J. Internet Comput. Serv.
ISSN 1598-0170 (Print) / ISSN 2287-1136 (Online)
http://www.jics.or.kr

타원곡선 디피헬만 기반 검증 토큰인증방식 구현 연구☆

Study on Elliptic Curve Diffie-Hellman based Verification Token Authentication Implementation

최 정 현[1*]
Cheong H. Choi

요 약

기존 서버기반 인증방식은 취약한 패스워드 기반 인증을 사용함으로 개인정보유출사고가 빈번했다. 이는 불법적 아이디 도용의 문제를 야기함으로 대체할 새로운 인증방안이 연구되었다. 최근 OAuth 2.0 및 JWT 토큰기반 인증을 웹사이트 인증에 사용하고 있지만 토큰 도청만으로 인증정보가 유출되는 취약점이 있어 보완책이 강구되고 있다. 이를 위해 본 논문은 유효시간이 포함된 암호화 인증코드를 담은 검증토큰을 사용한 인증방식을 제안한다. 이 방안의 검증은 검증토큰을 복호화 하여 나온 인증코드를 자신의 계산과 비교하면 된다. 본 토큰 암복호화 방식은 타원곡선 그룹연산 기반 DH(디피헬만)의 세션키 방식과 빠른 XOR 암호방식을 사용하므로 세션키 합의 오버헤드가 없고 암호화 계산이 빠르다. 본 논문의 인증은 기존 토큰인증의 장점, 빠른 XOR 암호화의 장점, 그리고 안전한 DH 세션키 방식을 사용하여 인증 취약점으로 인한 개인정보 유출위협에 대응하는 탁월한 인증방식이다.

☞ 주제어 : 인증, 디피헬만, 타원곡선암호

ABSTRACT

Since existing server-based authentications use vulnerable password-based authentication, illegal leak of personal data occurs frequently. Since this can cause illegal ID compromise, alternative authentications have been studied. Recently token-based authentications like OAuth 2.0 or JWT have been used in web sites, however, they have a weakness that if a hacker steals JWT token in the middle, they can obtain plain authentication data from the token, So we suggest a new authentication method using the verification token of authentic code to encrypt authentication data with effective time. The verification is to compare an authentication code from decryption of the verification-token with its own code. Its crypto-method is based on do XOR with ECDH session key, which is so fast and efficient without overhead of key agreement. Our method is outstanding in preventing the personal data leakage.

☞ keyword : Authentication, Diffie-Hellman, Elliptic Curve Crypto

1. 서 론

최근 인터넷 쇼핑이나 온라인 뱅킹의 경우 휴대기기나 PC에서 OTP 또는 지문 같은 인증코드를 실시간으로 입력받아 사용자 인증에 사용하는 경우가 많다. 그러나 개인정보 관점에서 보면 휴대기기, PC마다 인증정보가 저장되어야 하고 또 여러 서버마다 개인정보가 따로 등록됨으로 함으로 관리가 어려워 도난이나 해킹으로 인한 유출위험이 크다. 특히 시스템 보안이 허술한 서버에서 단순 해킹만으로 개인정보가 불법 유출되어 금융관련거래 등 다양한 사이버 거래에 불법적으로 도용되고 있다 [1][2][3].

이 때문에 서버 시스템의 보안강화를 의무화하는 개인정보보호법[3]은 위반에 따른 벌금도 규정하고 있음으로 기업은 개인정보보호에 각별한 주의를 기울여야 한다. 그럼에도 불구하고 2014년 개인정보 대량 유출사건이 발생했고 관련된 3개의 국내 유수 기관인 농협은행, 국민카드, 롯데카드 등은 개안정보 보호법 적용으로 유출사고에 대해서 법률 최고 벌금형을 받았고 2개 카드사는 가중 벌금까지 부과 받았다 [4].

다수 고객을 상대로 서비스를 제공하는 금융사, 통신사, 정부기관 등에서 개인정보 유출사고가 발생하면 그 양이 방대하고 그 피해가 추적하기 힘들 정도로 클 것으로 예상된다[1]. 또한 불법 유출된 개인정보는 불법적으로

1 Dept. of MIS, Kwangwoon Univ., Seoul, 0189, Korea
* Corresponding author (chchoi@kw.ac.kr)
[Received 20 February 2018, Reviewed 22 February 2018(R2 15 May 2018, R3 10 July 2018), Accepted 29 August 2018]
☆ The present research has been conducted by the research grant of Kwangwoon Univ. in 2012

거래되어 타인명의의 불법적 금융거래에 악용될 수 있어 그 파급효과가 크고 개인에게는 재정적 손실을 가져다주며 법률적 소송 등으로 많은 사회적 비용을 소모하게 된다 [2].

개인정보보안 관점에서 정보유출의 통로, 보안구멍(security hole)은 서버 시스템 보안의 다양한 취약점들과 연관되어 있다. 특히 개인정보 유출은 보안 취약점을 직접 이용하여 이루어지기보다, 취약점을 이용해 인증을 우회하여 시스템에 침투한 후 관리자 계정을 도용하여 유출한다. 빈번한 개인정보 유출 해킹은 패스워드기반 서버를 아이디 도용으로 침투하는 방식이다. 통계를 보면 사용자 아이디 도용 사례가 2006년 개인정보 관련 사건접수 건수의 36%를 차지한다. 또 탈취된 개인정보가 도용된 건수는 90%이다 [5].

따라서 개인정보 유출 해킹의 방어책은 해킹에 취약한 거래서버에 인증을 위한 기본적 개인정보만 두고 민감한 개인정보는 인증서버로 분산 저장하고, 사용자가 직접 인증서버에 토큰발행을 요청하는 방식이 적절하다고 판단하였다. 그러나 기존 토큰기반 인증방식은 토큰 도용에 취약하므로 개선안으로 암호화 기반 검증토큰 방식을 설계하였다.

보통 인증 종류는 인증방식과 인증정보로 나눠 분류한다. 인증방식이란 인증절차와 그 프로토콜, 그리고 인증코드 생성과 검증의 암복호방식 등으로 분류하다. 반면 인증정보란 사용자의 진위를 가리는 식별정보 종류로 분류한다. 인증정보는 무결성을 위해 해쉬값 인증코드로 변환된다. 그러나 사용자 식별정보가 패스워드이든 OTP 또는 생체코드이든지 크래킹의 난이도가 다를 뿐이지 본질적 보안성을 담보하지는 않는다[7].

따라서 본 논문은 인증정보보다는 인증방식에 더 비중을 두어 보안성을 높이려 하였다. 본 논문의 구성은 제2장 연구목적과 범위에서 좀 더 안전한 인증을 위한 우리의 새로운 방안 - 검증코드 방안을 설명하고, 제3장 관련 연구에서 기존 인증방식(그림 1, 2)과 그 취약점 다루고 ECDH의 타원곡선 그룹연산의 기초를 다룬다.

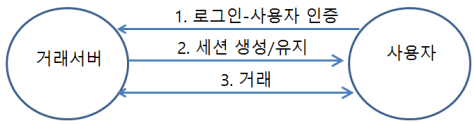
제4장에서 우리 인증방식의 핵심인 인증절차 프로토콜, 인증정보와 검증을 위한 암복호방식을 다룬다. 특별히 암복호방식은 보안성이 뛰어나고 빠른 ECDH(타원곡선 디피헬만) 방식을 어떻게 사용하는지를 설명한다. 보안성은 ECDH 이산대수 NP 문제에 근거한다. 제5장에서 ECDH 키-합의 방식을 이용한 검증토큰 인증방식의 안전성과 우수성을 증명할 것이고 마지막 제6장에서 본 인증방식의 우수성을 요약하고 추후 연구를 언급한다.

2. 연구목적과 범위

본 논문의 목적은 안전하고 빠른 새로운 토큰기반 인증을 제안하는 것이다. 먼저 기존 인증방식의 보안 취약점을 살펴보자.

2.1 시스템 보안

패스워드기반 인증에서 아이디를 도용하여 사용자 권한을 위장하는 공격은 다양하지만, 최근에 주로 웹쉘, 파라미터 변조, APT 공격과 SQL 인젝션 공격 등을 이용한다. 이 공격으로 해커는 사용자 인증을 거치지 않으므로 정상 사용자로 위장하여 시스템 침투할 수 있다 [6].



(그림 1) (서버기반 인증절차)
(Figure 1) (Server based Authentication)

이와 같이 패스워드 방식의 서버기반 인증(그림 1)에서 아이디 도용을 막는 방법은 시스템 보안을 강화하고 취약한 패스워드기반 인증에 다중요소 인증을 접목하는 것이다 [13].

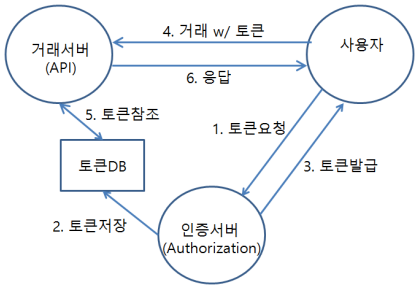
예를 들어 JWT 토큰인증(그림 2)의 경우 보안이 강화된 토큰발행 및 검증 겸용 서버를 둔다. JWT 방식의 목적은 확장성이 요구되는 모바일 환경에 적합하도록 기존 세션을 삭제하는 것이다 [13]. 반면 JWT의 단점은 토큰 유출만으로도 평문인 인증정보가 쉽게 노출된다는 점이다. 또한 다량의 개인정보를 가진 토큰발행서버는 DoS 공격으로 정상적 인증절차가 마비될 수 있는 문제가 있다 [6].

2.2 인증코드 암호화

일반 인증방식은 통신주체인 인증 요청자와 검증자의 신분위장과 재전송 공격의 위험으로부터 보호되어야 한다. 신분위장의 첫 단계는 인증정보를 탈취하여 해커가 합법적 신분으로 메시지의 생성, 삽입, 삭제 및 변조 행위이다[7].

토큰기반 인증의 안전성은 토큰 변조를 탐지하여 도용 가능성을 방지하는 무결성과 토큰 내용을 숨겨서 신분위장을 방지하는 기밀성이라 할 수 있다.

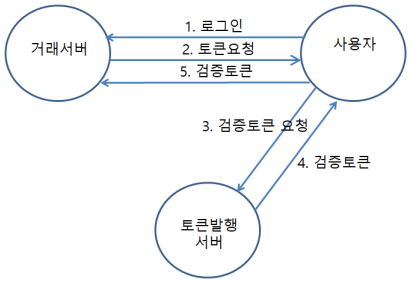
본 논문은 이런 위험을 극복하기 위해 해쉬함수를 생



(그림 2) (토큰기반 인증절차)
(Figure 2) (Token based Authentication)

성한 인증코드로 무결성을, 토큰 암호화로 기밀성을 달성한다. 검증자는 복호화한 인증코드를 자신의 것과 비교하는 기존 검증방안을 사용하여 인증서버로 인한 부담을 없앤다. 또한 인증 참여개체마다 다른 인증정보를 사용하여 개인정보 유출로 인한 위험을 또한 낮춘다.

우리가 제안한 인증절차는 두 흐름으로 나눈다. 첫째 토큰요청 흐름(그림 3의 2, 3)이고 거래서버가 사용자에게 토큰요청을 보내면, 사용자는 식별코드를 실시간으로 입력받아 토큰발행서버에게 검증토큰을 요청하는 것이고, 둘째 흐름(그림 3의 4, 5)은 검증토큰응답 흐름이다. 토큰발행서버에서 사용자를 거쳐 온 검증토큰을 복호하면 상대 검증이 가능하다.



(그림 3) (검증토큰기반 인증절차)
(Figure 3) (Verification Token based Authentication)

2.2.1 검증토큰 방식

기존 토큰방식은 토큰발행서버가 토큰요청 사용자를 검증하여 사용자 정보를 담은 토큰을 사용자에게 발행하고 사용자는 거래 시 토큰을 첨부하여 인증하는 방식이다. 토큰의 검증은 거래서버가 인증(토큰발행)서버에게 요청하여 이루어진다.[13]

본 논문이 제안한 검증토큰이란 거래서버가 인증서버에게 요청하지 않고도 직접 검증할 수 있는 토큰방식이다. 검증은 검증토큰에 실린 인증코드를 복호화 하여 자신이 계산한 인증코드와 비교하는 일반적 방식을 사용한다. 검증토큰에 실린 인증코드는 검증에만 사용함으로 검증코드라 부른다.

검증토큰 암복호에 사용하는 ECDH 세션키는 거래서버와 사용자, 사용자와 토큰발행서버 사이에 각각 생성된다. 따라서 검증토큰의 송수신자는 동일한 세션키를 가지고 암호화로 검증코드의 기밀성을 유지하고 복호화로 검증이 되는 것이다.

여기서 타원곡선 디피헬만(ECDH)의 키 합의를 위해 교환되는 공개값은 토큰요청 흐름과 검증토큰응답 흐름에 담겨진다. 구체적 계산과 전달방식은 제4장에서 논의하겠다.

2.2.2 인증정보 분산과 인증코드 검증

본 논문의 인증절차는 세 참여개체(그림 4) 간 두 번의 (토큰)요청과 (검증토큰)응답으로 이루어진다. 각 요청과 응답마다 부분적으로 다른 인증정보를 참여개체의 보안수준에 따라 분산 보관하여 개인정보 유출 위험을 낮춘다.

거래서버와 사용자 간 인증정보는 사용자 식별고유 아이디와 통신주소(e.g. 전화번호)이다. 사용자와 토큰발행서버 간 인증정보는 사용자 식별고유 아이디와 실시간 인증코드(e.g. OTP, 지문) 그리고 토큰발행서버 고유 아이디(인증등록시 거래서버에 기록). 인증정보는 유효시간이 결합되어 해쉬함수로 인증코드로 변환된다.

각 인증코드의 확인은 두 번 이루어진다(그림 3). 첫 번째는 그림 3의 토큰요청(2)과 검증토큰 응답(5)에서 사용자와 거래서버가 상호 확인하고, 둘 번째는 그림 3의 (검증)토큰요청(3)과 검정토큰 응답(4)에서 사용자와 토큰발행서버가 상호 확인한다. 구체적 구현은 제4장에서 논의하겠다.

3. 관련 연구

본 인증방식과 비교할 기존 인증방식을 논의하였다. 먼저 OTP 방식, 토큰기반 인증방식 그리고 티켓기반 인증방식 등을 언급한다. 마지막으로 ECDH 이론을 논의하였다.

3.1 OTP 인증방식

기존 패스워드기반 인증방식은 패스워드가 장기간 반복 사용되어야 함으로 사회 공학적 추측이 충분히 가능

하고 패스워드 크래킹 도구 등으로 노출되는 취약점을 보였다. 이로 인해 사용자 인증에서 일회용 패스워드를 생성하는 OTP (One Time Password) 방식을 사용하여 기존 패스워드기반 인증의 패스워드 크래킹 가능성을 낮추려 하였다.

OTP 방식의 인증과정을 보면, 개인이 소유한 OTP 장치는 인증센터에 등록 시 유일한 마스터 키가 주어지고 동일한 시드(seed) 생성 방식으로 세팅된다. OTP 생성 시 시드는 새롭게 갱신되어 매번 새로운 패스워드가 생성된다.

OTP 패스워드를 아이디와 함께 대상 서버에게 전송하면 이를 가지고 대상 서버는 OTP 인증 센터에 OTP 검증을 요청한다. OTP 센터는 등록된 마스터 키와 동일한 방식으로 생성된 시드를 활용하여 패스워드를 생성하고 비교하여 검증하게 된다.

여기서 시드(seed)값을 생성하는 방식에 따라서 동기화 방식과 비동기화 방식이 있다. 동기화 방식은 이벤트 동기화 방식, 시간 동기화 방식, 혼합 방식 등으로 나눌 수 있고 비동기화 방식은 서버가 제시하는 질의 값에 응답 값을 서버에 전송하는 징의 응답 방식이다[9].

OTP 방식 중 PC에 탑재된 OTP 방식은 OTP 사용 전에 PIN을 입력하여 OTP 사용자 인증을 받는 것과 동일하다. 이것은 스마트폰에 OTP 방식을 도입하여도 문제는 동일하다. 논문 [9]는 이런 방식의 취약점은 PIN, 마스터 키 값의 취약점으로 기인한다고 지적하였다. PIN의 저장방식이 SHA 해쉬함수를 통한 값으로 저장하고 있고 충분히 추측되는 취약점을 보인다. 위 방식도 OTP 자체의 문제라기보다는 패스워드 PIN의 문제이다.

3.2 SSO와 토큰기반 인증

SSO(Single Sign-On)는 사용자가 한 번의 인증 절차로 다수 서비스에 접속할 수 있는 인증 처리 과정이다. 일정한 유효시간 안에 어떤 관련 서비스는 추가적 인증 요구 없이 접속할 수 있다. 사용자 편의를 증대하고 여러 번 인증 절차가 불필요하므로 보안이 강화된다는 방식이다 [16].

SSO 모델에서 사용자, 서비스(RP), 인증대행(IdP) 등 세 개의 컴포넌트가 동작한다. 최초 인증을 받을 때 사용자는 IdP로부터 토큰을 할당받고 RP 접속 시 이 토큰을 사용한다. SSO는 구현시 사용하는 프로토콜은 보안 도메인 사이 인증과 권한 정보를 교환하기 위한 프로토콜인 SAML, 또는 다른 웹 사이트 간 로그인에서 MS, 구글, 페이스북 계정을 이용하게 하는 프로토콜인 OAuth 등은 토큰기반 인증방식이다[16].

SSO에 더하여 최근 구글, 페이스북 등의 OAuth 2.0 인증토큰을 이용하여 특정 서비스를 로그인 없이 접속하고 있다. OAuth의 동작방식을 살펴보면, 특정 사이트에 접속하려고 할 때 사용자 아이디와 패스워드, 사용자 정보를 남기는 위험을 없애고 구글, 페이스북 등 사용자 아이디, 패스워드와 사용자 정보를 가진 Resource Owner를 통해 인증을 하고 그 인증서버로부터 접근토큰(Access Token)을 발급받아 그 접근토큰을 사이트에 로그인을 대신하여 전달하므로서 접속이 이루어지는 인증방식이다[12].

기존 서버기반 인증 시스템은 서버 측에서 사용자의 정보를 가지고 있어야 한다. 또 로그인으로 인증이 이루어지면 세션을 생성하고 저장하고 그 이후 서비스 요청은 세션 정보를 검색하면서 이루어진다. 그러나 웹이나 모바일 웹 애플리케이션이 늘어나면서 이러한 서버기반 인증방식은 서버의 확장성과 성능에 문제를 야기한다 [13].

세션 사용은 서버에 접속한 사용자 수만큼 세션 정보를 서버에 유지해야 한다. 세션의 수가 증가할수록 서버의 처리능력에 상당한 부담을 준다. 또 세션의 사용은 분산 환경으로 서버를 재설계하는 것을 어렵게 하고, 세션 정보는 쿠키를 통해서 전달하게 되는데 이 쿠키 방식은 여러 도메인을 가로지르는 환경에서는 관리가 쉽지 않은 단점이 있다. 즉 기존 서버 인증방식은 서버의 확장성을 현격히 낮추는 결함이 있다[13].

동시에 동일한 사용자의 인증을 위한 접근토큰은 언제나 동일하므로 반복 사용되면 도청자의 공격에 취약할 수밖에 없다. 이를 막기 위해 TOTP(Time-based One-Time Password) 방식을 사용하는 토큰이 제안되었다. 즉 시간 난수를 사용하여 토큰생성에 적용함으로서 토큰의 반복 사용에도 불구하고 도청의 위험을 낮추었다[14].

3.3 토큰기반 JWT 인증방식

서버기반 인증의 세션을 사용하면 확장성이 떨어지는 분산 환경에 적합한 JWT (JSON Web Token) 토큰기반 인증방식이 있다. 웹 브라우저가 서버에 로그인 하면 JWT 토큰이 발행되고, 이를 브라우저는 저장한 후 다음 페이지 요청시 토큰을 첨부하면 서버는 JWT 토큰의 서명을 검증하여 응답하는 방식이다 [6]. 그림 3는 일반적 토큰기반 인증절차이다.

JWT 토큰의 구조는 (헤더, 페이로드, 서명) 등 세 부분으로 구성된다. 세 부분 모두는 Base64 인코딩을 한 스트링을 서명한다. 헤더는 서명방식을 기술하고, 페이로드는 JSON 클레임(Claim)이라는 사용자 인증정보를 담는다. 특별히 서명은 JSON 클레임에 대해서만 서명 처리한다.

서명방식은 비밀키를 사용하는 HMAC SHA-1 256, 384, 512 방식도 허용된다. 또한 공인 인증서를 사용한 개인키 서명방식인 HS256 방식도 허용된다. 서명에 사용된 비밀키는 토큰을 발행하는 인증서버에 보관되고, API 서버도 접근할 수 있다. 서명 방식은 헤더에 다음과 같이 기록한다 [6].

{“alg”:“HS256”, “typ::”JWT“}

3.3.1 JWT 인증의 취약점

JWT 토큰은 전송 중 도청에 취약하고 누출되면 공개된 Base64 알고리즘으로 JWT 토큰을 디코딩하여 토큰의 클레임(claim) 내용을 이용하여 신분 위장에 도용될 수 있는 심각한 취약점이 있다. 특히 JWT 방식은 모든 페이지 요청마다 JWT 토큰이 첨부되어야 하므로 JWT 토큰의 불법 노출을 방지하려면 웹 서비스 통신이 암호화 채널로 이루어질 필요가 있다 [17].

또한 웹 브라우저에 저장된 JWT 토큰도 해킹의 대상이 되므로 JWT 방식 웹 브라우저의 호스트 시스템 접근관리에 각별히 신경 써야 한다. 현재 JWT 방식의 보안 고려점은 토큰 변조를 막기 위해 무결성을 위한 서명방안뿐이다.

3.4 티켓기반 AAA 인증

티켓이란 인증에서 사용자 또는 서비스를 식별하는 정보조각이다. 사용자와 사용자 네트워크 접근 권한을 식별하는데 사용한다. 티켓기반 Kerberos 인증 메커니즘을 살펴보면, 그 구성은 인증을 얻으려는 Kerberos 사용자(Client)와 접속하려는 서버, 그리고 티켓을 발급해주는 TGS (Ticket Granting Server), 티켓, 사용자를 인증하는 인증서버로 이루어진다[15]. 그 동작을 보면 다음과 같다.

(1) 먼저 사용자가 자신 사용자 ID와 TGS ID를 인증서버에 전송하여 인증을 요청하고, 인증서버는 TGT(티켓부여 티켓)를 발행한다.
(2) 사용자는 TGT를 가지고 실제 접속하는 서비스를 받기 위해서 TGS에 서비스 승인 티켓을 요청한다. 이때 사용자의 패스워드를 요청하여 인증 후 발행된다.
(3) 서비스 승인 티켓을 서비스(거래)서버에 제공하여 사용자 ID와 티켓을 인증하여 서버에 접속을 허락한다.

위 TGT와 서비스 승인 티켓 모두 유효시간이 제한되어 있어 일정시간 동안만 유효하고 또 TGT의 보호를 위해서 인증서버는 비밀키 암호화 방식으로 기밀성을 유지한다.

티켓기반 방식의 가장 큰 취약점은 TGS의 DoS 공격으로 티켓의 발행과 승인을 못하는 경우 티켓기반 서비스가 마비된다는 취약점이다. 대칭키 방식으로 TGT와 티켓을 보호하기 때문에 비밀키 노출로 인한 티켓의 도용이 가능하다.

그러나 티켓기반 인증도 패스워드 인증과 본질적으로는 차이가 없다는 점이다. 만일 사용자 패스워드가 노출되면 티켓기반 인증은 침입이 이루어진다. 확장해서 생각하면 티켓기반 AAA 서버의 인증과정도 사용자 패스워드를 기반으로 한다는 점에서 동일한 취약점을 가진다.

3.5 타원곡선 디피헬만

디피헬만 키교환 방식은 대칭 암호방식에서 공통 비밀키를 교환 배포하는 취약점을 해결하는 방안으로 1976년 디피와 헬만에 의해서 발표되었다.

타원곡선의 순환군(Cyclic Group)이란 다음과 같이 형성된다. 표수(Characteristc)가 p인 유한체(Finite Field) F_q 상의 정수로 이루어진 타원곡선 E 상의 점 (x,y) $x,y \in F_q$들은 기저점(Generator) P, $P \subseteq E(F_q)$로부터 형성된 유한 순환군 G_n, 위수 n를 이룬다. 순환군 G_n의 점들을 구하는 모든 연산은 정수론의 나머지 연산 $\bmod n$이라는 것을 염두에 두어야 한다.

$$G_n = \{(x,y) | (x,y) = kP \bmod n, 1 \le k \le n-1\}$$

타원곡선 디피헬만 문제는 $R = rP$과 $S = sP$ 두 점과 기저점 P를 알고 rsP를 구하는 문제이다. 이 타원곡선 디피헬만 문제는 일반 유한체 상에서 정의된 순환그룹의 디피헬만 문제보다 어렵다고 알려져 있다. 여기서 도청자가 rsP를 구하려면 무작위수 r, s를 알아야 가능하고 P, R, S로부터 r, s를 구하는 문제는 NP에 속한 이산대수 문제이다.

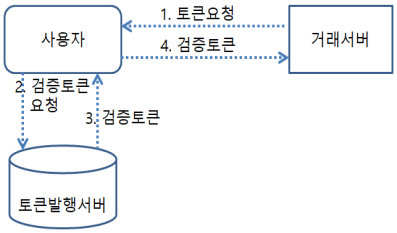
두 당사자 A, B가 디피헬만 방식으로 키 합의에 이르는 알고리즘은 다음과 같다.

① A: $r \in_R (1, n-1)$을 무작위 선택하고, $R = rP$를 계산해서 B에게 전송
② B: $s \in_R (1, n-1)$을 무작위 선택하고, $S = sP$를 계산해서 A에게 전송
③ 키 합의: A의 키는 $rS = rsP$가 되고, B의 키는 $sR = srP$가 된다. 따라서 두 키는 동일

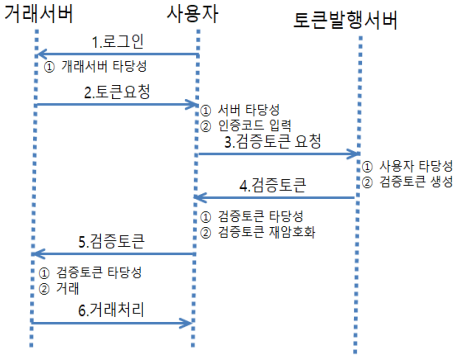
이상과 같은 알려져도 되는 값을 교환한 후 계산이 어려운 공통 세션키를 계산해 낸 것이다 [16][18][19].

4. 검증토큰 인증방식

우리가 새롭게 제안한 검증토큰 인증 (그림 3)을 메시지 중심으로 요약하여 정리하면 그림 4와 그림 5이다. 그림 5의 "2.토큰요청"은 인터넷 뱅킹에서 자금이체 때 OTP 번호로 이중 검증하는 것과 유사하다[17].



(그림 4) (검증토큰 인증참여개체)
(Figure 4) (Participating Entities of Verification Token)



(그림 5) (검증토큰 인증절차)
(Figure 5) (Authentication Process of Verification Token)

4.1 참여개체 인증정보

그림 4를 보면 본 인증방식에서 참여개체는 거래서버, 사용자, 토큰발행서버이다. 사용자가 거래서버에 아이디와 패스워드로 로그인을 하면 거래서버는 인증절차를 요구한다. 여기서 사용자가 토큰발행서버에 검증토큰을 요청하면서 사용자는 실시간으로 입력한 인증코드로 자신의 진위를 인증에 한다. 따라서 참여개체가 사용하는 인증 관련 정보는 부분 차이가 나지만 다음과 같다.

거래서버
① 사용자 고유식별 아이디: 인증등록 때 제공
② 사용자 통신주소: 인증등록 때 제공
③ 토큰발행서버 고유 아이디: 인증등록 때 제공

사용자
① 사용자 고유식별 아이디: 휴대기기 경우 IMEI (International Mobile Equipment Identity) 또는 MEDI (Mobile Equipment Identifier); PC 경우 등록 때 장치 아이디 생성
② 사용자 통신주소: 휴대기기는 전화번호; PC는 IP 주소.
③ 실시간 인증코드: OTP 또는 생체 코드값, 검증계산 방식
④ 토큰발행서버 고유 아이디: 인증등록 때 제공

토큰발행서버
⑤ 사용자 고유식별 아이디: 인증등록 때 제공
⑥ 실시간 인증코드: 인증등록 때 제공
⑦ 토큰발행서버 고유 아이디: 사용자 고유식별 아이디 마다 다른 아이디 생성

4.2 검증토큰 인증절차

그림 5에 보면 각 개체별 메시지의 종류와 처리 동작을 기술하고 있다. 사용되는 수학적 기호의 의미는 다음 표 1과 같다. 위 2.2절에서 언급한 두 흐름을 중심으로 나누어 기술한다.

(표 1) (수학적 기호표)
(Table 1) (Symbol Table)

분류	기호	의미
메시지	m_{s2d}	개체 $s \rightarrow d$ 메시지
참여개체	b, h, p	거래서버, 사용자, 토큰발행서버
ECDH	Q_d, n_d	d의 ECDH로 생성된 공개값과 무작위값
	K^s_{s2d}	s 생성 $s \leftrightarrow d$ ECDH 세션키
	G, q, n	ECDH 파라미터
인증코드	au_d	d의 생성 인증정보 다이제스트
	vr_d	d의 확인용 다이제스트
검증토큰	V_{s2d}	$s \leftrightarrow d$ 검증토큰
검증코드	vc_d	d의 검증토큰에 담길 검증코드
발행장치	P_{id}	토큰발행장치 고유 ID

시간값	t_{dur}	검증토큰 유효시간
	t_{now}	현재시간
	d_t	유효 시간간격
입력코드	ac_h	사용자 실시간 입력 코드
해쉬함수	$h()$	
개체고유	U_{id}	사용자 식별 고유 ID
	a_h	사용자 주소
	P_{id}	토큰발행서버 고유 ID
타원곡선	G	기저점 Generator
	n	순환그룹의 위수

4.2.1 토큰요청 흐름

그림 5의 흐름 2, 3으로 거래서버, 사용자, 토큰발행서버로 토큰요청 흐름이다. 번호는 그림 5에 나타난 번호와 일치한다.

사용자:

1. 거래서버에 아이디, 패스워드 로그인. 유효 시간간격 d_t을 결정하여 거래서버와 합의한다.

거래서버:

2. 인증요청
   - ① 유효시간(t_{dur}) = 현재시간(t_{now}) $\oplus$ 시간간격(d_t)
   - ② 인증코드 au_b = $h(<U_{id}, a_h, t_{dur}>)$ 계산
   - ③ 타원곡선 디피헬만 정수 $(0, n-1)$에서 무작위수 n_b 선정
   - ④ 공개값 $Q_b = n_b G \bmod n$ 타원곡선 그룹연산으로 계산
   - ⑤ 인증요청 메시지 $m_{b2h} = (Q_b, au_b)$ 전송

사용자:

- ○ 인증요청 메시지 $m_{b2h} = (Q_b, au_b)$ 처리
  - ⑥ 거래서버 타당성
    - 유효시간 $t^h_{dur} = t^h_{now} \oplus d_t$ 계산
    - 확인코드 $vr_h = h(<U_{id}, a_h, t^h_{dur}>)$ 계산
    - $vr_h \equiv au_b$ 이면, 타당성 확인
  - ⑦ 실시간 인증코드 ac_h 입력, TOTP는 유효시간값이 고유하게 설정.
3. 검증토큰 요청
   - ⑧ 인증코드 $au_h = h(<U_{id}, a_h, ac_h>)$ 계산, 유효시간값 제외(TOTP 포함)
   - ⑨ 타원곡선 디피헬만 정수 $(0, n-1)$에서 무작위수 n_h 선정
   - ⑩ 공개값 $Q_h = n_h G \bmod n$을 타원곡선 그룹연산으로 계산
   - ⑪ 검증토큰요청 메시지 $m_{h2p} = (Q_h, au_h)$ 전송

토큰발행서버:

- ○ 검증요청 메시지 $m_{t2h} = (Q_h, au_h)$ 처리
  - ⑫ 토큰요청 타당성
    - 확인코드 $vr_p = h(<U_{id}, a_h, ac_p>)$ 계산
    - $vr_p \equiv au_h$ 이면, 타당성 확인
  - ⑬ 검증토큰 생성
    - ECDH 방식 정수 $(0, n-1)$에서 무작위수 n_p 선정
    - 공개값 $Q_p = n_p G \bmod n$을 ECDH 그룹연산으로 계산
    - 토큰발행서버와 사용자 세션키 계산 $K_{p2h} = n_p Q_h (n_p n_h G \bmod n)$
    - 검증코드 vc_p = $h(<U_{id}, a_h, ac_p, P_{id}>)$
    - 검증토큰 $V_{p2h} = K_{p2h} \oplus vc_p$ 생성
4. 검증토큰 메시지 $m_{p2h} = (Q_p, V_{p2h})$ 전송

4.2.2 검증토큰 전달

사용자:

- ○ 검증토큰 메시지 $m_{p2h} = (Q_p, V_{p2h})$ 처리
  - ⑭ 토큰발행서버 타당성
    - 사용자와 토큰발행서버 세션키 계산, $K^h_{p2h} = n_h Q_p (= n_p n_h G \bmod n)$
    - 확인코드 $vr_h = h(<U_{id}, a_h, ac_p, P_{id}>)$ 계산
    - $V_{p2h} \oplus K^h_{p2h} \equiv vr_h$, 타당성 확인
  - ⑮ 검증토큰 변경
    - 사용자와 거래서버 세션키 계산, $K_{h2b} = n_h Q_b = n_h n_b G \bmod n$

- 검증코드 vc_h = $h(<U_{id}, t^p_{dur}, P_{id}>)$, ($ac_p$, a_h 제외, 유효시간 첨부) 변경
- 검증토큰 $V_{h2b} = K_{h2b} \oplus vc_h$ 생성, 추후 유효시간 동안 사용 위해 보관

5. 검증토큰 메시지 $m_{h2b} = (Q_h, V_{h2b})$ 전송

거래서버:

○ 검증토큰 메시지 $m_{h2b} = (Q_h, V_{h2b})$ 처리

⑯ 사용자 타당성 검사

- 사용자와 거래서버 세션키 계산, $K^b_{h2b} = n_b Q_h = n_h n_b G \bmod n$
- 확인코드 vr_b=$h(<U_{id}, t^b_{dur}, P_{id}>)$ 계산
- $V_{h2b} \oplus K^b_{h2b} \equiv vr_b$, 타당성 확인

6. 거래 처리

4.2.3 유효시간 계산

컴퓨터 시간표현법은 여러 가지가 있지만 DOS 시간 표기를 중심으로 설명하겠다. 시간표기는 '연월일:시분초'의 각 부분을 총 4바이트로 적합한 양의 비트를 아래와 같이 할당해서 표현한다.

- 연월일(2바이트) - 년: 7비트(1980년 시작 년도), 월: 4비트, 일: 5비트
- 시분초(2바이트) - 시: 5비트, 분: 6비트, 초: 5비트(2초 단위로 계산)

유효 시간간격은 위 표현방식으로 표현한 후 0와 1을 맞바꿈 마스크값을 취한다. 예를 들면 50초라는 간격은 (0)$_{(27)}$(11001) 으로 표현된다. 이것의 마스크값 d_t은 (1)$_{(27)}$(00110)가 된다. 이때 유효시간(t_{dur}) = 현재시간(t_{now}) ⊕ 시간간격(d_t)은 간격동안은 유효시간은 동일한 값이 된다 [20].

만일 유효시간이 지나면 인증절차에서 t^d_{dur}이 다른 값이 되고 타당성 검사에서 확인코드 vr_d가 달라져서 타당성 판정에 실패한다. 이때는 유효시간 만료로 판정하고 다시 인증절차를 시작해야 한다. 반면 현재 검증토큰 전달에서 타당성 판정에 실패하면 검증토큰 재발행 요청을 해야 한다.

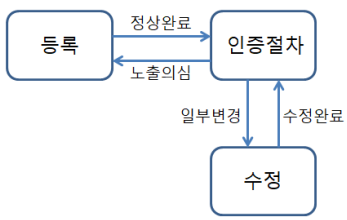
4.3 인증방식 등록 및 수정

본 검증토큰 인증이 적용되도록 최초 인증방식 등록 단계가 있다. 본 인증에 참여하는 개체들이 인증정보와 ECDH 알고리즘과 파라미터를 상호 일치시키는 동기화 단계이다.

등록단계는 안전채널을 확보하고 이루어져야 한다. 타원곡선 순환그룹, 해쉬함수, 실시간 인증코드의 종류와 처리 알고리즘 등을 검증토큰 인증앱 패키지에 담아서 서로 공유한다. 다음은 인증앱에 포함될 필수 공유정보이다.

- 타원곡선 순환그룹 EG_n, 그 순환그룹의 위수 n와 기저점(Generator) G
  $EG_n = \{(x,y) | (x,y) = kG \% n, 1 \le k \le n-1\}$
- 인증코드 계산 해쉬함수 $h()$
- 인증정보
  - 토큰발행서버 고유 아이디, P_{id}는 등록단계에 사용자별로 다르게 계산되어 생성된 후 설정된다. 재등록 과정에서 변경된다.
  - 사용자 식별고유 아이디, U_{id}는 사용자 기기에 따라 인증등록 때 다르게 생성 또는 결정된다. 재등록 과정에서 변경된다.
  - 사용자 통신주소, a_h는 주소가 변경되면 재등록을 수행한다.
  - 실시간 인증코드
    · 코드 종류 및 처리방식: OTP, 패스워드 또는 생체코드 등 처리 알고리즘이 다르다.
    · 코드값: 실제 채취될 고정 고유값

4.3.1 재등록 및 파라미터 수정

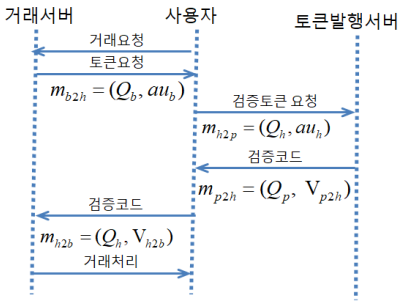


(그림 6) (재등록 및 수정)
(Figure 6) (Re-registration and Modification)

본 검증토큰 방식은 최초 등록된 공유정보가 변경되면 재등록 과정 또는 수정 과정이 필요하다. 사용자 또는

거래서버가 인증방식 공유정보의 불법적 노출을 탐지하거나 의심되는 경우 또는 주기적으로 재등록 과정을 수행한다. 재등록 절차는 초기 등록절차와 동일한 과정이다.

반면 인증방식의 함수 또는 파라미터의 일부 변경 또는 인증정보의 부분 변경이 발생하면 수정 과정을 수행한다. 예를 들면 타원곡선 순환그룹의 변경 또는 인증정보 a_h,ac_h,P_{id},U_{id}의 변경, 그리고 인증코드 종류 및 처리방식의 변경이 발생한 경우 수정 과정을 수행한다.



(그림 7) (참여개체 간 교환 메시지)
(Figure 7) (Message Exchange b/w Entities)

위 재등록이나 수정 과정은 모두 안전채널을 확보하고 일상적 거래는 중지된 상태에서 이루어진다.

5. 보안성과 효율성 분석

본 인증방식에서 해킹공격은 다음과 같다.

- 중간자(Man In the Middle) 공격: 메시지(그림 7) 도청으로 인증정보 노출, ECDH 공개값 노출로 세션키 계산, 검증토큰의 변조와 재전송 등의 공격
- 스푸핑 공격: 참여개체를 가장하는 공격
- 스니퍼링 공격: 참여개체를 해킹하여 인증정보를 불법 도용하는 공격

위 세 가지 공격위험에 대한 안전성을 증명한다.

5.1 중간자 공격

본 인증방식에서는 인증정보 평문의 노출은 불가능하다. 인증정보의 인증코드를 만들 때 일방향성(Trapdoor) 해쉬함수를 사용한다. Trapdoor 함수란 $h(x_d)=au_d$에서 인증코드 au_d만으로 인증정보 x_d를 알아낼 수 없다는 것을 의미이다. 다시 말해 인증코드로 인증정보를 역으로 구할 수는 없다 [20].

인증정보에 대한 다른 공격은 인증코드, 해쉬값 au_d을 재사용하여 재전송하는 것이다. 그러나 사용자와 거래서버 간 인증코드는 au_d = h(< U_{id}, a_d, ac_d, t_{dur}^d >)에서 보면 유효시간값 t_{dur}^d이 결합되어서 유효시간 내에서만 유효하다. 시간이 지난 인증코드는 재전송에 무효하다. 유효시간값이 제외된 사용자와 토큰발행서버 간 인증코드를 도청해도 수시로 변하는 시간제한 실시간 인증코드가 포함되어 있어서 재사용은 역시 어렵다.

ECDH 공개값 $Q_d = n_d G \bmod n$은 타원곡선 그룹연산으로 계산된다. 여기서 그룹 파라미터 G, n은 노출되지 않는다고 가정한다. 그룹연산 $Q_d = n_d G \bmod n$에서 세션키를 계산하려면 Q_d를 안다고 해도 n_d을 알아야 가능하다. Q_d를 알고 역으로 n_d를 계산하는 문제는 수학적으로 어려운 ECDH 이산대수 NP 문제이다. 따라서 공개값 노출은 큰 문제가 되지 않는다(3.6절 참조).

검증토큰 $V_{h2b} = K_{h2b} \oplus vc_p$는 검증코드 vc_p가 세션키로 XOR 되어 있어서 세션키를 모르고 토큰변조는 불가능하다. 검증토큰 재전송 공격도 유효시간 또는 실시간 인증코드의 유효기간만 유효함으로 불가능하다.

5.2 스푸핑 공격

본 인증방식에 대한 스푸핑 공격은 참여개체 즉 거래서버, 사용자 또는 토큰발행서버를 가장하여 인증절차에 참여하는 공격이다.

본 검증토큰 방식에서 참여개체는 자신을 의미하는 인증정보가 등록단계에서 안전채널을 통해 세팅된다. 또각 참여개체는 인증절차에서 필요한 개체별 ECDH 공개값 Q_b, Q_h, Q_p를 계산하고 인증코드 au_b, au_h, au_p를 생성해야 한다. 그러나 등록단계에서 세팅된 파라미터가 일치하지 않으면 유효한 공개값 계산이나 인증코드 생성은 불가능하다. 중간자 공격으로 도청한 공개값이나 인증코드는 유효시간 제한과 시간제한 실시간 인증코드로 재전송 공격에 사용될 수 없다.

5.3 스니퍼링 공격

본 인증방식에서 시스템 해킹 대상개체는 거래서버 또는 사용자이다. 토큰발행서버는 강력한 시스템 보안으로 해킹이 쉽지 않다고 가정한다.

토큰발행서버의 검증토큰은 시간제한 실시간 인증코드로 만들어진 토큰이고 중요한 것은 암호화 되어 있어

서 거짓 인증정보로 생성하기가 어렵고, 설혹 해킹한 인증정보로도 ECDH 파라미터까지 해킹하지 않으면 생성이 불가능하다.

근본적으로 거래서버와 사용자의 인증정보는 토큰을 요청한 사용자의 기초정보인 사용자 식별고유 아이디와 통신주소이다. 두 개체의 인증정보는 개인정보가 아니므로 인증이외의 불법적 용도로 사용되기에 부적합하다.

시스템 해킹으로 유출한 인증정보로 본 검증토큰 인증을 재현할 수 없다는 점에서 안전하다고 말할 수 있다.

5.4 효율성 분석

암호화에서 공개키 방식은 그 속도가 매우 느리다는 것이 큰 단점이다. 따라서 본 검증토큰 생성은 $V_{s2d} = K_{s2d} \oplus vc_s$으로 제한된 시간동안만 유효한 검증코드를 ECDH 세션키로 XOR 암호화 하여 토큰이 보호된다.

본 검증토큰 암호화 XOR 방식은 스트림 암호화에 사용되며 속도는 보면 매우 빠르지만 XOR 암호방식은 암호 알고리즘의 안전성 요소인 혼란과 확산에서 취약하므로 블록암호에 사용될 경우 분석시간이 충분하면 크래킹 될 수 있는 단점이 있다. 따라서 본 검증토큰의 유효시간이 제한되므로 안전하다고 할 수 있다[21].

XOR 암호화 성능은 논문 [21]에서 추출한 표 2을 보면 알 수 있다. 100(MB) 파일 공유에서 암호화 처리 연산 성능을 비교한 표로서, 평문처리 연산, AES 암호처리 연산, XOR 암호처리 연산의 속도를 비교하여 보여준다. 버퍼 크기 1(KB)에서 세 방식의 처리 시간은 각각 427, 2556, 482(ms)을 나타낸다. 이는 XOR 암호처리는 평문처리보다 12.8% 시간 부하가 요구되었지만 대칭키 AES 암호처리와 비교하면 1/5 시간 안에 이루어지는 빠른 성능을 보인다.

결론적으로 검증토큰이란 암호화 되어서 확인코드 vr_d와 세션키 K_{s2d}를 알고 있는 인증개체만이 복호화된 검증코드 vc_d를 비교할 수 있으므로 해커의 유효시간안에 합리적 검사는 어렵다.

(표 2) (암호화 방식 속도 비교표(AES vs XOR))
(Table 2) (Efficiency of AES vs XOR) [21]

	No ENC	AES 256	XOR	암호화 비교
버퍼(KB)	io(ms)	io(ms)	io(ms)	AES/XOR
1	427.8	2566.0	482.4	5.32
256	138.0	1038.0	184.4	5.63
512	134.2	996.8	194.3	5.13

(표 3) (기존 인증방식과 검증토큰 방식의 비교)
(Table 3) (Comparison between exiating authentication and verification token)

평가기준		검증토큰	서버기반	JWT	티켓기반
특징		개인정보 분산 BYOD 기반 인증	기존 패스워드 기반 방식	웹 확장성 토큰기반	다수 서비스 티켓 발행 인증
개인정보	분산	참여개체 분산	사용자/서버 중복	토큰발행 인증서버	TGS 인증서버
	노출	토큰암호화 매우 낮음	시스템 보안 취약	토큰노출 취약	패스워드 취약
인증정보	종류	개인정보	패스워드	개인정보	패스워드
	보호	해싱과 ECDH	패스워드 보호	토큰 인코딩 (Base64)	티켓 비밀키 암호화
검증방식		토큰복호 -> 해싱값 확인	패스워드 확인	서명복호/디코딩 확인	패스워드 확인
시간제약		검증토큰	OTP 사용시	미사용	티켓

6. 결　론

본 검증토큰 인증방식은 네 가지 장점을 가지고 있다.

첫째, 표 3을 참조하면 기존 토큰기반 인증(i.e. JWT 방식)은 확장성이 높은 모바일 환경에 적합한 방식이지만 토큰을 평문으로 유통하므로 노출 위험이 높아서 인증의 개인정보 노출이 빈번히 발생한다[13].

본 검증토큰 방식은 인증정보에 유효시간과 결합하여 해싱된 검증코드를 다시 세션키로 XOR 암호화 되어 있으므로 토큰 노출로 인한 인증정보 유출의 위험은 없다.

둘째, 수행속도가 빠른 타원곡선 순환그룹 기반 암복호 연산에 기반을 둔 디피헬만 방식이므로 타 디피헬만 방식보다 계산이 빠르다. 또 디피헬만 세션키 합의는 한 번의 Round가 요구된다. 그러나 본 방식은 세션키 사전 합의 없이 요청과 응답이 이루어지므로 합의 오버헤드가 없다. 그림 7을 보면 인증절차의 요청 메시지에 공개값을 실어 보내고 응답 메시지에 상대 공개값이 함께 수신됨으로서 세션키도 합의된다.

셋째, 검증은 합의된 세션키로 복호화 함으로 이루어진다. 따라서 기존 토큰인증에서 인증서버에게 다시 토큰의 검증을 요청하는 절차가 불필요하여 인증절차의 처리 속도가 빠르다(그림 2, 3 참조).

넷째, 검증코드의 유효성은 제한된 시간 안에만 검증이 가능하다. 그러므로 제한된 시간 내의 해킹은 어렵다. 검증토큰의 재전송 공격도 불가능하다.

추후 연구할 사항은 타원곡선 그룹연산 알고리즘의 효율성을 위한 방안을 연구하여 구현하는 것이다.

참고문헌(Reference)

[1] “Leakage of 1,230 Personal Info. in KT LGU+ SKT,” Moneytoday, MARCH 11, 2014. http://news.mt.co.kr/mtview.php?no=2014031111001655436&type=1

[2] “Leakage of 20 millin personal info. of drivers in Administrative Portal of Automotive Complaints,” MBCnews, March 26, 2014, http://imnews.imbc.com/replay/2014/nwdesk/article/3436559_18451.html

[3] Sun-Jae Sung, “Personal Information Protection Act,“ Seoul Economy and Management Pub., 2014. ISBN 9788997937172

[4] “*Personal Information Leakage Accident ‘Nonghyup, KB Card, Lotte Card’ Penalty*,” http://news.jtbc.joins.com/article/ArticlePrint.aspx?news_id=NB11273154, [JTBC] input 2016-07-15 17:16:05 modified

[5] Tae-Myung Jung, 2, “A study on the prevention of a privacy exposure and the protection of privacy information on the Internet,” Korea Information Security Agency, Technical Report KISA-WP-2007-0042, Dec. 2007, https://www.kisa.or.kr/jsp/common/libraryDown.jsp?folder=012271

[6] Kyung-Ae Kim, “The First cause of Personal Information is External Attack! Frequent 4 Hacking method is:,” Security News, http://www.boannews. com/media/view.asp?idx=56616, Aug-2, 2017. http://dx.doi.org/10.14257/jse.2017.04.02

[7] S.R,Cho, O.S. Choi, S.H, Jin and H.H. Lee, “Passwordless Authentication Technology-FIDO,” Electronics and Telecommunications Trends, Vol. 29, Issue 4, Aug. 2014. https://doi.org/10.22648/ETRI.2014.J.290411

[8] Byung-Rae Cha, Nam-Ho Kim, Jong-Won Kim, “Design of OTP based on Mobile Device using Voice Characteristic Parameter,” The journal of Korea Navigation Institute, Vol 14 Issue 4, Aug. 2010. http://kiss.kstudy.com/thesis/thesis-view.asp?key=2867972

[9] Woo Chan Hong, Kwang Woo Lee, Seung Joo Kim, Dong Ho Won, “Vulnerabilities Analysis of the OTP Implemented on a PC,” The KIPS Transactionsty C, Vol. 17 No. 4, pp.361-370, 2010. https://doi.org/ 10.3745/KIPSTC.2010.17C.4.361

[10] Nam Ho Kim, Bu Hyun Hwang, “Fingerprint overlay technique of mobile OTP to extent seed of password,” The journal of Korea Navigation Institute, Vol. 16 No. 2, pp.375-385, 2012. https://doi.org/10.12673/jkoni.2012.16.2.375

[11] Seo Il Kang, Im Yeong Lee, “A Study on PIN-based Authentication and ID Registration by Transfer in AAA System,” The KIPS transactions : Part C, Vol.13 No.3, 2006. http://www.riss.kr/link? id=A101859859

[12] Hardt, D., Ed., ″The OAuth 2.0 Authorization Framework″, RFC 6749, Oct. 2012. https://doi.org/10.17487/RFC6749

[13] velopert, “[JWT] Introduction on Token based Authentication,” Dec. 4 2016. https://velopert.com/2350

[14] Byoungcheon Lee, “Strengthening of Token Authentication Using Time-based Randomization,” Journal of Security Engineering, Vol.14, No.2, pp.103-114, 2017. http://dx.doi.org/10.14257/jse.2017.04.02

[15] Namho Kim, ChoongSeon Hong, “OAuth-Based Authentication with Kerberos for IoT Network,” 2016 Proceedings of the Korean Information Science Society Conference, 2016. https://www.dbpia.co.kr/Journal/ArticleDetail/NODE07017547

[16] Jin-Tak Choi, “A Study on Authentication Management Technique Used of SSO,” J. KSIAM IT series Vol.7, No.2, pp1-9, 2003. https://www.dbpia.co.kr/Journal/ArticleDetail/NODE00992033

[17] I. Liusvaara, “CFRG Elliptic Curve Diffie-Hellman (ECDH) and Signatures in JSON Object Signing and Encryption (JOSE),“ Request for Comments: 8037, January 2017. http://dx.doi.org/10.17487/RFC8037, https://tools.ietf.org/html/rfc8037

[18] Song, Bo-Yeon, Kim, Kwang-Jo, “Key Agreement Protocol based on Diffie-Hellman Problem Over Elliptic Curve ,” Proceedings of the Korea Information Processing Society Conference (1), pp.785-788, Oct. 13 2000. http://caislab.kaist.ac.kr/publication/paper_files/2000/bysong/kips14_song.pdf

[19] Kyoungsook Jung, TaeChoong Chung, “Hybrid Cryptosystem based on Diffie-Hellman over Elliptic Curve,” Journal of the Korea Society of Computer and Information Vol. 8 No. 4, pp104-110, Dec. 2003. http://insight.dbpia.co.kr/article/related.do?nodeId=NOD

E06535740
[20] Cheong H. Choi, “IBE based Mobile IP Security”, Proceedings for ICONI & APIC-IST 2010, Mactan Island, Philippines, pp115-118, 2010-12-17.
[21] Hyun-Wook You, “Design of a XOR encryption based file sharing system that provides efficient in cloud server environments ”, Thesis of MS, Hanyang Univ., Feb 2017. http://www.riss.kr/link?id=T14385365

◐ 저 자 소 개 ◑

최 정 현(Cheong H. Choi)
1984년 서울대학교 컴퓨터공학과(공학사)
1988년 조지아공과대학 컴퓨터 과학과(이학석사)
1992년 어번대학교 컴퓨터 공학과(공학박사)
1994년~현재 광운대학교 경영정보학과 교수
관심분야 : 컴퓨터 네트워크, 정보보안, etc.
E-mail : chchoi@kw.ac.kr