정 보 보 호 학 회 지
제 30 권 제5호, 2020. 10

LWE와 완전동형암호에 대한 분석 및 동향

유 준 수*, 윤 지 원**

요 약

동형암호(homomorphic encryption)는 암호화된 데이터 사이에서 임의의 연산을 가능하게 하는 유망한 암호학적 스킴(scheme)이다. 이를 활용하면 암호화된 데이터를 복호화하지 않고, 암호화된 상태에서 임의의 연산을 수행 할 수 있을 뿐만 아니라, 격자를 기반(lattice-based)으로 하여 양자 알고리즘에 내성(resistant)이 있어 안전하다. 하지만, 동형암호를 이해하기 위해서는 전문적인 암호 또는 계산적인 이론의 지식과 이해가 필요하다. 따라서 본 논문에서는 완전동형암호(fully homomorphic encryption)의 기저에 있는 LWE(learning with error) 문제에서부터 완전동형암호의 핵심인 NAND 게이트와 부트스트래핑(bootstrapping)까지의 과정을 어렵지 않게 설명하여 초보자들의 이해를 돕고자 한다.

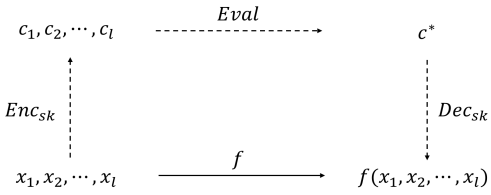
Ⅰ. 서 론

양자 컴퓨터가 향후 가까운 미래에 출현할 것에 대비하여 기존 암호체계에서 벗어난 post-quantum 암호에 대한 요구가 증대되고 있다. 현재 사용되는 대표적인 공개키 암호들의 경우, 소인수 분해(integer factorization)와 이산 대수(discrete logarithm)의 수학적 어려움을 두고 설계가 되었다. 하지만, post-quantum 시대에서의 무한한 계산복잡도를 갖는 공격자에게 기존 공개키 암호체계가 갖고 있는 취약성은 잘 알려져 있다. 대표적으로 Shor 알고리즘이 있다[1].

이를 대비하여 다양한 암호체계들이 제시되었는데, 이 중 격자를 기반(lattice-based)으로 하여 수학적 어려움을 통해 안전성을 제공하는 암호체계들이 각광 받고 있다. 동형암호는 격자를 기반으로 하여 안전성을 제공하고 더 나아가 (이론적으로)모든 연산들을 boolean 동형암호 게이트로 구성하여 암호화된 데이터 사이에서 연산을 가능하게 한다.

이를 이해하기 위한 대표적인 모델로 서버와 사용자 간에 동형암호를 이용하여 2-party 통신을 하는 것을 생각해 볼 수 있다[그림 1].

사용자가 원하는 것은 본인의 민감한 데이터 $x_1, x_2, \cdots, x_l$와 원하는 연산(또는 함수) f을 서버에



〔그림 1〕 완전동형암호 모식도

게 본인의 개인키(sk)로 암호화하여 보낸다. 서버는 사용자의 원하는 서비스를 암호화된 알고리즘(회로) $Eval$를 통해 c^*를 얻는다. 서버는 이후 사용자에게 c^*를 보내고 사용자는 본인의 개인키를 이용하여 복호화하여 원하는 서비스의 결과 $f(x_1, x_2, \cdots, x_l)$를 얻는다.

이 과정은 $Dec_{sk}(c^*) = f(x_1, x_2, \cdots, x_l)$를 만족시킨다. 즉, 서버는 $Eval$ 함수를 통해 암호화된 상태에서 사용자의 데이터를 처리하고, 암호화된 결과 값인 c^*를 사용자에게 보내줌으로써 서버는 사용자의 정보에 대해 알 수 없다. 결론적으로, 서버는 사용자의 데이터와 원하는 연산이 무엇인지 알 수 없고, 다만 사용자가 요구하는 연산만을 수행한다.

이처럼 동형암호를 활용하면 서버에 암호화된 데이

이 논문은 2020년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임(No.2017-0-00545, 암호화된 데이터상의 기계학습 라이브러리 설계 및 활용기법 연구).
* 고려대학교 정보보호대학원 (대학원생, sandiegojs@korea.ac.kr)
** 고려대학교 정보보호대학원 (교수, jiwon_yoon@korea.ac.kr)

터를 아웃소싱(outsourcing)하여 데이터를 암호화된 상태로 저장해놓고 사용자의 필요에 따라 원하는 데이터를 갖고 올 수 있다[2]. 이는 동형암호의 특징인 양자내성(quantum resistant)이 있기 때문이다. 또한 사용자는 서버에 저장된 암호화된 데이터에 대해서 암호화된 쿼리를 보내 서버로 하여금 사용자의 원하는 연산을 수행하도록 할 수 있다.

하지만, 이러한 동형암호의 다양한 장점에도 불구하고 가장 큰 문제점으로 암호화된 비트 사이의 $Eval$ 연산량이 많아 속도와 메모리에서 성능(performance)이 현저히 떨어진다. 이에 대한 보다 자세한 내용은 3.3장에서 확인할 수 있다.

최근 들어서는 이러한 동형암호의 장점을 이용하여 암호화된 데이터 간의 기계학습을 수행하는 연구가 활발히 진행되고 있다[3-5]. Hong et al[6]은 암호화된 데이터의 경향성을 가장 잘 설명할 수 있는 최적의 차수(degree)를 찾은 모델선택(model selection)을 설명하고, 이의 적용 모델로 암호화된 의료데이터를 택하여 적용한 연구결과도 있다. 비슷한 연구로 기계학습에서 사용되는 로지스틱 회귀분석을 암호화된 상태에서 수행하고, 이를 의료데이터에 적용한 연구 결과도 확인할 수 있다[3]. 또한 동형암호를 딥러닝(deep learning)에 적용하는 연구들도 확인할 수 있다[7].

이렇게 뚜렷한 장점을 갖는 동형암호에 대해 기존에 많은 연구들이 진행되었다. 하지만 대부분의 연구들은 수학 또는 계산복잡도 이론에 기반하여 전문적으로 서술되어 있어 입문자가 읽기 어렵거나 혹은 지엽적이여서 동형암호의 전반적인 내용을 이해하기 쉽지 않다.

따라서, 본 연구는 이러한 동형암호 기술에 대해 기저에 있는 LWE부터 최종적으로 완전동형암호의 NAND 게이트를 설계하는 방법과 부트스트래핑(bootstrapping)까지 어렵지 않게 설명하고자 한다. 완전동형암호 NAND 게이트까지 설명을 하게 되면, 이론적으로는 이 universal 게이트를 통해 모든 함수 f를 동형암호 스킴으로 설계할 수 있기 때문이다.

1.1. 논문구성과 자주 등장하는 기호(notation)

본 논문은 다음과 같이 구성된다. 2장에서는 동형암호의 안전성을 기반으로 하는 LWE 문제를 설명한다. 3장은 부스트래핑 없이 제한된 연산만을 가능하게 하는 somewhat 동형암호를 설명하고, 4장에서는 부트스트래핑까지 가능한 완전 동형암호에 대해 설명한다. 마지막으로, 5장에서는 동형암호의 적용 분야와 도움이 될만한 자료들을 소개한다.

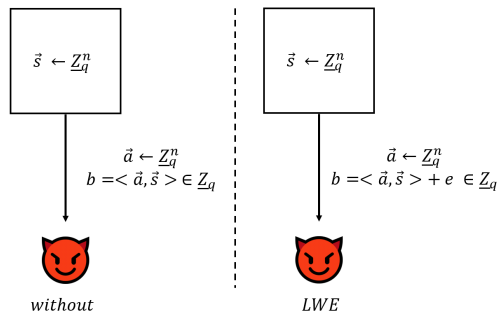
본 논문에서 벡터는 $\vec{v} = (v_1, v_2, \cdots, v_n)^T$와 같이 화살표로 표시하고 열벡터를 나타낸다. 행렬은 M과 같이 대문자로 표기한다. 상수(constant)는 c와 같이 소문자로 표기한다.

자주 사용하게 되는 기호로 $\underline{Z}$는 정수, 모듈러 q 공간상은 $\underline{Z}_q = \{0, 1, 2 \cdots, q-1\}$, $\underline{Z}_q^n$는 n개의 원소를 갖고 있는 열벡터이고 각각의 원소는 $\underline{Z}_q$의 공간에서 추출된 것을 의미한다. 또한 두 벡터 간 내적은 $< \vec{a}, \vec{b} >$ 로 표기하고 각 원소들의 곱의 합으로 정의한다.

Ⅱ. LWE

2.1. Learning without Error

그림 2(좌)는 본격적인 LWE를 설명하기 이전에 에러가 어떤 의미를 갖는지 설명하기 위한 시뮬레이션 모델이다. 여기서, 상자는 오라클을 의미하고 비밀키 $\vec{s} = (s_1, s_2, \cdots, s_n)$를 모듈러 공간 상에서인 $\underline{Z}_q^n$에서 임의로 선택하고 공격자에게 숨긴다. 반면, 공격자는 $\vec{s}$를 알아내기 위해 오라클에 질의(query)를 한다. 각 질의시 공격자는 한 쌍의 벡터 $\vec{a}$와 내적값 $< \vec{a}, \vec{s} > (= b)$을 받게 된다.



〔그림 2〕 (좌)에러가 없는 시뮬레이션 모델에서의 공격자가 학습하는 과정을 표현한 그림 (우)에러가 있는 시뮬레이션 모델에서의 공격자가 학습하는 과정을 표현한 그림

공격자는 $\vec{s}$를 알아내기 위해 n개 또는 그 이상 되는 질의를 오라클에 하게 되면 $\vec{s}$ 값을 알 수 있게 된다. 즉, 각각의 질의 $\vec{a}_1$, $\vec{a}_2$, ⋯, $\vec{a}_n$에 대해, 각 $\vec{a}_i$의 원소를 $a_{i1}, a_{i2}, \cdots, a_{in}$라고 한다면, 공격자는 다음과 같은 선형방정식계(system of linear equations)를 얻을 수 있게 된다.

$$\begin{aligned} <\vec{a}_1, \vec{s}> &= a_{11}s_1 + a_{12}s_2 + \cdots + a_{1n}s_n \\ <\vec{a}_2, \vec{s}> &= a_{21}s_1 + a_{22}s_2 + \cdots + a_{2n}s_n \\ &\vdots \\ <\vec{a}_n, \vec{s}> &= a_{n1}s_1 + a_{n2}s_2 + \cdots + a_{nn}s_n \end{aligned} \quad (1)$$

단, 여기서 n개의 선형방정식에 대해서 full rank를 얻을 수 있다는 것을 가정한다(만약, 얻지 못한다면 질의를 더 진행한다).

공격자는 이제 단순히 Gauss 소거법을 이용하여, $\vec{s}$ 값을 쉽게 얻을 수 있다. 따라서, 공격자는 $O(n)$ 의 계산복잡도로 일차다항식에 대한 해(비밀키)를 얻을 수 있다.

2.2. Search LWE(Learning With Error)

그림 2(우)에서의 에러가 있는 상황에서 공격자가 비밀키를 알아내는 방법은 다르다. 이번에 공격자에게 주어진 것은 $\vec{a}_i$와 여기에 비밀키 $\vec{s}$의 내적값에 에러 $\vec{e}$를 더해준 값이다. 즉, $b_i = <\vec{a}_i, \vec{s}> + e_i = a_{i1}s_1 + a_{i2}s_2 + \cdots + a_{in}s_i + e_i$이고 공격자는 (a_i, b_i)를 원하는 만큼 오라클로부터 얻을 수 있다. 여기서, e_i는 $\{-B, -B+1, \cdots, B-1, B\}$의 균일확률분포(uniform distribution)에서 추출한 에러값이다.

2.1장에서와 마찬가지로 가우스 소거법을 이용해 문제를 풀게 되면 공격자가 풀어야 하는 계산복잡도는 $O(n^{2B+1})$이 된다. 예를 들어, $B=1$인 경우, 에러는 $\{-1, 0, 1\}$의 세 가지 값 중 하나를 갖게 된다. 따라서, i번째 식 $b_i = <\vec{a}_i, \vec{s}> + e_i$에 대해서 e_i의 값은 세 가지를 갖기 때문에, 총 풀어야 하는 경우의 수는 최소 3^n가지, $O(3^n)$이 된다. 따라서 확률적 다항 연산 PPT(probabilistic polynomial time) 공격자는 문제를 풀 수 없게 된다.

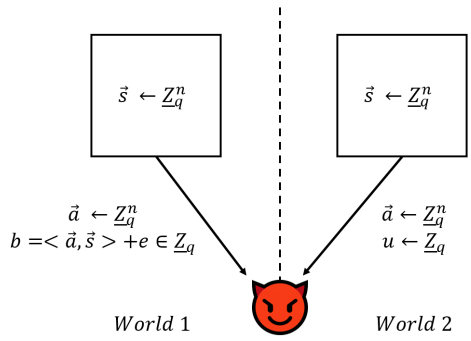
이와 같은 공격에 대해 Arora-Ge[8]에서 제안된 공격 방법이다. 이 방법의 아이디어를 간단하게 설명하면, 공격자는

$$\begin{aligned} &(b - <a, s> - B)(b - <a, s> - B + 1) \cdots \\ &(b - <a, s> + B - 1)(b - <a, s> + B) = 0 \end{aligned} \quad (2)$$

식(2)를 풀어야한다. 왜냐하면 $b - <a, s> = e$이고 각 에러는 $\{-B, -B+1, \cdots, B-1, B\}$의 값 중 하나를 갖기 때문이다. 식(2)는 차수(degree)가 $2B+1$인 다변수 다항식이고, (2)와 같이 한 쌍의 LWE 샘플 (a_i, b_i)마다 $2B+1$개의 변수에 대해서 푸는 일차방식으로 변환하여 총 변수의 개수가 n^{2B+1}에 대해 $O(n^{2B+1})$의 식을 풀어야 공격이 가능해져 ppt는 풀 수 없는 것을 증명하였다.

2.3. Decisional LWE

Decisional LWE는 동형암호가 기반으로 하는 수학적 어려움(hardness)이다. 그림3에서는 decisional LWE에서의 공격자와 공격자가 해야 하는 선택에 대한 시뮬레이션 모델에 대한 도식이다. 공격자는 두 개의 세계관(World 1 또는 World 2) 중 한 개를 선택해야 한다. 즉, 공격자는 World 1에서 쌍으로 받는 LWE 샘플들 $(\vec{a}_i, b_i)$과 World 2에서 샘플$(\vec{a}_i, u)$들을 통하여, 본인이 위치하고 있는 곳이 어디인지 판별할 수 있는지에 관한 게임이다. 여기서 u는 $\underline{Z}_q$의 균일확률분포에서 샘



〔그림 3〕 Decisional LWE 시뮬레이션 모델

플링한 의미 없는 값이다.

이 모델에서 공격자는 비밀키를 갖고 있지 않다면, 두 개의 세계관에서 오는 암호문(LWE 샘플)들이 모두 랜덤한 값으로 보이게 된다. 왜냐하면 World 1에서 $b_i = < \vec{a_i}, \vec{s} > + e_i$에 의해 각각의 a_i에 대해 내적 후 임의의 값 e_i이 더해지기 때문이다. 마찬가지로 World 2에서는 u값이 임의의 값이므로 두 세계관에서 오늘 모든 값들이 랜덤하게 보이게 된다. 즉, 이 암호화 스킴은 semantically 안전하게 된다.

반면에, 비밀키를 정당하게 아는 사람의 경우, $b_i - < a_i, s >$를 하여 e_i값들을 얻고, 이에 대해 $e_i \in \{-B, -B+1, \cdots, B-1, B\}$인지 판명하면 과정을 거치면 양쪽 세계관 중 한 개를 선택할 수 있다. 왜냐하면, 세계관 1에서는 e_i값들의 범위가 정해져 있는 반면, 세계관 2에서는 Z_q안에서 임의의 값이기 때문에 확률적으로 몇몇 경우 $e_i \notin \{-B, -B+1, \cdots, B-1, B\}$가 생기게 된다.

마찬가지로 동형암호의 암호화 스킴들은 이러한 decisional LWE의 어려움에 기반하여 암호화 방식을 구성하였다.

2.4. LWE 기반 암호화

2.3장에서 설명한 decisional LWE[9-10]를 기반으로 하는 암호화 기법 중 가장 기본적인 방법은 다음과 같다. 여기서 x는 한 비트, 즉 $x = 0$ 또는 1을 갖는다.

- **LWE.Keygen**: $\vec{s}$를 Z_q^n으로부터 임의의 값을 추출한다.
- **LWE.Enc($\vec{s}$, x)**: $x = 1$인 경우 $(\vec{a}, b = < \vec{a}, \vec{s} > + e)$를 생성하고, $x = 0$인 경우 $(\vec{a}, u)$를 생성한다.
- **LWE.Dec**: 위에서 설명한 대로 $< \vec{a}, \vec{s} >$ 값을 계산하여, b 또는 u의 샘플로부터 값을 빼준다. 뺀 오차가 설정한 에러범위 $\{-B, -B+1, \cdots, B-1, B\}$에 포함되면 $x = 1$이고, 균일(uniform)한 값을 지수(exponentially)적으로 높은 확률을 갖으면 $x = 0$이 된다.

위 방법은 가장 단순하면서 효과적인 LWE를 기반으로 암호화를 할 수 있다. 단, 복호화시 확률적으로 x값이 정해진다는 점인데, 이는 같은 x를 더 많이 암호화함으로써 x값이 0인지 아닌지 명확하게 구분해 낼 수 있게 된다. 완벽하게 구별(distinguish)하는 방법에 대해서는 다음 장에서 에러 교정(error correct) 개념과 함께 설명한다.

Ⅲ. 동형암호(homomorphic encryption)

3.1. 동형암호의 분류

완전(fully) 동형암호 이전에 나온 동형암호는 발표된 시간 순서에 따라 부분(partial) 동형암호, somewhat 동형암호 등이 있다. 전자의 경우는 완전동형암호가 덧셈, 곱셈 양쪽을 지원하는 것에 비해, 한쪽만 지원하는 암호화 방식을 말한다. 예를 들어 El Gamal[11], Paillier[12] 암호화 방식들도 부분 동형암호의 한 일종이다. 이외에도 이후 발표되 다양한 격자 기반의 암호화 방식들 Ajitai와 Dwork[13], Regev[14]이 한쪽만을 지원하는 것으로 볼 수 있다.

이렇게 한쪽 연산만 가능했던 동형암호는 2009년 Gentry[15-17]의 새로운 방식인 somewhat 동형암호가 제안되면서 덧셈, 곱셈 양쪽 모두 가능하게 되었다. 먼저, Gentry는 제한된 연산(차수가 낮은 다항식)만이 가능한 함수를 통해 두 연산이 가능함을 보이고, 이후 부트스트래핑이 복호화 회로를 통해 가능함을 보이면서 'leveled' 동형암호를 보였다. 마지막으로 암호화된 키를 통해 부트스트래핑이 되는 것을 보여주면서 완전 동형암호가 가능한 것을 증명하였다.

3.2. 부분 동형암호(PHE)

2.4장에서는 간단한 decisional LWE를 통해 한 비트를 암호화하고 복호화하는 방법에 대해서 알아 보았다. 이 내용에 error correct 개념을 추가한 부분 동형암호화(PHE: Partial Homomorphic Encryption) 방식은 다음과 같다.

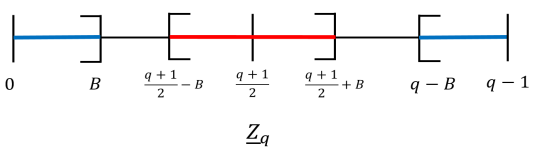
- **PHE.Keygen**: 2.4장과 동일하다.
- **PHE.Enc($\vec{s}$, x)**: $(\vec{a}, b = < \vec{a}, \vec{s} > + e$

$+x\left\lceil \frac{q}{2} \right\rceil$). 여기서 ⌈ ⌉ 는 라운딩 함수다. 마찬가지로 모든 비트에 에러값을 더해주었기 때문에, 이 암호화 방식은 semantic security를 따른다.

- **PHE.Dec:** b 에서 $<\vec{a},\vec{s}>$를 빼면 남는 값은 $x\left\lceil \frac{q}{2} \right\rceil + e \pmod q$ 가 된다. 따라서, $x=0$일 경우, e값만 남게 되고, 이는 Z_q 상에서 $[0,1,\cdots,B] \cup [q-B,\cdots,q-1]$의 범위가 된다. 반대로, $x=1$일 경우, $\frac{q+1}{2}+e \pmod q$가 되어 Z_q 상에서 $[\frac{q+1}{2}-B,\frac{q+1}{2}-B+1,\cdots,\frac{q+1}{2}+B]$의 범위가 된다. 즉, 그림 4에서 나타난 것과 같이 복호화 시 얻는 $x\left\lceil \frac{q}{2} \right\rceil + e \pmod q$에서 x값에 따라 다른 구간의 값을 얻게 된다.

그림 4의 파란 구역일 경우 $x=0$으로 복호화되고, 빨간 구역일 경우 $x=1$을 갖도록 완벽히 분리된다. 이것이 가능해진 이유는 error correct 값인 $\frac{q+1}{2}$를 x에 곱해주었기 때문이다. 따라서, 암호문 복호화 시 정확한(correct) 값을 갖도록 하기 위해서는 $4B<q$를 만족해야 한다.

중요한 점은 첫 번째로 이 암호화 방식의 안전성을 보장하기 위해서 파라메타 n값이 커야 된다. 일반적으로 $n=1024$ 또는 그 이상의 값을 갖는다. 두 번째로 SNR(signal to noise ratio)인 $\frac{q}{B}$가 적당한 값을 가져야 한다. 예를 들어, $\frac{q}{B}$가 큰 값을 갖게 되면, 암호문 간에 구별이 가능해져 정확성에 도움이 되지만 안전성



〔그림 4〕 PHE 복호화시 x값의 범위(빨강: $x=1$, 파랑: $x=0$)

은 떨어지게 된다. 가능해진다. 단, 유의 해야할 점은 $\frac{q}{B}$의 비율(ratio)이 안전성과 정확성에 관련한 것이다. 개별적으로 q 또는 B의 값이 크다고 안전해지는 것은 아니다.

이 암호화 방식은 동형적 덧셈을 가능하게 한다. 암호문 $c_1=(\vec{a}_1,b_1)$과 $c_2=(\vec{a}_2,b_2)$에 대해서 각각을 더하면 $c_1+c_2=(\vec{a}_1+\vec{a}_2,b_1+b_2)$가 된다. 여기서 $b_1+b_2= <\vec{a}_1+\vec{a}_2,\vec{s}>+e_1+e_2+(x_1+x_2)(\frac{q+1}{2})$이다. 따라서, b_1+b_2를 복호화하면, $e_1+e_2+(x_1+x_2)(\frac{q+1}{2})$가 되고, 이것은 다시 $e_1+e_2+e^*+(x_1 \oplus x_2)(\frac{q+1}{2})$이 된다. 여기서 $\oplus$은 XOR을 뜻하고 e^*는 x_1과 x_2가 모두 1이 되면 $(x_1+x_2)(\frac{q+1}{2})=q+1=1 \pmod q$이 되기 때문에 $x_1=x_2=1$일 때 $e^*=1$, 나머지 값일 때 $e^*=0$을 의미한다. 따라서, 이 방식은 그림 4와 같이 복호화할 때마다 $x_1 \oplus x_2$ 값에 따라 서로 다른 영역으로 분리된다.

여기서 확인할 수 있는 점은 두 암호문을 더할 때마다 에러 $e_1+e_2+e^*$가 증가한다는 것이다. 따라서, 에러 증가량 $|e_1+e_2+e^*|$이 $\frac{q}{4}$보다 작아야 덧셈으로 나오는 결괏값이 정확하게 복호화된다.

그러므로 많은 양의 암호문 간 동형적 덧셈을 해야 하는 함수가 있다면 에러 파라메타 값 e을 잘 조정해야 한다.

3.3. Somewhat 동형암호(SHE)

앞서 설명한 암호화 방식에서 동형적 곱셈을 정의하기는 어렵다. 따라서, 이전에 제시한 암호화 기법에서 조금 변경한 (격자 기반) somewhat 동형암호 방식을 통해 덧셈 곱셈을 가능하게 한다.

- **SHE.Keygen**: 이전과 동일하다.
- **SHE.Enc($\vec{s}$, x)**: $C=L+xG$

여기서, C는 암호문 행렬을 의미하고,

$$C = L + xG = \begin{bmatrix} | & | & & | \\ a_1 & a_2 & \cdots & a_m \\ | & | & & | \\ b_1 & b_2 & & b_m \end{bmatrix} + x \begin{bmatrix} 1\,2\cdots 2^{\lfloor \log_2 q \rfloor} & & & \\ & 1\,2\cdots 2^{\lfloor \log_2 q \rfloor} & & \\ & & \cdots & \\ & & & 1\,2\cdots 2^{\lfloor \log_2 q \rfloor} \end{bmatrix}$$

L 행렬은 LWE 샘플 m개의 $(\vec{a_i}, b_i)$ 값들에 대해, 열벡터로 행렬에 나열한 것이다. 여기서 $b_i = < \vec{a_i}, \vec{s} > + e$이고 L 행렬의 차원은 $(n+1) \times m$과 같다.

G 행렬은 gadget 행렬로, 블록 $g = [1, 2, \cdots, 2^{\log_2 q}]$를 단위행렬 $I_{(n+1)\times(n+1)}$과 대각곱($\otimes$)을 해준 것이다. 즉, $G = g \otimes I_{(n+1)\times(n+1)}$이고 차원은 $(n+1) \times (n+1)\lfloor \log_2 q \rfloor$ 가 된다. 암호문 C의 원소 b_i들은 모두 임의의 에러 값들이 더해지는 LWE 샘플들이기 때문에 semantic 안전성을 확보할 수 있다. 단, 암호화를 조금 특이한 방식으로 하는 이유에 대한 이해는 암호문간 곱셈를 통해 알 수 있다.

- **SHE.Dec**: 복호화키는 $\vec{s}$의 전치(transpose)를 한 벡터에 -1을 연속으로 잇는(concatenate) 벡터 $[\vec{s}^T \| -1]$를 이용한다.

$$[\vec{s}^T \| -1]\, C = \vec{e}^T + x\,[\vec{s}^T \| -1]\, G \tag{3}$$

식(3)에서 $\vec{e}^T = (e_1, e_2, \cdots, e_m)$으로 각 LWE 샘플들에 대한 에러 벡터이다. 각각의 e_i값들은 $[\vec{s}^T \| -1]$에 의해 $b_i - < \vec{a_i}, \vec{s} >$가 되어 $\vec{e}^T$의 원소들이 된다.

여기서 복호화 방법은 암호문 C에 키를 곱한 $[\vec{s}^T \| -1]\, C$ 행렬 마지막 블록의 $2^{\log_2 \frac{q+1}{2}}$ $(= \frac{q+1}{2})$되는 지점에서 $e_i + x(\frac{q+1}{2})$의 값이 (이전과 같은 방식으로) 어느 범주 안에 들어가는지를 보면 $x = 0$ 또는 1인지 알 수 있다.

위 방식으로 암호문 C_1과 C_2의 덧셈을 하면 $C_1 + C_2 = (L_1 + L_2) + (x_1 + x_2)G$ 이다. 이에 대해 복호화를 하게 되면, $[\vec{s}^T \| -1](C_1 + C_2) = \vec{e_1}^T + \vec{e_2}^T + (x_1 + x_2)[\vec{s}^T \| -1]\, G$가 되고 마찬가지로 행렬에서 $\frac{q+1}{2}$가 되는 지점에서 값이 갈리는 것을 통해 $x_1 \oplus x_2$의 값을 유추할 수 있게 된다.

암호문 간의 곱셈 $C_1 \times C_2$을 하기 위해서는 행렬의 크기에 대한 조정이 필요하다. 따라서, 새로운 함수 $G^{-1}: C \rightarrow G^{-1}(C)$를 다음과 같이 정의한다.

$$G \circ G^{-1}(C) = C$$

$G^{-1}(C)$의 원소는 0과 1의 값들로 구성되어 있다. 방법은 아래와 같다.

모든 원소 $c_{i,j} \in C$에 대해서 $c_{i,j}$의 2진 표시(binary representation)를 열벡터로 갖는다. 즉, 함수 G^{-1}는 $(n+1) \times (n+1)\log_2 q$의 차원을 갖는 C에서 $(n+1)\log_2 q \times (n+1)\log_2 q$의 차원을 갖는 $G^{-1}(C)$로 변환(transform)시킨다.

그러므로 암호문 간 동형적 곱셈은 $C_1 G^{-1}(C_2)$로 정의되고, 이를 복호화하여 correctness를 증명할 수 있다.

$$\begin{aligned} & [\vec{s}^T \| -1]\, C_1 G^{-1}(C_2) \\ &= (\vec{e_1}^T + x_1 [\vec{s}^T \| -1]\, G) G^{-1}(C_2) \\ &= \vec{e_1}^T G^{-1}(C_2) + x_1 [\vec{s}^T \| -1]\, C_2 \\ &= \vec{e_1}^T G^{-1}(C_2) + x_1 (\vec{e_2}^T + x_2 [\vec{s}^T \| -1]\, G) \\ &= (\vec{e_1}^T G^{-1}(C_2) + x_1 \vec{e_2}^T) + x_1 x_2 [\vec{s}^T \| -1]\, G \\ &= e^* + x_1 x_2 [\vec{s}^T \| -1]\, G \end{aligned} \tag{4}$$

앞의 $e^* (= \vec{e_1}^T G^{-1}(C_2) + x_1 \vec{e_2}^T)$가 에러가 되고, 위 식을 통해 $C_1 G^{-1}(C_2)$를 복호화하면 $x_1 x_2$의 비

트 곱인 AND 연산이 됨을 알 수 있다.

Ⅳ. 완전동형암호

앞서 3.3장에서 somewhat 동형암호의 동형 덧셈과 곱셈 연산이 가능함을 확인하였다. 이번 장에서는 동형암호 NAND 게이트를 이론적으로 만드는 방법에 대해 알아보고, 이를 토대로 에러 증가를 예측하고, 부트스트래핑 방법을 소개하여 완전동형암호가 이론적으로 가능함을 보인다.

4.1. NAND 게이트

C_{NAND}는 한 비트의 암호문인 C_1과 C_2의 NAND 연산의 암호문 결괏값이다. 이는 앞서 설명한 somewhat 동형암호 스킴에서 AND (비트)곱 연산의 결과 $C_1 G^{-1}(C_2)$를 G에서 빼준 값이다. 다시 말하면, $C_{NAND} = G - C_1 G^{-1}(C_2)$가 된다.

C_{NAND}를 복호화하여 x_1과 x_2의 NAND 값이 나오는 것을 확인해보면 다음과 같다.

$$\begin{aligned}&[\vec{s}^T \| -1] C_{NAND}\\ &= [\vec{s}^T \| -1] G - e^* - x_1 x_2 [\vec{s}^T \| -1] G\\ &= -e^* + (1 - x_1 x_2)[\vec{s}^T \| -1] G \end{aligned} \quad (6)$$

NAND 게이트 1개당 잡음(noise) e^*의 증가량을 계산하면, 고정된 파라메타 (q, n, B)에서 어느 정도 깊이(depth)까지 계산할 수 있는지 알 수 있게 된다. 따라서, e^* 벡터 내에서 원소 중 최대 증가량은 다음 부등식을 만족한다.

$$\|e^*\|_\infty \leqq Bm + B \leqq B(m+1) \leqq (2m)B \quad (7)$$

왜냐하면, $e^* = \overrightarrow{e_1}^T G^{-1}(C_2) + x_1 \overrightarrow{e_2}^T$이고 $\|\overrightarrow{e_1}^T\|_\infty \leqq B$, $G^{-1}(C)$는 0과 1으로 구성된 $m \times (n+1)\log_2 q$ 행렬이다(단, $m = (n+1)\log_2 q$이지만 편의상 m으로 표기한다).

4.2. Leveled 동형암호

모든 연산은 NAND 게이트로 구성하여 만들 수 있다. 마찬가지로, 동형암호 NAND 게이트를 사용하여 어떤 함수 f를 만들 수 있는데, 이때 잡음(noise) 값을 계산하여 깊이(depth) d를 예측할 수 있다. 즉, 이렇게 제한된 연산만이 가능한 동형암호를 leveled 동형암호라고 한다.

d까지 계산한 최대 잡음을 $\|e_d\|_\infty$라고 하면, $\|e_d\|_\infty \leqq (2m)^d B \leqq \frac{q}{4}$을 만족한다. 이유는 correctness가 보장되기 위해서 noise의 최대값은 $\frac{q}{4}$를 넘어서는 안되기 때문이다. 위 식을 변형하면 SNR의 lower bound가 $\frac{q}{B} \geqq 4(2m)^d$임을 알 수 있다. 또한, 안전성을 위해서는 $\frac{q}{B} \leqq 2^{n^\epsilon}$, $(\epsilon < 1)$ 부등식을 만족해야 한다.

따라서 $4(2m)^d \leqq \frac{q}{B} \leqq 2^{n^\epsilon}$가 되고, 여기서 $4(2m)^d \leqq 2^{n^\epsilon}$을 d에 관해 풀면, $d \leqq c\frac{n^\epsilon}{\log_2 n}$(c는 상수)의 관계식을 얻을 수 있다.

4.3. 부트스트래핑(bootstrapping)

가장 우선으로 부트스트래핑이 필요한 이유는 이 과정을 하지 않고서는 잡음의 증가로 인해 더 이상 correctness를 보장할 수 없게 된다. Leveled 동형암호를 유지하면서 위와 같은 문제에 대한 대책으로 서버가 연산 중간값을 사용자에게 보내고 사용자는 이를 복호화하여 잡음을 제거한 뒤 서버에게 다시 암호화된 중간값을 송신하는 방법을 생각할 수 있다.

하지만 위와 같은 방법은 크게 두 가지 문제가 있다. 첫 번째로 문제가 되는 것은 양쪽에서 지속적으로 통신을 해야 한다는 것이다. 또한 semi-honest 또는 malicious한 사용자는 서버가 계산한 중간값의 정보를 알게 되면서 서버가 만든 동형함수 f가 어떻게 구성되었는지에 대한 정보를 알게 된다. 결국 이는 서버의 회로 프라이버시(circuit privacy) 침해 문제가 된다.

부트스트래핑의 아이디어는 사용자가 서버에게 암호화된 비밀키 $Enc_{sk}(\vec{s})$를 본인의 비밀키 $sk = \vec{s}$로 암호화하여 보낸다. 여기서 서버는 사용자에게 이미 전에 받은 암호화된 x, 즉, 암호문 $c = Enc_{sk}(x)$에 대해서 사용자가 원하는 연산 f을 깊이 d만큼 진행하였다고 가정하자. 여기서 쌓인 잡음의 값은 $(2m)^d B$가 될 것이고 d만큼 진행된 암호문의 중간값을 c'이라고 하자. 부트스트래핑은 이 c'을 기존에 있는 복호화 회로를 돌려 노이즈를 제거하는 것이다. 즉, 복호화 회로는 $\vec{s}$를 받아 $Dec(\vec{s}, c') = x'$의 평문을 내보내는 것이 아닌 암호화된 비밀키 $Enc_s(\vec{s})$를 입력으로 받고 $Dec(Enc(\vec{s}), c') = c''$을 내보내게 된다. 여기서 c''은 x'의 암호화된 값에 Dec 회로를 돌고 난 후 추가된 노이즈 $(2m)^{d_{dec}} B$ 값을 의미한다. 따라서, 이 노이즈 $(2m)^d B$가 제거되면서 생기는 새로운 노이즈 $(2m)^{d_{dec}} B$가 더 작다면, 부트스트래핑 과정에 따른 시간이 다소 걸리겠지만, 서버는 이론적으로 무한정 연산을 할 수 있게 된다.

Ⅴ. 결 론

본 논문에서는 각광 받는 기술인 완전동형암호를 기저의 LWE 어려움 문제에서부터 출발하여 동형적 덧셈과 곱셈, 부트스프래핑까지의 개념에 대해 간략히 알아보았다. 이 내용들을 밑바탕으로 하는 다양한 동형암호 스킴들이[18-19] 파라메타, 메모리, 속도 등의 최적화와 관련되어 활발한 연구가 진행되고 있다. 내부적 알고리즘뿐만 아니라 다양한 응용분야인 의료[20] 또는 금융[21] 등에서도 동형암호를 접목하여 암호화된 데이터를 분석하는 시도가 점점 늘고 있다.

끝으로, 동형암호 또는 격자에 관한 심도 있는 내용은 UC Berkley 대학의 Simons 기관에서 주최하는 웹사이트에서 다양한 강의를 참조[22]하거나 Vaikuntanathan[23]의 동형암호 리뷰에 관한 내용을 읽으면 많은 도움이 될 것으로 생각된다.

참 고 문 헌

[1] Shor, Peter W. "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer." *SIAM review 41*, no. 2, 303-332, 1999

[2] E. Kushilevitz and R. Ostrovsky, “Replication is not needed: Single database, computationally-private information retrieval,” in *FOCS*, pp. 364‐373, 1997

[3] Yoo, Joon Soo, et al. "A Bitwise Logistic Regression Using Binary Approximation and Real Number Division in Homomorphic Encryption Scheme." *International Conference on Information Security Practice and Experience*. Springer, Cham, 2019

[4] Yoo, Joon Soo, Baek Kyung Song, and Ji Won Yoon. "Logarithm design on encrypted data with bitwise operation." *International Workshop on Information Security Applications*. Springer, Cham, 2018.

[5] Song, Baek Kyung, et al. "A Bitwise Design and Implementation for Privacy-Preserving Data Mining: From Atomic Operations to Advanced Algorithms." *Security and Communication Networks 2019*

[6] Hong, Mi Yeon, Joon Soo Yoo, and Ji Won Yoon. "Homomorphic Model Selection for Data Analysis in an Encrypted Domain." *Applied Sciences* 10.18, 2020

[7] Boura, Christina, Nicolas Gama, and Mariya Georgieva. "Chimera: a unified framework for B/FV, TFHE and HEAAN fully homomorphic encryption and predictions for deep learning." *IACR Cryptol*. ePrint Arch. 2018

[8] S. Arora and R. Ge. “New algorithms for learning in presence of errors.” In L. Aceto, M. Henzinger, and J. Sgall, editors, *ICALP*, volume 6755 of Lecture Notes in Computer Science, pages 403‐415. Springer Verlag, 2011

[9] Regev, Oded. "The learning with errors problem." Invited survey in *CCC 7*, 2010

[10] Regev, Oded. On lattices, learning with errors, random linear codes, and cryptography. *Journal*

of the ACM, 56(6):34, 2009

[11] T. E. Gamal, “A public key cryptosystem and a signature scheme based on discrete logarithms,” in *CRYPTO*, pp. 10 – 18, 1984

[12] P. Paillier, “Public-key cryptosystems based on composite degree residuosity classes,” in *EUROCRYPT*, pp. 223 – 238, 1999

[13] M. Ajtai and C. Dwork, “A public-key cryptosystem with worst-case/average-case equivalence,” in *STOC*, pp. 284 – 293, 1997

[14] Regev, Oded. “New lattice-based cryptographic constructions,” J. *ACM*, vol. 51, no. 6, pp. 899 – 942, 2004.

[15] C. Gentry, “A fully homomorphic encryption scheme,” Ph.D. dissertation, Stanford University, http://crypto.stanford.edu/craig, 2009

[16] C. Gentry, “Fully homomorphic encryption using ideal lattices,” in *STOC*, pp. 169 – 178, 2009

[17] C. Gentry, “Implementing gentry’s fully-homomorphic encryption scheme,” in EUROCRYPT, ser. Lecture Notes in Computer Science, K. G. Paterson, Ed., vol. 6632. Springer, pp. 129 – 148, 2011

[18] Z. Brakerski and V. Vaikuntanathan, “ Fully homomorphic encryption from ring-LWE and security for key dependent messages,” in *CRYPTO*, vol. 6841, p.501, 2011

[19] Chillotti, I., Gama, N., Georgieva, M. and Izabachène, M., 2020. TFHE: fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1), pp.34-91.

[20] Dowlin, N., Gilad-Bachrach, R., Laine, K., Lauter, K., Naehrig, M. and Wernsing, J., Manual for using homomorphic encryption for bioinformatics. *Proceedings of the IEEE*, 105(3), pp.552-567, 2017

[21] Brenner, Michael, Tyler Moore, and Matthew Smith, eds. *Financial cryptography and data security*. Springer, 2014

[22] https://simons.berkeley.edu/

[23] Vaikuntanathan, Vinod. "Computing blindfolded: New developments in fully homomorphic encryption." 2011 IEEE 52nd *Annual Symposium on Foundations of Computer Science*. IEEE, 2011

〈저자소개〉

유 준 수 (Joon Soo Yoo)

2017년 9월 : 고려대학교 수학과 졸업

2020년 2월 : 고려대학교 정보보호학과 석사

2020년 9월~현재 : 고려대학교 정보보호학과 박사과정

<관심분야> 동형암호, 암호분석, 기계학습 등

윤 지 원 (Jiwon Yoon)

1995~2003년 : 성균관대학교 정보공학 학사

2003~2004년 : School of Informatics, University of Edinburgh 석사

2005～2008년 : Statistical Signal Processing, University of Cambridge 박사

2016년 3월~현재 : 고려대학교 정보보호대학원 부교수

<관심분야> 신호분석, 암호분석, 인공지능, OSINT