논문 21-46-05-01 The Journal of Korean Institute of Communications and Information Sciences '21-05 Vol.46 No.05
https://doi.org/10.7840/kics.2021.46.5.777

SGX 기반의 키워드 탐색 공개키 암호 기법

윤 현 도*, 허 준 범°

SGX-Based Public Key Encryption with Keyword Search

Hyundo Yoon*, Junbeom Hur°

요 약

키워드 탐색이 가능한 공개키 암호화 기법(이하 PEKS)은 외부 서버에 저장되어 있는 암호화된 데이터를 검색 가능하도록 한다. 외부에 저장되어 있는 데이터를 갱신하는 경우에 정보 유출이 기존 쿼리에 의해 발생할 수 있는 가능성이 있다. 기존 연구는 이와 같은 정보 유출을 막기 위해 순방향 안전성을 보장하는 기법을 사용하지만 성능 저하에 대한 문제를 가지고 있다. 본 논문에서는 Intel SGX의 Software Guard Extension(이하 SGX)를 활용 순방향 안전성을 보장하는 PEKS 기법을 소개한다. 제안된 기법은 기존 기법들보다 좋은 성능을 보여준다. 특히, n개의 데이터에 대한 쿼리 처리 비용을 O(n)에서 O(1)로 줄였다. 성능 분석에 의하면 전체적인 연산 비용이 평균적으로 80% 줄었다. 그리고 SGX 기반 순방향 안전한 PEKS의 보안적 정의를 하고 제안 기법의 안전성을 증명하였다.

키워드 : 신뢰 실행 환경, 키워드 탐색 공개키 암호 기법, 인텔 SGX
Key Words : Public Key Encryption with Keyword Search, Trusted Execution Environment, Intel SGX

ABSTRACT

Public key encryption with keyword search (PEKS) enables users to search over encrypted data outsourced to an untrusted server. Unfortunately, updates to the outsourced data may incur information leakage by exploiting the previously submitted queries. Prior works addressed this issue by means of forward privacy, but most of them suffer from significant performance degradation. In this paper, we present a novel forward private PEKS scheme leveraging Software Guard Extension(SGX), a trusted execution environment provided by Intel. The proposed scheme presents substantial performance improvements over prior work. Specifically, we reduce the query processing cost from O(n) to O(1), where n is the number of encrypted data. According to our performance analysis, the overall computation time is reduced by 80% on average. Lastly, we provide a formal security definition of SGX-based forward private PEKS, as well as a rigorous security proof of the proposed scheme.

Ⅰ. 서 론

클라우드 서비스 제공자에게 데이터 아웃소싱을 맡기는 것은 데이터 관리 측면에서 여러가지 이점이 있지만, 데이터 안전성과 프라이버시 유출 같은 문제들이 있다. 아웃소싱을 하기 전에 데이터를 암호화는 것

※ 본 연구는 방위사업청과 국방과학연구소가 지원하는 미래 전투체계 네트워크기술 특화연구센터 사업의 일환으로 수행 되었습니다.(UD190033ED)
※ [29] 논문의 확장 논문
* First Author : Korea University Department of Informatics, hdyoon@isslab.korea.ac.kr, 학생회원
° Corresponding Author : Korea University Department of Informatics, jbhur@korea.ackr, 정회원
논문번호 : 202102-040-A-RN, Received February 18, 2021; Revised February 21, 2021; February 21, 2021

은 데이터 프라이버시 문제들을 해결할 수 있지만, 데이터를 암호화하는 것은 단순한 작업이 아니며, 아웃소싱한 데이터를 탐색하는 것과 같은 데이터관리 기능 중 중요한 기능들에서 문제가 발생할 수 있다. PEKS는 이 딜레마를 해결해준다. PEKS는 데이터 송신자가 공개키를 통해 암호화를 한 암호문을 공개키와 관련된 비밀키를 가지고 있는 데이터 수신자만이 탐색이 가능하게 하는 기법이다.[1]

불행하게도 이전 PEKS 기법들은 쿼리 유출 공격에 취약했다. 예를 들어 파일 삽입 공격에서는[2] 공격자인 데이터 송신자가 악의적으로 파일들을 취사선택해 만들어서 데이터 수신자의 공개키로 암호화를 한 후, 클라우드 저장소에 아웃소싱한다. 이 후 공격자는 특정 수신자가 쿼리에 반환하는 파일들을 감시하는 방식으로 파일 액세스 패턴을 관측해 수신자의 쿼리를 유출시킨다.

이에 대항책으로 순방향 안전성을 보장하는 PEKS가 발표되었다[3]. 이 새로운 기법은 과거의 탐색 쿼리들을 새로 입력되는 파일에 사용하지 않게 하는 기법이다. 하지만 대부분의 클라우드 기반의 어플리케이션들에서는 여러 데이터 송신자들이 수신자 각각에게 데이터를 보내는 다중 사용자 환경이므로, 이전 기법들은 이러한 환경과 맞지 않았다(이 연구에서는 다중 수신자 환경을 고려하지 않았다. 따라서 이후로 이 연구에서 언급하는 다중 사용자 환경은 오직 다중 송신자 환경만을 의미한다). 따라서 대용량의 다중 사용자 설정을 지원할 수 있는 새로운 순방향 안전성을 보장하는 PEKS를 고안하는 것은 PEKS에 관련된 연구의 맥락에서 의미 있는 목표이다.

또한 이전 연구들은 실용성을 하락 시키는 높은 통신 오버헤드로 문제가 많았다. 예를 들어, Zhang의 연구[4]에서는 키 폐기 기법을 사용해서 순방향 안전성을 보장했는데, 쿼리가 진행될 때마다 키 갱신 메시지를 분배하는 키 관리 작업들이 많은 비용이 필요해지는 결과를 초래했다. Zeng의 연구[5]에서는 이런 키 폐기 기법을 사용하지 않고 순방향 안전성을 보장하는 기법을 발표했다. 하지만 이 기법은 실제 환경에서는 감당하기 힘든 연산비용을 초래하는 광범위한 암호학적 요소들을 기반하고 있다. 더하여 눈에 띄는 통신 오버헤드를 발생할 수 있는 시간 주기로 쿼리의 크기를 정하고 있다.

안전하고 효율적인 기법을 위해 인텔의 Software Guard Extension(SGX)[6-10]같은 신뢰 실행 환경(TEE)를 사용하는 방법이 있다. TEE는 (신뢰할 수 없는)메모리에서 (신뢰할 수 있는) 격리된 메모리 공간이나 엔클레이브로 데이터를 불러오거나 코딩할 수 있도록 메모리 격리를 제공한다. TEE는 데이터나 프로세스를 엔클레이브를 사용해 운영체제나 하이퍼바이저로부터 보호하기 때문에, 운영체제나 하이퍼바이저가 관련해 있어도 기밀성과 무결성을 보장할 수 있다. 최근 Amjad의 연구[11]에서는 순방향 안전성을 보장하는 SGX기반 동적탐색 대칭암호 기법을 소개했다. 하지만 이 기법 또한 다중 사용자 환경에는 적용할 수 없었다.

본 연구에서는 키워드 탐색 기법을 통해 순방향 안전성을 보장하는 SGX기반 공개키 암호화 기법(forward private SGX-based public key encryption with keyword search scheme 이하 SPEKS)을 소개한다. 본 연구가 진행된 시점에서는 이 기법은 다중 사용자 환경에서 순방향 안전성을 보장할 수 있는 첫 번째 SGX기반 PEKS이다. 이 기법은 현재 데이터 상태와 이전 쿼리들의 연결을 해제하여 순방향 안전성을 보장하기 위해 탐색 카운터를 사용한다. 자세하게 말하자면, 데이터 갱신 때마다 갱신하는 탐색 카운터를 데이터 수신자와 클라우드 서버 둘 다 공유한다. 가장 최근의 탐색 카운터를 사용해 현재 데이터를 암호화하기 때문에 이전 쿼리들은 나중에 갱신한 데이터와 관련이 있을 수 없다. 덧붙여서 SPEKS는 이전 연구들[4,5,12]에 능가하는 성능을 보여주며 인텔 SGX를 사용하여 순방향 안전성을 위협하는 더 강한 공격들로부터 보호한다.

이 연구의 의미는 다음과 같다

- 인텔 SGX를 사용해 키워드 탐색을 사용하는 공개키 암호화를 통해 다중 사용자 환경에서도 순방향 안전성을 보장하는 첫번째 SGX기반 PEKS 기법이다.
- 이전 연구들은 암호화한 데이터에 비례해 여러 쿼리들을 요구하지만, 하나의 쿼리로 여러 암호화한 데이터를 탐색하기에 충분해서 통신 비용을 눈에 띄게 줄였다.
- SGX기반 순방향 안전성을 보장하는 PEKS의 보안모델을 정의했으며 이 기법의 보안성을 증명했다.
- SGX를 사용해 이 기법을 실행해서, 이전 기법들과 우리 기법의 성능을 평가했다. 본 실험에 의하면 이 기법은 보안성 하락 없이 이전 기법들에 비해 훨씬 효율적이었다.

Ⅱ. 배 경

인텔 SGX는 순방향 안전성을 보장한 암호화기법을 설계하기 위해 사용하였다. SGX는 x86 명령어 집합 기계(이하 ISA)의 확장으로 인텔 6세대 스카이레이크 프로세서부터 사용할 수 있다[28]. SGX는 메모리 격리, 엔클레이브 페이지 캐시, 소프트웨어 증명 같은, 이 기법을 설계하는 데에 기반이 되는 중요한 기능들을 제공한다. 이 장에서는, SGX 구조와 제안기법의 바탕이 되는 기본 PEKS 알고리즘을 소개한다.

2.1 Intel Software Guard Extension(SGX)

2.1.1 Memory Isolation(메모리 격리)

SGX 플랫폼은 신뢰할 수 있는 부분과 신뢰할 수 없는 부분으로 나눌 수 있다. 엔클레이브는 신뢰할 수 있는 부분 혹은 내용이 보호되는 물리적 공간의 독립적 영역이라 할 수 있다. 엔클레이브의 메모리 공간은 더 높은 계층에서 실행되는 프로세스를 포함한 엔클레이브 외부의 프로세스로부터 격리되어 있다. 따라서 우선되는 프로세스를 가지는 운영체제, 펌웨어, 하이퍼바이저, 시스템관리 모드(SMM)의 코드 같은 다른 프로세스들의 엔클레이브 메모리 접근이 허가되지 않는다.

엔클레이브 메모리는 가상 메모리의 신뢰할 수 없는 부분으로 매핑되고, 신뢰할 수 없는 부분은 가상 주소 공간 하에 보통의 프로세스 위에서 실행된다. 엔클레이브가 호스트 프로세스의 전체 가상 메모리로 접근이 가능해지기 위해서 엔클레이브 메모리의 매핑은 매우 중요하다. 하지만 호스트 프로세스는 특정 인터페이스를 통해 유일하게 엔클레이브를 호출할 수 있다. 덧붙여 엔클레이브 안에 있는 실행코드와 데이터는 메모리의 신뢰할 수 없는 부분에 있을 때 암호화된다. 한편으로는 엔클레이브로 불러질 때 엔클레이브는 CPU에서 즉시 해독된다. 따라서 프로세서는 잠재적으로 위협이 되는 다른 프로세스들이 코드를 검사할 수 없게 보호한다.

2.1.2 Enclave Page Cache(EPC)

EPC는 엔클레이브 코드와 데이터가 위치하는 메모리 영역이다. 메모리 암호화 엔진(MEE)를 사용해서, EPC는 암호화되고 해당 메모리 버스의 외부읽기는 암호화된 데이터만 모니터링 할 수 있다. EPC로 사용하기 위해 최대 128MB까지 시스템의 메인 메모리의 고정된 부분은 엔클레이브와 메타데이터를 저장하는 데에 할당된다. 할당된 메모리는 엔클레이브와 관련 메타데이터가 공유하기 때문에, 엔클레이브 캐시는 평균적으로 96MB를 사용할 수 있다[13]. 메모리 제한 때문에 할당된 메모리보다 큰 데이터를 다룰 때에는 페이지를 바꿀 필요가 있다. 부팅 페이즈 동안에는 SGX 메모리는 시스템의 런타임동안 정적으로 차지한다. 만약 여러 개의 엔클레이브가 있다면, 메모리는 운영체제에 의해 동적으로 관리되고, 각 엔클레이브에 메모리가 할당된다. 페이지 교체가 일어나면, 부팅 시간에 생성된 키는 페이지의 암호화와 복호화에 사용한다.

2.1.3 Software Attestation[14]

SGX는 가까이 혹은 멀리 생성된 엔클레이브들이 유효한지 증명하는 소프트웨어 증명을 지원한다. 엔클레이브가 생길 때 시작되는 엔클레이브 판단은 엔클레이브가 올바른지 검증한다. SGX 증명 기능성 덕분에 이러한 판단은 악성 엔클레이브를 찾아낼 수 있고 맞는 엔클레이브인지 인증할 수 있다. 가까이 있는 엔클레이브를 증명하기 위해선, EREPORT와 EGETKEY 명령어들이 목표 엔클레이브가 맞는지 인증서를 생성하고 증명하는데 사용된다. 원격 증명을 위해서는 SGX의 한 부분인 인용 엔클레이브(QE)에 의해 인증서를 생성한다. 인증서를 생성하기 전에, QE는 타당한 엔클레이브인지 보장할 수 있는 하드웨어에서만의 측정만을 인정한다.

2.2 공개키 기반 키워드 탐색 기법(PEKS)

Boneh의 연구[1]에서 처음으로 키워드 탐색이 가능한 공개키 암호화를 소개했다(PEKS). 대칭 탐색가능 암호화와 비교하여 PEKS는 데이터 공유에 더 나은 성능을 보여주었다. Abdalla의 연구[15]에서는 PEKS의 일반적인 구조를 제안하였고 어떻게 익명 ID기반 암호화에서 탐색가능 암호화에 기반한 공개키를 얻을 수 있는지 보여주었다. PEKS 기법에서는 데이터 수신자가 키워드를 위한 트랩도어 함수를 통해 게이트웨이를 제공한다. 그리고 데이터 송신자가 데이터의 수신자의 공개키를 사용해 키워드를 암호화하고 암호문을 서버나 게이트웨이에 보낸다. 서버나 게이트웨이는 탐색토큰과 암호문에 탐색 함수나 검증 함수를 적용한다. 탐색토큰과 암호문이 맞게 확인된 키워드들은, 탐색함수나 검증함수가 1을 반환하고 아닐 경우 0을 반환한다. 이 기법은 표준 모델에서 안전성을 검증받았지만, 특정 수치보다 적은 악의적 사용자가 있는 조건에서만을 상정한다.

Ⅲ. SPEKS 개요 및 정의

이 장에서는 제안 SPEKS 기법의 기본적인 구조와 암호화된 데이터에 대한 사용자의 탐색 과정을 설명한다. 그리고 SPEKS의 알고리즘과 보안 모델을 설명한다.

3.1 SPEKS 개요

Fig 1.은 제안 기법의 기본적인 디자인이다. 제안 기법에는 데이터 수신자, 데이터 송신자, SGX가 활성화된 클라우드 서버, 이렇게 3개의 다른 개체들로 구성되어 있다.

Setup 단계에서는 데이터 수신자가 개인키 SK_u, 공개키 PK_u, 대칭키 K_u를 생성한다. 데이터 수신자는 클라우드 서버의 엔클레이브와 SGX Attestation 프로토콜을 사용하여 보안채널을 생성한다(Step 1). 엔클레이브는 두 개의 프로비저닝된 비밀 키를 저장하고 있다. 저장되어 있는 키는 엔클레이브가 사용되지 않을 때, 로컬 메모리에 암호화되어 저장된다.

PEKS 알고리즘 단계에서 데이터 수신자는 데이터 송신자의 탐색 카운터를 받아 사전에 정의된 키워드와 함께 데이터를 암호화하여 서버에 업로드 한다(Step 2와 3). 데이터 수신자는 생성한 탐색 쿼리를 대칭키 SK_u로 암호화하여 Trapdoor 알고리즘(Step 4)을 통해 클라우드 서버의 엔클레이브에 보낸다. 엔클레이브는 복호화와 탐색 쿼리에 필요한 키를 소유하고 있다. 탐색 알고리즘에서는 클라우드 서버에 저장되어 있는 데이터 레코드를 엔클레이브로 불러온다. 불러온 데이터 레코드의 크기가 EPC보다 크면, 레코드는 작은 조각으로 나누어지고 일부 레코드를 엔클레이브로 여러 번 불러온다. 엔클레이브는 탐색 쿼리를 복호화하여 매칭되는 레코드를 탐색한다(Step 6). 마지막으로 매칭되는 결과가 있으면 데이터 수신자에게 결과 값을 반환한다(Step 7).

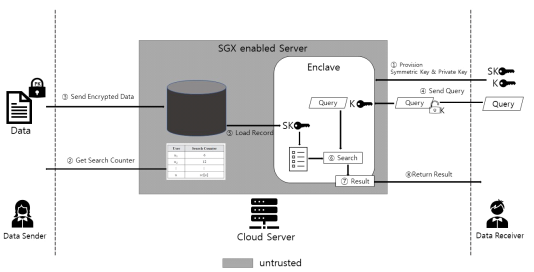

그림. 1. SPEKS의 기본 디자인
Fig. 1. SPEKS's Design Overview

3.2 알고리즘 및 보안성 정의

SPEKS는 네 개의 polynomial-time 알고리즘으로 구성되어 있다.

$$SPEKS=(Setup(1^{\lambda},\lambda),PEKS(PK_u,w,u,F),\\ Trapdoor(K_u,w,sc[u]),Search(SK_u,R,t_w))$$

Definition 1. (SPEKS) SPEKS는 네 개의 polynomial-time 알고리즘(Setup, PEKS, Trapdoor, Search)들로 구성되어 있다.

- $(PK_u,SK_u,K_u)\leftarrow Setup(1^{\lambda},\lambda)$: 보안 파라미터 1^{λ}를 받아 개인키 SK_u와 공개키 PK_u를 생성하고, λ를 받아 대칭키 K_u를 생성한다.
- $t_w\leftarrow Trapdoor(K_u,w,sc[u])$: 대칭키 K_u, 키워드 w, 탐색 카운터 $sc[u]$를 입력 값으로 받아 탐색 토큰 t_w를 생성한다.
- $R\leftarrow PEKS(PK_u,w,u,F)$: 공개키 PK_u, 키워드 w, 사용자 인덱스 u, 데이터 셋 F를 입력 값으로 받아 레코드 R를 생성한다.
- $(F/\perp)\leftarrow Search(SK_u,R,t_w)$: 개인키 SK_u, 레코드 R, 탐색 토큰 t_w를 입력 값으로 받아 탐색에 성공하면 F를 반환, 탐색 결과가 없다면 $\perp$를 반환한다.

Definition 2. (Correctness) D는 Definition 1에 정의된 4개의 알고리즘으로 구성되어 있는 SPEKS를 의미한다. 암호문 $R\leftarrow PEKS(PK_u,w,u,F)$와 $t_w\leftarrow Trapdoor(K_u,w,sc[u])$가 있다. 올바르게 생성된 데이터 수신자의 공개키 쌍 (PK,SK), 대칭키 K가 있을 때, $Search(SK_u,R,t_w)=1$일 확률은 1이다.

본 연구에서는 보안 모델을 [16]에서 소개한 three steps framework를 기반으로 정의했다. 첫 번째, 적응형 공격자가 얻을 수 있는 정보의 상한선을 정해 formulated leakage를 정의한다. 두 번째, 적응형 공격자 A와 polynomial-time simulator S에 대하여 $Real_A(\lambda)$와 $Ideal_{A,S}(\lambda)$ 게임을 정의한다. $Real_A(\lambda)$은 실제 프로토콜이다. $Ideal_{A,S}(\lambda)$ 은 formulated leakage를 이용하여 S을 사용해 real game에 대한 모의 프로토콜이다. 이전에 실행된 프로토콜에서 얻은 정보를 적응형 공격자는 이후 쿼리에 사용할 수 있다. 마지막으로 제안 기법이 CKA2-secure 하다는 것은 A가 결과 값을 구분할 확

률이 0에 가깝다는 것을 보인다. 확률이 0에 가깝다면 A는 앞서 정의한 leakage이외에는 더 많은 정보를 얻을 수 없다.

제안 기법은 [13]에서 소개된 연구와 비슷하게 클라우드 서버와 SGX간의 추가의 처리과정이 있다. 이 처리과정은 공격자가 모니터링 가능하다. 그렇기 때문에 기존의 보안 모델에서 하드웨어-보안 모델로 확장했다. L_{hw}는 CKA2-HW-security의 leakage를 의미한다.

Definition 3. (CKA2-HW-security) D는 Definition 1에 정의된 4개의 알고리즘으로 구성되어 있는 SPEKS를 의미한다. A는 정적인 소극적 공격자, S는 leakage 함수 L_{PEKS}, L_{hw}을 받는 정적 시뮬레이터이다. 두 확률론적 실험 $Real_A(\lambda)$와 $Ideal_{A,S}(\lambda)$는 아래와 같다.

- $Real_A(\lambda)$: 데이터 수신자는 $setup(1^\lambda,\lambda)$를 실행하여 공개키 쌍 (PK,SK), 대칭키 K을 생성한다. A는 사용자 u의 탐색 카운터를 반환한다. 데이터 송신자는 $R \leftarrow PEKS(PK_u,w,u,F)$를 연산하여 A에게 R를 전달한다. 데이터 수신자는 탐색 토큰 t_w를 A에게 반환한다. A는 R과 반환된 탐색 토큰을 사용에 신뢰 하드웨어에 쿼리를 보낼 수 있다. 신뢰 하드웨어는 $(F/\perp) \leftarrow Search(SK_u,R,t_w)$를 통해 쿼리에 대한 응답을 한다. 쿼리가 매칭되는 결과가 있다면 탐색 카운터는 증가하고 A는 결과값 비트 b를 반환한다.
- $Ideal_{A,S}(\lambda)$: 공격자 A는 데이터 송신자에게 탐색 카운터 sc를 준다. 데이터 송신자는 L_{PEKS}를 이용하여 R를 생성해 A에게 보낸다. 시뮬레이터 S는 탐색 토큰 t_w를 생성하여 A에게 전달한다. A는 R과 탐색 토큰 t_w를 이용해 신뢰 하드웨어를 시뮬레이션하는 S에게 쿼리를 만들어 보낸다. L_{hw}가 주어졌을 때, S는 탐색 결과를 반환한다. 공격자 A는 결과 값 비트 b를 반환한다.

$$|\Pr[Real_A(\lambda)=1]-\Pr[Ideal_{A,S}(\lambda)=1]| \leq \neg l(\lambda) \quad (1)$$

(1)과 같다면 D는 적응형 키워드 선택 공격에서 (L_{PEKS}, L_{hw})-secure하다.

Ⅳ. 기법설계

4.1 Cryptographic Primitive

Enc_{PKE}와 Dec_{PKE}는 IND-CPA에 안전한 공개키 암호화 알고리즘과 복호화 알고리즘이다. Enc_{SKE}와 Dec_{SKE}는 IND-CPA에 안전한 대칭키 암호화 알고리즘과 복호화 알고리즘이다. 제안 기법은 위 공개키, 대칭키 암호화/복호화 알고리즘을 기반으로 설계되었다.

4.2 Provisioning

데이터 수신자와 엔클레이브 간의 키 공유를 위해 데이터 수신자의 개인키와 대칭키가 엔클레이브에 프로비져닝 된다. 키가 다른 개체에 유출이 되면 안 되기 때문에 데이터 수신자와 엔클레이브는 SGX의 attestation 기능을 이용하여 보안 채널을 생성한다. 생성된 보안 채널로 통신 할 때 사용되는 암호 키는 엔클레이브 생성 과정에서 발급한 공개키 쌍 sk_E와 pk_E를 사용한다. 하드웨어의 random number generator (rdrand[17])는 키 생성 과정에 필요한 무작위성을 보장한다.

4.3 알고리즘

본 논문에서 제안하는 순방향 안전성을 보장하는 키워드 탐색 기반 탐색 가능한 암호 기법(SPEKS)은 네 개의 알고리즘($Setup, PEKS, Trapdoor, Search$)로 구성되어 있다.

Algorithm 1은 SPEKS의 Setup 알고리즘을 설명하고 있다. $(PK_u,SK_u,K) \leftarrow Setup(1^\lambda,\lambda)$ 알고리즘에서 데이터 수신자(이하 DR)은 PK_u, SK_u, K_u을 생성한다. 그리고 DR은 SK_u와 K_u를 클라우드 서버의 엔클레이브(이하 $Enclave_{CS}$)에 보안 채널을 통해 프로비져닝한다.

Algorithm 2는 PEKS 알고리즘을 설명하고 있다.

Algorithm 1: $(PK_u,SK_u,K) \leftarrow Setup(1^\lambda,\lambda)$
DR: $(PK_u,SK_u) \leftarrow Setup(\lambda)$ symmetric key $K_u \leftarrow Setup(1^\lambda)$ Provision Private Key SK_u and Symmetric Key K_u to $Enclave_{CS}$

$R \leftarrow PEKS(PK_u, w, u, F)$에서 F는 데이터 셋을 의미한다. 데이터 송신자(이하 DS)는 클라우드 서버(이하 CS)에게 사용자 u의 탐색 카운터 $sc[u]$를 요청한다. 그리고 DS는 $Enc_{PKE}(PK_u, (w, sc[u]))$를 통해 탐색 가능한 암호문 ct를 생성한다. 레코드 R은 세 개의 요소 (d, ind, ct)로 구성되어 있다. d는 데이터, ind는 인덱스, ct는 탐색 가능한 암호문을 의미한다. 생성된 R은 CS에 저장된다.

Algorithm 2: $R <- PEKS(PK_u, w, u, F)$

DS:

Request the search counter of u from the CS

CS:

Return $sc[u]$ to DS

DS:

for $i = 1$ to |F| **do**
  $ct_i \leftarrow Enc_{PKE}(PK_u, (w, sc[u]))$
  $R \leftarrow R \cup \{(d_i, ind_i, ct_i)\}$
end for
Transfer R to CS

CS:

for $i = 1$ to $|R|$ **do**
  $ED \leftarrow ED \cup \{(d_i, ind_i, ct_i)\}$
end for

Algorithm 3은 Trapdoor 알고리즘을 설명하고 있다. DR은 키워드를 이용하여 탐색 토큰을 생성한다. 탐색 토큰을 생성하기 위해서 DR은 탐색 카운터 $sc[u]$와 키워드를 사용하고, 대칭키 K_u를 사용해 탐색 토큰을 암호화한다. 암호화된 탐색 토큰 t_w를 $Enclave_{CS}$로 전송하고 탐색 카운터의 값을 1 증가시킨다.

Algorithm 4는 제안 기법의 Search 함수를 설명한다. $Enclave_{CS}$는 탐색 토큰 t_w와 매칭되는 R가 존재하는지 확인한다. $Enclave_{CS}$는 $Dec_{SKE}(K_u, t_w)$를 통해 keyword w'을 받는다. 그리고 SK를 이용하여 ct를 복호화하여 키워드 w와 탐색 카운터 sc를 받는다. 매칭되는 결과가 있으면 ind_i 값을 R로부터 알아내어 DR에게 반환하고 매칭 되는 결과가 없다면 $\perp$을 반환한다.

Algorithm 3:
$t_w \leftarrow Trapdoor(K_u, w, sc[u])$

DR:

$\tau_w \leftarrow (w, sc[u])$
$t_w \leftarrow Enc_{SKE}(K_u, \tau_w)$
Transfer t_w to $Enclave_{CS}$
$sc[u] \leftarrow sc[u] + 1$

Algorithm 4: $(F/\perp) \leftarrow Search(SK_u, R, t_w)$

CS:

$Enclave_{CS}$:
$(w', sc') \leftarrow Dec_{SKE}(K_u, t_w)$
for $i = 1$ to sc' **do**
  $(w, sc) \leftarrow Dec_{PKE}(SK_u, ct_i)$
  if $(w = w')$ **and** $(i = sc)$ **then**
    return ind_i
  end if
end for
$sc[u] \leftarrow sc[u] + 1$
Return F to DR if match; else, $\perp$

V. 분 석

5.1 안전성 검증

본 장에서는 access pattern과 관련된 leakage function을 정의하여 제안 기법의 안전성을 증명하고 제안 기법 SPEKS가 순방향 안전성을 보장하는지 설명한다.

먼저 두 개의 leakage function $L_{PEKS}(sc)$와 $L_{hw}(sc, R)$을 정의한다. $L_{PEKS}(sc)$는 탐색 카운터 sc를 인자로 받아 레코드R을 출력한다. 레코드R은 암호화된 데이터, 인덱스, 탐색 가능한 암호문으로 이루어져 있다. $L_{hw}(sc, R)$은 sc와 R이 주어졌을 때, 엑세스 패턴 $P(sc, R)$을 출력한다.

Access pattern $P(sc, R)$는 서버의 비신뢰 영역에 메모리에 저장되어 있는 탐색 카운터 sc와 레코드 R에 대한 정보를 가지고 있다. 서버는 데이터 수신자가 sc를 요청할 때 반환되는 값을 확인할 수 있다. R 또한 어떠한 레코드가 엔클레이브와 데이터 수신자에게 전송되는지 유출된다. 본 연구에서는 value access pattern $\triangle(sc, R)$[13] 분석에 활용한다. Value access

pattern은 인덱스 ind를 가지고 있는 결과 값의 레코드의 포인터를 의미한다.

Theorem 1. (Security) SPEKS는 (L_{PEKS}, L_{HW})-secure하게 설계되었다.

증명. 다항 시간 시뮬레이터 S에서 확률론적이고 다항 시간적인 공격자 A는 무시 가능한 확률로 $Real_A(\lambda)$와 $Ideal_{A,S}(\lambda)$를 구분할 수 있다.

- Setup : S는 새로운 랜덤 키 $(\widetilde{PK}, \widetilde{SK}, \widetilde{K}) = Setup(1^{\lambda}, \lambda)$을 생성하여 저장한다.
- R 시뮬레이션 : S는 L_{PEKS}와 탐색 카운터 sc를 받는다. S는 $Enc_{PKE}(PK,(w,sc))$을 이용하여 데이터 셋의 크기 $|F|$만큼의 키워드에 대한 암호화 $C=(C_1,...,C_{|F|})$을 한다. 각 암호화된 값들은 서로 다른 인덱스 값 ind가 부여된다. S는 $R=(ind_i, C_i)$을 출력한다. 위 연산들은 탐색 카운터와 레코드의 크기가 leakage에 포함되어 있기 때문에 가능하다. R 시뮬레이션은 $Real_A(\lambda)$의 출력 값의 크기와 동일하다. 기법에서 사용된 공개키 암호는 IND-CPA-secure 하기 때문에 시뮬레이션의 결과 값과 $Real_A(\lambda)$ 결과 값은 구분할 수 없다.
- t_w 시뮬레이션: 시뮬레이터 S는 $\tau_w=(w,sc)$ 값을 암호화 $Enc_{SKE}(K_u, \tau_w)$ 한다. S는 탐색 토큰 t_w을 결과 값으로 가진다. Enc_{SKE}는 IND-CPA secure하기 때문에 시뮬레이션된 t_w는 $Real_A(\lambda)$의 값과 구분할 수 없다.
- 시큐어 하드웨어 시뮬레이션 : 시간 t에 대하여 S는 탐색 토큰과 L_{hw}를 받는다. S는 $P(sc,R,t)$를 이용하여 access pattern을 시뮬레이션한다. S는 P의 첫 번째 레코드부터 시작하여 R에서 주어진 인덱스를 조사한다. Leakage $\triangle$가 특정 인덱스의 레코드를 결정한다.

표 1. 제안 기법과 기존 기법의 비교 분석 ($|sc|$는 탐색 카운터의 값, n_d는 데이터 수, $|id|$ 식별자의 크기, $|K|$ 키 셋의 크기, $|T|$는 인코딩된 시간 주기)
Table 1. Scheme Comparison

Scheme	Search Time	Query Size	Index Size
Zeng et al.[5]	$O(\|T\| \cdot n_d)$	$O(\|T\|)$	$O(n_d)$
Kim et al.[12]	$O(\|sc\| \cdot n_d)$	$O(\|sc\|)$	$O(n_d+\|id\|)$
Boneh et al.[1]	$O(\|K\| \cdot n_d)$	$O(\|K\|)$	$O(n_d \cdot \|id\|)$
Our scheme	$O(\|sc\| \cdot n_d)$	$O(1)$	$O(n_d)$

공격자 A는 $Real_A(\lambda)$의 액세스와 시뮬레이션의 액세스를 구별하지 못한다. 같은 키워드에 대해서 매번 요청에 따라 결과는 일정하다. $\triangle(sc,R)$의 분명하기 때문에 매칭되는 포인터의 개수와 포인터는 일정하다. 포인터가 가리키는 값은 IND-CCA secure하게 암호화되어 있어 구분 불가능하다.

5.2 성능 분석

본 장에서는 제안 기법의 성능을 분석하고 기존 기법들과 비교 분석을 했다.

실험 환경은 다음과 같다. Intel(R) Core(TM) i7-9700K CPU 3.60 GHz, 16G DDR4 RAM, SGX가 활성화된 64-bit Ubuntu 18.04.4 LTS 운영체제에서 실험을 했다. 그리고 제안 기법은 인텔 SGX Software Development Kit(SDK)를 활용하여 구현했다.

5.2.1 키 생성 및 관리 비용

다중 사용자 환경에서는 키 생성과 키 관리의 초기 비용에 대한 분석이 중요하다. 일반적인 Symmetric Searchable Encryption(이하 SSE)에서 다중 사용자 환경을 지원하기 위해서는 키를 사용자의 수만큼 생성해야 한다. 예를 들어, 사용자의 수를 u라 하고 Diffie-Hellman 키 교환 프로토콜의 초기 비용 값을 $|DH|$라 할 때, 키 생성의 초기 비용은 아래와 같다.

$$\text{(initial key setup cost)} = u \cdot |DH| \quad (2)$$

하지만 제안 기법은 데이터 송신자마다 키 교환 프로토콜을 실행하지 않아도 된다. 그렇기 때문에 키 생성 비용은 일정하다.

기존 기법에서는 데이터 수신자는 데이터 송신자의 수만큼의 키 관리를 해야한다. 따라서, 저장 오버헤드는 데이터 송신자의 수에 따라 증가한다. SSE 기법에서는 데이터 수신자가 데이터 송신자 각각의 키를 저장해야 한다. 키의 사이즈를 $|K|$라 했을 때, 저장 오버헤드는 아래의 수식과 같다.

$$\text{(storage cost)} = u \cdot |K| \quad (3)$$

제안 기법에서는 데이터 수신자가 모든 데이터 송신자들의 키를 저장할 필요가 없기 때문에 데이터 송신자의 수와 상관없이 저장 오버헤드는 일정한 값을 가진다. 따라서, 다중 사용자 환경에서 높은 확장성을 가지는 것을 보인다.

5.2.2 연산 비용

본 연구에서 제안하는 기법은 표 2에 나타난 것처럼 기존의 PEKS 기법들보다 낮은 연산 비용을 보인다. PEKS 알고리즘의 연산 시간의 경우, [12]의 기법은 3.958 ms, [5]의 기법은 8.123 ms이다. 제안 기법은 0.0919 ms로 연산 속도가 큰 차이를 보이며 낮은 연산 오버헤드를 가진다. 탐색 토큰을 생성하는 속도는 제안 기법의 경우 0.02 ms이고 [12]의 기법은 4.85 ms이다. [5]의 기법은 12.11 ms로 연산 시간이 제일 높다. 제안 기법은 탐색 카운터의 값과 상관없이 일정한 연산 비용을 보인다. Zhang의 연구[27]의 기법은 키워드의 크기에 따라 탐색 토큰의 생성하는 시간이 증가한다. 제안 기법의 Search 알고리즘의 연산 시간 또한 제일 낮다. 제안 기법과 [12]의 기법의 탐색 복잡도는 Table 1.에 보인 것처럼 같다. 하지만 실제 연산 시간에서는 차이를 보인다. 평균적으로 제안 기법의 연산 시간은 0.0436 ms이고 [12]의 기법은 0.863 ms이다. 그리고 [5] 기법의 연산 복잡도는 인코드된 시간 주기에 영향을 받는다. [1]의 기법의 탐색 복잡도는 키의 크기에 영향을 받는다.

제안 기법의 연산 오버헤드의 감소는 Intel SGX와 같은 신뢰 실행 환경의 특징을 이용했기 때문이다. 기존 기법들은 연산 오버헤드가 높은 페어링 연산을 기반으로 설계됐다. 예를 들어, 제안 기법의 Trapdoor 알고리즘에서 탐색 토큰을 SGX SDK에서 제공하는 AES-GCM을 활용하여 암호화한다. 기존의 기법들은 RSA나 ECC와 같은 암호 기법을 활용할 수 없다. 왜냐하면 RSA나 ECC로 만든 암호문을 복호화하지 않고 탐색을 할 수 없기 때문이다. 하지만 신뢰 실행 환경에서는 데이터의 안전성을 보장하면서 평문 탐색이 가능하다.

5.2.3 통신 비용

키 폐기 기반의 기법[5]과 탐색 카운터 기반의 기법[12]은 다수의 탐색 토큰을 생성한다. 공개키 쌍이 폐기되고 재발급됨에 따라 기존의 암호문에 대한 탐색 토큰들을 생성해야 한다. 그래서 탐색 토큰의 크기는 누적된 탐색 횟수에 영향을 받는다. 예를 들어, Fig 3.에 나타난 것과 같이 [12] 기법에서 이전에 1000회의 탐색을 했다면, 다음 탐색에서 1000개의 탐색 토큰을 생성해야 한다. 그리고 키 폐기 기반의 기법의 경우 암호문을 매번 갱신해줘야 한다. [27]의 기법 또한 암호문을 갱신한다. 모든 암호문을 갱신하는 과정에서 큰 통신 오버헤드가 발생하여 확장성이 좋지 않다. 표 1의 복잡도 분석에 나온 것처럼 본 연구에서 제안한 기법은 탐색 토큰의 크기는 $O(1)$로 일정하다. 하지만 [5]의 기법은 인코딩된 시간 주기, [12]의 기법은 탐색 카운터, [1]의 기법은 키의 크기에 따라 탐색 토큰의 크기가 변한다.

표 2. 제안기법과 기존 기법의 알고리즘 별 시간 비용
Table 2. Computation cost of schemes

Scheme	PEKS	Trapdoor	Search
Zeng et al.[5]	8.123 ms	12.11 ms	1.546 ms
Kim et al.[12]	3.958 ms	4.854 ms	0.863 ms
Our scheme	0.0919 ms	0.02 ms	0.0436 ms

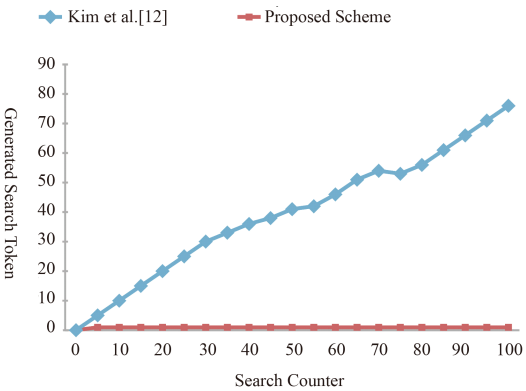

그림. 2. 탐색 카운터 값에 따른 탐색 토큰 생성 개수
Fig. 2. Relationship between search counter and search token

Ⅵ. 관련 연구

본 장에서는 탐색 가능한 암호 기법과 신뢰 실행 환경(TEE)에 대해 설명한다.

6.1 탐색 가능한 암호 기법

탐색 가능한 암호 기법(이하 SE)은 Song[18]에 의해 처음 소개되었다. SE는 지속적으로 연구되어 왔다. SE 기법은 대칭키 기반 탐색 가능한 암호 기법(SSE)과 공개키 기반 탐색 가능한 암호 기법(PEKS) 두 가지로 나누어진다. 하지만 SSE 기법은 대칭키 원시 요소[18-21] 기반으로 PEKS보다 일반적으로 효율적이지만 다중 송신자 환경에는 적합하지 않다. PEKS는 공개키 원시 요소[22,23] 기반으로 Boneh[1]에 의해 처음 소개되었다. PEKS는 효율적인 키 관리에 의해 다중 송신자 환경에 적합하다. PEKS에서 데이터 송신자가 탐색 가능한 암호문을 특정 수신자의 공개키를 이용해 생성한다. 그리고 데이터 수신자는 탐색 쿼리를 전

송하여 데이터를 반환 받는다.

6.2 신뢰 실행 환경 기반 기법 설계

Fisch[24]는 Intel SGX를 이용하여 처음으로 함수 암호를 설계했고 보안 모델을 정의했다. 이후, Intel SGX를 활용한 암호 기법 설계에 대한 연구가 활발히 진행되었다. Fuhry[13]는 Intel SGX를 활용하여 암호화된 데이터 베이스 인덱스 HardIDX를 설계했다. HardIDX는 탐색 프로세스를 엔클레이브 안에 설계했지만 업데이트 프로세스는 지원하지 않는다. 또한, Intel SGX를 이용하여 효율성을 높인 ORAM 기법의 Zerotrace[25]도 제안되었다. Oblix[26]는 Intel SGX를 활용하여 액세스 패턴과 결과 값의 크기에 대한 유출을 최소화하는 기법이다. 신뢰 실행 환경을 이용하여 SE 기법을 설계하는 것은 기법의 효율성과 높은 보안성을 보장하는 데에 효과적이다.

Ⅶ. 결 론

본 연구에서는 Intel SGX를 이용한 순반향 안전성을 보장하는 공개키 기반 키워드 탐색 기법을 제안한다. 제안 기법의 보안 모델을 정의하고 안전성 검증을 했다. 기존 기법들과 비교했을 때, 제안 기법은 높은 효율성을 보인다. 다른 기법들의 경우 다수의 탐색 토큰을 생성하는 반면, 제안 기법은 조건과 상관없이 단일 탐색 토큰을 생성한다. 그리고 인덱스 생성, 탐색 토큰 생성, 탐색하는 프로세스의 연산 시간이 많이 감소되었다.

References

[1] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, “Public key encryption with keyword search,” in *Int. Conf. Theory and Appl. of Cryptographic Techniques,* pp. 506-522, Berlin/Heidelberg, Germany, 2004.

[2] Y. Zhang, J. Katz, and C. Papamanthou, “All your queries are belong to us: The power of file-injection attacks on searchable encryption,” in *Proc. 25th fUSENIXg Security 16*, pp. 707-720, Austin, TX, USA, Aug. 2016.

[3] R. Bost, “Σ οφος: Forward secure searchable encryption,” in *Proc. 2016 ACM SIGSAC Conf. Comput. and Commun. Secur.*, pp. 1143-1154, Vienna, Austria, Oct. 2016.

[4] X. Zhang, C. Xu, H. Wang, Y. Zhang, and S. Wang, “FS-PEKS: Lattice-based forward secure public-key encryption with keyword search for cloud-assisted industrial Internet of Things,” *IEEE Trans. Depend. Secur. Comput.*, Apr. 2019.

[5] M. Zeng, H. F. Qian, J. Chen, and K. Zhang, “Forward secure public key encryption with keyword search for outsourced cloud storage,” *IEEE Trans. Cloud Comput.*, Sep. 2019.

[6] I. Anati, S. Gueron, S. Johnson, and V. Scarlata, “Innovative technology for CPU based attestation and sealing,” in *Proc. 2nd Int. Wkshp. Hardw. and Architectural Support for Secur. and Privacy*, vol. 13, p. 7, Tel-Aviv, Israel, Jun. 2013.

[7] V. Costan and S. Devadas, *Intel SGX Explained*, IACR Cryptol, EPrint Arch., pp. 1-118, 2016.

[8] M. Hoekstra, R. Lal, P. Pappachan, V. Phegade, and J. Del Cuvillo, *Using innovative instructions to create trustworthy software solutions*, HASP@ ISCA 2013, 11, 2487726-2488370.

[9] Intel, *I. Software Guard Extensions Programming Reference*, Revision 2, 2014. Available online: https://software.intel.com/sites/default/files/managed/48/88/329298-002.pdf (accessed on 15 Sep. 2020).

[10] Intel, *R. Software Guard Extensions (Intel R SGX)*. 2018. Available online: https://www.intel.com/content/www/us/en/architecture-and-technology/software-guard-extensions.html (accessed on 15 Sep. 2020).

[11] G. Amjad, S. Kamara, and T. Moataz, “Forward and backward private searchable encryption with SGX,” in *Proc. 12th Eur. Wkshps. Syst. Secur.*, pp. 1-6, Dresden, Germany, Mar. 2019.

[12] H. Kim, C. Hahn, and J. Hur, “Forward secure public key encryption with keyword search for cloud-assisted IoT,” in *Proc. 2020 IEEE Int. Conf. Cloud Comput.*, Beijing, China, Oct. 2020.

[13] B. Fuhry, R. Bahmani, F. Brasser, F. Hahn, F. Kerschbaum, and A. R. Sadeghi, "HardIDX: Practical and secure index with SGX," in *IFIP Annu. Conf. Data and Appl. Secur. and Privacy*, pp. 386-408, Springer: Berlin/ Heidelberg, Germany, 2017.

[14] S. Johnson, V. Scarlata, C. Rozas, E. Brickell, and F. Mckeen, *Intel® software guard extensions: Epid provisioning and attestation services*, White Paper, 119, 1, 2016.

[15] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi, "Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions," in *Annu. Int. Cryptology Conf.*, pp. 205-222, Springer: Berlin/Heidelberg, Germany, 2005.

[16] E. Stefanov, C. Papamanthou, and E. Shi, *Practical Dynamic Searchable Encryption with Small Leakage*, pp. 72-75, 71, NDSS 2014.

[17] P. Guide, *Intel® 64 and ia-32 Architectures Software Developer's Manual*, vol. 3B: System Programming Guide Part. 2011, vol. 2, p. 11, Available online: file:///C:/Users/MDPI/ AppData/Local/Temp/253669-sdm-vol-3b.pdf (accessed on 15 Sep. 2020).

[18] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. S&P 2000*, pp. 44-55, Berkeley, CA, USA, May 2000, IEEE: Piscataway, NJ, USA, 2000.

[19] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," *J. Comput. Secur.*, vol. 19, pp. 895-934, 2011.

[20] M. S. Islam, M. Kuzu, and M. Kantarcioglu, "Access pattern disclosure on searchable encryption: Ramification, attack and mitigation," *Ndss Symp. 2012*, Feb. 2012.

[21] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M. C. Roşu, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for boolean queries," in *Annu. Cryptology Conf.*, pp. 353-373, Springer: Berlin/ Heidelberg, Germany, 2013.

[22] P. Xu, H. Jin, Q. Wu, and W. Wang, "Public-key encryption with fuzzy keyword search: A provably secure scheme under keyword guessing attack," *IEEE Trans. Comput.*, vol. 62, no. 11, pp. 2266-2277, Sep. 2012.

[23] C. Bösch, P. Hartel, W. Jonker, and A. Peter, "A survey of provably secure searchable encryption," *ACM Comput. Surv. (CSUR)*, pp. 1-51, Aug. 2014.

[24] B. Fisch, D. Vinayagamurthy, D. Boneh, and S. Gorbunov, "Iron: Functional encryption using Intel SGX," in *Proc. 2017 ACM SIGSAC Conf. Comput. and Commun. Secur.*, pp. 765-782, Dallas, TX, USA, Oct.-Nov. 2017.

[25] S. Sasy, S. Gorbunov, and C. W. Fletcher, "ZeroTrace: Oblivious memory primitives from Intel SGX," *IACR Cryptol. EPrint Arch. 2018*, 549, Jan. 2018.

[26] P. ishra, R. Poddar, J. Chen, A. Chiesa, R. A. Popa, "Oblix: An efficient oblivious search index," in *Proc. 2018 IEEE Symp. Secur. and Privacy (SP)*, pp. 279-296, San Francisco, CA, USA, May 2018; IEEE: Piscataway, NJ, USA, 2018.

[27] W. Zhang, et al., "Privacy preserving ranked multi-keyword search for multiple data owners in cloud computing," *IEEE Trans. Comput.*, pp. 1566-1577, vol. 65, no. 5, 2015.

[28] H. D. Yoon and J. B. Hur, "A survey of searchable encryption using intel SGX," in *Proc. CISC-S'20*, pp. 346-347, Online, Korea, Jul. 2020.

[29] H. Yoon, S. Moon, Y. Kim, C. Hahn, W. Lee, and J. Hur, "SPEKS: Forward private SGX-based public key encryption with keyword search," *Appl. Sci.*, vol. 10, no. 21, 7842, Nov. 2020.

윤 현 도 (Hyundo Yoon)

2019년 2월 : 고려대학교 컴퓨터학과 졸업

2019년 3월~현재 : 고려대학교 컴퓨터학과 석박사통합과정

<관심분야> 암호프로토콜, 컴퓨터보안, 컴퓨터네트워크

허 준 범 (Junbeom Hur)

2001년 2월 : 고려대학교 컴퓨터학과 졸업

2005년 8월 : 한국과학기술원 전산학 석사

2009년 8월 : 한국과학기술원 전산학 박사

2009년 9월~2011년 8월 : University of Illinois at Urbana-Champaign 박사후 연구원.

2011년 9월~2015년 2월 : 중앙대학교 컴퓨터공학부 조교수

2015년 3월~현재 : 고려대학교 컴퓨터학과 부교수

<관심분야> 클라우드 보안, 빅데이터 보안, 네트워크 보안, 응용 암호학