디지털산업정보학회 논문지 제16권 제4호-2020년 12월
http://dx.doi.org/10.17662/ksdim.2020.16.4.043

다양한 장치에서 JWT 토큰을 이용한 FIDO UAF 연계 인증 연구*

김 형 겸**·김 기 천***

A Study on FIDO UAF Federated Authentication Using JWT Token in Various Devices

Kim HyeongGyeom·Kim KiCheon

〈Abstract〉

There are three standards for FIDO1 authentication technology: Universal Second Factor (U2F), Universal Authentication Framework (UAF), and Client to Authenticator Protocols (CTAP). FIDO2 refers to the WebAuthn standard established by W3C for the creation and use of a certificate in a web application that complements the existing CTAP.

In Korea, the FIDO certified market is dominated by UAF, which deals with standards for smartphone (Android, iOS) apps owned by the majority of the people. As the market requires certification through FIDO on PCs, FIDO Alliance and W3C established standards that can be certified on the platform-independent Web and published 『Web Authentication: An API for Accessing Public Key Credentials Level 1』 on March 4, 2019. Most PC do not contain biometrics, so they are not being utilized contrary to expectations.

In this paper, we intend to present a model that allows login in PC environment through biometric recognition of smartphone and FIDO UAF authentication. We propose a model in which a user requests login from a PC and performs FIDO authentication on a smartphone, and authentication is completed on the PC without any other user's additional gesture.

Key Words : FIDO(Fast IDentify Online), UAF(Universal Authentication Framework), Biometric Authentication, JWT(Json Web Token), PC(Personal Computer), WebAuthn(Web Authentication)

Ⅰ. 서론

FIDO1의 UAF 지원범위는 Android, iOS 앱에 국한되었으므로 시장에서는 PC 환경에서도 스마트폰과 같이 생체인식을 통한 인증이 요구되었다. FIDO2에서는 플랫폼(OS)에 독립적인 Web Browser에서 사용할 수 있도록 규격을 제정하고 W3C는 『Web Authentication: An API for accessing Public Key Credentials Level1』[1]을 2019년 3월 4일에 게시했

* 본 논문은 김형겸(2020)의 석사학위논문을 일부 수정한 것임.

** 건국대학교 IT융합정보보호학과 석사과정(주저자)
*** 건국대학교 IT융합정보보호학과 교수(교신저자)

다. 이로써 PC Web 환경에서 WebAuthn API를 통한 인증을 기대했으나 생체인식을 탑재한 PC가 보편화 되지 않았으므로 활성화되지 못하고 있다.

제안 모델에서는 이러한 요구사항에 부합하고자 스마트폰의 생체인식기와 FIDO UAF Protocol[2]을 활용해서 PC 환경에서 인증을 수행하기 위한 일련의 과정들을 기술한다. 이를 구현하기 위해 크게 세 가지 측면을 고려하고 설계되었다. 첫째, FIDO 서버의 인증 결과를 비즈니스 서버에서 어떻게 신뢰 할 것인가? 둘째, FIDO를 적용하기 위해 레거시 시스템의 아키텍처 변경을 최소한으로 할 수 있는 방안은 무엇인가? 셋째, 두개의 요소(PC, 스마트폰)를 이용하게 되므로 사용자 제스처를 최소화하는 방한은 무엇인가?

목표한 시스템은 비즈니스 서버와 FIDO 서버가 서로 다른 네트워크에 위치할 수 있다고 가정하였다. 이는 메시지 전달 과정에서 취약점이 발생할 수 있으므로 적절한 보안요구사항을 충족하지 않으면 공격자에 의해서 원치 않은 결과를 가지게 된다. 취약점을 해소하고 FIDO 서버의 인증결과를 검증하기 위한 방안으로 JWT 토큰[3]을 활용하는 과정을 설명한다.

일반적인 레거시 시스템의 인증방식은 클라이언트에 인증에 필요한 값(ID/Password)을 보내고 서버에서 검증하는 방식을 사용하고 있다. 따라서 JWT 토큰을 사용자 PC에서 비즈니스 서버로 전송하기 위해 Token GW 서버를 활용하는 방법에 대해서 설명한다.

온라인에서 스마트폰을 통한 인증방식에서 사용자의 제스처를 비교하고 사용하면 일반적으로 다음과 같은 순서로 진행된다.

1) PC에서 인증요청
2) 스마트폰에서 인증수행
3) 스마트폰에서 PC에서 수행해야 할 내용 공지
4) PC에서 추가 제스처

사용자는 인증을 수행하기 위해 4단계의 제스처가 필요하다.

본 논문에서는 이러한 불편함을 해소하고자 Token GW 서버를 통해 2단계의 제스처로 완료하도록 설계하였다.

1) PC에서 인증요청
2) 스마트폰에서 인증수행

Ⅱ. 관련연구

2.1 FIDO(Fast IDentity Online)

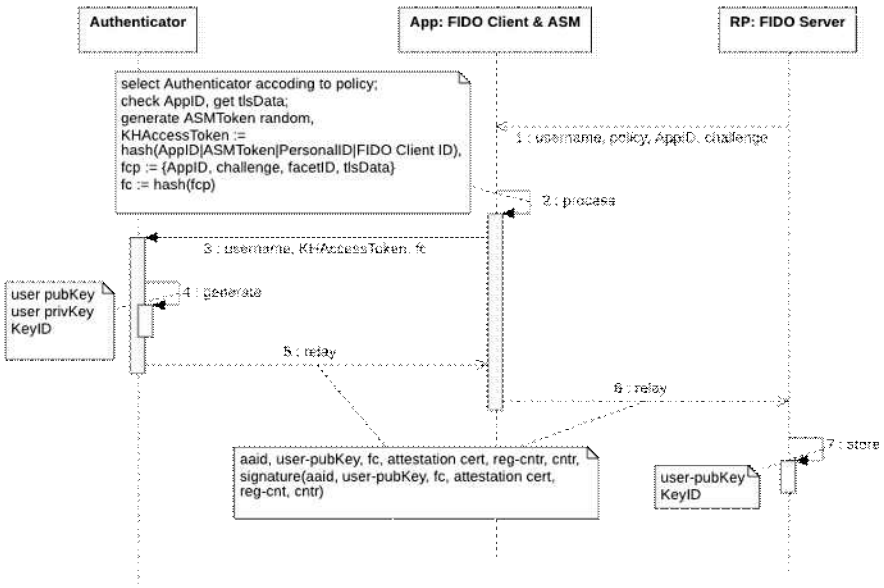
FIDO(Fast IDentity Online) 기술은 FIDO Alliance에서 온라인상의 인증과정에서 비밀번호를 대체 또는 더 강화된 인증을 하기 위해서 제정한 기술 표준으로 FIDO1 Architecture에는 두 가지 주요 프로토콜이 포함되어 있으며 비즈니스 성격에 맞게 선택되어 사용된다.

- **UAF(Universal Authentication Framework) Protocol[4]:** 사용자는 지문인증, 안면인증 등과 같은 로컬 인증을 수행해서 온라인 서비스에 자신의 Authenticator를 등록하고 이후 로그인할 때 로컬 인증을 수행함으로써 password를 대체할 수 있는 프로토콜
- **U2F(Universal 2nd Factor) Protocol[5]:** 온라인 서비스에 로그인할 때 사용자는 id/password로 로그인 후 두 번째 요소의 장치를 제출함으로써 보안을 강화하는 프로토콜

본 논문에서는 UAF 인증에 대해서 설명하며 3가지의 단계를 가진다.

1) Authenticator 등록

<그림 1>은 FIDO Authenticator를 등록하는 과정을 보여주고 있다. 사용자는 등록에 앞서 Relying

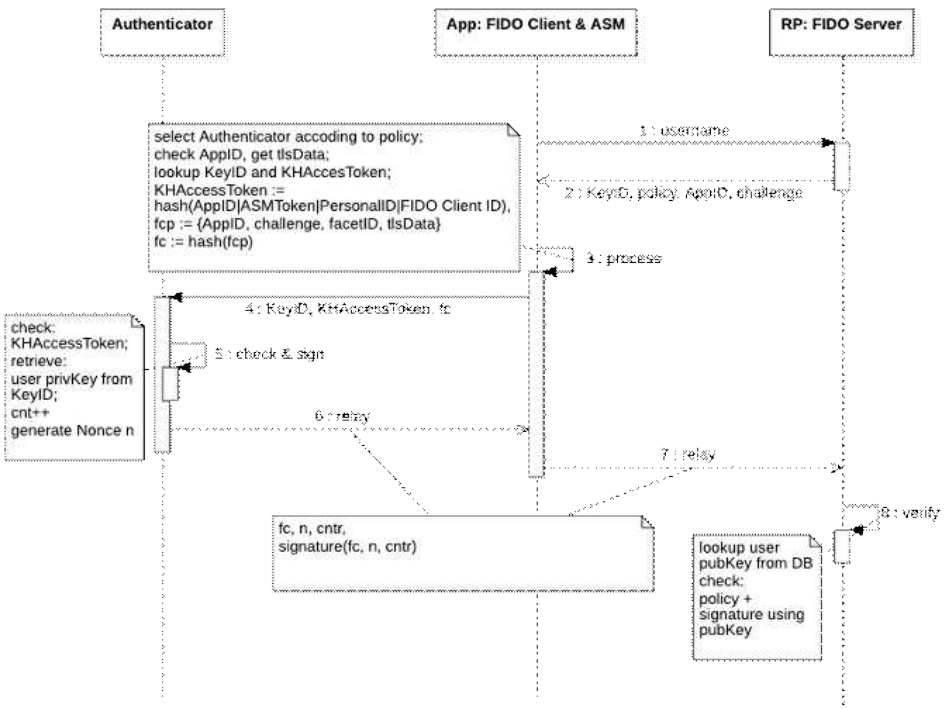


<그림 1> Authenticator Registration 과정

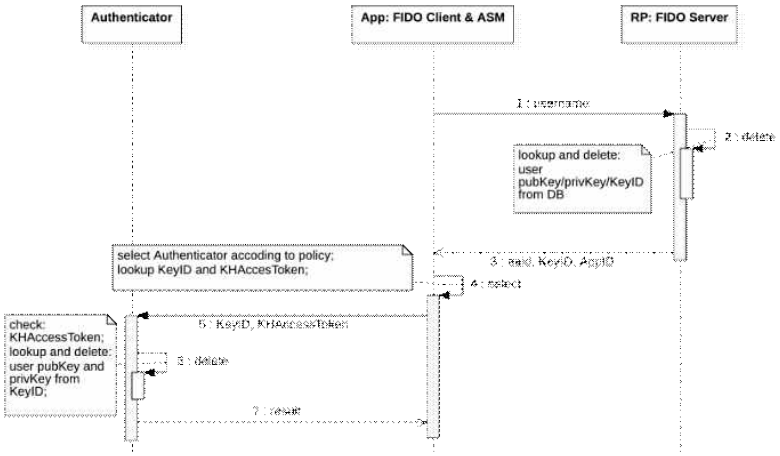
Party 서버로부터 로그인이 수행되어야 하고 그 이후 단계부터 설명한다. FIDO 서버는 username, policy, AppID, challenge를 값으로 Authenticator 등록을 요청한다. FIDO Client는 스마트폰에 있는 Authenticator를 검색하고 policy에 해당하는 Authenticator를 선택한다. 그리고 FIDO 서버에서 신뢰하는 App목록을 AppID를 통해서 체크하고 전송구간의 3자 개입 여부를 탐지하기 위해 TLS 인증서로 tlsData를 생성한다. ASM은 유일한 ID를 갖기 위해 최초 1회 ASMToken를 생성하고 KHAccessToken := hash(AppId|ASMToken|PersonalId|FIDO Client ID)을 만든다 - 이는 인증/해지 과정의 Relying Party, FIDO Client, ASM가 등록할 때와 같은지를 확인하기위한 용도. 이후 AppId, challenge, facetID, tlsData를 해시한 값 fc와 username, KHAccessToken을 Authenticator에게 전달한다. Authenticator는 사용자에게 로컬 인증(생체 인증)을 요청해서 인가된 사용자임을 확인하고 Relying Party에 연결된, username에 해당하는 비대칭키 쌍을 생성하고 이에 대한 식별자 KeyID를 만든다. reg-cntr(등록 카운트)를 1증가 후 aaid, fc, reg-cntr, cntr(인증 카운트), user pubKey를 attestation 개인키로 서명하고 FIDO 서버에서 서명검증 할 수 있도록 aaid, user pubKey, fc, KeyID, attestation cert, reg-cntr, cntr을 ASM → FIDO Client → App → Replying Party → FIDO 서버로 전달한다. FIDO 서버는 fc를 생성할 정보를 가지고 있으므로 FIDO Client에서 생성한 값과 비교해서 3자의 개입여부를 확인하고 Authenticator에서 서명한 값을 attestation 인증서를 통해서 검증한다. 그리고 aaid에 해당하는 Metadata에서 인증서 체인을 획득하고 attestation 인증서 검증을 수행한다. 위 과정이 모두 유효하면 user pubKey와 이에 대한 식별자 KeyID을 저장한다.

2) 인증

<그림 2>는 FIDO 서버에 등록된 사용자를 인증하는 과정을 보여주고 있다. App은 Relying Party를 통해 username에 해당하는 정책을 질의한다. FIDO 서버는 KeyID, policy, AppID, challenge를 값으로 인증을 요청한다. FIDO Client는 스마트폰에 있는 Authenticator를 검색하고 policy, KeyID에 해당하는 Authenticator를 선택한다. 그리고 App이 FIDO 서버에서 신뢰하는지를 AppID를 통해서 체크하고 전송구간의 3자 개입 여부를 탐지하기 위해 TLS 인증서로 tlsData를 생성한다. ASM은 KHAccessToken := hash(AppId|ASMToken|PersonalId|FIDO Client ID)을 만든다 - 이는 등록 과정의 Relying Party, FIDO Client, ASM가 인증할 때와 변경이 없음을 확인하기위한 용도. 이후 AppId, challenge, facetID, tlsData를 해시한 값 fc와 KeyID, KHAccessToken을 Authenticator에게 전달한다. Authenticator는 사용자에게 로컬 인증(생체 인증)을 요청해서 인가된 사용자임을 확인하고 등록 및 인증의 KHAccessToken의 일치여부를 비교한다. cntr(인증 카운트)를 1증가 후 fc, nonce, cntr(인증 카운트) username 계정의 개인키로 서명하고 FIDO 서버에서 서명검증 할 수 있도록 fc, nonce, cntr을 ASM → FIDO Client → App → Replying Party → FIDO 서버로 전달한다. FIDO 서버는 스스로 fc를 생성해서 FIDO Client에서 생성한 값과 비교해서 3자의 개입여부를 확인하고 서명 값을 username 계정의 공개키를 통해서 검증한다.



<그림 2> Authentication 과정



<그림 3> Authenticator Deregistration 과정

3) Authenticator 해지

위의 <그림 3>은 Authenticator를 해지하는 과정을 보여주고 있다 App은 Relying Party를 통해 username에 해당하는 인증장치 해제를 요청한다. FIDO 서버는 username에 해당하는 pubKey, privKey, KeyID를 DB에서 삭제 후 aaid, KeyID, AppID를 값으로 해제를 요청한다. FIDO Client는 스마트폰에 있는 Authenticator를 검색하고 aaid, KeyID에 해당하는 Authenticator를 선택한다. 그리고 App이 FIDO 서버에서 신뢰하는지를 AppID를 통해서 체크한다. ASM은 KHAccessToken := hash(AppId|ASMToken|PersonalId|FIDO Client ID)을 만든다 - 이는 등록 과정의 Relying Party, FIDO Client, ASM가 해지할 때와 변경이 없음을 확인하기위한 용도. 이후 KeyID, KHAccessToken을 Authenticator에게 전달한다. Authenticator는 등록 및 해지의 KHAccessToken의 일치여부를 비교 하고 KeyID에 해당하는 user pubKey, privKey를 삭제한다.

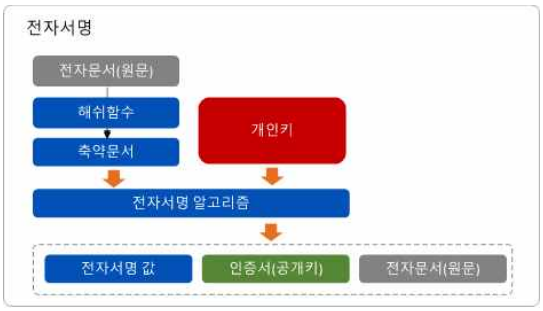
2.2 PKI(Public Key Infrastructure)

PKI(Public Key Infrastructure)는 비대칭키 알고리즘(Asymmetric Key Algorithm) 특성을 이용하여 공개키(Public Key)와 개인키(Private Key)를 기반으로 메시지를 서명·검증, 암·복호화 하는 특징을 가지고 있다. 대표적인 비대칭키 알고리즘에는 Ron Rivest, Adi Shamir, Leonard Adleman 발명자의 이름 앞 글자를 딴 RSA가 있으며, 두개의 소수의 곱을 구하는 것은 쉽지만, 구해진 값에서 원래의 두개의 소수를 구하는 것은 어렵다는 것을 근간으로 한다. 두개의 소수 중 하나를 공개키, 나머지 하나를 개인키로 하여 공개키로 암호화한 값을 개인키로 복호화하고, 개인키로 서명한 값을 공개키로 검증하는 방식을 취한다.

2.2.1 전자서명

전자서명은 전자문서 원문을 해시 함수를 이용해서 축약을 한다. 개인키로 축약문서를 전자서명 후

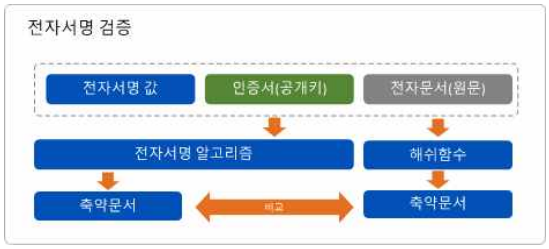
인증서, 전자문서 원문과 함께 전달한다.



<그림 4> 전자서명 과정

2.2.2 전자서명 검증

전자서명 검증 과정은 전자서명 값을 공개키로 복호화해서 축약문서를 획득하고 전자서명 원문을 해시 함수를 이용해서 축약한다. 이 두개의 축약문서를 비교해서 같은지 확인한다.



<그림 5> 전자서명 검증 과정

2.3 JWT(JSON Web Token)

JWT는 두 개체 간에 전송할 claim을 JSON을 이용하여 안전성 있게 전달하는 수단이다. JWT의 claim은 JWS(JSON Web Signature) 구조 또는 JWE(JSON Web Encryption) 구조의 JSON 개체로 인코딩되어 claim을 디지털 서명하거나 무결성을 보호할 수 있다. JWT는 마침표(.)를 구분자로 Header, Payload, Signature 구조로 되어있다(ex. hhhhh.ppppp.sssssss).

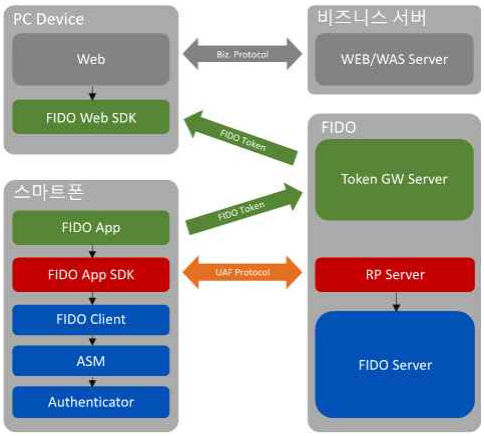
<표 1> JWT 구조

구조	설명
Header	토큰의 type과 알고리즘 정보를 가진다.
Payload	토큰에 포함할 Claim을 가지고 Claim은 name, value쌍으로 이루어져 있다. Claim은 Registered, Public, Private 3가지로 구성되어 있다.
Signature	Header와 Payload가 3자에 의해 변조되지 않았음을 증명하는 값을 가진다. Header의 알고리즘에 따라서 Signature의 형태가 지정된다.

- **Header:** JOSE(Javascript Object Signing and Encryption) Header로 토큰의 Type과 알고리즘을 지정한다. typ은 "JWT"이고, alg는 "HS256", "RS256" 등이 있다(ex. {"typ":"JWT","alg":"RS256"})
- **Payload:** 두 개체간에 전송할 정보를 가진다.
  - Registered Claim Names: IANA "JSON Web Token Claims"에 이미 등록되어 있는 Claim으로 토큰의 iss(발급자), nbf(유효기간 시작), exp(유효기간 끝), aud(대상자) 등이 있다.
  - Public Claim Names: Claim Name은 JWT를 사용자가 임의로 정의할 수 있다. 그러나 충돌을 방지하기 위해서는 IANA "JSON Web Token Claims"에 등록되거나 충돌방지를 위한 공용 이름이어야 한다. 일반적으로 충돌을 방지하기 위해 URI형식의 이름을 사용한다.
  - Private Claim Names: Registered/Public Claim Names도 아닌 두 개체간에 필요에 의해서 정의된 Claim Name으로 제안 모델에서는 nonce를 정의해서 Token 재사용 방지 용도로 사용한다.
- **Signature:** JOSE Header에 정의된 형태에 따라서 JWS(JSON Web Signature) or JWE(JSON Web Encryption) 형태의 서명 값을 사용한다.

Ⅲ. 제안모델 구현

3.1 제안모델 아키텍처



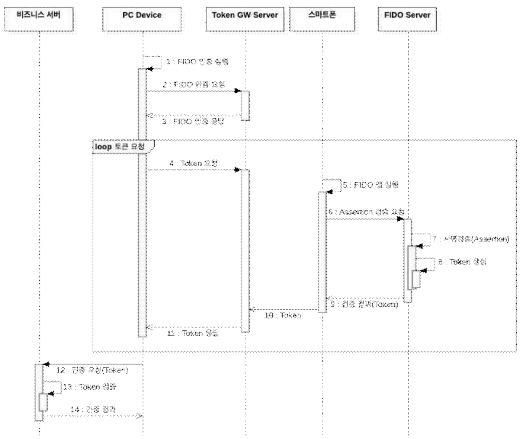
<그림 6> 시스템 아키텍처

<그림 6>은 FIDO UAF 아키텍처에서 제안모델을 구연하기 위해 필요한 추가 구성은 녹색상자로 구분했고 각 요소에 대한 기능은 다음과 같다.

- **FIDO Web SDK**: PC Web Browser에서 시작한 인증 요청을 Token GW 서버에 접속해서 FIDO Token을 수신하고 Web Browser에게 전달해주는 모듈
- **Token GW Server**: Web의 인증 요청과 FIDO App에서 수신한 FIDO Token을 중계하는 서버
- **FIDO App**: FIDO App SDK로부터 FIDO Token을 수신하고 Token GW Server에 전달하기 위한 App

Token이 전달되는 일련의 과정을 표현하면 다음과 같다.

1) 사용자가 Web Brower에서 FIDO 인증을 시작하면

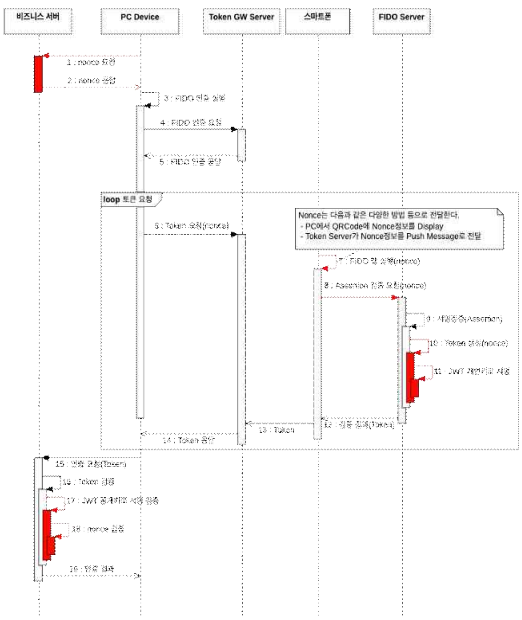


<그림 7> Token 전달 흐름도

2) FIDO Web SDK는 Token GW 서버에 FIDO 인증을 요청하고
3) 응답을 수신한다.
4) FIDO Web SDK는 일정 간격으로 Token GW 서버에게 주기적으로 Token을 요청한다.
5) 사용자는 FIDO App을 실행하고
6) 인증을 수행한 후 Assertion 생성 및 검증을 요청한다.
7) FIDO 서버는 Assertion을 검증하고
8) 정상적이면 Token을 생성
9) 검증 결과를 FIDO App에 반환한다.
10) FIDO App은 Token을 Token GW 서버에 전송하고 Token이 수신될 경우
11) Token을 FIDO Web SDK에 전달한다.
12) Token을 전달받은 Web Browser는 비즈니스 서버에 Token을 통한 인증을 요청한다.
13) 비즈니스 서버는 Token을 검증하고
14) 인증 결과를 응답한다.

3.2 보안성

제안 모델은 Token을 통해서 FIDO 인증 결과를 신뢰하고 있고 비즈니스 서버와 Token을 발행한 FIDO 서버의 위치가 물리적으로 다른 네트워크라는 가정으로 설계되었다. 적절한 보안 요구사항이 충족되지 않을 경우 공격자에 의해서 원치 않은 결과를 가지게 되므로 Token은 위·변조 및 재사용에 대한 신뢰를 확보해야 한다.



<그림 8> Token의 신뢰성 확보를 위한 흐름도

<그림 8>은 Token의 신뢰성을 확보하기 위한 내용을 표현한다. 앞선 시나리오(<그림 7>)와 다르게 FIDO 인증을 실행하기 전에 1), 2)번에서 비즈니스 서버로부터 nonce를 받아온다. 그리고 nonce는 FIDO 서버로 전달이 되고 JWT 형태의 Token을 발급할 때 포함되어진다. 이후 비즈니스 서버는 두 가지 형태로 JWT를 검증한다.

3.2.1 JWT를 이용한 검증

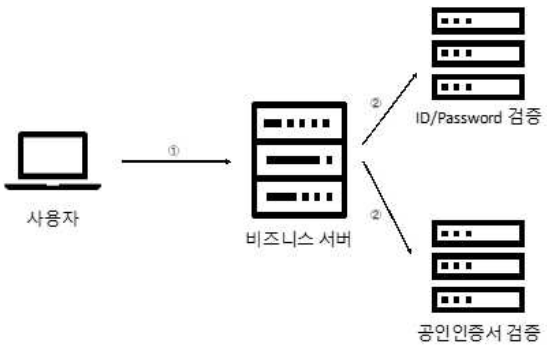
JWT는 알고리즘을 RS256을 선택하여 PKI 기반으로 관리되고 있고 FIDO 서버는 Token을 검증하기 위한 공개키를 사전에 비즈니스 서버에 배포한다. Token은 10), 11) 과정에서 생성할 때 RSA 개인키에 의해 전자서명 되고, 17) 이를 공개키로 전자서명 검증 하므로 중간에 공격자가 다른 개인키로 서명한 Token으로 변조했다면 비즈니스 서버는 전자서명 검증이 불가능하다. 또한 Token의 발급시간과 timeout이 현재 시간의 범위 내에 있는지 확인함으로써 만료된 Token을 확인할 수 있다. 그러나 서명 검증과 유효기간 판단으로 Token의 신뢰성을 담보할 수 없어서 추가적인 보안이 더 필요하다.

3.2.2 Replay Attack 방지모델

비즈니스 서버는 Token이 재사용 되었는지 확인해야 한다. 공격자가 정상적인 Token을 사전에 수집하고 인증이 필요한 시점에 사용하게 되면 RS256 PKI 기반의 서명 검증으로는 부족하다. 따라서 해당 세션에서 발행한 Token 여부를 검증하는 방안이 필요한다. JWT는 Payload 영역에 Private Claim Names을 통해서 양단간 서로 약속한 항목을 추가할 수 있다. 비즈니스 서버는 세션이 생성되는 시점 1), 2) 과정에서 nonce을 생성하고 FIDO 서버 Token에 nonce를 추가하여 서명한다. 비즈니스 서버는 18) 세션이 시작했을 때 발행한 nonce가 Token을 통해서 되돌 오는지를 확인한다. 이로써 재사용된 Token 여부를 확인할 수 있으므로 Replay Attack으로부터 방어할 수 있다.

3.3 온라인상의 모바일인증 방법과 비교

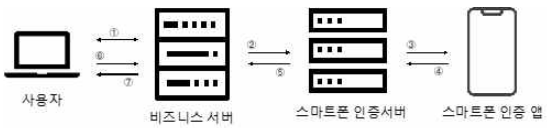
3.3.1 PC Web 로그인 방식



<그림 9> 일반적인 로그인 방식

사용자는 ID/Password 또는 공인인증서로 로그인 방식을 선택하고 비즈니스 서버는 사용자가 선택한 로그인 형태에 따라서 각 인증방식 선택하고 검증한다.

3.3.2 모바일 장치를 연계한 PC Web 로그인 방식

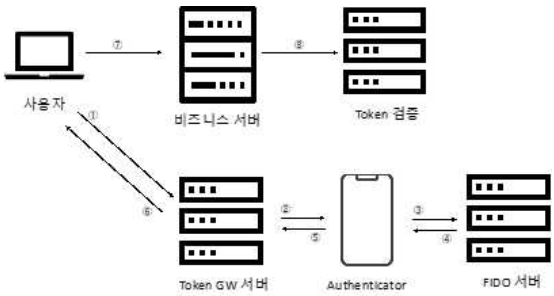


<그림 10> 모바일 장치를 연계한 로그인 방식

1) 사용자는 PC에서 인증을 요청하면,
2) 비즈니스 서버는 스마트폰 인증서버에 인증을 요청한다.
3) 스마트폰 인증서버는 Push를 통해 스마트폰으로 인증을 요청하고
4) 스마트폰은 인증결과를 리턴한다.
5) 스마트폰 인증서버는 인증 값을 비즈니스 서버에 전달하고,
6) 7) 사용자 PC는 인증이 성공했는지 비즈니스 서버에 질의함으로써 인증이 수행된다.

위의 연계 인증을 수행하기 위해서 비즈니스 서버는 사용자 PC와 스마트폰 인증서버의 세션을 연결하고 사용자 PC에서 인증결과를 확인하기까지 세션을 관리해야 한다. 이는 스마트폰 인증을 통해 PC와 연계하기 위해서는 레거시 시스템을 구조적으로 변경해야 하는 부담이 생긴다.

3.3.3 제안모델 로그인 방식



<그림 11> Token GW를 활용한 로그인 방식

제안 모델에서는 Legacy 시스템에서 사용하는 검증 방식에 구조적인 변화를 최소한으로 설계되었다. 사용자가 ID를 입력하고 Token GW 서버에 FIDO 인증을 요청하면 Token을 응답 받고 비즈니스 서버에게 로그인을 요청한다. 비즈니스 서버는 ID/Password, 공인인증서 방식을 사용하듯 요청 형태에 맞춰 Token을 검증할 수 있다.

Ⅳ. 결론

본 논문은 FIDO UAF를 활용해서 모바일 기기에서 수행한 인증을 PC Web 환경에서 로그인하는 방법을 설명하고 있다. 다른 장치에서 인증한 결과를 신뢰하기 위해 필요한 절차와 기존 Legacy 시스템의

구조변경을 최소화하는 방법을 제시하였다. 또한 개인이 소유하고 있는 Authenticator를 통해 Web Application에서 인증이 가능하다는 것을 보여주었다.

Legacy 시스템의 구조변경을 최소화 한다는 것은 손쉽게 적용이 가능하므로 이식성이 높다는 것이다. Legacy 시스템의 기준으로 바라보면 Web Application에 FIDO Token을 요청하는 API와 비즈니스 서버에 Token 검증모듈을 추가하는 것으로 마무리 된다. 비즈니스 서버의 구조적 변경이 필요하지 않다.

타 사례에서는 FIDO 인증 Token을 단순히 랜덤한 식별자로 생성해서 사용한다. Token의 검증방법으로 비즈니스 서버가 FIDO 서버에게 Token 발급 여부를 질의하는 것으로 확인한다. 이는 Token의 재사용, 발급대상을 확인할 수 없을 뿐더러 네트워크 트래픽이 증가한다. 이러한 문제점을 JWT Token을 활용해서 보안성을 확보했다.

불특정 다수가 사용하는 공용 PC에서 공인인증서 파일을 설치해야하는 부담스러운 환경이나 인증을 위해 플러그인 설치를 불편하게 여기는 사용자, 그리고 사이트 별로 id/password 외워야 하는 불편함을 국민 대다수가 소유하고 있는 스마트폰의 생체인식 Authenticator를 통해 인증이 가능하므로 편리하게 사용할 수 있다.

참고문헌

[1] Dirk Balfanz, Alexei Czeskis, Jeff Hodges, J.C. Jones, Michael B. Jones, Akshay Kumar, Angelo Liao, Rolf Lindemann, Emil Lundberg, Web Authentication: An API for accessing Public Key Credentials Level 1 W3C Recommendation, 4 March 2019.

[2] Salah Machani, RSA, the Security Division of EMC, Rob Philpott, RSA, the Security, Division of EMC, Sampath Srinivas, Google, Inc., John Kemp, FIDO Alliance, Jeff Hodges, PayPal, Inc., FIDO UAF Architectural Overview FIDO Alliance Proposed Standard, 02 February 2017.

[3] M. Jones, JSON Web Token (JWT), RFC 7519, May 2015.

[4] Dr. Rolf Lindemann, Nok Nok Labs, Inc., Eric Tiffany, FIDO Alliance, FIDO UAF Protocol Specification FIDO Alliance Proposed Standard, 02 February 2017.

[5] Sampath Srinivas, Google, Inc., Dirk Balfanz, Google, Inc., Eric Tiffany, FIDO Alliance, Universal 2nd Factor (U2F) Overview FIDO Alliance Proposed Standard, 11 April 2017.

■ 저자소개 ■

김 형 겸
Kim Hyeong Gyeom

2009년 9월~현재
(주)드림시큐리티 서비스사업본부
개발1팀 부장

2019년 3월~현재
건국대학교 IT융합정보보호학과
석사과정

관심분야 : 인증, 정보보호
E-mail : kaykim@dreamsecurity.com

김 기 천
Kim Ki Cheon

1992년 7월 Northwestern University 박사
1988년 2월 서울대학교(학사)

현 건국대학교 정보통신대학원 원장
현 건국대학교 소프트웨어 연구센터장
현 건국대학교 일반대학원 IT융합정보보호학과장
현 한국인터넷윤리학회 이사
현 한국정보과학회 조직위원
전 미 Sprint Nextel 사 Core Network Division Director
전 미 FCC NRIC 위원회 Sprint 대표 Delegate

관심분야 : 통신공학, 사이버보안, 미래인터넷, IoT
E-mail : kckim@konkuk.ac.kr

논문접수일 : 2020년 11월 25일
수 정 일 : 2020년 12월 9일
게재확정일 : 2020년 12월 18일