독립 인증 서버를 이용한 Eddystone-EID 검증 시스템 구현

강진수, 서정훈, 송호종, 원유재*
충남대학교 컴퓨터공학과
e-mail : kangjs1798@naver.com, sjh7497@naver.com, shj931014@naver.com
yjwon@cnu.ac.kr

Implementation of Eddystone-EID verification system using independent authentication server

Jin Su Kang, Jeong Hun Seo, Ho Jong Song, Yoo Jae Won
Department. Computer Science, Chung-nam National University

요 약

최근 비콘을 이용하여 근거리에 있는 사용자를 찾아 다양한 혜택을 지원해주는 서비스가 제공되고 있다. 2016년에 Google은 보안성 강화를 위해 비콘의 고유 ID를 사용하는 것이 아닌, 주기적으로 변경되는 임시 ID를 사용하여 이를 암호화하고 Broadcasting 하는 Eddystone-EID(Ephemeral Identity)를 공개하였다. EID를 사용하기 위해서는 Google에 비콘의 정보를 등록하는 과정과 EID를 수신 후 Google을 통해 검증하는 과정이 필요하다. 이러한 이유로 외부 업체들도 EID를 사용하기 위해서는 Google에 비콘을 등록해야 하는 문제점이 발생한다. 이러한 문제점을 해결하기 위해 본 논문에서는 Eddystone-EID를 Google을 이용하지 않고 자체적으로 검증하는 방법을 제시한다.

1. 서론

최근 블루투스 기반 근거리 무선 통신 장치인 비콘을 이용하여 반경 50~70m 범위 내에 있는 사용자의 위치를 찾아 메시지 전송, 모바일 결제 등을 가능하게 해주는 서비스가 다양한 서비스업에서 제공되고 있다. 비콘을 이용한 서비스 시장은 2015년 400만 달러에서 2018년에는 10배 이상 성장한 4500만 달러에 이를 것으로 전망되고 있으며[1] 세계 비콘 시장은 연평균 17%의 성장을 할 것으로 예측되고 있다.[2]

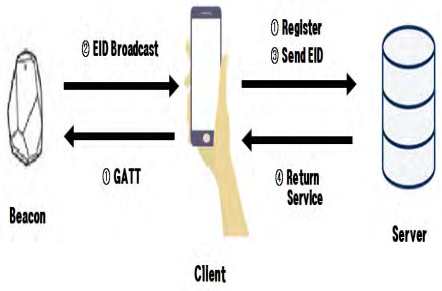
비콘의 플랫폼은 애플에서 2013년에 발표한 iBeacon과 2015년에 Google에서 발표한 Eddystone이 대표적이다. 2016년 Google은 Eddystone-EID(Ephemeral Identity, 이하 EID)라는 새로운 비콘 프레임을 발표하였다. EID 프레임 패킷은 비콘의 고유 ID가 아닌 주기적으로 변경되는 임시 ID를 사용하여 암호화 한 뒤 Broadcasting함으로써, Spoofing, 악의적 자산 추적, 재전송 공격 등의 해킹 공격을 효과적으로 방지할 수 있다.[3]

현재 EID를 이용하는 비콘 서비스의 통신과정은 그림 1과 같다. Client가 비콘이 Broadcasting하는 EID를 수신하여 Google 서버에 전송해주면 Google 서버는 비콘을 식별하여 고유 아이디를 다시 Client에게 전송해준다. 이 결과를 다시 서비스 제공자의 서버로 전송하면 비콘에 해당하는 서비스를 Client에게 제공해주는 방식이다.

EID로 비콘을 식별하기 위해서는 EID를 수신하고 검증하는 과정이 필요한데 검증을 위해서는 사전에 EID 검증에 필요한 비콘의 정보와 데이터들을 검증 서버에 등록하는 과정이 선행되어야 한다. 그림 1과 같이 현재 EID를 검증하는 과정은 Google 서버를 이용하는 형태이기 때문에 EID를 사용하려면 비콘 서비스 제공자가 사전에 소유하고 있는 비콘들의 모든 정보를 Google에 제공해야한다. 이는 서비스 제공자가 소유한 비콘 들이 자체적인 통신망을 구성하지 못하고 Google에 종속될 수밖에 없는 문제를 발생시킬 수 있으며 소유하고 있는 비콘의 정보를 모두 제공하는 것은 사업적으로 민감한 문제가 될 수 있다. 이러한 문제를 해결하기 위해서는 Google 서버가 수행하고 있는 EID를 검증하여 비콘을 식별하는 기능을 서비스 제공자의 서버에서 수행할 수 있게 만들어 그림2와 같이 Google을 거치지 않고 EID를 사용할 수 있게 만들어야 된다. 따라서 본 논문에서는 EID를 생성하는 과정과 검증하는 방법을 연구하여 자체적으로 EID를 검증하고 서비스를 제공할 수 있는 시스템을 구축하고자 한다.

연구를 진행하는데 있어 주변 비콘의 신호를 스캔하고 비콘과 통신하여 데이터를 주고받는 부분은 Google에서 공개한 오픈소스를 사용하였고, EID 등록 및 검증에 관련된 부분은 서버를 구축하여 구현하였다.

(그림 1) 기존의 비콘 서비스 통신과정



(그림 2) 독립적인 비콘 서비스 통신 과정

2. Eddystone-EID 생성

비콘이 EID를 Broadcasting하기 위해서 비콘은 먼저 Client로부터 받은 Resolver의 데이터를 이용하여 EID를 계산하는 과정을 거친다. 이 때 비콘이 EID를 계산하기 위해서는 세 가지 데이터가 필요하다.

첫째로 Identity key이다. 이 값의 크기는 16Byte이며 비콘과 Resolver간의 공유된 key 값으로 Resolver가 바뀐다면 이 값도 달라진다.

두 번째는 Rotation Period Exponent Scalar값으로 서비스 제공자가 0에서 15사이의 값 중 하나를 선택하여 비콘에 전송한다. 이 값을 K라 할 때, 비콘은 2^K 초의 시간이 지났을 때 새로운 EID가 발생한다. 즉 EID가 변하는 주기를 결정한다는 뜻이다.

마지막으로 Timecounter가 필요하다. 이 값은 비콘 자체적으로 생성하는 값이며, Timestamp와 동일하다.

(그림 3) EID 생성과정

세 가지 데이터를 가지고 EID를 계산하는 과정은 그림 3과 같다. EID 계산의 가장 첫 번째 단계는 비콘과 Resolver간의 키 교환 과정이 필요하다. 이 때 비콘과 Resolver의 키 교환은 타원 곡선 디피-헬만(Elliptic Curve Diffie - Hellman, 이하 ECDH) 알고리즘을 기반으로 한다.[4] 즉 비콘과 Resolver는 각각 ECDH 공개키와 개인키를 가지고 있고, 비콘은 Resolver의 ECDH 공개키를 전송받아, 자신의 ECDH 개인키와 연산하여 Shared Secret을 생성한다. ECDH 알고리즘 특성상, 비콘의 공개키와 Resolver의 개인키를 연산하여도 같은 값의 Shared Secret을 얻을 수 있다.

Shared Secret을 생성했다면 이 값을 HKDF-256 알고리즘으로 암호화 하는 과정을 거치는데, 이 때 Salt의 값은 Resolver의 공개키와 비콘의 공개키를 더하여 만든다. HKDF-256 암호화의 결과로 얻은 32Byte(256bit)의 데이터중 상위 16Byte(128bit)을 취하여 Identity key를 만든다.

Identity key를 통해 EID를 생성하는 과정은 두 단계로 이루어져 있다. 첫 째 Identity key를 통해 temporary key를 생성하는 단계, 둘 째 temporary key를 통해 EID를 생성하는 단계이다. 표 1, 표 2 는 각각의 단계에서 필요한 데이터의 구조를 보여준다.

Byte offset	Field	Description
0 - 10	Padding	0x00
11	Salt	0xFF
12 - 13	Padding	0x00
14 - 15	TC[0], TC[1]	Timecounter의 상위 16bit

<표 1> temporary key의 생성을 위한 데이터의 구조

Byte offset	Field	Description
0 - 10	Padding	0x00
11	K	rotation period exponent
12 - 15	TC[0]…TC[3]	Timecounter 32bit

<표 2> EID의 생성을 위한 데이터의 구조

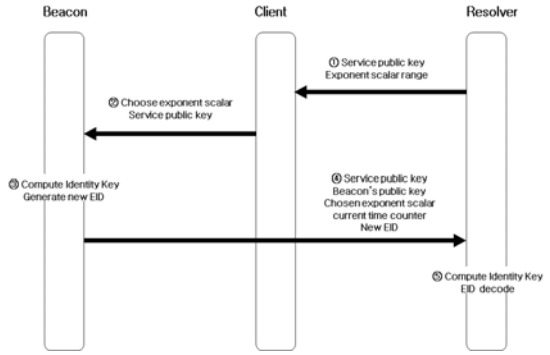
먼저 Temporary key는 표 1 과 같이 구성된 16Byte 크기의 데이터를 AES-128 알고리즘으로 암호화 하여 만든다. 이 때 AES-128 알고리즘의 암호화 키는 Identity key가 된다.

마지막으로 EID는 표 2와 같이 구성된 16Byte 크기의 데이터를 마찬가지로 AES-128 알고리즘으로 암호화 한다. 이 때 사용되는 암호화 키는 앞서 만들었던 Temporary key가 된다. 결과적으로 16Byte 크기의 암호화된 데이터가 나오는데 이 값의 상위 8Byte를 취하면 EID 계산이 완료된다.[5]

3. Eddystone-EID 등록 및 검증

3.1 등록

지금까지 비콘의 입장에서 EID를 계산하는 과정을 보았다. Resolver의 입장에서 EID를 계산 하는 것은, Shared Secret을 만들 때 사용되는 ECDH 키가 달라지는 것 말고는 차이점이 없다. 하지만 Resolver의 입장에서 비콘의 공개키, 선택된 Rotation Period Exponent Scalar, 비콘의 Timecounter 값 등은 알 수 없기 때문에 EID를 계산할 수 없다. 그래서 서비스 제공자는 비콘이 EID를 Broadcasting 하는데 사용된 정보들을 Resolver에게도 알려야 할 필요가 있다. 이 과정이 그림 4에 나타나 있다. 이 것을 등록이라 하며 이 과정에서 추가적으로 비콘의 위도, 경도, floor, description, 제공할 서비스 등을 Resolver에게 알린다.

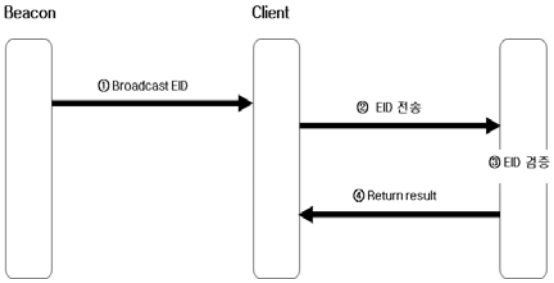


(그림 4) EID 생성 및 등록과정

3.2 검증

EID의 최대 장점은 비콘의 고유 ID를 고정적으로 사용하는 것이 아닌, 임시 ID를 주기적으로 바꾸며 암호화 하여 Broadcasting하는데 있다. 하지만 이런 보안성 향상의 trade-off로 수신 받은 EID 신호가 어떤 비콘으로부터 왔는지 단번에 알 수 없어, 사용자의 편의성이 감소하였다. 즉 사용자 앞에 여러 개의 비콘이 있을 때, 사용자의 스마트기기에 EID신호를 보내는 비콘이 어떤 비콘인지 알 수 없고, 또한 EID만으로는 서비스도 제공 받을 수 없다. 따라서 어떤 비콘으로부터 EID 신호를 받았고, 그에 따른 서비스는 무엇인지 알 기 위해선 검증의 과정이 필요하다.

그림 5에는 EID 검증의 과정이 나타나 있다. EID 검증이란, 암호화된 EID를 다시 복호화 하는 것이 아닌, 등록 과정에 Resolver에게 저장되어있는 비콘의 정보들과 검증 시의 Timestamp를 이용하여 EID를 만들어 수신 받은 EID와 같은 값을 만들어 내는 비콘이 있는지 비교하는 과정이다.



(그림 5) EID 검증과정

4. 연구결과

비콘이 EID를 Broadcasting하기 위해서는 Resolver의 ECDH 키 정보를 전송해야 하는데 이 때 두 가지 전송 방법이 있다.

하나는 34Byte 크기의 데이터를 전송하는 것이고, 하나는 18Byte 크기의 데이터를 전송하는 것이다.

34Byte 데이터를 전송하는 경우 이 데이터는 1Byte의 프레임 타입과 32Byte의 Resolver 서버의 ECDH 공개키, 그리고 1Byte의 rotation period exponent로 구성한다. 이 경우 Resolver의 공개키를 전송하여 비콘이 직접 Shared Secret을 계산한다.

18Byte 데이터를 전송하는 경우 이 데이터는 1Byte의 프레임 타입과 16Byte의 Identity key, 그리고 1Byte의 rotation period exponent로 구성한다. 이 경우 Resolver가 Identity key를 직접 계산하여 전송하는 것으로, Identity key가 외부에 공개될 가능성이 있으므로 34Byte 데이터 전송의 경우보다 안전성이 떨어지는 문제가 있다.[6]

본 연구에서는 그 중 34Byte의 데이터를 전송하는 방식을 선택하였다. 명시된 구조대로 데이터를 만들어 비콘에 전송하였더니 비콘 내부에서 그림 3의 과정을 거쳐 EID를 Broadcasting 하는 것을 확인 할 수 있었다.

비콘이 EID를 Broadcasting 하는 것을 확인함과 동시에 Generate 시키는데 사용했던 데이터들과 비콘의 데이터, 그리고 방금 생성한 EID를 Resolver 서버에 전송하도록 하였다. Resolver 서버에서는 전송 받은 데이터들과 자신의 데이터를 이용하여 EID를 계산하여, 전송받은 EID와 비교 하는 과정을 거친다. EID 계산 결과가 같다면 데이터베이스에 이 정보들을 모두 저장하고, 계산한 EID가 서로 다르거나 데이터베이스에 동일한 EID가 있다면 등록을 거절하도록 하였다.

등록이 완료되면 비콘 스캔 리스트에서 수신 받은 EID에 대해 검증 요청을 할 수 있다. 검증 요청을 하게 되면 해당 EID를 Resolver 서버에 전송한다. Resolver 서버에서는 데이터베이스의 모든 레코드들에 대해 2장에서 설명한 과정과 같이 EID를 계산하여 보고 비콘으로부터 받은 EID와 같은지 비교하는 과정을 거쳐 해당 비콘이 어떤 비콘인지 식별 결과를 다시 어플리케이션으로 전송한다.

(그림 6) EID 검증 요청 결과 (좌 - 성공, 우- 실패)

그림 6은 EID 검증 요청 결과를 보여준다. 이렇게 비콘에 EID를 Generate 시키고, 해당 비콘을 우리의 Resolver 서버에 등록하고, 검증까지 Resolver 서버에서 가능 한 것을 확인 할 수 있었다. 하지만 본 연구는 Resolver 서버로부터 EID를 검증하고, 해당 비콘의 식별 결과만을 받는데 까지만 진행된 상태이다. 이제 더 나아가 비콘 식별 결과가 아닌 자신이 원하는 서비스를 제공 할 수 있도록 한다면 비로소 Google이 만든 Eddystone을 사용하면서도, Google에 종속되지 않고 자체적으로 비콘 통신망을 구성하여 서비스를 제공 할 수 있을 것이라 기대한다.

〔Acknowledgement〕

본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 SW중심대학지원사업의 연구결과로 수행 되었음 (2015-0-00930)

참 고 문 헌

[1] Tony Danova “BEACONS: The Technology That Could Transform How People Use Their Phones In Stores, Events, And Their Homes”, Business insider, 2013

[2] “Global Bluetooth Beacons Market in Retail 2016-2020” ,Technavio. 2015

[3] Avinatan Hassidim, Yossi Matias, Moti Yung, and Alon Ziv. “Ephemeral Identifiers: Mitigating Tracking & Spoofing Threats to BLE Beacons”, Google Inc. April 14, 2016

[4] Daniel J. Bernstein “Curve 25519: new Diffie-Hellman speed recoeds ”, Public Key Cryptography - PKC 2006, 9th International Conference on Theory and Practice of Public-Key Cryptography p207-228

[5] GitHub - google/eddystone: EID Computation https://github.com/google/eddystone/blob/master/eddystone-eid/eid-computation.md

[6] GitHub - google/eddystone: Eddystone configurarion GATT Service https://github.com/google/eddystone/blob/master/configuration-service/README.md