https://doi.org/10.33778/kcsa.2019.19.4.043

FIDO 2.0 기반의 웹 브라우저 안전 저장소를 이용하는 군 정보체계 사용자 인증 시스템 설계 및 구현★

박 재 연*, 이 재 영**, 이 형 석**, 강 지 원**, 권 혁 진**, 신 동 일*, 신 동 규*

요 약

최근 북한 소행으로 의심되는 군 인트라넷 침투 정황이 다수 발견되고 있다. 기존의 군 정보체계에 접근할 수 있는 사용자 인증 데이터 변조가 가능하여 취약점이 발생할 수 있다는 문제점이 존재했다. 본 논문에서는 FIDO(Fast IDentity Online) 표준을 따르는 웹 브라우저에서 인증 취약점을 해결하기 위하여 상호 검증 기법과 API(Application Programming Interface) 위/변조 차단 및 난독화를 적용하였다. 또한 별도의 프로그램 설치를 요구하지 않는 No-Plugin을 구현함으로써 사용자의 편의성도 향상된다. 성능 테스트 결과 RSA 키 생성 속도 기준으로 대부분의 브라우저에서 약 0.1ms의 성능을 보인다. 또한 서버의 전자서명 검증 속도에서도 0.1초 이하의 성능을 보여 상용화에 사용할 수 있음을 검증하였다. 해당 서비스는 안전한 웹 저장소를 구축하여 브라우저 인증이라는 대체방안으로서 군 정보체계 보안 향상에 유용하게 사용될 것으로 예상한다.

Design of Military Information System User Authentication System Using FIDO 2.0-based Web Browser Secure Storage

Park Jaeyeon*, Lee Jaeyoung**, Lee Hyoungseok**, Kang Jiwon**, Kwon Hyukjin
Shin Dongil*, Shin Dongkyoo*

ABSTRACT

Recently, a number of military intranet infiltrations suspected of North Korea have been discovered. There was a problem that a vulnerability could occur due to the modification of user authentication data that can access existing military information systems. In this paper, we applied mutual verification technique and API (Application Programming Interface) forgery / forgery blocking and obfuscation to solve the authentication weakness in web browsers that comply with FIDO (Fast IDentity Online) standard. In addition, user convenience is improved by implementing No-Plugin that does not require separate program installation. Performance tests show that most browsers perform about 0.1ms based on the RSA key generation rate. In addition, it proved that it can be used for commercialization by showing performance of less than 0.1 second even in the digital signature verification speed of the server. The service is expected to be useful for improving military information system security as an alternative to browser authentication by building a web secure storage.

Key words : Authentication, Secure Storage, Military, FIDO 2.0

접수일(2019년 10월 1일), 게재확정일(2019년 10월 28일)

★ 이 논문은 2018년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No. 2018R1D1A1B07047395).

* 박재연 세종대학교 컴퓨터공학과 석사과정
신동일 세종대학교 컴퓨터공학과 교수
신동규 세종대학교 컴퓨터공학과 교수 (교신저자)
** 이재영 ㈜아이리플
이형석 ㈜아이리플
강지원 세종대학교 정보보호학과 교수
권혁진 국방부 정보화기획관

1. 서 론

최근 북한 소행으로 의심되는 군 인트라넷 침투 정황이 다수 발견되고 있다. 2016년 8월에는 군 인터넷 망을 통하여 악성코드를 주입시킨 후 중계서버를 이용하여 인트라넷과의 망 분리를 무력화했던 사건이 있었다[1]. 뿐만 아니라 2017년 8월에 실시했던 한·미 연합훈련인 을지프리덤가디언(UFG) 기간 중에 훈련 정보가 유출될 뻔한 사례가 있었다[2]. 현재 업무 프로세스 상 군 정보체계에 접근할 수 있는 사용자 인증 데이터가 탈취 및 재사용이 가능하다는 문제점이 존재한다[3].

본 논문에서는 인증시스템에 적용할 수 있는 FIDO 기반의 웹 브라우저 안전 저장소를 개발하기 위하여 WebCrypto를 적용하였으며 No-Plugin을 구현하여 편의성을 확보한다. 또한 자체 개발한 상호 인증 무결성 검증과 위/변조 차단 및 난독화를 적용함으로써 데이터의 탈취를 방어한다.

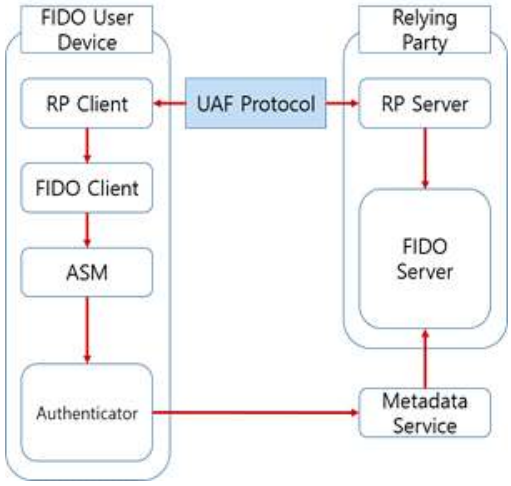
2. 관련 연구

다양한 인증 수단의 등장에 따라 브라우저 측면의 보안성 이슈를 해결하기 위한 연구가 다양하게 진행되고 있다. 해당 이슈를 해결하기 위하여 FIDO 기술에 더불어 클라이언트와 서버 간 데이터를 분할하여 저장하는 형태가 등장하며 클라이언트에 정보를 안전하게 저장 및 보관하는 방법이 연구되고 있다[3]. 이를 위하여 W3C에서 웹 표준화 작업을 진행중이며 키 쌍 생성시 추출 불가능한 개인키를 생성하는 방법을 제공한다. 또한 대부분의 브라우저에서 Webcrypto를 이용하여 전자서명, 암복호화, 해시 등을 지원하여 안전하게 자료를 저장할 수 있는 기술적 기반이 제공되고 있다[4].

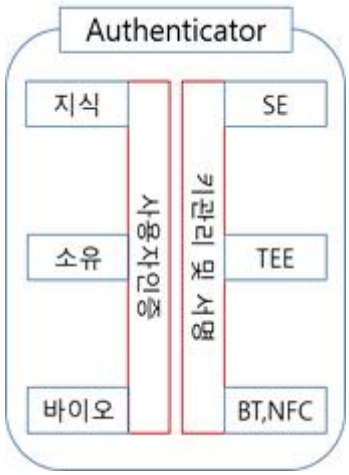
2.1 FIDO 2.0 기술

FIDO client모듈이 웹 브라우저에 내장 될 예정이기에 브라우저 기반의 인증이 점차 확대될 전망이다. 해외에서는 이미 Web Secure Storage에 대한 논의가 활발히 진행되고 있는 상태이다[5]. 이에 따라 해당 인증정보에 대한 새로운 보안 정책 역시 중요성이 높아지고 있다. FIDO란 Fast Identity Online의 줄임말로서 비밀번호 방식뿐만 아니라 지문, 홍재, 얼굴 인식 등의 생체 정보를 활용하여 인증시스템으로 사용하는 기술표준이다. 시스템 특징으로는 인증 프로토콜과 인증 수단을 분리함으로써 보안과 편리성을 확보한다. 일반적으로 인증 프로토콜은 2가지로 제안되었다[6].

첫 번째 모델은 UAF로 Universal Authentication Framework의 줄임말이다. 사용자의 기기에서 제공하는 인증방법을 기반으로 온라인 서비스와 연동하여 권한을 인증하는 기술이다. 대표적으로 지문인식을 통하여 결제 서비스를 진행하는 예시가 있다. (그림 1)에서는 UAF 프로토콜의 상세한 작동 모델이다. (그림 2)에서는 UAF 프로토콜 모델 중에 Authenticator를 구성하는 요소들을 보여준다.



(그림 1) UAF protocol model



(그림 2) UAF Authenticator

두 번째로 모델은 U2F로 Universal 2nd Factor의 줄임말이다. 기존의 아이디와 비밀번호 기반 인증서비스에서 추가적인 사용자 인증을 요구하는 방법이다. USB 보안키를 이용한 2차 인증에 사용하는 것이 대표적인 예이다.

2.2 Webcrypto 알고리즘

Webcrypto는 브라우저 별로 지원하는 알고리즘이 상이하다. 해당 API는 2014년 12월 W3C에서 최종 권고안을 만들었다. 암호 관련 기술을 표준화함으로써 기존 비표준 방식으로 개발된 기능들을 웹에서 별도 설치 없이 처리하는 것을 지향한다. WebCrypto는 실제 키에 대한 접근 요청을 제외한 전자서명 생성, 검증, 해시, 암호화, 복호화, 키 생성, 키 유도, 키 불러오기, 키 내보내기, 키 탐색 등을 수행할 수 있는 인터페이스의 집합으로 구성되어 있다[7][8]. 기능 별로 사용 가능한 알고리즘은 <표 1>에서 확인할 수 있다.

WebCrypto는 암호관련 기능을 표준으로 처리하는 것을 지향하여 브라우저 별 제공사의 목적하는 바에 따라 지원하는 알고리즘의 범위가 다소 차이가 있다. 서비스가 아니라 표준이기 때문에 특정 업체의 환경이나 기술을 추가적으로 강요하거나 권장하지 않는다. 따라서 인증 시스템이 브라우저에 따라 각기 다른 방법으로 작동할 수 있도록 구현한다. <표 2>에서는 브라우저 별 지원하는 알고리즘의 현황이다.

<표 1> Webcrypto에서 지원하는 알고리즘

기능	사용 가능 알고리즘
암호화	RSA-OEAEP, AES-CTR, AES-CBC, AES-GCM, AES-CFB
복호화	RSA-OEAEP, AES-CTR, AES-CBC, AES-GCM, AES-CFB
서명	RSASSA-PKCS1-V1_5, RSA-PSS, ECDSA, AES-CMAC, HMAC
검증	RSASSA-PKCS1-V1_5, RSA-PSS, ECDSA, AES-CMAC, HMAC
해시	SHA-1, SHA-256, SHA-384, SHA-512
키 생성	RSASSA-PKCS1-V1_5, RSA-PSS, RSA-OEAEP, ECDSA, ECDH, AES-CTR, AES-CBC, AES-CMAC, AES-CFB, AES-KW, HMAC, DH, PBKDF2
키 유도	ECDH, DH, CONCAT, HKDF-CTR, PBKDF2
비트 유도	ECDH, DH, CONCAT, HKDF-CTR, PBKDF2
키 로드	RSASSA-PKCS1-V1_5, RSA-PSS, RSA-OEAEP, ECDSA, ECDH, AES-CTR, AES-CBC, AES-CMAC, AES-CFB, AES-KW, HMAC, DH
키 싸기	RSA-OEAEP, AES-CTR, AES-CBC, AES-GCM, AES-CFB, AES-KW
키 풀기	RSA-OEAEP, AES-CTR, AES-CBC, AES-GCM, AES-CFB, AES-KW

<표 2> 브라우저별 지원하는 알고리즘

	Webcryto (PBKDF2)	Webcryto (AES-CVBC)	Webcryto (RSA PKCS1)	CrytoJS (PBKDF2)
IE 11	X	O	O	O
IE Edge	X	O	O	O
Chrome	O	O	O	-
Firefox	O	O	O	-
Opera	O	O	O	-
Safari (sierra)	O	O	O	-

2.3 군 정보체계 운용 특성

현재 군 정보체계로 운용중인 전산망은 전용선을 기반으로 하며 성능 향상을 위하여 일부를 상

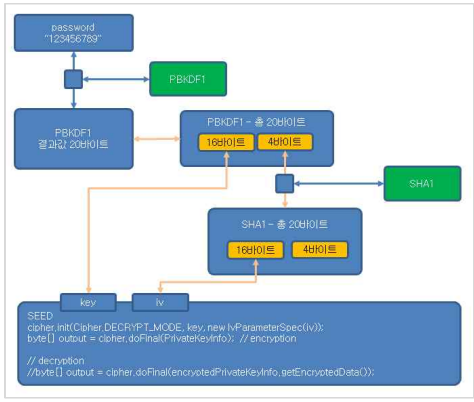
용선으로 분산시키는 구조이기에 개방성을 지니는 한계점을 가진다[9]. 또한 현재 대부분의 데이터가 시스템 성능상의 문제로 인하여 암호화가 이루어지지 않아 유출될 시 심각한 피해가 예상된다. 따라서 정보보호 취약점을 해결하기 위하여 정보 접근이 가능한 사용자의 본인 인증을 강화함으로써 해결되어야 한다. 일반적으로 사용되는 사용자 인증 기법으로는 패스워드, 보안토큰, 생체정보 등의 방법과 공개키 기반구조가 있다. 하지만 해당 기법들은 악성프로그램을 통하여 계정 정보나 토큰이 탈취될 수도 있으며, 국내 인증 시스템의 구조상 완전한 보안이 성립되지 않기 때문에 인증 정보가 탈취되어 악용될 수 있는 취약점이 존재한다. 따라서 새로운 방식의 사용자 인증 기법이 필요한 시점이다.

2.4 국내 PKI 인증 시스템의 취약점

국내의 PKI(Public Key Infrastructure)구조에서 전자서명에 사용되는 비밀키는 국내 암호알고리즘인 SEED를 사용하며 패스워드기반의 키 유도 알고리즘은 PBKDF를 사용한다. 전자서명 알고리즘은 RSA를 사용하며 보안강도는 현재 2048비트의 보안강도를 지원하고 있다. (그림 3)은 국내 PKI인증 시스템의 구조이다. 기본적으로 PKI 인증은 보안상 문제가 없다. 하지만 국내에서 사용하는 인증시스템은 인증서와 개인키가 로컬에 일괄 저장되는 형식으로 간단하게 복사함으로써 탈취 가능한 문제가 있다[10].

2.5 키 생성 알고리즘

키를 생성하는 다양한 방법이 있지만 각각의 방법들이 지원하는 기능도 다르며 동일 방법이라도 버전에 따라 구조적인 측면에서 차이가 있기에



(그림 3) 국내 공인 인증 형식

선택에 따라 구현하는데 차이가 발생한다.

PKCS8[11]는 암호화된 개인키를 저장하는 방법을 정의한다. 공개키와 인증서를 전달할 때 사용하는 공개 형식인 X.509를 사용하여 인코딩을 진행하며 개인키는 PKCS#8로 인코딩을 진행한다. PKCS #7 또는 P7B 형식[12]은 Base64 ASCII 형식으로 저장되며, P7B 파일에는 개인키가 아닌 인증서와 체인 인증서 (중개 CA) 만 포함한다. PKCS # 12 또는 PFX / P12 형식[13]은 서버 인증서, 중간 인증서 및 개인키를 하나의 암호화 가능 파일에 저장하기 위한 이진 형식이다. 보통 개인 컴퓨터에서 인증서 및 개인키를 가져오고 내보내는 데 사용된다. CMS / PKCS # 7은 메시지 서명 및 봉인 기능을 제공. 전자 서명 및 전자 메일 첨부는 SignedCms[14] 메시지 및 EnvelopedCms[15] 메시지 항목에서 별도로 설명한 모든 보안 서비스를 적용하여 보호한다. PKCS5[16]은 개인키의 패스워드를 비밀키로 암호화하는 방법에 대한 정의한다. 비밀키를 생성하는 알고리즘과 비밀키 패스워드를 생성하는 방법에 대하여 정의하며 PKCS8과 함께 개인키와 공개키의 개인키 생성방법과 저장방법에 대한 정의를 기술하고 있다.

2.6 Web Storage : Indexed DB

IndexedDB는 브라우저에 데이터를 저장하기 위한 여러 가지 방법 중에 가장 유연한 접근성을

제공하는 저장소이다. W3C 명세서에는 '많은 양의 데이터를 클라이언트 측에 저장하기 위한 저수준의 API이다' 라고 정의하고 있으며 이는 자바스크립트를 이용하여 데이터를 저장하거나 인출할 수 있도록 하는 객체지향 데이터베이스이다. <표 3>는 브라우저 저장소 간의 특성을 비교한다.

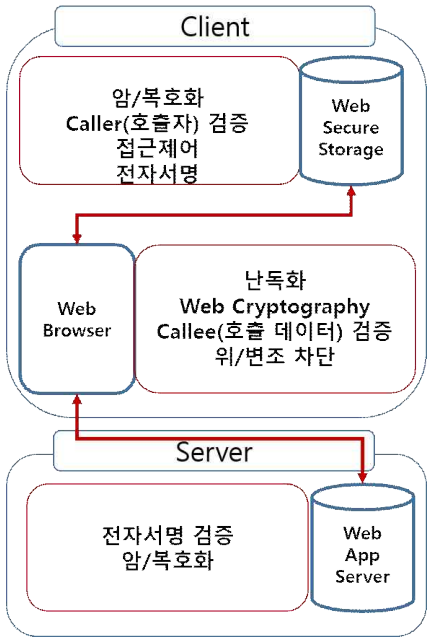
<표 3> Local Storage 특성

특징	쿠키	로컬 저장소	세션 저장소	Indexed DB
저장소 제한	~4KB	~5MB	~5MB	HDD의 절반
영구적 저장	YES	YES	NO	YES
저장 형식	문자열	문자열	문자열	모든 형태
인덱스화	NO	NO	NO	YES

Indexed DB는 IE11을 포함한 거의 모든 브라우저에서 지원한다[17][18]. Indexed DB는 상대적으로 저장용량이 매우 크며 String과 Binary 타입 등 모든 데이터 형태를 사용할 수 있으며 DB와 같이 테이블 형태로 데이터를 구분하여 저장할 수 있다. 저장소가 생성된 후 제공되는 API로 입력, 수정, 삭제를 편리하게 할 수 있으나 지원되는 API가 많은 만큼 다른 저장소에 비해 사용법이 상대적으로 복잡하다.

3. 웹 브라우저 안전 저장소 시스템 설계와 구현

본 연구에서는 안전한 웹 저장소를 활용하여 FIDO 기반의 웹 표준 인증 시스템을 구현하였다. 클라이언트에서는 Javascript 언어 자체의 특성을 보완하기 위하여 위/변조 차단 기능과 상호 검증 기능을 구현하여 인증 정보 탈취를 예방하였다. 또한 인증 정보 자체에 대한 보안 강화를 위하여 Pin인증에 사용하는 정보를 자체 개발한 인코딩 알고리즘을 적용하여 난독화를 진행하였다. 서버에서는 브라우저로부터 전송받은 데이터에 대한 서명 검증 절차를 거쳐 서비스를 제공한다. (그림 4)은 시스템의 전체적인 구조도이다.



(그림 4) 시스템 구조도

3.1 클라이언트

클라이언트는 인터넷 브라우저를 통해 접근을 하며 PC와 모바일에서 제공되는 브라우저다. 서비스는 Tomcat WAS를 사용하며 WAS에서는 안전저장소를 구현한 JS파일 배포, Pin 인증서비스, DB를 이용한 Pin인증 등록 서비스, 전자서명 및 암호화 서비스를 제공하는 기능을 한다.

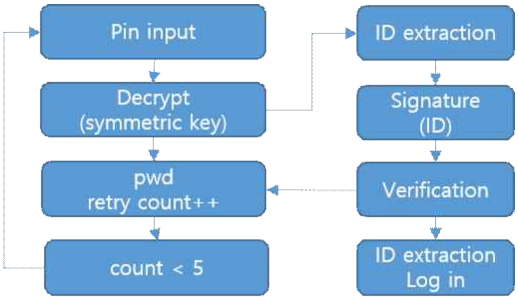
3.1.1 Pin인증 구현

클라이언트 브라우저에서 Pin 정보로 로그인하기 위해 Pin 등록 절차가 필요하다. Pin등록 절차에서는 브라우저에서 생성한 공개키, 개인키 정보가 클라이언트와 서버에 각각 저장되며 특히 클라이언트에서는 ID정보가 암호화되어 저장된다.

개인키와 공개키 쌍은 브라우저 저장소에 저장되며 공개키는 등록 시 서버로 전송되어 저장된다. 등록될 ID는 전자서명을 한 후 서버로 전송되며 해당 ID와 공개키는 쌍으로 서버에 저장된다.

Pin인증의 패스워드는 비밀키를 사용하여 암호화된 ID를 복호화하여 ID를 추출하고 복호화에 성공했

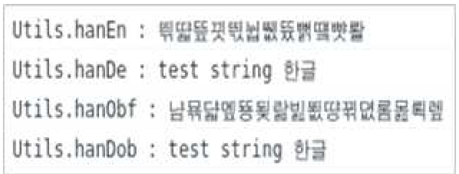
을 경우 미리 생성된 개인키로 전자서명 되어 서버로 전송된다. 전송된 데이터는 서버에 저장된 공개키로 서명 검증을 진행하고 등록된 정보가 정상적인 것임을 확인한 후 로그인 절차를 진행한다. 간편 로그인의 Pin 등록은 간단한 id/pwd검증을 거쳐 등록 절차로 이동하나 실제로는 보안 강도를 높이기 위하여 본인인증 등을 추가로 하여 등록 절차를 실행한다. Pin 패스워드가 틀렸을 경우 암호화된 가변데이터(retry count)가 안전저장소의 데이터 값으로 안전하게 저장된다. 오류 횟수가 증가하며 특정 횟수 이상 틀렸을 경우 이는 정책에 의하여 Pin 패스워드 강제 변경 절차를 진행한다. (그림 5)는 Pin 인증의 흐름도이다.



(그림 5) 핀 인증 구조

3.1.2 한글 인코딩 난독화

자체 개발한 한글 인코딩 알고리즘을 사용하여 저장소에 저당되는 name, value를 쉽게 접근할 수 없도록 적용한다. (그림 6)는 난독화 예이다.



(그림 6) 인코딩 난독화

3.1.3 Javascript 위/변조 차단 기능 개발

Javascript는 언어의 특징상 소스코드가 노출되는 것이 외에도 위/변조가 가능하다. 이를 방지하기 위하여 난독화를 적용하여 소스코드를 보호하는 것에 추가로 무결성 검증을 진행하여 소스코드가 임의로 변조되지 않았음을 확인한다. 자체적으로 고안한 위/변조 방지 코드를 삽입함으로써 function에 대한 임의 변경을 방지하였다.

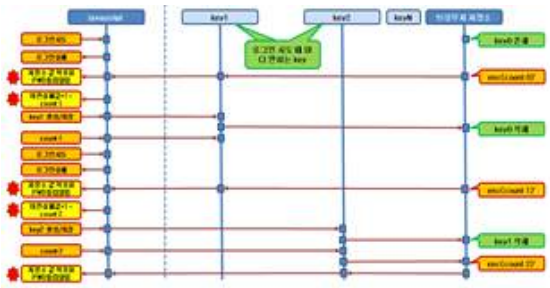
3.1.4 Javascript API 상호검증 기능 개발

일반적으로 javascript 함수는 누구나 호출이 가능하다. 예를 들어 'go()'라는 함수가 있다고 가정하면, A라는 함수에서도 'go()'를 호출할 수 있으며, B라는 함수에서도 호출이 가능하다. API 상호 검증 기능이란, 특정 javascript 함수를 지정된 javascript함수에서만 호출할 수 있도록 접근을 제어하는 기능으로 인가되지 않은 scope에서의 호출은 ALERT Message를 통해 접근제어 한다. 해당 기능은 javascript 위/변조 차단 기능을 보완하는 역할을 수행하며, 공격자에 의해 정의된 함수에서의 호출을 차단한다. (그림 7)는 접근제어가 적용된 위치를 도식화하였다.

(그림 7) 상호인증 구조

3.1.5 가변데이터(retry count) 저장

Retry count 값은 Pin인증 패스워드에 오류가 발생했을 때 횟수를 하나씩 증가시키도록 갱신되어 저장되는 값으로 고정 값을 저장하는 상수 저장 방식이 아닌 다른 로직을 실행하도록 구현한다. 한번 사용된 비밀키와 IV 값은 다시 사용할 수 없도록 로그인 시도 때마다 키가 변경되고, 변경된 키를 이용하여 retry count는 암호화되어 저장된다. 재시도 시 기존의 키 값으로 복호화한 값을 이용하여 재설정할 때는 또 새로운 키를 생성하여 값을 설정한다. 기존의 키 값은 exportable 될 수 없어 복사하여 재사용이 불가능하다. (그림 8)은 가변데이터가 변경되는 흐름도이다.



(그림 8) Retry Count 흐름

3.1.6 안전저장소 구조

클라이언트는 크게 indexedDB에 저장/추출 하는 부분과 브라우저 저장소 부분으로 나뉠 수 있다. indexedDB에 데이터를 저장하거나 안전저장소로부터 데이터를 추출할 때 지정된 API를 이용해야만 수행할 수 있다. 대칭키와 공개키 쌍은 다른 브라우저로 가져갈 수 없으며, 동일한 브라우저에서도 복사할 수 없다. 가변 데이터(retry count)는 매번 키를 갱신하므로 재사용이 불가하다. 또한 저장 로직에 난독화가 적용되어 export API를 사용하여도 키 재사용과 데이터 재사용도 불가하다.

3.2 서버

서버 부분은 javascript를 배포하며 Web을 통하여 수신된 데이터를 처리한다. 또한 전자 서명을 검증하는 역할을 수행한다.

3.2.1 검증 서비스

공개키가 저장된 사용자는 Pin 인증 시 제공되는 전자서명 값을 통해 사용자를 추출하고 전자서명 값과 공개키 값, 원문을 비교하여 전자서명의 유효성을 검증한다. 검증 요청시 ID와 안전저장소에서 개인키를 추출하여 ID를 전자서명 한다. 검증에 성공한 전자서명은 AuthLog에 인증 성공/실패 로그를 저장한다. (그림 9)은 전자서명 검증 서비스의 흐름도이다.

(그림 9) 전자 서명 검증 구조

4. 실험 결과

4.1 실험환경 구성

클라이언트에서는 WebCrypto로 개발된 RSA 키를 생성한다. <표 5>은 클라이언트 실험에 사용한 PC사양이다.

<표 4> 클라이언트 사양

항목	사양
CPU	intel i7-6500U CPU @ 2.50GHz
RAM	8GB
HDD	256GB
NIC	Ethernet 100 Mbps * 1 Port
OS	windows10(professional or home) 64비트 운영체제
브라우저	ie (11이상) ie edge 크롬 (60이상) 파이어폭스 (62이상) 오페라 (57이상) 사파리(10이상)

서버 사양은 아래 <표 6>과 같다.

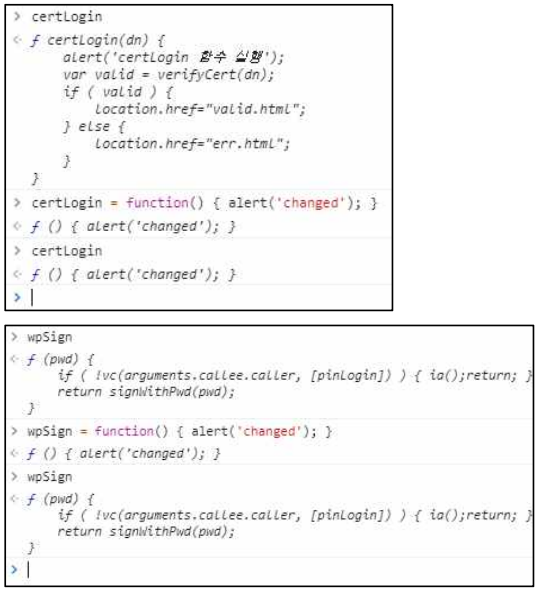
<표 5> 서버 사양

항목	사양
CPU	intel i7-6500U CPU @ 2.50GHz
RAM	8GB
HDD	256GB
NIC	Ethernet 100 Mbps * 1 Port
OS	linux ubuntu 64비트 운영체제 Linux kernel 4.4.0

4.2 안전성 실험 결과

자체적으로 개발한 Javascript 위/변조 차단 및 상호검증 기능과 난독화를 검증 실험을 진행하기

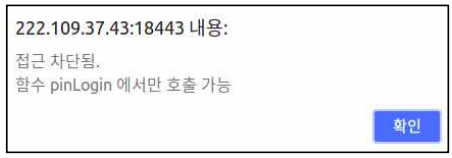
위하여 Javascript의 특징을 악용하는 모의 변조를 진행하였다. 첫 번째로 언어 특성상 소스코드의 노출로 쉽게 위/변조가 가능한 것을 차단하기 위하여 (그림 10)는 방지코드를 적용하여 공격자의 시도를 차단하는 실험이다. 위쪽 시도는 방지코드가 적용되기 전으로 임의의 조작으로 함수의 변경이 가능함을 보여준다. 아래쪽 시도는 방지코드가 적용된 것으로 동일한 조작을 수행하였지만 함수의 내용이 변조되지 않음을 확인할 수 있다.



(그림 10) 위변조 차단 실험

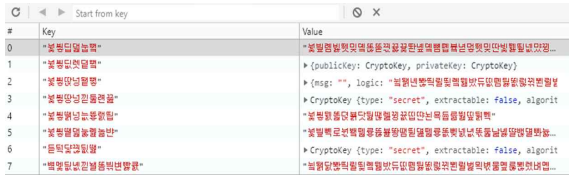
두 번째로 API 상호검증 기능 검증을 위하여 비인가된 함수에서 시료함수를 호출하는 실험을 진행하였다. (그림 11)은 일반 함수와 자체 제작한 검증기능이 적용된 함수를 각각 호출할 수 있도록 구성한 웹 페이지이다. 함수 별로 인가된 접근과 인가되지 않은 접근을 시도할 수 있으며 시료 함수에 인가되지 않은 접근을 시도한다면 (그림 12)과 같은 경고를 확인 할 수 있다.

(그림 11) API 상호검증 실험



(그림 12) 시료함수의 비인가 접근

마지막으로 난독화 인코딩이 적용되어 Indexed DB에 저장되는 값을 실험한다. (그림 13)는 난독화가 적용되어 저장된 사용자 인증 값을 의미한다. 해당 값은 3.1.5의 가변데이터 취급 방식을 적용하여 매 사용마다 새롭게 난독화되며 항상 다른 값을 가져 유출되더라도 해독하기 어렵다.



(그림 13) 난독화 실험

4.3 키 생성 시간 실험 결과

(그림 14)과 (그림 15)는 Chrome과 IE11에서 각각 실험을 진행한 결과이다. WebCrypto로 개발된 RSA 키 생성 속도를 측정 하였으며 양호한 키 생성 속도를 확인하였다. IE11를 제외한 모든 브라우저에서 Chrome과 비슷한 키 생성 속도 0.1 ms를 보였으며 IE11에서는 상대적으로 키 생성속도가 늦는 것을 확인 하였다. 그러나 이 또한 약 1초 미만의 속도를 나타내어 실제 적용의 가능 범

위에 들어가는 적정한 수준을 유지하였다.

키 생성 후 Pin 인증 시 전자서명을 서버에서 검증하는 속도를 측정하였다. 클라이언트에서 ID 값을 전자서명한 서명 값은 서버로 전송되고 서버에서는 전자서명 값을 검증할 수 있는 공개키를 DB로부터 가져와서 서명을 검증한다. 클라이언트에서는 ID값을 전자서명과 함께 전송한다. 이때 ID값은 전자서명에 사용한 원문의 데이터로 사용된다. DB에서 가져온 공개키는 클라이언트의 안전저장소에 저장된 개인키와 쌍을 이루는 키로 전자서명 검증은 빠르게 실행된다. 대략 0.1초 이하의 속도로 전자서명 검증이 실행되었으며 상용화에 적절한 속도를 확보하엿다. (그림 16)는 서버 검증 실험이다.

(그림 14) 키 생성 속도 실험 : Chorme

(그림 15) 키 생성 속도 실험 : IE11

(그림 16) 서버 검증속도 실험

5. 결 론

본 논문에서는 FIDO 기술 기반의 웹 브라우저 안전 저장소를 구축하여 사용자 인증을 수행하는 시스템 연구를 진행하였다. 기존의 설치프로그램 형태를 벗어나기 위하여 Javascript를 기반으로 하였다. Javascript의 특성인 접근 제어 문제를 자체적으로 해결하여 취약점을 해결할 수 있을 뿐만 아니라 사용자 편의성도 확보 하였다. 또한 난독화와 동적 Hash를 통하여 유출된 정보에 대한 해석을 불가능하게 함으로써 OWASP Top 10에서 지속적으로 언급되었던 유출 정보 재사용 문제의 해결을 돕는다.

해당 시스템을 군 정보체계에 접목시킨다면 사용자의 인증정보를 클라이언트에 안전하게 보관하여 사용자 편의성을 확보할 뿐만 아니라 다양한 기법을 적용하여 정보 재사용도 방지할 수 있기에 군 인트라넷 보안 향상에 기여할 것으로 기대한다.

참고문헌

[1] 국방일보, 국방부, “국방망 해킹, 北해커조직 추정세력 소행”, 2017.

[2] 중앙일보, “군 작전 정보체계도 바이러스 감염…핵심 군사기밀 북한에 유출됐는데도 여전히 허술한 군 보안”, 2017.

[3] 김두한, “군보안상 해킹대응방안에 관한 연구”, 융합보안논문지, 17.5호, pp.133-142, 2017.

[4] Halpin, Harry. "The W3C web cryptography API: motivation and overview." Proceedings of the 23rd International Conference on World Wide Web. ACM, 2014.

[5] M. Jemel and A. Serhrouchni, “Security assurance of local data stored by HTML5 web application,” in 2014 10th International Conference on Information Assurance and Security, 2014.

[6] Hwa-Gun. Cho, and Hae-Sool. Yang, “FIDO 생체기술과 안전영역을 연계한 공인인증서 효율화 방법,” 디지털융복합연구, vol. 15, no. 8, pp. 183-193, Aug. 2017.

[7] Ryan Sleevi and Mark Watson. 2014. Web cryptography API.W3C candidaterecommendation, W3C, Dec(2014).

[8] Cairns, Kelsey, Harry Halpin, and Graham Steel. "Security analysis of the W3C web cryptography API." International Conference on Research in Security Standardisation. Springer, Cham, 2016.

[9] 우희철, 김용훈, 정석균. "미래 정보전에 대비한 육군전술지휘정보체계 (C4I) 정보보호대책 연구." 디지털융복합연구, 제10권, 제9호, pp. 1-13, 2012.

[10] Datanet.co.kr, “PKI는 공인인증서가 아니다.” 2016.

[11] B. Kaliski, “Public-Key Cryptography Standards (PKCS) #8: Private-Key Information Syntax Specification Version 1.2,” RFC Editor, May 2008.

[12] B. Kaliski, “PKCS #7: Cryptographic Message Syntax Version 1.5,” RFC Editor, Mar. 1998.

[13] M. Nystrom, S. Parkinson, A. Rusch, and M. Scott, “PKCS #12: Personal Information Exchange Syntax v1.1,” RFC Editor, Jul. 2014.

[14] R. Housley, “Cryptographic Message Syntax,” RFC Editor, Jun. 1999.

[15] R. Housley, “Cryptographic Message Syntax (CMS) Authenticated-Enveloped-Data Content Type,” RFC Editor, Nov. 2007.

[16] B. Kaliski, “PKCS #5: Password-Based Cryptography Specification Version 2.0,” RFC Editor, Sep. 2000.

[17] Kimak, Stefan, Jeremy Ellman, and Christopher Laing. "Some potential issues with the security of HTML5 indexedDB." (2014): 2-2.

[18] Kimak, Stefan, and Jeremy Ellman. "HTML5 IndexedDB Encryption: Prevention against Potential Attacks." International Journal of Intelligent Computing Research 6.4 (2015): 621-630.

〔저자소개〕

박 재 연 (Jaeyeon Park)
2018년 8월 학사

email : rollingjae@sju.ac.kr

이 재 영 (Jaeyeon Park)
2001년 2월 학사

email : kwangbul@irriple.co.kr

이 형 석 (Hyoungseok Lee)
2001년 2월 학사
2003년 2월 학사

email : hslee@irriple.co.kr

강 지 원 (Jiwon Kang)
1988년 2월 학사
1997년 2월 석사
2012년 8월 박사
email : jwkang@sejong.ac.kr

권 혁 진 (Hyukjin Kwon)
1989년 2월 학사
1991년 2월 석사
2000년 8월 박사
email : khjsjy2001@daum.net

신 동 일 (Dongil Shin)
1988년 2월 학사
1993년 2월 석사
1997년 8월 박사
email : dshin@sejong.ac.kr

신 동 규 (DongKyoo Shin)
1986년 2월 학사
1992년 2월 석사
1997년 8월 박사
email : shindk@sejong.ac.kr