Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.27, NO.3, Jun. 2017 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2017.27.3.477

부분동형암호와 외부서버를 이용한 효율적인 다자간 연산 기법*

은 하 수,† 우바이둘라, 오 희 국‡
한양대학교

Efficient Outsourced Multiparty Computations Based on Partially Homomorphic Encryption*

Hasoo Eun,† Ubaidullah, Heekuck Oh‡
Hanyang University

요 약

MPC(multiparty computation) 프로토콜이란 다수의 사용자가 각각 데이터를 제공하고, 이를 이용하여 협력적으로 연산을 수행하는 기법이다. 기존의 MPC 프로토콜은 사용자 사이의 상호작용에 의존했기 때문에, 연산이 끝날 때까지 모든 사용자가 온라인 상태를 유지해야 했다. 이를 개선하기 위한 연구 중 하나로서, 공모하지 않은 두 서버에 연산을 위임하는 기법이 연구되고 있다. 사용자의 참여를 완전히 배제한 최초의 기법이 Peter 등에 의해 제안되었으나, 서버 간 통신량이 매우 높다는 단점이 있다. 본 논문에서는 Peter 등의 기법에서 문제가 되었던 서버 간 통신량을 PRE(proxy re-encryption)를 이용하여 개선하였다. 제안하는 기법과 유사한 기법이 두 차례 제안되었으나, 복호화 과정에서 이산대수 문제를 해결해야 하거나, 서버와 사용자 사이의 공모공격에 취약한 등 다양한 문제점이 존재한다. 본 논문에서는 기존 기법의 문제점을 분석하고 이를 바탕으로 안전하고 효율적인 MPC 프로토콜을 제안한다. 제안하는 기법은 PRE를 이용하여 서버 간 통신량을 낮추었으며, 연산과정에서 사용자의 참여를 완전히 배제하였고, 복호화 과정에서 이산대수문제를 풀지 않고도 연산결과를 얻을 수 있다.

ABSTRACT

Multiparty computation (MPC) is a computation technique where many participants provide their data and jointly compute operations to get a computation result. Earlier MPC protocols were mostly depended on communication between the users. Several schemes have been presented that mainly work by delegating operations to two non-colluding servers. Peter et al. propose a protocol that perfectly eliminates the need of users' participation during the whole computation process. However, the drawback of their scheme is the excessive dependence on the server communication. To cater this issue, we propose a protocol that reduce server communication overhead using the proxy re-encryption (PRE). Recently, some authors have put forward their efforts based on the PRE. However, these schemes do not achieve the desired goals and suffer from attacks that are based on the collusion between users and server. This paper, first presents a comprehensive analysis of the existing schemes and then proposes a secure and efficient MPC protocol. The proposed protocol completely eliminates the need of users' participation, incurs less communication overhead and does not need to solve the discrete logarithm problem (DLP) in order to get the computation results.

Keywords: Multiparty Computations, Outsourced MPCs, Proxy Re-Encryptions, Delegating Computations

Received(05. 15. 2017), Modified(05. 30. 2017), Accepted(06. 01. 2017)

* 이 논문은 2017년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(NRF-2015R1D1A1A09058200)

* 본 연구는 미래창조과학부 및 정보통신기술진흥센터의 대학ICT연구센터육성 지원사업의 연구결과로 수행되었음 (IITP-2017-2014-0-00636)

† 주저자, hseun@hanyang.ac.kr

‡ 교신저자, hkoh@hanyang.ac.kr(Corresponding author)

I. 서 론

MPC(Multiparty Computation) 프로토콜이란 다수의 사용자가 각각 데이터를 제공하고, 이를 이용하여 협력적으로 연산을 수행하되 각자의 데이터는 노출하지 않는 암호학적 기초기술이다. Yao 등에 의해 처음 제안한 MPC는 두 사용자가 자신의 재산을 공개하지 않은 채로 누가 부자인지 계산하는 것으로서 백만장자 문제라 불린다[1]. 과거의 MPC 프로토콜들은 데이터를 제공한 사용자가 직접 연산을 수행했기 때문에 한 명이라도 오프라인인 경우 연산이 불가능했다. 즉, 연산이 수행되는 동안 모든 사용자가 온라인 상태를 유지해야 했다.

Lòpez-Alt 등은 이와 같은 문제점을 지적하며 OMPC(On-the-Fly MPC) 프로토콜을 제안했다[2]. OMPC 프로토콜을 이용하면 기존 MPC 프로토콜의 연산을 외부의 서버에 위임할 수 있다. 사용자의 단말을 대신하여 외부의 서버가 연산을 수행하기 때문에, 연산이 진행되는 동안 사용자의 단말이 온라인 상태를 유지할 필요가 없으며, IoT 기기와 같은 저전력 단말에도 쉽게 적용할 수 있다. 하지만 Lòpez-Alt 등의 기법도 복호화 과정에서는 모든 사용자가 연산에 참여해야 하는 한계가 있다.

이후, Peter 등은 BCP(Bresson, Catalano, and Pointcheval) 암호시스템과 데이터 블라인딩(data blinding) 기법을 이용하여 사용자의 참여를 완전히 배제한 최초의 OMPC 프로토콜을 제안했다[3, 4]. Peter 등의 기법은 ① 사용자의 데이터를 관리하는 서버가 암호화된 데이터를 블라인딩 한 후 ② 트랩도어를 관리하는 서버에게 전달, 해당 개체가 이를 ③ 복호화, ④ 연산, ⑤ 다시 암호화하여 ⑥ 반환하는 형태로 진행된다. 즉, 데이터를 다루는 모든 과정을 서버 간 통신에 의존하고 있다.

Wang 등은 Peter 등의 기법이 서버 간 통신에 과도하게 의존한 결과 효율성이 떨어지는 문제점을 지적하였으며, PRE(proxy re-encryption)를 이용하여 해당 문제를 해결한 프로토콜을 제안했다[5]. PRE를 이용하면 사용자 i의 공개키로 암호화된 암호문을 사용자 j가 자신의 개인키로 복호화 할 수 있다. Wang 등은 BBS PRE(Blaze, Bleumer, and Strauss' proxy re-encryption)의 양방향 특성을 이용하여 곱셈연산을 제외한 모든 과정에서 서버 간 통신이 필요 없는 OMPC 기법을 제안하였다[6]. 하지만, BBS PRE는 양방향 특성으로 인해 프록시(=재암호화를 수행하는 개체)와 사용자 사이의 공모공격에 취약하다. Wang 등의 기법은 BBS PRE를 그대로 적용하였으므로 해당 취약점 또한 그대로 계승하고 있다.

이후, 단방향 특성을 이용하여 프록시와 사용자 사이의 공모공격 취약점을 해결한 기법이 제안되었다[7, 8]. 특히, [7]에서는 AFGH PRE(Ateniese, Fu, Green, and Hohenberger's proxy re-encryption)를 덧셈동형암호 형태로 변형하여 적용한 Banana+와 AFGH PRE를 그대로 적용한 Banana*가 제안되었다. 하지만, Banana+는 복호화 과정에서 이산대수문제를 해결해야하는 한계가 있다. Banana*는 곱셈동형특성을 이용하여 데이터를 블라인딩하는데, 차후 이를 효율적으로 제거하기 위해 연산에 함께 사용되는 암호문에 동일한 블라인딩 요소(blinding factor)를 적용해야한다. 동작은 효율적으로 이루어지지만, 데이터를 제공한 사용자 중 한 명과 연산을 담당하는 서버가 공모하게 된다면 나머지 모든 사용자의 데이터가 노출된다.

본 논문에서는 앞서 언급한 OMPC 프로토콜의 문제점을 상세히 분석함으로써 보안요구사항을 도출하고, 이를 기반으로 안전하고 효율적인 OMPC 프로토콜을 제안한다. 제안하는 기법은 BCP 암호시스템을 일부 수정하여 단방향 PRE와 결합하는 것이다. BCP 암호시스템의 덧셈동형특성을 이용하면 각 암호문에 독립적인 블라인딩 요소를 사용할 수 있다. 더불어 단방향 PRE를 적용함으로써 공모공격을 방지하고 암호문에 사용된 키를 단일화하는 과정에 필요한 서버 간 통신량을 제거할 수 있다.

본 논문의 구성은 다음과 같다. 2장은 사전지식으로서, OMPC 프로토콜을 이해하는데 필요한 암호기법들에 대해 다룬다. 3장은 관련연구로서 앞서 언급한 기법들에 대해 상세히 소개하고, 이들의 문제점을 분석한다. 4장에서는 앞서 분석한 내용을 기반으로 안전하고 효율적인 OMPC 프로토콜을 제안하고, 5장에서 제안하는 기법의 안전성 및 효율성을 분석한다. 마지막으로 6장에서 결론을 맺는다.

II. 사전지식

본 장에서는 OMPC 프로토콜을 이해하는데 필요한 암호기법들에 대해 다룬다. 본 논문에서 다루는 다양한 프로토콜을 일관성 있게 설명하기 위해, 공통적으로 사용되는 기호에 대해서는 [표 1]에 제시된

Table 1. Notations

Notation	Description
(pk_x, sk_x)	Public key pair for entity x.
msk	Master secret key.
$rk_{i\to j}$	Re-encryption key from entity i to entity j.
m_i	Plaintext for entity i.
c_i	Ciphertext for entity i.
n	Number of users.
G	Group.
$\odot$	Component-wise multiplication.
$\hat{e}(.)$	Bilinear map.
$Enc(.)$	Encryption function.
$Dec(.)$	Decryption function.
$ReEnc(.)$	Re-encryption function.
$mDec(.)$	Master decryption function.
$Add(.)$	Evaluation function for addition.
$Mult(.)$	Evaluation function for multiplication

표기법을 따른다.

2.1 겹선형 쌍곡선(bilinear pairing)

겹선형 쌍곡선은 타원곡선상의 이산대수 문제를 유한체 상의 이산대수 문제로 축소함으로써, 타원곡선 암호 시스템을 효과적으로 공격하기 위해 개발된 도구이다. 2001년 Boneh 등이 겹선형 쌍곡선에 기반을 둔 신원기반 암호시스템을 발표하였고, 이후 다양한 암호들이 연구되고 있다[9]. 겹선형 쌍곡선은 동일한 위수(order)를 갖는 곱셈군 G_1, G_2와 곱셈군 G_T 사이의 사상함수 $\hat{e}: G_1 \times G_2 \to G_T$로 정의된다. 세 곱셈군의 위수를 N라 하고, G_1과 G_2의 생성자(generator)를 각각 $P \in G_1$, $Q \in G_2$라 했을 때, 겹선형 쌍곡선은 다음과 같은 성질을 갖는다.

1. **Bilinearity**: 임의의 $a,b \in Z_N^*$에 대하여, $\hat{e}(P^a, Q^b) = \hat{e}(P,Q)^{ab}$를 만족한다.
2. **Non-degeneracy**: $\hat{e}$은 G_T의 항등원으로 사상되지 않는다. 즉, $\hat{e}(P,Q) \neq 1$을 만족한다.
3. **Computability**: $\hat{e}$을 효율적으로 계산할 수 있는 알고리즘이 존재한다.

2.2 프록시 재암호화(proxy re-encryption, PRE)

일반적인 공개키 암호시스템에서 사용자는 자신의 공개키로 암호화된 암호문만 복호화 할 수 있다. 하지만 PRE를 이용하면 사용자 i의 공개키로 암호화된 데이터 m_i를 사용자 j의 개인키로 복호화 할 수 있다. 초기 암호문의 복호화 권한을 가진 사용자를 i, 그 권한을 위임받아 복호화를 수행하려는 사용자를 j라 할 때, PRE는 다음과 같이 정의할 수 있다.

$$Dec(sk_j, ReEnc(rk_{i\to j}, Enc(pk_i, m_i))) = m_i.$$

2.3 동형암호(homomorphic encryption)

사용자는 기밀성을 유지하기 위해 데이터를 암호화하여 서버에 저장할 수 있다. 하지만 이를 연산에 활용하기 위해 서버에서 복호화 한다면, 그 과정에서 기밀성을 해치게 된다. 동형암호는 이와 같은 문제의 해결책으로서, 암호문 입력을 받아 암호문 형태로 결과 값을 반환하는 기법이다. 평문 도메인 연산 f에 대응하는 동형 연산 $eval_f$는 다음과 같이 정의된다.

$$eval_f(c_i, c_j) = Enc(pk_S, f(m_i, m_j)).$$

임의의 함수를 계산할 수 있는 기법을 완전동형암호(Fully Homomorphic Encryption, FHE)라 한다. 안전성이 증명된 최초의 FHE는 2009년 Gentry에 의해 제안되었다[10]. FHE를 이용하면 사용자 데이터의 노출 없이 외부에 저장된 데이터를 연산할 수 있지만, 아직까지 연산속도 측면에서 실용적이지 못하다.

FHE와 달리 몇몇 연산만 가능한 기법을 부분동형암호(Partially Homomorphic Encryption, PHE)라 한다. 예를 들어, RSA는 곱셈동형성질을 갖으며, BCP 암호기법은 덧셈동형성질을 갖는다.

2.4 데이터 블라인딩 기법

데이터 블라인딩 기법이란 암호화된 데이터에 임의의 값을 더하거나 곱함으로써 복호화하더라도 원래의 데이터가 무엇인지 알 수 없도록 만드는 기법이다. 예를 들어, 다음과 같은 암호문을 복호화하는 상황을 고려해보자.

$$c_i' = Add(c_i, Enc(pk_S, r_i)) = Enc(pk_S, m_i + r_i).$$

c_i'을 복호화하면 $m_i + r_i$를 얻게 되므로, r_i를 알지 못하면 m_i를 얻을 수 없다. 이때 r_i를 블라인딩 요소라 한다. 데이터 블라인딩 기법은 PHE 기반의 OMPC 프로토콜에서 다룰 수 없는 연산을 처리할 때 주로 사용된다. 하지만 서버 간 통신량이 증가하는 단점이 있다.

III. 관련 연구

본 장에서는 최근 제안된 OMPC 기법들을 소개한다. OMPC 개념이 처음 제안된 것은 Lòpez-alt 등의 기법이지만, 복호화 과정에서 모든 사용자의 참여가 필요하므로 완전한 OMPC 프로토콜이라 보기 어렵다. 본 논문에서는 사용자의 참여를 완전히 배제한 최초의 연구인 Peter 등의 기법부터 소개한다.

3.1 OMPC 프로토콜의 구성 및 동작

OMPC 프로토콜에는 사용자, 데이터를 관리하는 서버 C, 연산을 담당하는 S 등 총 3가지 개체가 등장한다. 사용자는 서버 C에게 데이터를 업로드하고, 서버 C와 서버 S는 협력적으로 연산을 수행한다.

OMPC 프로토콜들의 동작은 ① 초기화, ② 키통합, ③ 덧셈연산, ④ 곱셈연산, ⑤ 검색 등 5가지 단계로 구성된다. 초기화 단계에서는 프로토콜을 수행하는데 필요한 전역 파라미터와 공개키를 생성하고, 이를 이용하여 암호화한 데이터를 서버 C에게 업로드 한다. 키통합 단계에서는 암호문에 사용된 여러 사용자의 키를 하나의 키로 통합하는 작업을 수행한다. 덧셈연산과 곱셈연산 단계에서는 키통합을 거친 암호문을 이용하여 서버 C와 서버 S가 협력적으로 연산을 수행한다. 마지막으로 검색 단계에서는 통합키로 암호화되어 있는 연산결과를 사용자가 복호화할 수 있도록 재변환하고, 이를 사용자에게 반환한다.

3.2 Peter 등의 기법

Peter 등은 BCP 암호시스템의 트랩도어(=마스터키 msk)와 덧셈동형특성을 이용한 프로토콜을 제안하였다[3, 4]. 모든 사용자의 암호문은 서버 C에 저장되어 있으며, 모든 암호문을 복호화 할 수 있는 msk는 서버 S에 저장되어 있다. 만일 이들이 공모하게 된다면, 결과적으로 아무 제약 없이 모든 사용자의 데이터를 열람할 수 있게 된다. 따라서 두 서버가 공모하지 않는다는 가정이 필요하다. 이 기법에서 서버 S는 사용자와의 직접적인 접점이 없으며, 오직 서버 C만이 사용자에게 파라미터를 전달하고, 데이터를 주고받는 역할을 수행한다. 이는 각 서버의 역할을 더욱 공고히 하려는 것으로 해석될 수 있다. 하지만 서버 C가 별도로 전역 파라미터 PP'을 생성하여 사용자에게 배포한다면, 서버 C는 이 과정에서 새로운 마스터키를 얻을 수 있으며, 이를 이용하여 MITM (Man-in-the-Middle) attack을 할 수 있게 된다. 따라서 전역 파라미터의 경우 서버 S가 사용자에게 직접 전달할 필요가 있다.

Table 2. Communication cost for Peter et al.'s scheme

	C to S	S to C
Unification	nk	nk
Addition	0	0
Multiplication	nk	$nk/2$
Retrieval	$nk/2$	nk
Total	$5nk/2$	$5nk/2$

Peter 등의 기법에서 사용자의 암호문을 다루기 위해 ① 서버 C가 블라인딩 된 데이터를 서버 S에게 전달, ② 서버 S가 이를 msk로 복호화하여 가공한 후 서버 C에게 전달, ③ 서버 C가 블라인딩 요소를 제거하는 동작을 반복하게 된다. 이는 초기화와 덧셈을 제외한 모든 과정에서 사용된다. 예를 들어 n명의 사용자가 두 명씩 짝지어서 k-bit 데이터를 곱한다고 가정하면, 이때 필요한 서버 간 통신량은 [표 2]와 같다. 즉, 한 번의 곱셈을 위해서 사용자와 데이터 크기의 곱에 약 2.5배 만큼의 서버 간 통신량이 요구된다.

3.3 Vitamin

Wang 등은 Peter 등이 제안한 OMPC 프로토콜의 서버 간 통신량을 낮추기 위한 방법으로써 BBS PRE에 기반을 둔 Vitamin+와 Vitamin*를 제안하였다[5, 6]. BBS PRE는 양방향 PRE로서, 이를 이용하면 키 통합과 검색 단계에서 서버 간 통신이 필요 없어지므로, 단방향 통신비용을 nk로

낮출 수 있다. 하지만 Vitamin은 BBS PRE의 단점 또한 그대로 계승하고 있다. 즉, 서버 C가 사용자 중 하나와 공모하는 경우 모든 사용자의 데이터를 열람할 수 있게 된다. 예를 들어, 서버 C가 사용자 i와 공모하여, 사용자 j의 암호문을 열람하고 싶다고 가정해보자. 이때 서버 C는 두 사용자의 재암호화키 $rk_{i\to S}=sk_S/sk_i$, $rk_{j\to S}=sk_S/sk_j$를 모두 가지고 있다. 서버 C는 두 재암호화키를 이용하여 $rk_{j\to i}$를 얻을 수 있으며, 이를 이용하면 사용자 i를 통해 사용자 j의 모든 암호문을 복호화 할 수 있다.

이와 같은 취약점은 BBS PRE의 재암호화키에 존재하는 이행성(transitivity) 때문에 발생한다. 이후 PRE의 연구는 서버가 임의로 재암호화키를 확장할 수 없도록 이행성을 제거하거나, 조건에 맞는 암호문만 재암호화 할 수 있게 하는 등 프록시의 역할을 제한하는 방향으로 진행되고 있다.

3.4 Banana

기존 Vitamin에 존재하는 서버 C와 사용자간 공모공격 취약점을 해결하기 위해 제안된 것으로서, AFGH PRE를 적용한 기법이다[7, 8]. AFGH PRE의 재암호화키는 g^{sk_i/sk_j}의 형태를 띠며, 단방향성질을 갖기 때문에 유효한 해결책이 될 수 있다. 하지만 검색 단계에서 재암호화를 할 수 없게 되므로 그만큼 서버 간 통신비용이 증가한다. 해당 논문에는 AFGH PRE의 곱셈동형특성을 그대로 적용한 Banana*와 AFGH PRE를 덧셈동형특성으로 변환하여 적용한 Banana+가 함께 제안되어 있다.

Banana*은 곱셈동형특성을 가지므로, 곱셈을 이용하여 데이터 블라인딩을 수행한다. 하지만 이 경우, 함께 연산되는 암호문에 동일한 블라인딩 요소를 사용해야하는 제약이 발생한다. 두 암호문에 서로 다른 블라인딩 요소를 적용한다면, 평문 형태는 각각 r_im_i, r_jm_j가 되며, 두 평문을 더하면 $r_im_i+r_jm_j$가 된다. 서버 C는 곱셈동형특성을 이용하여 연산결과에 포함된 블라인딩 요소를 제거해야 하는데, 곱셈만으로 r_i, r_j를 동시에 효율적으로 제거할 수 있는 방법은 알려져 있지 않다. 반면, 두 암호문에 동일한 블라인딩 요소 r_k를 적용하면 r_k^{-1}을 곱하는 것만으로 쉽게 제거할 수 있다. 하지만 여기에는 사용자와 서버 S가 공모하는 경우 연산에 사용된 모든 데이터가 열람되는 문제점이 존재한다. 만일 사용자 i가 m_i를 서버 S에게 제공한다면, 서버 S는 이를 이용하여 r_k를 얻을 수 있다. 모든 데이터는 r_k로 블라인딩 되어 있으므로, 전달받은 암호문을 복호화 후 r_k로 약분하는 것만으로 데이터를 열람할 수 있다. 이와 같은 취약점은 Vitamin*에서도 나타난다.

Banana+는 덧셈동형특성을 가지므로 각 암호문에 서로 다른 블라인딩 요소를 적용할 수 있긴 하지만, 암호문이 $c_i=(\rho^{m_i}\cdot\rho^{r_i}, g^{x_ir_i})$의 형태를 띠게 되므로 연산결과를 얻기 위해 이산대수 문제를 해결해야 한다. 이산대수문제를 효율적으로 해결할 수 있다면, 디피헬만문제 또한 효율적으로 해결할 수 있게 된다. 하지만 AFGH PRE는 디피헬만문제에 안전성을 의존하고 있다. 즉, 암호문을 효율적으로 복호화 할 수 있다면 기법 자체의 안전성이 무너진다.

IV. 안전하고 효율적인 OMPC 프로토콜

본 논문에서 제안하는 기법에는 n명의 사용자가 존재하며 이들을 $U_1,\cdots,U_n$으로 표기한다. 또한, 데이터를 관리하는 서버 C와 연산을 담당하는 서버 S가 존재하며 이들 간에는 공모하지 않지만, 임의의 사용자와는 공모할 수 있다고 가정한다. 이와 같은 가정은 앞서 제안된 기법들과 동일하다. 본 장에서는 이와 같은 가정을 바탕으로 안전하고 효율적인 OMPC 프로토콜을 제안한다. 프로토콜을 제안하기에 앞서, 안전하고 효율적인 OMPC 프로토콜이 갖추어야 할 보안요구사항에 대해 서술한다.

4.1 보안요구사항

기존 연구 결과를 분석하였을 때, 안전하고 효율적인 OMPC 프로토콜을 구성하기 위해서는 다음과 같은 보안요구사항이 필요하다.

SR 1. 연산에 참여하는 사용자는 자신의 데이터와 연산결과 이외에 다른 사용자의 평문 데이터에는 접근할 수 없어야 한다.

이는 MPC 프로토콜의 궁극적인 목표로서, 가장 우선시 되어야 할 사항이며, OMPC 프로토콜에도 동일하게 적용된다. 기존의 MPC 프로토콜에서는 사용자가 직접 연산에 참여하기 때문에 비밀분산기법(secret sharing) 등을 통해 이를 보호하였다. 반

면, OMPC 프로토콜에서는 사용자가 직접 연산에 참여하지 않기 때문에 다른 사용자의 데이터에 대한 접근 가능성은 희박하다. 다만, 서버와 사용자 사이의 공모공격을 통해 다른 사용자의 평문 데이터가 노출 될 수 있는 문제에 대해서는 주의를 기울일 필요가 있다.

SR 2. 연산에 참여하는 두 서버가 임의로 사용자의 평문 데이터에 접근할 수 없어야 한다.

OMPC 프로토콜에는 두 서버가 존재한다. 부분동형암호에 기반을 두고 있기 때문에, 동형특성으로 해결할 수 없는 연산의 경우 복호화하여 직접 계산 후, 다시 암호화하는 방법을 취하고 있다. 복호화가 가능하다는 것은 평문에 대한 접근 가능성이 높아짐을 의미하므로, 이 부분에 대해 보호기술이 적용되어야 한다. 기존 연구에서는 이를 방지하기 위해 두 서버가 공모하지 않는다고 가정하고 있으며, 데이터 블라인딩을 사용하는 등 대비책을 두고 있다. [3]에서 공개 파라미터 PP의 생성을 서버 S가 담당하는 이유도 동일한 맥락으로 볼 수 있다.

SR 3. 재암호화를 사용하는 경우, 프록시 역할을 담당하는 개체가 임의로 재암호화의 방향을 변경할 수 없어야 한다.

BBS 암호기법의 경우 프록시가 이행성과 재암호화키의 역수 등을 이용하여 재암호화의 방향을 변경할 수 있다. 이것이 가능하다면, 프록시가 임의의 사용자와 공모하여 다른 사용자의 데이터에 접근할 수 있게 된다. 이를 방지하기 위해서는 AFGH와 같은 단방향 PRE 기법을 사용하거나 조건에 맞는 암호문만 재암호화 할 수 있도록 제한하는 등의 조치가 필요하다.

SR 4. 데이터 블라인딩을 사용하는 경우, 각 암호문에 별도의 블라인딩 요소를 적용해야 한다.

동일한 블라인딩 요소를 사용할 경우, 서버 S와 사용자의 공모공격에 의해 다른 사용자의 데이터가 노출될 수 있다. 따라서 모든 암호문에는 서로 다른 블라인딩 요소를 적용해야 한다. III-4에서 언급한 바와 같이 곱셈동형특성으로는 서로 다른 블라인딩 요소를 제거할 수 없다. 이는 덧셈동형특성을 갖는 암호시스템을 사용해야함을 의미한다.

4.2 제안하는 기법

두 안전한 소수(safe prime) p, q가 존재하고, $N=pq$라 하자. 즉, $p=2p'+1$, $q=2q'+1$이며, p'과 q' 또한 소수이다. 이차잉여법(Quadratic residue modulo) N^2을 갖는 순환군의 원소 중에서 위수가 N인 원소는 $\alpha=(1+kN) \bmod N^2$의 형태를 띤다. 본 논문에서는 α의 형태를 띠는 원소를 사용한다. 이와 더불어 위수가 N이며, 생성자가 g인 두 곱셈 순환군 G와 G_T를 사용한다. 두 곱셈군의 위수가 같으므로 겹선형 쌍함수를 정의할 수 있다.

$$\hat{e}: G \times G \rightarrow G_T \tag{1}$$

본 논문에서는 수식 (1)을 바탕으로 PRE를 구성하고, 이를 이용하여 사용자의 공개키로 암호화된 암호문을 서버 S가 복호화 할 수 있도록 변환한다. 다음 페이지의 (그림 1)은 제안하는 기법의 대략적인 동작을 나타낸 것이며, 세부적인 동작은 다음과 같다.

- **초기화** $PP \leftarrow Init(\lambda)$: 서버 S는 비트열의 길이가 λ인 두 safe prime p와 q를 선택한 후, $N=pq$를 계산한다. 위수가 N인 순환군 G에 대하여 임의의 원소 $g \in Z_N^*$를 선택하고, $\rho=\hat{e}(g,g)$를 계산한다. 이렇게 계산한 공개파라미터 $PP=(N, g, \rho, G, G_T, \hat{e})$를 서버 C와 사용자들에게 배포한다.

- **키 생성** $(pk_i, sk_i) \leftarrow KeyGen(PP)$: 모든 개체는 임의의 비밀키 $sk_i \in Z_N^*$을 선택하고, 공개키 $pk_i=g^{sk_i}$를 계산한다. 서버 S의 경우 초기화 단계에서 키를 생성하여 PP와 함께 배포한다.

- **재암호화키 생성** $rk_{i \rightarrow S} \leftarrow ReKeyGen(pk_S, sk_i)$: 사용자들은 서버 S의 공개키 pk_S를 이용하여 $rk_{i \rightarrow S}=pk_S^{1/sk_i}$를 계산한다.

- **2단계 암호화** $c_i \leftarrow Enc_2(pk_i, m_i)$: 사용자들은

임의의 값 $a_i \in Z_{N^2}$를 선택한 후, 재암호화를 허용할 데이터 m_i에 대하여 다음과 같이 암호문 $c_i = (A_i, B_i)$를 계산한다. 암호화가 완료되면 $(pk_i, rk_{i\to S}, c_i)$를 서버 C에게 전달한다.

$$A_i = pk_i^{a_i} \bmod N^2,\ B_i = \rho^{a_i}(1+m_iN) \bmod N^2.$$

- **1단계 암호화**: $c_i \leftarrow Enc_1(pk_i, m_i)$: 사용자들은 임의의 값 $a_i \in Z_{N^2}$를 선택한 후, 재암호화 없이 사용 할 데이터 m_i에 대하여 다음과 같이 암호문 $c_i = (A_i, B_i)$를 계산한다.

$$A_i = \hat{e}(pk_i, g)^{a_i} \bmod N^2,$$
$$B_i = \rho^{a_i}(1+m_iN) \bmod N^2.$$

- **재암호화** $c_{i\to S} \leftarrow ReEnc(rk_{i\to S}, c_i)$: 서버 C는 $A_{i\to S} = \hat{e}(A_i, rk_{i\to S})$를 계산한다. 즉, 재암호화 결과는 $c_{i\to S} = (A_{i\to S}, B_i)$이다.

- **덧셈연산** $c_{i+j} \leftarrow Add(c_{i\to S}, c_{j\to S})$: 서버 C는 $c_{i+j} = c_{i\to S} \odot c_{j\to S}$를 계산한다.

- **곱셈연산** $c_{i\times j} \leftarrow Mult(c_{i\to S}, c_{j\to S})$: 서버 C는 임의의 r_i, r_j를 선택하여 다음과 같이 데이터를 블라인딩 한 후 서버 S에게 전달한다.

$$c'_{x\to S} = Add(c_{x\to S}, Enc_1(pk_S, -r_x))_{[x \in \{i,j\}]}.$$

서버 S는 $m'_x = Dec(sk_S, c'_{x\to S})_{[x \in \{i,j\}]}$를 계산한 후, $c'_{i\times j} = (A'_{i\times j}, B'_{i\times j}) = Enc_1(pk_S, m'_i m'_j)$를 계산하여 서버 C에게 전달한다. 서버 C는 다음과 같이 블라인딩 요소를 제거한다.

$$c_\gamma = (A_\gamma, B_\gamma) = Enc_1(pk_S, -r_i r_j),$$
$$c_{i\times j} = (A_{i\times j}, B_{i\times j}),$$
$$A_{i\times j} = A'_{i\times j} A_{i\to S}^{r_j} A_{j\to S}^{r_i} A_\gamma \bmod N^2,$$
$$B_{i\times j} = B'_{i\times j} B_{i\to S}^{r_j} B_{j\to S}^{r_i} B_\gamma \bmod N^2.$$

- **검색** $Enc_1(pk_i, y) \leftarrow Ret(pk_i, Enc_1(pk_S, y))$: 사용자가 요구한 연산의 결과를 y라 하였을 때, 이 값은 $c_y = Enc_1(pk_S, y)$ 형태로 서버 C에 저장되어 있다. 서버 C는 임의의 값 r_i을 선택한 후, $c'_y = Add(c_y, Enc_1(pk_S, r_i))$을 계산한다. 서버 S는 (pk_i, c'_y)을 받아 $c''_y = Enc_1(pk_i, Dec(sk_S, c'_y))$

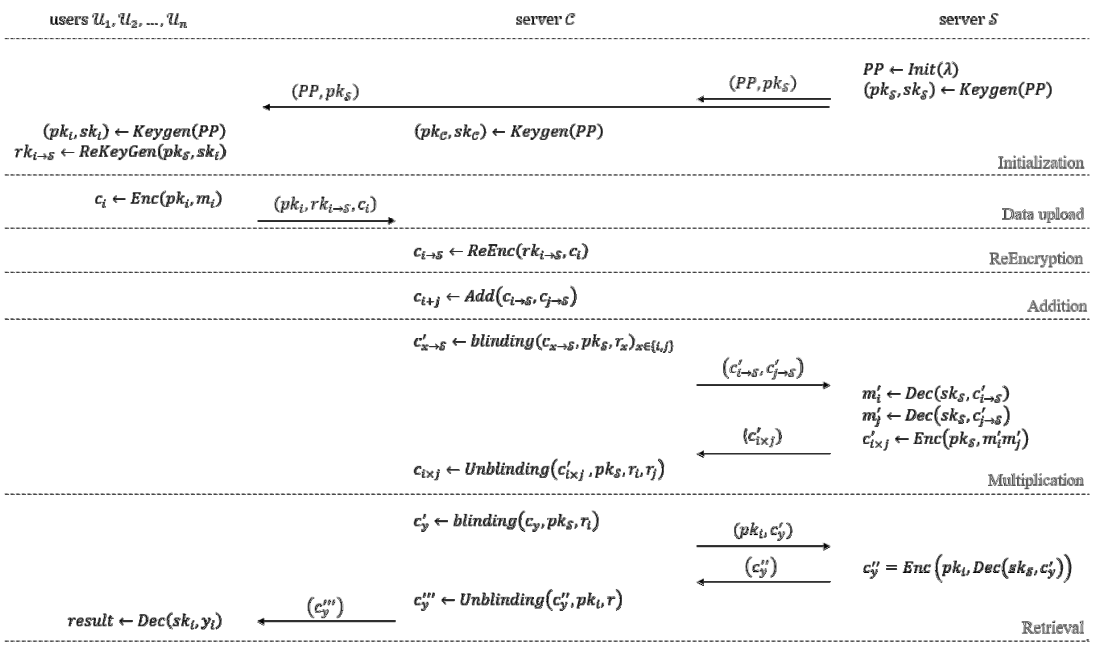


Fig. 1. Protocol overview

을 계산하여 서버 C에게 반환한다. 서버 C는 $c_y^{'''} = Add(c_y^{''}, Enc_1(pk_i, -r_i))$를 계산하여 사용자에게 제공한다.

V. 정확성 및 안전성 분석

본 장에서는 앞서 제안한 기법이 온전히 동작함을 보이고, 보안 요구사항 충족 여부를 분석한다. 본 논문에서는 초기화, 키 생성, 재암호화키 생성이 올바르게 이루어지고, 서버 C에게 안전하게 전달되었다고 가정한다. 즉, 재암호화, 덧셈연산, 곱셈연산, 검색, 복호화가 올바르게 동작하는지 검증한다. 안전성에 대해서는 제안하는 기법에 앞서 언급한 보안 요구사항을 만족하는지 분석한다.

5.1 정확성 (correctness)

- **재암호화 및 서버 복호화**: 제안하는 기법의 재암호화는 암호문 두 파트 중 A에 포함된 사용자의 개인키를 서버 S의 개인키로 치환하는 것이다. $A_i = g^{sk_i \cdot a_i}$이며, $rk_{i\to S} = g^{sk_S/sk_i}$이다. 이들을 이용하여 겹선형 쌍곡선을 계산하면 다음과 같은 결과를 얻을 수 있다.

$$A_{i\to S} = \hat{e}(A, rk_{i\to S}) = \hat{e}(g,g)^{sk_S \cdot a_i} = \rho^{sk_S \cdot a_i}.$$

서버 S는 재암호화 된 암호문을 다음과 같이 복호화함으로써 평문 데이터를 얻을 수 있다.

$$m_i = ((B_i^{sk_S}/A_i) - 1 \bmod N^2)/(sk_S N).$$

- **덧셈연산**: 제안하는 기법은 BCP 암호시스템에 기반을 두고 있다. BCP 암호시스템은 덧셈동형 특성을 가지고 있으며, 이를 재구성한 본 기법 또한 덧셈동형특성을 가지고 있다. 서버 S의 공개키로 재암호화 된 두 사용자 i, j의 암호문을 각각 (A_i, B_i), (A_j, B_j)라 하자. 이때, 두 암호문의 구성요소 단위 곱셈(component-wise multiplication)은 $(A_{i\to S}A_{j\to S}, B_iB_j)$와 같이 이루어진다. 해당 수식의 각 구성요소는 다음과 같이 풀어 쓸 수 있다.

$$A_{i\to S}A_{j\to S} = \rho^{sk_S(a_i+a_j)},$$
$$B_iB_j = \rho^{(a_i+a_j)}(1+m_iN)(1+m_jN) \bmod N^2$$
$$= \rho^{(a_i+a_j)}(1+(m_i+m_j)N) \bmod N^2.$$

$a_\delta = a_i + a_j$라 생각하면, 이는 $m_i + m_j$를 암호화한 후, 서버 S의 공개키로 재암호화 한 것과 동일하다.

- **곱셈연산**: 서버 C가 블라인딩하고, 서버 S가 이를 복호화하여 연산한 결과를 올바르게 반환하였다고 가정하자. 그렇다면 서버 C에게 주어진 블라인드 된 연산결과는 다음과 같은 형태가 된다.

$$c_{i\times j}^{'} = (A_{i\times j}^{'}, B_{i\times j}^{'}),$$
$$A_{i\times j}^{'} = pk_S^{a_S} \bmod N^2,$$
$$B_{i\times j}^{'} = \rho^{a_S}(1+(m_i-r_i)(m_j-r_j)N) \bmod N^2.$$

서버 C는 다음과 같은 연산을 통해 $B_{i\times j}^{'}$에서 $-m_ir_j - m_jr_i + r_ir_j$를 제거함으로써 곱셈의 결과값을 얻을 수 있다.

$$c_\delta = (A_\delta, B_\delta) = Enc_1(\rho^{sk_S}, -r_ir_j),$$
$$c_{i\times j} = (A_{i\times j}, B_{i\times j}),$$
$$A_{i\times j} = A_{i\times j}^{'} A_{i\to S}^{r_j} A_{j\to S}^{r_i} A_\delta \bmod N^2$$
$$= pk_S^{a_S(r_j+r_i+2)},$$
$$B_{i\times j} = B_{i\times j}^{'} B_{i\to S}^{r_j} B_{j\to S}^{r_i} B_\delta \bmod N^2$$
$$= \rho^{a_s(r_j+r_i+2)}(1+m_im_jN) \bmod N^2.$$

$a_\delta = a_S(r_j+r_i+2)$라 생각하면, 이는 서버 S의 공개키를 이용하여 m_im_j를 암호화한 것과 동일하다.

- **검색 및 사용자 복호화**: 사용자 i가 서버 C에게 연산결과를 요구하면, 서버 S의 도움을 받아 사용자 i의 공개키로 암호화된 연산결과를 반환한다. 이 과정은 앞서 언급한 블라인드 요소 제거와 서버 복호화를 그대로 사용한다. 하지만, 사용자가 연산결과를 복호화하는 과정은 서버 복호화와 차이점이 존재한다. 서버가 복호화 하는 암호문은 $c = (A, B)$에서 A 파트가 $\rho \in G_T$를 사용

Table 3. Comparison between proposed scheme and existing schemes

	Peter et al.'s scheme [3]	*Vitamin+ [5]*	*Vitamin* [5]*	*Banana+ [7]*	*Banana* [7]*	*Proposed scheme*
properties						
base scheme	BCP Enc.	BBS PRE	BBS PRE	AFGH PRE	AFGH PRE	BCP Enc./ AFGH PRE
homomorphic property	additive	additive	multiplicative	additive	multiplicative	additive
security and correctness						
collusion attack resistance	safe	C and user	(C or S) and user	safe	S and user	safe
correctness	O	O	O	X	O	O
unidirectional communication cost for multiplication using two encrypted data						
C to S	$5nk/2$	nk	nk	$3nk/2$	$3nk/2$	$3nk/2$
S to C	$5nk/2$	$nk/2$	$nk/2$	$3nk/2$	$3nk/2$	$3nk/2$

하지만, 사용자가 복호화 할 암호문은 $g \in G$를 사용한다. 사용자의 암호문을 서버가 복호화 할 수 있도록 재암호화 과정에서 A파트가 G_T로 맵핑되기 때문이다. 따라서 사용자는 A 파트를 G에서 G_T로 맵핑한 후 복호화를 수행해야 한다. 이때의 복호화는 다음과 같이 이루어진다.

$$y = ((B_i^{sk_i} / \hat{e}(A_i, g)) - 1 \bmod N^2) / (sk_i N).$$

5.2 안전성 (security)

제안하는 기법의 안전성은 데이터 블라인딩 기법에 의존하고 있다. 즉, 서버 S는 마스터키를 통해 임의의 암호문을 복호화 할 수 있지만, 서버 C가 제공하는 암호문에는 블라인딩 기법이 적용되어 있기 때문에 사용자의 평문 데이터는 얻을 수 없다. 이와 같은 논리가 성립하기 위해서는 제안하는 기법도 peter 등의 기법과 동일하게 두 서버 간 공모하지 않는다는 가정이 필요하다. 본 절의 나머지 부분에서는 이 가정을 기반으로 각 단계가 안전하게 동작하는지 검증한다.

초기화 단계에서 서버 S는 공개 파라미터와 마스터키를 생성한다. 공개 파라미터의 배포는 서버 S에 의해 직접 이루어지므로, 서버 C에 의한 공개 파라미터 기반 MITM attack을 방지할 수 있다. 또한, 공개 파라미터를 생성하는 과정에서 충분히 큰 소수 p와 q를 사용한다면, 인수분해로 인한 마스터키 노출에 대응할 수 있다. 사용자들은 각자의 공개키로 데이터를 암호화하여 서버 C에게 전달한다. 서버 C는 각 사용자의 개인키를 알 수 없으며, 공모하지 않는다는 가정에 의하여 마스터키를 얻을 수 없으므로, 사용자의 데이터가 서버 C에게 노출되지 않는다.

키를 통합하는 단계에서 서버 C는 AFGH PRE를 이용하여 암호문에 포함된 사용자들의 개인키를 서버 S의 개인키로 대치한다. AFGH PRE는 BBS PRE와 달리 단방향 성질을 가지므로 재암호화키 수정을 통한 이행성을 제공하지 않는다. 따라서 사용자와 서버 C의 공모공격으로부터 데이터를 보호할 수 있다.

덧셈연산 단계에서는 서버 C가 BCP 암호시스템의 덧셈동형특성을 이용하여 자체적으로 연산을 수행한다. 이 단계는 단순히 두 암호문을 곱하는 것으로써, 기반이 되는 BCP 암호시스템이 안전하다면 연산과정에서 데이터 노출이 발생하지 않는다.

곱셈연산 단계에서 서버 C는 서버 S에게 블라인딩 된 암호문을 제공한다. 두 서버가 서로 공모하지 않고 데이터 블라인딩이 정상적으로 이루어졌다면, 서버 S는 단지 복호화만으로는 사용자의 평문 데이터를 얻을 수 없다. 제안하는 기법은 덧셈동형특성을 이용하여 암호문을 블라인딩하므로, 함께 연산되는

암호문에 서로 다른 블라인딩 요소를 적용하여도 차후에 문제없이 제거할 수 있다. 더불어 이를 통해 서버 S가 특정 사용자와 공모하여 해당 사용자의 블라인딩 요소를 알게 되더라도 다른 사용자들의 평문 데이터 노출을 방지할 수 있다.

검색 단계는 곱셈연산 단계와 동일하게 진행된다. 서버 C는 서버 S에게 임의의 값으로 블라인딩 된 결과 값을 제공하므로, 두 서버가 공모하지 않고, 데이터 블라인딩이 정상적으로 이루어졌다면, 서버 S는 단지 복호화만으로는 결과 값을 얻을 수 없다. 만일 서버 S가 블라인딩 요소나 결과 값 둘 중에 하나를 알 수 있다면, 결과적으로 다른 하나도 알게 된다. 하지만 블라인딩 요소는 오직 서버 C만 알고 있으며, 연산의 결과는 서버 C 조차 알 수 없다. 따라서 서버 C가 서버 S와 공모하지 않는다면 현 시점에서 두 서버가 결과 값에 접근할 수 없다.

다만 각 서버는 연산이 완료된 후, 공모한 사용자들을 통해 결과 값을 알 수 있다. 이 경우는 해당 서버와 공모한 사용자를 하나의 개체로 볼 수 있으며, 해당 서버가 사용자로서 프로토콜에 참여한 것으로 바꾸어 생각할 수 있다. 이는 프로토콜이 완료된 후, 해당 서버 역시 결과 값에 접근할 정당한 권한이 있음을 의미한다.

앞서 보인 바와 같이 두 서버가 서로 공모하지 않고, 데이터 블라인딩 기법이 온전하게 적용된다면, 제안하는 기법은 프로토콜의 각 단계에서 사용자의 데이터가 노출되지 않으므로, 프로토콜 전 과정에서 기밀성이 유지된다. [표 3]은 제안하는 기법과 기존 기법들의 안전성과 효율성을 비교한 것이다.

VI. 결 론

OMPC 프로토콜이란 기존 MPC 프로토콜에서 사용자가 직접 수행하던 연산을 외부에 위임한 것으로서, 데이터를 제공한 사용자가 연산이 수행되는 동안 온라인 상태를 유지하거나 연산에 직접 참여할 필요 없이 결과 값을 얻을 수 있는 기법이다. 클라이언트 측면에서는 오직 단 한 차례의 암호화 및 복호화만 수행되므로, 기존의 MPC 프로토콜에 비하여 연산량이 낮으면서도 동일한 결과를 얻을 수 있는 기법이다. 본 논문에서는 최근 제안된 프로토콜들을 분석하고, 이를 기반으로 안전하고 효율적인 OMPC 프로토콜을 제안하였다. 제안된 기법은 BCP 암호시스템에 단방향 PRE를 적용한 것이다. 이를 위해 기존 BCP 암호시스템을 일부 수정하였다. BCP 암호시스템을 이용한다는 면에서는 Peter 등의 기법과 동일하지만, 단방향 PRE를 적용함으로써 서버 간 통신량을 낮추었다. 제안된 기법은 BCP 암호시스템과 동일하게 덧셈동형특성을 갖고 있으므로, 서로 다른 블라인딩 요소를 적용할 수 있다. 또한, 기존 Wang 등이 제안했던 기법과 달리, PRE를 사용하더라도 복호화 과정에서 이산대수문제를 해결할 필요가 없으며, 단방향 특성을 가지므로 서버 C와 사용자 사이의 공모공격으로부터도 안전하다. 제안된 기법은 기존의 MPC 프로토콜에 비하여 클라이언트의 부담이 적으므로, 최근 확산되고 있는 IoT 환경 등에 널리 활용될 수 있을 것으로 기대된다.

References

[1] A.C. Yao, "Protocols for secure computations," *in Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, pp. 160-164, Nov. 1982.

[2] A. Lòpez-Alt, E. Tromer, and V. Vaikuntanathan, "On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption," *in Proceedings of the 44th Annual ACM Symposium on Theory of Computing* (*STOC `12*), pp. 1219-1234, May 2012.

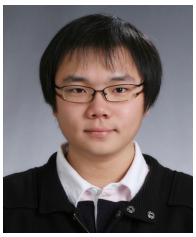
[3] A. Peter, E. Tews, and S. Katzenbeisser, "Efficiently Outsourcing Multiparty Computation Under Multiple Keys," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 12, pp. 2046 - 2058, Nov. 2013.

[4] E. Bresson, D. Catalano, and D. Pointcheval, "A Simple Public-Key Cryptosystem with a Double Trapdoor Decryption Mechanism and Its Applications," *ASIACRYPT 2003*, LNCS 2894, pp. 37 - 54, Nov. 2003.

[5] W. Boyang et al., "Computing encrypted cloud data efficiently under multiple keys," *in Proceedings of IEEE Conference on Communications and Network Security* (*CNS 2013*), pp. 504 - 513, Oct.

2013.
〔6〕 M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," *EUROCRYPT' `98*, pp. 127 - 144, May 1998.
〔7〕 B. Wang et al., "A tale of two clouds: Computing on data encrypted under multiple keys," *in Proceedings of IEEE Conference on Communications and Network Security* (*CNS 2014*), pp. 337-345, Oct. 2014.
〔8〕 G. Ateniese et al., "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Transactions on Information and System Security*, vol. 9, no. 1, pp. 1-30, Feb. 2006.
〔9〕 D. Boneh, and M. Franklin, "Identity-Based Encryption from the Weil Pairing," *CRYPTO 2001*, LNCS 2139, pp. 213-229, Aug. 2001.
〔10〕 C. Gentry, "Fully homomorphic encryption using ideal lattices," *in Proceedings of the 41st Annual ACM Symposium on Theory of Computing* (*STOC `09*), pp. 169-178, May 2009.

〈저자소개〉



은 하 수 (Hasoo Eun) 학생회원
2010년 2월: 한양대학교 컴퓨터공학과 졸업
2010년 3월~현재: 한양대학교 컴퓨터공학과 석박사통합과정
〈관심분야〉 암호기술응용, 암호학

우바이둘라 (Ubaidullah) 학생회원
2005년: Quaid-e-Awam 대학교, 공학/과학/기술학과 졸업
2011년: NUST 컴퓨터시스템공학과 석사
2011년 3월~현재: 한양대학교 컴퓨터공학과 석박사통합과정
〈관심분야〉 암호화폐, 차량형무선네트워크, IoT, 모바일소셜네트워크, 클라우드컴퓨팅

오 희 국 (Heekuck Oh) 종신회원
1982년: 한양대학교 전자공학과 졸업
1989년 아이오와주립대학 전자계산학과 석사
1992년 아이오와주립대학 전자계산학과 박사
1993년~1994년: 한국전자통신연구원 선임연구원
1995년 3월~현재: 한양대학교 컴퓨터공학과 교수
〈관심분야〉 암호기술응용, 시스템보안