참조