http://dx.doi.org/10.5392/JKCA.2013.13.08.036

완전동형암호 기술의 연구 동향

Technical Trend of Fully Homomorphic Encryption

정명인
육군사관학교 수학과

Myoung In Jeong(mangjjj@naver.com)

요약

완전동형암호는 암호화된 자료를 복호화 하는 과정을 거치지 않고 원하는 자료를 검색 및 통계처리 할 수 있도록 하는 암호기반 기술이다. 완전동형암호는 암호화 및 복호화에 소요되는 시간을 줄여줌으로써 검색 속도를 향상시키고 통계 처리를 위해 복호화 된 자료의 유출로 인한 피해를 막을 수 있는 기술로 주목받고 있다. 또한 최근 보편화 되고 있는 클라우드 컴퓨팅 환경에서 개인 정보가 외부의 저장 공간에 저장됨으로써 발생할 수 있는 여러 가지 문제점을 해결해 줄 것으로 기대를 모으고 있다. 완전동형암호가 처음 제안된 70년대 이후 지금까지 효율성과 기능성을 만족시키는 알고리즘을 개발하기 위한 연구가 계속되어 왔다. 본 고에서는 최근 활발한 연구가 진행되고 있는 완전동형암호의 연구 방향에 대해서 살펴보고자 한다.

■ 중심어 : | 완전동형암호 | 클라우드 컴퓨팅 |

Abstract

Fully homomorphic encryption is a cryptography system in which coded data can be searched and statistically processed without decryption. Fully homomorphic encryption has accelerated searching speed by minimizing time spent on encryption and decryption. In addition, it is also known to prevent leakage of any data decoded for statistical reasons. Also, it is expected to protect personal information stored in the cloud computing environment which is becoming commercialized. Since the 1970s when fully homomorphic encryption was first introduced, it has been researched to develop the algorithm that satisfy effectiveness and functionality. We will take the reader through a journey of these developments and provide a glimpse of the exciting research directions that lie ahead.

■ keyword : | Fully Homomorphic Encryption | Cloud Computing |

I. 서 론

최근 스마트폰 및 다양한 스마트 기기의 사용이 확산되면서 클라우드 컴퓨팅이 보편화 되어가고 있다. 클라우드 컴퓨팅은 정보가 인터넷 상의 서버에 영구적으로 저장되고, 데스크톱·태블릿컴퓨터·노트북·넷북·스마트폰 등의 IT 기기 등과 같은 클라이언트에는 일시적으로 보관되는 컴퓨터 환경을 뜻한다. 즉 이용자의 모든 정보를 인터넷 상의 서버에 저장하고, 이 정보를 각종 IT 기기를 통하여 언제 어디서든 이용할 수 있게 하는 개

접수일자 : 2013년 05월 31일
수정일자 : 2013년 07월 04일
심사완료일 : 2013년 07월 15일
교신저자 : 정명인, e-mail : mangjjj@naver.com

념이다. 클라우드 컴퓨팅을 도입하면 컴퓨터 시스템을 유지·보수·관리하기 위하여 들어가는 비용과 서버의 구매 및 설치 비용, 업데이트 비용, 소프트웨어 구매 비용 등의 엄청난 비용과 시간·인력을 줄일 수 있다는 장점이 있다. 그러나 서비스 업체 관계자는 언제든지 제한 없이 자료에 접근이 가능하고, 자료가 인터넷 상의 서버에 저장되기 때문에 해커에 의해 해킹당할 수도 있다는 취약점이 있다. 이러한 단점을 보완하기 위해 사용자가 자신의 자료를 암호화해서 서버에 저장할 수 있는데, 이럴 경우 암호화된 상태로는 필요한 자료를 찾거나 자료를 통계적으로 처리하고 가공할 수 없다는 단점이 있다. 그리고 클라우드 컴퓨팅 환경에서 모든 데이터 처리는 서버에서 이루어지기 때문에 일반적인 암호체계를 이용하여 클라우드 서버에서 작업을 하기 위해서는 사전에 복호화 과정을 거쳐야 한다. 이럴 경우 자료를 복호화 하여 처리하는 과정에서 임시로 클라우드 서버에 복호화 된 자료가 임의로 저장되거나 유출될 수 있는 위험이 존재한다.

최근에 빈번하게 발생하고 있는 기업체의 고객 개인정보 유출이나 공공기관 데이터베이스 유출 등의 사건을 보면, 현재의 데이터베이스 보안 기술이 개인 정보를 보호하는데 충분하지 않다는 것을 알 수 있다.

표 1. 주요 기업 개인정보 유출사고 **(단위=만명)**

연도	기업	규모
2006~2008	하나로텔레콤	650
2008년 2월	옥션	1863
2008년 9월	GS 칼텍스	1119
20010년 3월	신세계몰	330
2011년 4월	현대캐피탈	175
2011년 7월	SK 컴즈	3500
2011년 11월	넥슨	1320
2012년 4월	EBS	400
2012년 7월	KT	870

현재 개인정보 보호법상으로 모든 개인정보는 암호화 하여 저장하고 인가받지 않은 인원의 접근을 통제하여 안전하게 보호하도록 규정되어 있다. 데이터를 암호화하여 저장하는 경우 해킹을 통해서 외부에 자료가 유출되더라도 개인이 암호화에 사용된 비밀키만 안전하게 보관하고 있으면 자료 유출에 대해 걱정하지 않아도 된다. 그러나 자료를 통계적으로 처리하거나 키워드를 검색하기 위해서는 대용량의 데이터를 모두 복호화 하여 처리해야 하기 때문에 많은 시간이 소요되고, 또 그 과정에서 복호화 된 자료가 유출될 수 있다는 위험이 있다. 따라서 데이터베이스를 암호화 한 상태로 사용자가 원하는 통계적 처리를 하거나 기본적인 키워드 검색을 할 수 있는 암호체계의 필요성이 대두되었고 현재 활발한 연구가 진행되고 있다.

암호화된 데이터를 복호화 하지 않은 상태에서 원하는 횟수만큼 검색하고 통계처리를 할 수 있도록 하는 대표적인 암호는 fully homomorphic encryption으로 이것은 앞으로의 컴퓨팅 환경에 큰 영향을 미칠 기술로 주목을 받고 있다. fully homomorphic encryption은 암호화를 통해서 데이터의 안전성은 보장하면서 데이터 복호화 없이 다양한 서비스를 가능하게 하는 새로운 기술이다. MIT는 매년 향후 큰 영향력을 끼칠 것으로 예상되는 10대 유망 기술을 선정하여 발표하는데, homomorphic encryption은 2011년에 여기에 이미 선정된 바 있다. 사실 암호화된 데이터에 연산을 수행하는 문제는 공개키 암호가 처음 제안되었던 1970년대 후반부터 큰 관심을 받은 기술로 그 이후에도 다양한 연구결과들이 발표되었다. 하지만 안전성을 보장하면서도 사용자가 원하는 다양한 기능을 제공하는 완전동형암호 기술은 오랜 시간 제안되지 못했으며, 각국은 이러한 기술을 개발하기 위해 노력을 계속하였다. 미국의 경우 2011년부터 5년 동안 동형 암호를 개발하는 연구에만 2,000만 달러를 지원하였고, 영국은 2011년에만 100만 파운드를 지원하였다고 알려져 있다. 그러나 현재까지 만족할 만한 수준의 완전동형암호는 제안되지 못했으며, 제한된 횟수의 연산을 보존하는 기법만 제안되어 전자 투표 등에 국한된 문제에 대한 해결책을 제시하는 방안으로 연구가 수행되었다.

2009년, Gentry는 제한된 횟수의 연산만 가능하던 그때까지의 대부분의 연구 방향과는 달리 연산 횟수에 제한을 받지 않는 fully homomorphic encryption을 제안하였다. Gentry는 기존의 제한된 횟수의 연산을 보존하는 부분적인 동형암호(somewhat homomorphic

encryption)을 바탕으로 부트스트랩핑(bootstrapping)과 스쿼싱(squashing) 방법을 이용하여 fully homomorphic encryption을 완성하는 방식을 제안하였다. 그 후의 동형암호에 대한 연구는 이전과는 달리 Gentry의 방법을 바탕으로 효율성을 개선해 나가는 방향으로 주로 연구가 진행되었다.

II. 배경지식

homomorphic encryption에서 homomorphic 이라는 용어는 수학에서 사용하는 homomorphism에서 온 것으로, homomorphism은 연산이 정의된 두 개의 집합 사이에서의 연산을 보존하는 맵핑을 의미한다. 따라서 homomorphic encryption은 평문 공간과 암호문 공간이라는 두 개의 집합에서 대표적인 연산인 덧셈, 곱셈을 보존하는 암호체계이다. 쉽게 말해 평문에 연산을 하여 암호화한 것이나 평문을 암호화 한 것에 연산을 한 것이 같게 되는 암호체계이다. 덧셈과 곱셈 연산을 보존하는 homomorphic encryption을 식으로 표현하면 다음과 같다.

$$E(m_1)+E(m_2)=E(m_1+m_2)$$
$$E(m_1)\cdot E(m_2)=E(m_1\cdot m_2)$$

여기서 발전하여 상수곱, XOR, AND 등의 모든 논리연산을 보존하는 암호체계를 fully homomorphic Encryption이라고 한다.

연산을 보존하는 암호기법이 가장 먼저 제안된 시기는 공개키 암호가 발표된 직후인 1978년이다. Rivest, Adleman, Dertouzous가 제안한 privacy homomorphism을 시작으로 연구가 계속되어 2005년 Boneh, Goh, Nissm이 제안한 덧셈 연산과 한 번의 곱셈 연산이 가능한 somewhat homomorphism까지 활발한 연구가 진행되었다.

1. Privacy Homomorphism[RAD78]

Rivest, Adleman, Dertouzous은 공개키암호가 제안된 직후인 1978년에 연산을 보존하는 암호기법인 privacy homomorphism을 처음으로 발표하였다. 이 발표에는 RSA 암호시스템을 변형한 방법을 포함해 5가지 방법이 포함되었으나 안전성 문제가 해결되지 않아서 실제 사용은 제한되었다. RSA 암호시스템을 변형한 한 가지 형태의 Homomorphic Encryption 기법을 소개하면 다음과 같다.

- 두 개의 큰 소수 p와 q를 선택
- 공개키 : $n=pq$
- 암호화 :

$$E_k(a)=(a(\bmod p)+r_1\times p, a(\bmod q)+r_2\times q)$$
$$=(c_1, c_2)\quad (a\in Z_n)$$

- 복호화 : p와 q를 알기 때문에 난수 r_1과 r_2 제거 가능

$$D_{p,q}(c_1, c_2)=(c_1(\bmod p), c_2(\bmod q))=(a(\bmod p), a(\bmod q))$$

암호화 과정에서는 임의의 난수 r_1과 r_2가 사용되며, 복호화 과정에서 중국인의 나머지 정리를 적용하면 a를 계산할 수 있다. 그러나 위의 암호기법은 동일한 평문에 대해 서로 다른 두 개의 암호문 (c_{11}, c_{12})와 (c_{21}, c_{22})가 주어졌을 경우 두 암호문의 차를 계산하면 각각 (r_1p, r_2q) 형태가 되어 비밀키를 알 수 있다는 취약점이 있다. 따라서 안전성의 문제로 인해 현실에서의 사용은 제한되었다.

2. Paillier Cryptosystem[Pai99]

Privacy Homomorphism에 이어 부분적인 연산을 보존하는 암호(somewhat homomorphic encryption)에 대한 연구가 계속되어 XOR, AND, 덧셈, 덧셈과 곱셈 등의 연산을 일부 보존하는 암호가 잇따라 발표되었다. 1999년 Paillier은 덧셈과 상수곱 연산을 가능하게 하는 암호체계인 Paillier Cryptosystem을 제안하였다.

- 평문 m, 난수 r
- 암호화 : $c=g^m r^n \pmod{n^2}$, $g=1+n$

평문 m_1, m_2에 대한 두 암호문 c_1, c_2에 대해 다음과 같이 덧셈과 상수곱 연산을 가능하게 하여 m_1+m_2, $m \cdot a$에 대한 암호문을 얻을 수 있다.

$$c_1 \times c_2 = g^{m_1} r_1^n \times g^{m_2} r_2^n = g^{m_1+m_2} \cdot (r_1 r_2)^n \pmod{n^2}$$
$$c^a = (g^m r^n)^a = g^{ma}(r^a)^n \pmod{n^2}$$

그 후 Paillier의 방법을 타원곡선 상에 적용한 연구가 2002년 S. Galbraith에 의해 발표되었고, 2001년에는 Damgård와 Jurik이 Paillier cryptosystem을 일반화시키는 등 homomorphic encryption에 대한 연구가 활발하게 이루어졌다.

3. 2-DNV formulars[BGN05]

덧셈과 상수곱 연산을 보존하는 homomorphism에 이어 2005년 Boneh, Goh, Nissim이 덧셈과 한 번의 곱셈 연산이 가능한 homomorphic encryption을 발표하였다. 이들은 타원곡선상에서 정의된 겹선형 사상을 이용하여 덧셈과 한 번의 곱셈 연산이 가능하도록 하였다. 이들이 제안한 암호기법에서 암호화는 겹선형사상이 정의된 그룹에서 다음과 같이 정의된다.

- 암호화 : $c = g^m h^r \pmod{n}$

여기에서 두 암호문 c_1과 c_2에 대해 겹선형사상 연산을 시행하여 평문 m_1과 m_2에 대해 다음과 같은 연산을 수행하여 $m_1 \times m_2$의 암호문을 얻을 수 있다.

$$e(c_1, c_2) = e(g^{m_1} h^{r_1}, g^{m_2} h^{r_2}) = e(g^{m_1} g^{\alpha r_1}, g^{m_2} g^{\alpha r_2})$$
$$= e(g, g)^{(m_1+\alpha r_1)(m_2+\alpha r_2)} = g_1^{m_1 m_2} h_1^r$$

여기에서 e는 겹선형 사상이고 $g_1 = e(g, g)$, $h_1 = e(g, h)$이며, r은 계산을 통해 정해진 난수이다. 그러나 겹선형 사상을 이용하여 곱셈을 가능하게 한 암호기법은 겹선형사상의 특성으로 인해 단 한 번의 곱셈만이 가능하다는 제한점을 가지고 있다.

표 2. Homomorphic Encryption의 특징에 따른 구분

구 분	특 징
RAD78	- 곱셈 연산 보존 가능 - 안전성이 보장되지 않기 때문에 실제 사용은 불가능
Pai99	- 덧셈 연산과 상수곱 보존 가능
BGN05	- 겹선형사상을 이용하여 연산 보존 - 덧셈과 한 번의 곱셈연산 보존 가능

III. 최근 연구 동향

homomorphic encryption에 대한 연구는 과거의 제한된 연산을 할 수 있는 somewhat homomorphic encryption 연구에서 2000년대 후반부터 안전하면서도 기능적으로 충분히 만족할 만한, 즉 원하는 횟수 만큼의 연산을 보존 가능한 fully homomorphic encryption을 연구하는 방향으로 진행되고 있다. 2009년 Gentry는 최초로 안전성이 증명된 fully homomorphic encryption을 제안하였다.

1. The Gentry Encryption[Gen09]

Gentry는 lattice에서 간단한 연산을 사용하여 somewhat homomorphic encryption 시스템을 설계하였다. 이 시스템은 난수화 된 에러를 사용하며, 에러는 연산을 수행할 때마다 증폭되는 성질을 가지고 있다. 몇 번의 연산을 수행하여 증폭된 에러가 일정 수준을 넘어서게 되면 암호문을 복호화 할 수 없게 된다.

Gentry는 앞서 설명한 somewhat homomorphic encryption을 이용해서 원하는 횟수만큼 연산이 가능한 fully homomorphic encryption을 설계하였다. 이 과정을 부트스트래핑(bootstrapping)이라고 부른다. somewhat homomorphic encryption은 연산 횟수가 많아질수록 에러가 증가하기 때문에 제한된 횟수의 연산에 한해서 시행할 수 있다. 부트스트래핑 과정은 공개키 k_1을 이용하여 메시지m_1, m_2를 암호화 하여 암호문 c_1, c_2를 생성한다.

$$c_1 = E_{k_1}(m_1),\ c_2 = E_{k_1}(m_2)$$

이 경우 각각의 암호문에 에러 n_1과 n_2가 포함되게 된다. 그러고 난 후 두 암호문을 곱하면 somewhat homomorphic encryption의 성질에 의해

$$c_3 = E_{k_1}(m_1 \times m_2)$$

를 얻을 수 있으며, c_3에는 증폭된 에러 $n_1 \times n_2$가 포함되어 있다. 부트스트래핑은 다음과 같은 과정을 통해 여기에서 에러를 제거하는 과정이다.

먼저 공개키 k_1을 이용하여 암호문 c_3와 비밀키 k_2를 암호화 한다.

$$c_4 = E_{k_1}(c_3) = E_{k_1}(E_{k_1}(m_1 \times m_2))$$
$$c_5 = E_{k_1}(k_2)$$

c_4와 c_5가 주어지면 우리는 c_5를 이용하여 c_4를 복호화 할 수 있다. 다시 말하면, c_4에 포함되어있는 $c_3 = E_{k_1}(m_1 \times m_2)$를 비밀키 k_2를 이용하여 복호하 할 수 있다.

그러나 부트스트래핑이 가능한 somewhat homomorphic encryption을 설계하는 것은 매우 어려운 일이기 때문에 Gentry는 매우 복잡한 방법을 사용했다. lattice를 이용하는 것만으로는 부트스트래핑 가능한 암호를 설계하는 데 충분하지 않았기 때문에 이를 극복하기 위해 스쿼싱(squashing)이 제안되었다. 스쿼싱은 복호화에 필요한 일부과정을 사전 계산을 통해 미리 수행하는 것으로 복호화 과정의 복잡도를 크게 줄여주기 때문에 somewhat homomorphic encryption을 이용하여 fully homomorphic encryption을 구성할 수 있게 해 준다.

2009년 Gentry의 발표 이후 homomorphic encryption에 대한 연구 방향은 부트스트래핑과 스쿼싱을 이용하여 암호화, 복호화 계산을 단순화 하고 계산의 효율성을 개선하는 방향으로 진행되어 왔다. 그러나 최근에는 사전 계산이 필요한 스쿼싱의 비효율성 문제로 스쿼싱과 부트스트래핑 과정 없이 fully homomorphism encryption을 구성하는 방법이 주로 제안되고 있다.

2. Fully Homomorphic Encryption[DGHV10]

Dijk, Gentry, Halevi, Vaikuntanathan은 2009년에 처음 제안되었던 Gentry의 방법을 lattice가 아닌 정수 집합에 유사한 방법으로 적용한 새로운 somewhat homomorphic encryption을 발표하였다. 그들이 제안한 시스템은 다음과 같다.

- 키 설정 : 비밀키는 길이가 λ비트인 홀수 p이고, 공개키는 임의의 정수 q_i 및 에러값 r_i를 이용하여

$$x_i = q_i p + r_i \tag{1}$$

를 만족하는 x_i들을 결정하여 x_i들을 가장 큰 값으로부터 크기순으로 정렬하면 공개키 $(x_0, x_1, \cdots, x_n)$를 구할 수 있다.

- 암호화 : $m \in \{0,1\}$인 m을 임의의 난수 r과 공개키 $(x_0, x_1, \cdots, x_n)$ 중에 임의로 선택한 j개의 x_i들인 $(x_1', x_2', \cdots, x_j')$을 이용하여 다음과 같이 계산한다.

$$c \equiv m + 2r + 2\sum_{i=0}^{j} x_i' \pmod{x_0} \tag{2}$$

- 복호화 : 비밀키 p를 이용하여 다음과 같이 계산한다.
$m \equiv (c \bmod p) \pmod 2$

평문 m은 암호화 단계에서 난수 r과 공개키 $(x_1', x_2', \cdots, x_j')$을 이용하여 식 (2)에 의해 암호화 된다. x_i들은 식 (1)을 이용하여 암호화되었기 때문에 복호화 단계의 $\bmod p$에 의해 $q_i p$부분이 사라지고 에러 r_i만 남게 되며, 암호화 단계의 에러부분에는 모두 상수 2가 곱해져 있기 때문에 복호화 단계의 $\bmod 2$에 의해서 소거된다.

새로운 암호체계는 다음과 같은 과정을 통해 덧셈 및 곱셈 연산이 가능하다.

$$c_1+c_2=m_1+2r_1+2\sum x_i(\mathrm{mod}\ x_0)+m_2+2r_2+2\sum x_i(\mathrm{mod}\ x_0)$$
$$=(m_1+m_2)+2(r_1+r_2)+4\sum x_i(\mathrm{mod}\ x_0)$$
$$c_1\times c_2=(m_1+2r_1+2\sum x_i\ (\mathrm{mod}\ x_0))\times(m_2+2r_2+2\sum x_i\ (\mathrm{mod}\ x_0))$$
$$=(m_1\times m_2)+2(m_1r_2+m_2r_1+2r_1r_2+a_i\sum x_i)\ (\mathrm{mod}\ x_0)$$

이들이 제시한 방법에서도 09년 Gentry에 의해 제안된 방법과 같이 위와 같이 연산을 거듭할수록 에러가 증폭된다. 이러한 에러가 일정수준을 벗어나게 되면 복호화를 할 수 없게 되어 만족할 만큼의 연산을 수행하기 어렵다. 이 방식은 Gentry가 제안한 부트스트래핑이 가능하지 않기 때문에 스퀴싱 방식을 적용하여 복호화 함수를 단순하게 하는 방법을 통하여 fully homomorphic encryption을 구성할 수 있다. 또한 스퀴싱 방식을 사용할 경우 사전 계산이 필요하기 때문에 계산에 소요되는 시간이 추가되어 한 비트를 처리하는데 오랜 시간이 걸리게 된다는 단점이 있다.

3. CRT-Based Fully Homomorphic Encryption

1978년 Rivest, Adleman, Dertouzos에 의해 처음 homomorphic encryption이 제안되었을 때 중국인의 나머지 정리(Chinese Remainder Theorem)에 기반을 둔 방법도 함께 제안되었다. 그러나 그 당시의 방법은 알려진 평문 공격에 취약하여 실생활에서 활용할 수 없었다. 그러나 2013년 국내 연구진에 의해서 CRT에 기반해 암호문에 대응하는 유일한 원문을 복구해내는 방식의 안전한 fully homomorphic encryption이 개발되었다. 또한 지금까지의 완전동형암호는 1비트의 정보량을 암호화하기 위해 매우 긴 암호문을 생성하여 모든 연산을 비트 단위로 수행해야 했기 때문에 효율성 면에서 실제 구현에 한계가 있었다. 그러나 새로운 방법은 비트 단위가 아닌 큰 숫자 단위로 연산을 실시하여 평문 공간을 확장할 수 있는 장점을 가지고 있다.

중국인의 나머지 정리란 1세기경 중국의 수학자가 증명한 것으로, 서로소인 두 양의 정수 p와 q가 있고 어떤 수 x를 각각의 정수로 나눴을 때의 나머지를 알 경우, 이 수를 $n=pq$로 나눴을 때의 나머지는 유일하며 p와 q를 알면 쉽게 구할 수 있다는 정리이다. 새로 개발된 방식은 정보를 비트 단위로 암호화 하는 것이 아니라 큰 숫자 단위로 암호화 하기 때문에 기존의 방식보다 더 많은 정보를 저장할 수 있을 뿐만 아니라, 정보를 비트 단위로 변환하여 연산을 수행하지 않고 직접 연산을 수행하여 훨씬 효율성을 높일 수 있다는 장점이 있다.

CRT-based fully homomorphic encryption을 간단히 소개하면, 이것은 메시지를 특정한 수로 나눈 나머지를 상대방에게 보내는 방식이다.

* 암호화
- 쌍마다 서로소인 큰 정수 $p_i(i=1,\cdots,k)$, 비교적 작은 쌍마다 서로소인 정수 $q_i(i=1,\cdots,k)$, 임의의 정수 e
- 비밀키 : $(p_0,\cdots,p_k)$
- $n=\prod_{i=0}^{k}p_i,\ q=\prod_{i=1}^{k}q_i$
- 공개 parameter : $n,q_1,q_2,\cdots,q_k$
- 메시지 m을 q_i로 나눈 나머지 m_i 생성
- 임의의 정수 e를 이용하여 암호문 c 생성
  $c=CRT_{(p_0,\cdots,p_k)}(e,m_1,m_2,\cdots,m_k)$

* 복호화
- $d_i=(c \bmod p_i) \bmod q_i$ 계산
- $m=CRT_{(q_1,\cdots,q_k)}(d_1,\cdots,d_k)$

위의 방식을 적용하면 CRT를 이용하여 유일한 평문을 구할 수 있다. 이 방법은 기존의 [DGHV10]에서 정의한 Approximate GCD가 아닌 새로운 Decisional Approximate GCD를 정의하고, 이 문제가 풀리기 어렵다는 사실에 기반하여 암호 알고리즘을 만들어 낸 것이다.

현재까지 fully homomorphic encryption의 연구는 somewhat homomorphic encryption을 부트스트래핑과 스퀴싱을 통해 fully homomorphic encryption으로 구성하는 양식을 따르는 것이 일반적이었다. 이 틀 안에서 효율성을 개선하거나 단계를 단순화 시키는 방향으로 연구를 진행하거나 기존에 제안된 방식의 안전성을 분석하는 연구가 대부분이었다. 여기에 효율성을 더욱 개

선하기 위해 부트스트래핑과 스퀴싱을 제외하고 fully homomorphic encryption을 구성하는 방법이 제안되었으며, 최근에는 아예 새로운 수학적 난제를 제시하여 이를 기반으로 중국인의 나머지 정리를 통해 암호화/복호화를 하는 알고리즘을 개발하는 방향으로 연구가 진행되고 있다.

표 3. Fully Homomorphic Encryption의 특징에 따른 구분

구 분	특 징
Gen09	- ideal lattice 상의 난제에 기반하여 덧셈과 일정 횟수의 곱셈 연산 보존 - 난수화 된 에러를 사용하여 일정횟수의 연산 시행시 에러가 증폭되어 복호화 불가능 - 부트스트래핑과 스퀴싱을 이용하여 완전동형암호 구현
DGHV10	- 정수 집합에 Gen09의 방법 사용하여 덧셈과 일정 횟수의 곱셈 연산 보존 - 일정횟수의 연산 후 에러가 증폭되어 복호화 불가능
CRT-Based	- 중국인의 나머지정리를 이용 - 비트 단위가 아닌 큰 숫자단위로 연산하여 많은 정보 저장 가능 및 효율성 증대

IV. 발전 방향

최근 제안된 완전동형암호는 기존의 효율성이 떨어지는 문제를 개선한 방법으로 기존에 비해 저장할 수 있는 정보의 양이 늘고 효율성이 개선되어 실생활에서 실용화 가능할 것으로 기대된다. 완전동형암호가 실용화 될 경우 개인의 병원 진료기록이나 의료보험공단, 은행데이터 등 보호가 필요한 대량의 데이터를 복호화하지 않고 암호화된 상태에서 검색 및 처리할 수 있기 때문에 현재와 같이 빈번하게 일어나고 있는 개인정보 유출 사고를 확실하게 막을 수 있는 방법이 될 것이다. 또한 자료 처리의 효율성을 더욱 향상시킨다면 핸드폰이나 개인 컴퓨터를 이용해서도 정보 처리가 충분히 가능하게 되어 편리하게 자료의 유출에 대한 걱정 없이 정보를 이용 가능하게 될 것이다. 이렇게 된다면 인터넷이 발달하면서 빠르게 늘고 있는 핸드폰, 태블릿 PC 등을 이용한 클라우드 컴퓨팅 환경에서도 콘텐츠의 유출에 대한 걱정 없이 클라우드를 활용하여 언제 어디서 어떤 기기로도 콘텐츠에 접근하고 저장하고 가공할 수 있어 더욱 편리한 생활이 가능할 것으로 기대된다.

V. 결론

fully homomorphic encryption은 사용자가 암호화된 데이터를 복호화하는 단계를 거치지 않고 검색, 통계처리 및 연산을 수행할 수 있게 하는 암호기법이다. 이것은 클라우드 컴퓨팅 환경을 비롯한 여러 가지 개인 정보 보호가 요구되는 분야에 다양하게 응용될 수 있는 까닭에 큰 주목을 받고 있다. 이것은 공개키 암호와 함께 1970년대 후반에 제안되어 XOR, 덧셈, AND, 한 번의 곱셈을 제공하는 알고리즘으로 점차 발전하였다. 그러던 중 09년에 Gentry에 의해 fully homomorphism encryption이 처음으로 제안되었다. 그러나 이것은 효율성 측면에서 약점이 있었기 때문에 부트스트래핑과 스퀴싱을 점차 배제하고 시스템을 설계하는 방향으로 연구되었다. 그러던 중 최근 Gentry가 정의한 문제가 아닌 완전히 새로운 문제를 기반으로 CRT를 이용하는 fully homomorphism encryption이 제안되었다.

이전까지 제안된 방식들은 실제적으로 활용함에 있어서 그 효율성이 매우 떨어진다는 문제가 있었지만 최근 연구로 효율성이 대폭 개선되어 앞으로 완전동형 암호의 실용화를 이루기 위한 여건이 마련되었다. 향후 연구 과제로 연산과정을 더욱 효율적으로 수행할 수 있는 방법에 대한 연구가 필요하고, 구현을 통해 실생활에서 적용시 안전성 측면에서의 검증이 필요하다.

참 고 문 헌

[1] 송유진, 박광용, “데이터베이스 아웃소싱을 위한 준동형성 암호기술”, 정보보호학회지, 제19권, 제3호, pp.80-89, 2009.

[2] 김선주, 김진묵, 조인준, “모바일 클라우드 서비스 상에서 준동형 암호 기반의 형상 관리 방안”, 한국정보통신학회논문지, 제16권, 제10호,

pp.2217-2223, 2012.

[3] R. Rivest, L. Addleman, and M. Dertouzos, "On data banks and privacy homomorphism," In Foundations of Secure Computation, pp.169-177, 1978.

[4] I. Damgård and M. Jurik, "A Generalization, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System," Public Key Cryptography - PKC, pp.119-136, 2001.

[5] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," EUROCRYPT, pp.223-238, 1999.

[6] S. Galbraith, "Elliptic curve Paillier schemes," Journal of Cryptology - JOC, Vol.15, No.2, pp.129-138, 2002.

[7] C. Gentry, "Fully homomorphic encryption using ideal lattices," ACM Symposium on Theory of Computing - STOC, pp.169-178, 2009.

[8] D. Boneh, E. Goh, and K. Nissm, "Evaluating 2-DNF formulars on ciphertexts," Theory of Cryptography, pp.325-341, 2005.

[9] J. H. Cheon, J. S. Coron, J. Kim, M. S. Lee, T. Lepoint, M. Tibouchi, and A. Yun, "Batch fully hoomorphic encryption over the integers," Advances in Cryptology - EUROCRYPT, Lecture Notes in Computer Science, Vol.7881, pp.315-335, 2013.

[10] M. V. Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "fully homomorphic encryption over the integers," Advances in Cryptology - EUROCRYPT, Lecture Notes in Computer Science, Vol.6110, pp.24-43, 2010.

[11] C. Gentry and S. Halevi, "Implementing gentry's fully homomorphic encryption scheme," Advances in Cryptology - EUROCRYPT, Lecture Notes in Computer Science, Vol.6632, pp.129-148, 2011.

[12] http://terms.naver.com/entry.nhn?cid=200000000&docId=1350825&mobile&categoryId=200000756

[13] http://outsourcedbits.org/2012/06/26/applying-fully-homomorphic-encryption-part-1/

저 자 소 개

정 명 인(Myoung In Jeong) 종신회원

- 2004년 3월 : 육군사관학교 운영분석(이학사)
- 2008년 2월 : 서울대학교 수학과(이학석사)
- 2008년 3월 ~ 현재 : 육군사관학교 수학과 교수

<관심분야> : Cryptography, Information Security