정확성을 보장하는 결정적 Private Matching

홍정대[0,1] 김진일[1] 천정희[2] 박근수[1]
[1] 서울대학교 컴퓨터공학부
[2] 서울대학교 수리과학부
{jdhong, jikim, kpark}@theory.snu.ac.kr
{jhcheon}@snu.ac.kr

Deterministic Private Matching with Perfect Correctness

Jeongdae Hong[0,1], Jinil Kim[1], Jung Hee Cheon[2], and Kunsoo Park[1]
[1] School of Computer Science and Engineering, Seoul National University
[2] Department of Mathematics and ISaC-RIM, Seoul National University

요 약

Private Matching은 각기 다른 두 참여자 (two-party)가 가진 데이터의 교집합 (intersection)을 구하는 문제이다. Private matching은 보험사기 방지시스템 (Insurance fraud detection system), 의료정보 검색, 항공기 탑승 금지자 목록 (Do-not-fly list) 검색 등에 이용될 수 있으며 다자간의 계산 (multiparty computation)으로 확장하면 전자투표, 온라인 게임 등에도 이용될 수 있다. 2004년 Freedman 등은 이 문제를 확률적 (probabilistic)으로 해결하는 프로토콜 (protocol) [1]을 제안하고 악의적인 공격자 (malicious adversary) 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.

1. 서 론

특정 자료를 저장하고 있는 데이터 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 (privacy preserving) 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 데이터를 다항식으로 표현하여 전달하는 방법이 제안되었고 [1], Kissner 등에 의해 일반적인 집합 연산과 다자간 모델로 확장되었다 [4]. Kissner 등은 Freedman 등의 결과를 확장하기 위해 참여자의 데이터를 표현한 다항식에 차수가 같거나 큰 랜덤 다항식을 곱하여 이들을 더하는 방법을 사용하였다.

하지만 이들의 연구는 확률적 (probabilistic)인 방법을 사용하였기 때문에 낮은 확률로 부정확한 결과를 낼 수 있는 단점이 있었으며, 정확도를 높이기 위해 실제 데이터의 개수에 비해 큰 정의역을 사용해야 했기 때문에 불필요한 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 제안한다.

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 두 참여자인 클라이언트 (client) C 와 서버(server) S 가 각각 자신의 데이터로 구성된 집합 $X = \{x_1,...,x_{k_c}\}$ 와 $Y = \{y_1,...,y_{k_s}\}$ 를 가지고 있을 때, 프로토콜 (protocol)을 수행하여 C 가 $X \cap Y$ 을 private하게 알아내는 것을 말한다. 여기서 private하다는 의미는 프로토콜 수행 후 C 는 $X \cap Y$ 외의 Y 의 원소를 알지 못해야 하며 S 는 C 의 원소를 알지 못해야 한다는 것을 의미한다.

2.2 준동형 암호시스템 (Homomorphic Cryptosystem)

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

- 공개키 pk 에 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 c 와 암호문 $Enc_{pk}(m)$ 이 주어졌을 때 이를 곱한 암호문 $Enc_{pk}(cm)$ 을 계산할 수 있다.
- 차수가 k 인 다항식 $P(y) = a_0 + a_1 y + \cdots + a_k y^k$ 의 계수들의 암호문 $\{Enc_{pk}(a_0),..., Enc_{pk}(a_k)\}$ 이 주어졌을 때, 다항식에 임의의 값 y 를 대입한 $Enc_{pk}(P(y))$ 을 계산할 수 있다.

대표적인 준동형 암호시스템으로는 Paillier 암호시스템 [2, 3]이 있다.

2.3 공격자 모델

우리는 프로토콜을 C 와 S 가 모두 Semi-Honest한 Semi-Honest 모델, C 는 악의적이고 S 는 Semi-Honest한 악의적인 클라이언트 (Malicious-Client) 모델, S 는 악의적이고 C 는 Semi-Honest한 악의적인 서버 (Malicious-Server) 모델, C 와 S 가 모두 악의적인 모델의 4가지 경우로 구분하여 분석한다.

각각의 모델의 의미는 다음과 같다. 먼저 Semi-Honest 모델은 프로토콜에 참여하는 S 와 C 가 프로토콜의 모든 규칙을 준수하되 서로의 정보를 알려고 하는 상황을 의미한다. 두 번째로 악의적인 클라이언트 모델에서는 C 가 임의의 정보를 보내어 S 의 정보를 알려고 하는 상황을 의미한다. (이 때 C 가 S 의 잘못된 계산을 유도하는 것은 의미가 없다.) 세 번째로 악의적인 서버 모델에서는 S 가 의도적으로 C 가 잘못된 결과를 계산하게 하는 상황을 의미한다. 여기서 S 의 공격은 이상적인 TTP (Trusted Third Party)가 존재하여 S 와 C 의 집합을 입력으로 받아 C 에 교집합을 알려주는 TTP 모델과 비교하여 TTP 모델에서는 존재할 수 없는 공격으로 제한한다. 마지막으로 모두 악의적인 모델에서는 악의적인 클라이언트 모델과 악의적인 서버 모델을 결합하여 양쪽이 악의적인 상황을 의미한다.

3. FNP 방법

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

3.1 Semi-Honest 모델에서 PM 계산

Semi-Honest 모델에서는 다음과 같은 방법으로 PM을 계산한다. 먼저 클라이언트 C 는 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 각각의 계수를 자신의 공개키(public key)를 사용하여 암호화한 값 $\{Enc_{pk}(\alpha_0),..., Enc_{pk}(\alpha_{k_c})\}$ 를 서버 S 에게 보낸다. S 는 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_i 을 곱한 후, 다시 대입했던 원소를 더한 $Enc_{pk}(r_i \cdot P(y_i) + y_i)$ 를 C 에게 보낸다.

C 는 S 로부터 받은 k_S 개의 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_i 는 랜덤 값이므로 높은 확률로 $d_i \notin X$ 이다. 그러므로 $d_i \in X$ 이면 d_i 를 $X \cap Y$ 의 원소로 인식하고 아니면 랜덤 값으로 인식한다. 계산된 집합이 $X \cap Y$ 값일 확률을 충분히 크게 하기 위해 X 와 Y 의 원소들이 충분히 큰 정의역에서 표현되어야 한다.

X 와 Y 의 원소의 개수가 많은 경우 다항식의 차수를 낮추기 위해 균등 분배 함수 (balanced allocation hash function)를 이용할 수 있다. 이 경우 C 가 균등 분배 함수를 이용하여 X 를 최대 M 개의 원소를 갖는 B 개의

부분집합 $X_1, X_2, ..., X_B$ 로 나눈 다음 각각의 부분집합에 대해 별도로 위의 프로토콜을 수행한다. 그러면 사용하는 다항식의 차수가 M 으로 제한되어 다항식을 전개하는 비용과 대입하는 비용을 줄일 수 있다.

3.2 악의적인 클라이언트 모델에서 PM 계산

악의적인 클라이언트 C 는 서버 S 의 집합 Y 의 원소 중 교집합에 속하지 않는 원소를 알아내거나 S 가 잘못된 계산을 수행하도록 할 수 있다. 그런데 S 는 C 로부터 얻고자 하는 정보가 없으므로 C 가 S 로 하여금 잘못된 계산을 수행하도록 하더라도 얻을 수 있는 이익이 없다. 그러므로 여기서 악의적인 C 는 Y 의 원소를 알아내려고 하는 것으로 제한한다.

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

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

그런데 이 방법은 균등 분배 함수를 이용한 방법에는 확장할 수 없기 때문에 FNP는 cut-and-choose 방법을 사용하여 L 개의 다항식 집합을 만들어서 $L/2$ 개로부터 C 의 근의 개수를 확인하고 나머지 $L/2$ 개로는 PM을 확인하는 방법을 사용하였다.

3.3 악의적인 서버 모델에서 PM 계산

악의적인 서버 S 의 공격은 S 가 의도적으로 C 가 $X \cap Y$ 를 잘못 계산하도록 하는 것이다. 그런데 S 가 자신의 집합 Y 를 임의의 집합으로 대체해버리는 것과 같이 이상적인 TTP가 존재하는 경우에도 가능한 공격은 현실적으로 막기 어렵다. 그러므로 여기서는 이상적인 TTP를 통하는 경우 존재할 수 없는 공격으로 S 의 공격을 제한하였다. 예를 들어 S 가 $Enc_{pk}(r_i \cdot P(y'') + y'')$ 대신 $Enc_{pk}(r \cdot (P(y) + P(y')) + y'')$ 를 계산하여 C 에게 보내면 C 는 y 와 y' 이 X 에 속한 경우에만 y'' 을 $X \cap Y$ 의 원소로 인식하게 된다.

FNP는 랜덤 오라클 (random oracle)을 사용하여 y, y' 과 y'' 사이의 의존성을 피하는 방법을 제안하였다. 즉 S 와 C 모두 접근이 가능한 랜덤 오라클 H_1, H_2 가 있어서 S 가 임의의 랜덤값 s 를 선택하여 $r = H_1(s)$ 를 얻고 자신의 데이터를 다항식에 대입한 암호문과 그 데이터를 H_2 에 대입한 $(e,h) \leftarrow (Enc_{pk}(r' \cdot P(y) + s), H_2(r'', y))$ 를 C 에게 보낸다. 이때 암호시스템에 사용되는 랜덤값은 H_1 으로부터 얻은 r 을 사용하며 r', r'' 은 r 로부터 추출한다. C 는 e 를 복호화하여 $\tilde{s}$ 을 얻고 $\tilde{r} = H_1(\tilde{s})$ 를 계산하여 $\tilde{r}', \tilde{r}''$ 를 추출한다. 이후 S 의 연산을 재구성하는 방법으로 $(\tilde{e}, \tilde{h})$ 를 계산, 그 값이 (e,h) 와 일치하는 자신의 데이터 x_j 가 있는지를 확인하여 PM을 구한다.

3.4 상호 악의적인 모델

S 와 C 가 모두 악의적인 모델은 악의적인 클라이언트 모델과 악의적인 서버 모델을 결합함으로써 해결할 수 있다.

4. 결정적 (Deterministic) Private Matching 프로토콜

4.1 DPM-Semi-Honest 프로토콜

FNP는 Semi-Honest 모델에서 X 와 Y 의 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 1 이다.)

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

그리고 프로토콜 수행에서는 S 가 $P(y)$ 의 암호화된 계수들을 받은 후 Y 의 원소 y_i 에 대하여 랜덤값 r_i 를 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 같을 때에만 y_i를 $X \cap Y$의 원소로 받아들인다.

DPM 프로토콜은 FNP와 같이 균등 분배 함수를 사용한 경우에도 확장 가능하다. 자세한 프로토콜은 다음과 같다.

Protocol DPM-Semi-Honest
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) 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
   $\{Enc_{pk}(\alpha_0), ..., Enc_{pk}(\alpha_{k_C})\}$ S에게 전송.

2. 서버 S
   (a) 집합 $Y = \{y_1, ..., y_{k_S}\}$의 모든 원소 y_i에 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 C에게 전송

3. 클라이언트 C
   (a) $Result := \phi$
   (b) k_s개의 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 $Result := Result \cup \{d_{i,1}\}$

4.2 DPM-Malicious Client

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 다항식에 상수를 곱하는 비용을 줄일 수 있을 뿐 아니라 균등 분배 함수를 사용하여 각각의 다항식의 차수를 M 차로 제한하는 경우에도 확장 가능하다.

악의적인 클라이언트 모델에서도 DPM 방법은 결과의 정확성을 보장할 수 있다. 악의적인 클라이언트 모델에서 균등 분배 함수를 사용한 프로토콜은 다음과 같다.

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 B개의 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
   $$\begin{pmatrix} e_{j,1} \\ e_{j,2} \end{pmatrix} := \begin{pmatrix} Enc_{pk}(r_{j,1} \cdot P_h(y_j) + y_j) \\ Enc_{pk}(s_{j,2} \cdot P_h(y_j) + y_j) \end{pmatrix}$$
   (b) $\{(e_{j,1}, e_{j,2}) \mid 1 \le j \le k_S\}$를 C에게 보냄

3. 클라이언트 C
   (a) $Result := \phi$
   (b) k_s개의 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 $Result = Result \cup \{d_{j,1}\}$

4.3 DPM-Malicious-Server

우리는 FNP의 방법에 DPM을 적용하여 연산시간을 단축하였다. 즉 기존의 방법에서는 C가 PM을 확인하기

위해 S 로부터 받은 (e,h) 각 쌍에 자신의 모든 원소로 S 의 연산을 재구성한 $(\tilde{e},\tilde{h})$ 을 k_c 번 비교해야 했는데, 우리는 (e,h) 각 쌍에 해당하는 S 의 연산을 한번만 재구성하도록 개선하였다. 자세한 프로토콜은 아래와 같다.

Protocol DPM-Malicious-Server

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

Output : $Result = X \cap Y$

Random Oracle : H_1, H_2

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

2. 서버 S
   (a) 데이터 $Y=\{y_1,\dots,y_{k_s}\}$ 의 모든 원소 y_i 에
   (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 S 의 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 C 에게 전송

3. 클라이언트 C
   (a) $Result := \phi$
   (b) S 로부터 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_1 에 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) S 의 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
   $Result := Result \cup \{\tilde{y}_{i,1}\}$

4.4 DPM-Malicious

S 와 C 가 모두 악의적인 모델의 경우는 DPM-Malicious-Client 프로토콜과 DPM-Malicious-Server를 결합하여 확장 가능하다.

5. 분 석

5.1 정확도 (Correctness)

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

Theorem 1. S 와 C 가 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 C 가 프로토콜의 결과값 $X\cap Y$ 이 자신의 데이터에 포함되는지 확인할 필요가 없다. 즉 $d_{i,1}=d_{i,2}$ 이면 $d_{i,1}\in X$ 임이 보장되기 때문에 별도로 X 에 포함되는지 여부를 확인할 필요가 없다.

5.2 안전도 (Security)

클라이언트의 보안성 (client's security)은 C 가 데이터를

변경해가면서 프로토콜을 수행할 때 서버가 데이터의 변경여부를 알지 못하는 (indistinguishable) 것을 의미한다. 그리고 서버의 보안성 (server's security)은 프로토콜 수행 후 C 가 얻은 PM이 가상의 신뢰할 만한 제3자 (Trusted Third Party: TTP)가 만들어낸 정확한 PM과 구분할 수 없이 같다는 것을 의미한다.

DPM은 클라이언트의 보안성과 서버의 보안성을 모두 만족시키므로 FNP 프로토콜과 동일한 수준의 안전도를 가지고 있다고 할 수 있다. 특히 DPM에서 사용하는 암호시스템은 Semantically Secure한 특성을 가지고 있기 때문에 중간에 도청자가 암호문 쌍을 가로채더라도 복호화했을 때 같은 값인지 여부를 알 수 없다.

5.3 복잡도 (Complexity)

5.3.1 공간 (Space) 복잡도

각각의 원소가 n 비트로 표현될 때 FNP에서 높은 확률 (에러률 $\epsilon = 1-(1-k_c/2^n)^{k_s}$)로 정확한 PM을 계산하기 위해서는 n 이 충분히 커야 한다. 이들의 프로토콜에서는 각각의 원소를 저장하는데 공간이 많이 필요하거나 프로토콜 수행 전 별도의 전처리 (pre-processing)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 $\epsilon = 0.63$, $n = 30$ 비트일 때 $\epsilon = 9.76 \times 10^{-4}$ 의 에러 확률로 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 C 가 데이터를 다항식으로 표현하여 전개하는 시간 $O(k_c^2)$ 과 S 가 데이터를 다항식에 대입하는데 걸리는 시간 $O(k_s k_c)$ 은 FNP와 동일하다. 한편, Kissner 등의 방법으로 PM을 구하려고 하면 각 참여자의 다항식 차수가 2배 이상이 되기 때문에 다항식을 전개하는 시간은 4배 이상으로 증가한다. 따라서 PM의 계산에 있어서는 FNP와 DPM이 Kissner 등의 방법 보다 더 효율적이다.

통신량을 살펴보면 DPM 프로토콜은 모두 S 가 C 에게 데이터 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 C 가 일치하는 (e,h) 쌍을 찾기 위해 서버의 연산을 재구성하는 연산을 한번만 수행하기 때문에 연산량이 크게 줄어든다.

따라서 DPM은 S 가 C 에게 보내는 통신량이 증가하지만 단위 데이터의 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 데이터의 크기를 줄일 수 있었으며, 결정적인 결과를 산출하는 특성을 가지고 있다. 현재 DPM 프로토콜은 참여자가 둘인 경우를 가정하였는데 이를 효율적으로 다자간의 계산으로 확장하는 것은 이후 좋은 연구 주제가 될 것이다.

참고문헌

1. Michael Freedmain, Kobi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In Proc. Of Eurocrypt, volume LNCS 3027, pages 1-19. Springer-Verlag, May 2004

2. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Advances in Cyptology-EUROCYTPT '99, pages 223-238, Prague, Czech Republic, May 1999

3. 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, Cheju Island, Korea, February 2001.

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