Homomorphic Encryption 의 기술 동향 및 전망

조남수
한국전자통신연구원 선임연구원
nsjho@etri.re.kr
장구영
한국전자통신연구원 책임연구원

1. 개요

매년 MIT 는 향후 큰 영향력을 끼칠 것으로 예상하는 10 대 최신 기술을 선정하여 발표하며, 2011 년에는 cancer genomics, cloud streaming, crash-proof code, gestural interfaces, homomorphic encryption, social indexing, smart transformers, solid-state batteries, separating chromosomes, synthetic cells 의 10 가지 기술이 선정되었다[1]. 이 기술들을 살펴보면 현재 세계적으로 가장 큰 이슈가 되고 있는 소셜 인덱싱과 클라우드 스트리밍 등의 기술을 비롯한 전기 자동차를 위한 배터리 기술 등이 미래 사회에 큰 영향을 끼칠 것으로 여겨지는 중요한 기술들에 포함되어 있다. 하지만, 이 중 homomorphic encryption 의 경우 많은 사람들에게 생소할 것으로 예상된다. 본 고는 이러한 homomorphic encryption 의 정의와 연구 동향 그리고 앞으로의 전망을 소개하려고 한다.

최근 스마트폰으로 대표되는 스마트 기기의 사용이 급증하면서 컴퓨팅 환경이 급속하게 변화해 가고 있다. 또한, 이와 맞물려 아마존, Google 과 IBM 등의 선도적 기업을 중심으로 클라우드 컴퓨팅 환경 구축을 위한 움직임이 활발히 진행되고 있다. 클라우드 컴퓨팅 환경은 현재 개인 컴퓨터가 제공하는 모든 작업 환경을 클라우드 서버를 통해 서비스로 제공하는 것을 목표로 한다. 따라서, 미래의 컴퓨팅 환경은 현재 개개인이 개별적으로 PC 시스템을 구축하고 관리하는 컴퓨팅 환경에서 클라우드 서버에 개인의 모든 자료를 저장하고

* 본 내용과 관련된 사항은 한국전자통신연구원 조남수 선임연구원 (☎ 042-860-1860)에게 문의하시기 바랍니다.
** 본 내용은 필자의 주관적인 의견이며 NIPA 의 공식적인 입장이 아님을 밝힙니다.

개인이 컴퓨팅 환경을 필요로 하는 경우에 스마트폰 등과 같은 단말기를 통해 언제 어디에서라도 서버를 통해 저장된 자료에 접속하고, 또한 서버의 컴퓨팅 능력을 서비스로 제공받는 환경으로 진화할 것으로 예상된다.

하지만, 이러한 목표를 이루기 위해서 우선적으로 해결해야 하는 큰 문제가 있다. 최근 빈번히 발생하는 기업체의 고객 정보 유출 및 개인 사용자들의 위치 정보를 비롯한 여러 개인 정보 유출 등의 일련의 사건을 통해서 현재 데이터베이스 서버가 제공하는 데이터 보호 기술에는 한계가 있다는 것을 알 수 있다. 이러한 문제를 해결하기 위해서 일부 데이터베이스 서버는 데이터를 암호화하고 있다. 데이터를 암호화하는 경우에는 해킹 등을 통해서 데이터가 외부에 유출되는 경우에도 개인 또는 데이터 서버에서 복호화를 위한 비밀키를 안전하게 보관하는 것 만으로 데이터의 기밀성을 유지할 수 있다. 그러나, 암호화된 데이터에 대한 연산을 비롯한 기본적인 검색 기능 조차 완벽하게 제공하는 암호화 기법은 현재 존재하지 않기 때문에 단순 데이터 저장 이외의 용도로 사용하기에는 데이터의 활용 측면에서 큰 부작용이 생겨나게 된다. 그리고 클라우드 컴퓨팅 환경을 고려해 보면, 모든 데이터에 대한 작업을 서버에서 수행하기 때문에 일반적인 암호화 기법을 사용하여 암호화된 데이터를 가지고 클라우드 서버에서 작업을 수행하기 위해서는 복호화 과정이 선행되어야 한다. 이렇게 임시로 복호화된 데이터가 클라우드 서버에 의해서 임의로 저장되거나 또는 작업 도중에 복호화된 데이터가 유출될 수 있는 등의 여러 문제점을 지니고 있어 일반적인 암호화 기법을 사용한 방식은 근본적인 해결책이라 할 수 없다.

이러한 배경에서 암호화된 데이터에 대해 복호화하지 않고 마음대로 연산을 수행할 수 있는 homomorphic encryption 이 최근 큰 주목을 받고 있으며, 미래 컴퓨팅 환경에 큰 영향을 끼칠 기술로 고려되고 있다. 사실 암호화된 데이터에 대한 연산을 수행하는 문제는 공개키 암호화 기법이 제안된 직후인 70 년대 후반부터 암호학자들 사이에서 큰 관심을 받아 왔으며 다양한 연구 결과들이 발표되었다. 하지만, 안전하면서도 만족할만한 기능을 제공하는 homomorphic encryption 기법은 오랫동안 제안되지 못했고, 이에 따라 homomorphic encryption 의 존재 가능성에 대한 부정적인 시각이 지배적이었다. 최근까지의 연구 결과는 제한된 연산 기능을 제공하는 부분적인 homomorphic encryption 이 전부였으며, 전자투표 등과 같은 특정 문제에 대한 해결책으로 연구가 수행되어 왔다.

이러한 homomorphic encryption 에 대한 연구 방향은 2009 년 Gentry 에 의해서 처음

으로 fully homomorphic encryption 기법이 제안되면서 크게 바뀌게 되었다[2]. Gentry 는 제한된 회수의 연산을 보존할 수 있는 부분적인 homomorphic encryption 기법(somewhat homomorphic encryption)을 바탕으로 부트스트랩핑(bootstrapping)과 스쿼싱(squashing) 방식을 통해 fully homomorphic encryption 을 구성하는 방식을 제안하였다. 그 이후 모든 homomorphic encryption 에 대한 연구는 Gentry 가 제시한 기본 틀을 바탕으로 효율성을 개선하는 것으로 초점이 맞춰져 있다.

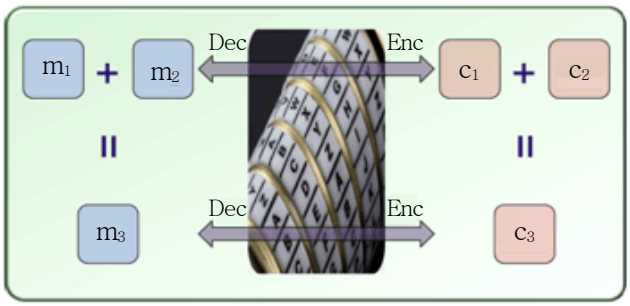
본 고의 이 후 부분에서는 homomorphic encryption 의 명확한 정의와 과거에 제안되었던 homomorphic encryption 기법에 대해 살펴보고, 최근의 연구 경향을 통해 앞으로의 homomorphic encryption 에 대한 연구 전망을 살펴보고자 한다.

2. 정의 및 기존의 연구

수학 분야에서 homomorphism 은 연산이 정의된 두 집합 사이의 맵핑(mapping)으로 두 집합에서 정의된 연산을 보존하는 맵핑을 의미한다. 암호화 함수 또한 평문 공간과 암호문 공간 사이의 맵핑으로 생각할 수 있으며, homomorphic encryption 이란 이러한 암호화 함수 중에서 평문 공간과 암호문 공간에 정의된 연산을 보존하는 암호화 함수라 할 수 있다. Homomorphic encryption 은 기본적으로는 대표적인 산술 연산인 덧셈 연산과 곱셈 연산을 암호문에 적용하여 평문에 대한 연산을 수행할 수 있도록 하는 암호화 기법이라 할 수 있다. 이것을 식으로 표현하면 다음과 같다.

$$E_k(m_1) + E_k(m_2) = E_k(m_1 + m_2), \quad E_k(m_1) \cdot E_k(m_2) = E_k(m_1 \cdot m_2)$$

이러한 기본적인 homomorphic encryption 과 차별화하여 모든 임의의 논리 연산을 보



(그림 1) Homomorphic Encryption 개념도

존하는 homomorphic encryption을 fully homomorphic encryption이라 부른다. 비트 단위의 'AND'와 'XOR' 연산이 주어지면 두 연산을 조합하여 임의의 논리 연산을 구성할 수 있기 때문에 결국 fully homomorphic encryption이란 비트 단위의 AND와 XOR 연산을 복호화 없이 암호문을 통해 수행할 수 있는 암호화 기법이라 할 수 있다. 모든 연산을 서버에서 수행하는 클라우드 컴퓨팅 환경 등의 데이터 프라이버시를 보호하기 위해서는 이러한 fully homomorphic encryption이 요구된다.

연산을 보존하는 암호화 기법은 RSA 공개키 암호 기법이 발표된 직후인 1978년 Rivest, Adleman, Dertouzous에 의해서 privacy homomorphism이라는 이름으로 처음 제안되었다[3]. Rivest 등은 RSA 암호 시스템을 변형한 기법을 포함한 다섯 가지 기법을 발표하였으나, 현실에 사용하기에는 안전성에 문제가 있었다. 여기에서는 우선 homomorphic encryption에 대한 이해를 돕기 위해 RSA 기법을 변형한 가장 간단한 형태의 homomorphic encryption 기법을 소개하도록 한다.

키 설정은 RSA 암호 시스템과 동일하게 두 개의 큰 소수 p, q를 선택하고, n = pq로 설정한다. 평문 $a \in Z_n$에 대한 암호화 과정은 다음과 같다.

$$E_k(a) = (c_1, c_2) = (a \pmod p + r_1 \times p,\ a \pmod q + r_2 \times q)$$

여기에서 r_1, r_2는 임의의 난수이다. p, q를 알고 있는 사용자는 (c_1, c_2)으로부터 난수 부분을 제거하여 $(c_1 \bmod p, c_2 \bmod q) = (a \bmod p, a \bmod q)$을 얻을 수 있고, 여기에서 중국인의 나머지 정리를 이용하여 a를 계산할 수 있다.

이 암호화 방법의 homomorphic 성질을 살펴보기로 하자. 두 평문 a_1과 a_2에 대한 두 암호문 $E_k(a_1) = (c_{11}, c_{12})$와 $E_k(a_2) = (c_{21}, c_{22})$가 주어진 경우, 두 암호문에 대해서 다음과 같이 연산을 수행할 수 있다.

$$\begin{aligned}
&- (c_{11}, c_{12}) + (c_{21}, c_{22}) = (c_{11} + c_{21},\ c_{12} + c_{22}) \\
&\qquad = ((a_1 + a_2) \pmod p + r_1' \times p,\ (a_1 + a_2) \pmod q + r_2' \times q) \\
&- (c_{11}, c_{12}) \times (c_{21}, c_{22}) = (c_{11} \times c_{21},\ c_{12} \times c_{22}) \\
&\qquad = ((a_1 \times a_2) \pmod p + r_1'' \times p,\ (a_1 \times a_2) \pmod q + r_2'' \times q)
\end{aligned}$$

하지만, 이 방식은 동일 평문에 대한 암호문이 다수 주어진 경우, $(c_{11}, c_{12}) - (c_{21}, c_{22})$를 계산하면 각각 $(r_1 p, r_2 q)$의 형태가 되어 쉽게 비밀키가 노출되는 단점을 지니고 있다. 이 후, 몇 차례 이와 비슷한 방식의 homomorhpic encryption이 제안되었으나, 모두 안전성에 문

제가 있는 것으로 판명되었다.

부분적인 homomorphic encryption 에 대한 연구도 진행되었다. 대표적인 예로 1999 년 Paillier 에 의해서 제안된 Paillier cryptosystem 를 들 수 있는데, 덧셈 연산과 상수곱 연산 기능을 제공한다[4]. Paillier cryptosystem 에서의 평문 m 에 대한 암호문은 난수 r 을 포함하여

$$c = g^m \cdot r^n \bmod n^2$$

의 형태로 이루어져 있으며, 두 암호문 c_1, c_2에 대해서 다음과 같은 연산을 수행하여,

$$c_1 \times c_2 = g^{m1} \cdot r_1^{\,n} \times g^{m2} \cdot r_2^{\,n} = g^{m1+m2} \cdot (r_1 \cdot r_2)n \bmod n^2$$

$$c^a = (g^m \cdot r^n)^a = g^{ma} \cdot (r^a)^n \bmod n^2$$

각각, m_1+m_2 와 $m \cdot a$ 에 대한 암호문을 얻을 수 있다. 또한, 2005 년에는 Boneh, Goh, Nissim 이 타원 곡선에서 정의된 겹선형 사상(bilinear map)을 이용하여 덧셈 연산과 한 번의 곱셈 연산이 가능한 homomorphic encryption 을 발표하였다[5]. Boneh 등이 제안한 암호화 방식에서 평문 m 에 대한 암호문은 겹선형 사상이 정의된 그룹에서

$$c = g^m \cdot h^r \bmod n$$

의 형태를 지니고 있는데, 두 암호문 c_1, c_2에 대해서 겹선형 사상 연산을 수행하여

$$e(c_1, c_2) = e(g^{m1} \cdot h^{r1}\ ,\ g^{m2} \cdot h^{r2}) = g_1^{\,m1m2} \cdot h_1^{\,r}$$

$m_1 \times m_2$에 대한 암호문을 얻을 수 있다, 여기에서 e 는 겹선형 사상, $g_1=e(g,g)$, $h_1=e(g,h)$, 그리고 r 은 계산에 의해 결정된 난수이다. 하지만, Boneh 등이 제안한 방식은 겹선형 사상의 특성에 의해서 단 한 번의 곱셈 연산만을 수행할 수 있는 제한적인 기법이다.

3. 최근 연구 동향 및 전망

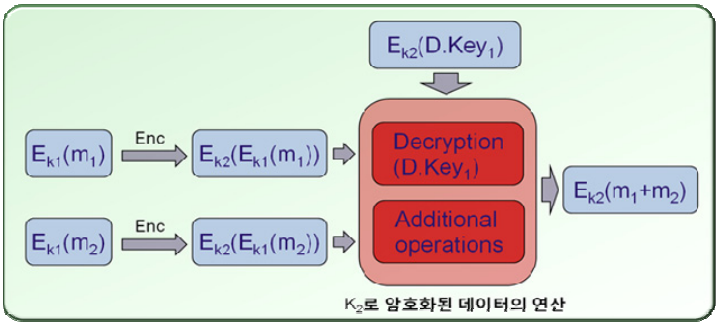
Homomorpic encryption 에 대한 최근의 연구는 기존의 기법들과 전혀 다른 방향으로 전개되고 있다. 2009 년 Gentry 는 최초로 안전성이 증명된 fully homomorphic encryption 기법을 제안하였다[2].

Gentry 는 우선 homomorphic 성질을 자연스럽게 만족하도록 래티스(lattice)에서의 간단한 연산을 사용한 somewhat homomorphic encryption 을 구성하였다. 여기에서 somewhat homomorphic encryption 이란, 몇 번의 연산만이 보존되는 homomorphic encryption 이

다. Gentry 가 구성한 somewhat homomorphic encryption 은 간단한 연산을 바탕으로 encryption 을 구성하였으며, 암호 시스템에서 기밀성을 유지하기 위해서 난수화된 에러를 포함하는 방식을 사용하였다. 이러한 구성을 통해서 암호문 사이의 연산이 자연스럽게 정의가 된다. 하지만, 여러 번의 연산을 수행하는 과정에서 암호문에 포함된 에러가 점차 증폭되게 되고, 이 에러의 크기가 일정 수준을 넘어서면 정확한 복호화가 불가능하게 된다.

Gentry 는 이러한 somewhat homomorphic encryption 을 바탕으로 하여 fully homomorphic encryption 을 구성하는 방식을 제안하였으며, 이 과정을 부트스트랩핑이라 부른다. 부트스트랩핑은 somewhat homomorphic encryption 의 경우, 암호화된 상태에서 일정 회수까지 자유로운 연산이 가능하기 때문에 만약 복호화에 필요한 연산이 somewhat homomorphic encryption 에서 제공하는 연산 회수보다 작은 경우, 복호화 연산 또한 homomorphic 하게 수행할 수 있다는 결론을 얻게 된다. 이러한 과정을 위해 복호화에 필요한 복호화 키를 새로운 암호화 키로 암호화하여 제공하고 연산하고자 하는 암호문 또한 새로운 암호화 키로 재 암호화한 상태에서 내부 연산을 수행하게 된다. 이러한 부트스트랩핑 과정을 반복적으로 수행하여 원하는 임의의 연산을 수행할 수 있는 fully homomorhpic encryption 을 구성할 수 있는 것이다.

하지만, 현재까지 제안된 somewhat homomorphic encryption 은 복호화에 필요한 연산 회수보다 작은 연산만을 보존할 수 있기 때문에 부트스트랩핑 성질을 만족시키지 못하고 있다. 이러한 면을 극복하기 위한 방법이 스쿼싱 과정이다. 스쿼싱 과정은 복호화에 필요한 정보 및 복호화의 일부 과정을 미리 수행하여 공개키와 암호문으로 제공하는 것으로 복호화에 필요한 연산의 계산 복잡도를 크게 줄여주기 때문에 적은 연산을 제공하는



(그림 2) Bootstrapping

somewhat homomorphic encryption에 대해서도 fully homomorphic encryption을 구성할 수 있도록 한다.

Gentry가 제안한 homomorphic encryption 구성 방법은 그 후의 모든 homomorphic encryption 연구의 기본 골격으로 활용되고 있다. 하지만, Gentry가 초기에 제안한 기법은 ideal lattice 등을 사용하여 somewhat homomorphic encryption을 구성하였기 때문에 관련 기반 지식 없이 이해하기에는 매우 난해하여 본 고에서는 Gentry가 제안한 기법에 대한 세부 설명은 생략하기로 한다. 대신, 2010년 Dijk, Gentry, Halevi, Vaikuntanathan이 Gentry의 구성 방법을 정수 집합에 유사하게 적용하여 구성한 보다 간략한 기법을 소개하기로 한다[6].

다음은 Dijk 등이 제안한 somewhat homomorphic encryption이다.

① 키 설정

- 주어진 안전성 파라미터 λ로부터 다음 변수들을 결정한다.
  (η: 비밀키의 비트 길이, ρ: 에러의 비트 길이, γ: 에러를 포함하는 공개키 x_i들의 비트 길이, τ: 공개키인 x_i들의 수, ρ': 암호화에 사용되는 이차 에러의 비트 길이)
- 임의의 η-비트 길이의 홀수 p를 선택한다.
- $0 \le i \le \tau$ 에 대해서 공개키로 사용할 정수 x_i를 다음과 같이 결정한다.

$$x_i = q_i \cdot p + r_i$$

  여기에서, q_i는 $2^{\gamma-\eta}$ 보다 작은 임의의 정수이고, r_i는 $2^{-\rho} \le i \le 2^{\rho}$를 만족하는 에러 값이다.
- 선택된 x_i들을 크기 순으로 재정렬하여 x_0가 가장 큰 값이 되도록 한다, 여기에서 x_0는 홀수이며 '$x_0 \pmod p$ = 짝수'를 만족하는 것을 가정한다.
- 최종 공개키는 $(x_0, x_1, \cdots, x_\tau)$이고 비밀키는 p이다.

② 암호화 단계

- 평문 m은 0 또는 1의 값으로 가정한다. 즉, $m \in \{0,1\}$이다.
- $1 \le j \le \tau$ 를 만족하는 j를 임의로 선택하고, 공개키 $(x_1, \cdots, x_\tau)$ 중에의 j개의 원소들을 임의로 선택한다. 선택한 원소들을 $(x_1', \cdots, x_j')$라 가정한다.
- 또한, ρ'-비트 길이를 가지는 임의의 난수 r을 선택한다.
- 마지막으로, m에 대한 암호문은 다음과 같이 계산된다.

$$c = m + 2r + 2\Sigma_{0\le i\le j} x_i' \pmod{x_0}$$

③ 복호화 단계

- 주어진 암호문 c 에 대해서 비밀키 p 를 이용하여 다음과 같이 복호화를 수행한다.

$$m = (c \bmod p) \bmod 2$$

여기서 소개한 somewhat homomorphic encryption 의 암/복호화 과정을 좀 더 자세히 살펴보면 다음과 같다. 암호화 단계에서 평문 m 은 난수인 r 과 임의로 선택된 공개키들의 합에 의해서 은닉된다. 공개키인 x_i 들은 각각 $x_i = q_i \cdot p + r_i$ 으로 선택되어 있기 때문에 복호화 과정의 첫 번째 단계인 (c mod p) 계산에 의해서 $q_i \cdot p$ 부분이 소멸되고 에러 부분인 r_i 만 남게 된다. 암호화 단계를 살펴보면 이러한 에러 부분은 모두 상수 2 가 곱해져 있음을 알 수 있으며, 따라서 복호화의 두 번째 단계인 mod 2 계산에 의해서 모두 소거된다.

또한, 두 평문 m_1, m_2 에 대한 암호문 $c_1 = m_1 + 2r_1 + 2\Sigma x_i \pmod{x_0}$, $c_2 = m_2 + 2r_2 + 2\Sigma x_i \pmod{x_0}$ 에 대해서 다음과 같이 연산을 수행할 수 있게 된다.

$$\begin{aligned} c_1 + c_2 &= m_1 + 2r_1 + 2\Sigma x_i \pmod{x_0} + m_2 + 2r_2 + 2\Sigma x_i \pmod{x_0} \\ &= (m_1 + m_2) + 2(r_1 + r_2) + 2\Sigma x_i \pmod{x_0} \\ c_1 \times c_2 &= (m_1 + 2r_1 + 2\Sigma x_i \pmod{x_0}) \times (m_2 + 2r_2 + 2\Sigma x_i \pmod{x_0}) \\ &= (m_1 \times m_2) + 2(m_2 r_1 + m_1 r_2 + 2r_1 r_2 + \Sigma a_i x_i) \pmod{x_0} \end{aligned}$$

위 수식에서 알 수 있듯이, 이러한 연산을 수행하는 과정을 통해서 암호문에 포함된 에러의 크기가 점점 커지게 된다. 특히, 곱셈 연산의 경우에는 이 에러가 매우 급격히 증가하고, 이 에러의 크기가 일정 범위를 넘어서게 되면 정확한 복호화가 불가능하게 되어 극히 제한된 회수의 연산 만이 가능하다. 정확한 계산이 필요하기는 하지만, 이 somewhat homomorphic encryption 기법 또한 부트스트랩핑의 성질을 만족하지 못하고, fully homomorphic encryption 을 구성하기 위해서는 Gentry 가 2009 년에 소개한 스쿼싱 방식을 사용하여 복호화 함수를 단순하게 변형할 필요가 있다.

스쿼싱 과정은 복호화에 필요한 특정 정보를 공개키 및 암호문의 일부로 제공하여 복호화 과정을 단순화하고, 이 단순화를 통해 somewhat homomorphic encryption 이 부트스트랩핑의 성질을 만족하게 변형하는 작업이다. 앞에서 소개된 somewhat homomorphic encryption 의 복호화 계산 과정을 살펴보면

$$m = (c \bmod p) \bmod 2 = (c - p \cdot [c/p]) \bmod 2$$

이고, p 가 홀수이므로, (c − [c/p]) mod 2 로 정리할 수 있다, 단 여기에서 [x]는 x 에 가장 가까운 정수를 의미한다. 스퀴싱 과정을 통해 이 복호화 과정에서 가장 많은 연산이 요구되는 [c/p]의 계산을 보다 단순한 계산으로 대체한다. 기본 아이디어는 다음과 같다. 만약 $\Sigma y_i = 1/p \pmod 2$을 만족하는 y_i가 주어진다면, $c/p = c \cdot \Sigma y_i \pmod 2 = \Sigma c y_i \pmod 2$ 계산을 통해서 c/p 의 값을 계산할 수 있으므로, 이러한 y_i들을 공개키로 제공하고 암호화 단계에서는 cy_i 들을 추가로 계산하여 암호문에 포함시킨다. 복호화 과정에서는 주어진 암호문으로부터 쉽게 c/p 를 계산할 수 있다. 실제 스퀴싱 과정에서는 공개키인 y_i 들로부터 비밀키인 p 를 유추할 수 없도록 좀 더 복잡한 과정을 거치게 된다.

구체적인 스퀴싱 과정을 살펴보면, 키 설정 단계에서 y_i를 계산하는 부분과 암호화 단계에서 z_i 의 계산이 추가되며, 마지막 복호화 방법이 새로운 방법으로 대체되는 것을 제외하고 나머지 부분의 경우는 somewhat homomorphic encryption 과 기본적으로 동일하다.

키 설정 단계에 추가되는 부분은 다음과 같다. 안전성 변수 Θ, κ를 정하고 κ-bit 로 표현되는 실수 $y_i \in [0,2)$를 Θ개 임의로 선택한다, 단 집합 $S \subset \{1, 2, \cdots, \Theta\}$에 대해서 $|\Sigma_{i \in S} y_i = 1/p| < 2^{-\kappa}$를 만족하는 것을 가정한다. y_i들을 공개키로 공개하고, 집합 S 는 비밀키가 된다. 암호화 단계에서는 암호문 c 를 계산한 후에, 각각의 y_i에 대해서 $z_i = c \cdot y_i \pmod 2$ 를 계산하는 과정이 추가되며 암호문으로 c 와 $\{z_1, z_2, \cdots, z_\Theta\}$를 동시에 제공한다. 마지막으로 복호화 과정은 $c-[\Sigma_{i \in S} z_i] \pmod 2$의 연산으로 대체된다. 이 스퀴싱 과정을 통해서 주어진 somewhat homomorphic encryption 이 부트스트랩핑의 성질을 만족하게 되며, 여기에서는 엄밀한 증명 과정은 생략한다.

하지만, 이러한 스퀴싱 과정을 통해 y_i 가 공개키로 제공되어 안전성 수준에 따라 수십 MB 에서 많게는 수 GB 에 이르는 공개키가 요구된다. 암호문의 경우에도 추가적인 z_i들을 암호문의 일부로 제공하여야 하기 때문에 암호문의 크기도 문제가 된다. 또한, 스퀴싱 과정을 위한 z_i 를 계산하기 위해 소요되는 시간 또한 수 초에서 수십 분이 추가로 소요되어 1bit 단위로 처리되는 데이터 연산을 위한 처리 시간이 극단적으로 비효율적이라는 문제점을 지니고 있다.

2009 년 이후 현재까지의 주요 연구 방향은 Gentry 가 제안한 스퀴싱 및 부트스트랩핑 방식을 통한 fully homomorphic encryption 의 구성이라는 큰 틀을 벗어나지 않고 보다 간단한 연산 등을 사용하여 암/복호화 단계를 단순화시키고 이를 통해 효율성 개선하는 방향과

Gentry 등이 somewhat homomorphic encryption 구성에 사용한 새로운 수학적 난제의 안전성 분석 등에 대한 연구가 주를 이루었다[7],[8]. 하지만, 최근에는 Gentry 의 방식에서 스쿼싱 과정에 의한 근본적인 비효율성 문제를 극복하기 위해서 부트스트랩핑 및 스쿼싱 과정을 제거하는 구성 방법이 제안되고 있으며, 앞으로의 homomorphic encryption 연구의 주요 연구 방향이 될 것으로 전망된다[9],[10].

4. 결언

Homomorphic encryption 은 암호화된 상태의 데이터에 대한 연산을 추가적인 복호화 과정 없이 수행할 수 있도록 하는 암호화 기법으로, 미래 클라우드 컴퓨팅을 비롯한 여러 데이터 프라이버시 보호가 요구되는 응용 분야에 널리 사용될 수 있는 기반 기술로 현재 여러 분야의 큰 주목을 받고 있다. 1978 년 처음으로 제안되었으나, 최근까지 부분적인 homomorphic 연산을 제공하는 수준에 머물러 있었다. 2009 년 Gentry 에 의해서 처음으로 모든 연산을 보존하는 fully homomorphic encryption 이 제안되고 또한 fully homomorphic encryption 을 구성하는 일반적인 방법이 제시되면서 이러한 기법을 바탕으로 다시 homomorphic encryption 에 대한 연구가 활발히 진행되었다. 하지만, 지금까지 알려진 homomorphic encryption 은 모두 극단적인 효율성 문제를 안고 있으며 실제 응용 분야에서 활용하기는 불가능하다고 할 수 있다. 현재에는 이러한 효율성 문제를 극복하기 위해서 다양한 방법이 연구되고, 특히 Gentry 가 제시한 구성 방법을 벗어난 새로운 구성 방법들에 대한 연구가 주요 연구 주제가 될 것으로 전망된다.

<참 고 문 헌>

[1] http://www.technologyreview.com/tr10/

[2] Ronald L. Rivest, Len Adleman, and Michael L. Dertouzos, “On data bank and privacy homomorphisms”, in Proceedings of the 19th Annual Symposium on Foundations of Secure Computation – FSC 1978, Academic Press, 1978, pp.169-180.

[3] Pascal Paillier, “Public-key cryptosystems based on composite degree residuosity classes”, LNCS vol.1592 Advances in Cryptology – Eurocrypt 1999, Springer, 1999. 5, pp.223-238.

[4] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim, “Evaluating 2-DNF formulas on ciphertexts”, LNCS vol.3378 Advances in Cryptology – Crypto 2005, Springer, 2005. 8, pp.325-341

[5] Craig Gentry, “Fully homomorphic encryption using ideal lattices”, in Proceedings of the 41st

ACM Symposium on Theory of Computing – STOC 2009, ACM, 2009. 05, pp.169-178.

[6] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan, "Fully homomorphic encryption over the integers", LNCS vol.6110 Advances in Cryptology–Eurocrypt 2010, Springer, 2010. 5, pp.24-43.

[7] Craig Gentry, and Shai Halevi, "Implementing Gentry's fully homomorphic encryption scheme", LNCS vol. 6632 Advances in Cryptology – Eurocrypt 2011, Springer, 2011. 5, pp.129-149.

[8] Zvika Brakerski, and Vinod Vaikuntanathan, "Fully homomorphic encryption from ring-LWE and security for key dependent messages", LNCS vol. 6841 Advances in Cryptology - Crypto 2011, Springer, 2011. 8, pp.505-524.

[9] Craig Gentry, and Shai Halevi, "Fully homomorphic encryption without squashing using depth-3 arithmetic circuits", in Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science – FOCS 2011, IEEE, 2011. 10, pp.107-116.

[10] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan, "Fully homomorphic encryption without bootstrapping", cryptology eprint archive - http://eprint.iacr.org/2011/277.pdf