Check for updates

Journal of KIIT. Vol. 23, No. 6, pp. 107-117, Jun. 30, 2025. pISSN 1598-8619, eISSN 2093-7571
http://dx.doi.org/10.14801/jkiit.2025.23.6.107

가변쿠키를 활용한 웹 보안 및 성능 개선에 관한 연구

정진호*, 차영욱**

A Study on Improving Web Security and Performance using Variable Cookie

Jin-Ho Jeong*, Young-Wook Cha**

이 논문은 2024학년도 국립경국대학교 학술연구조성비에 의하여 연구되었음

요 약

HTTP는 비상태 프로토콜로, 상태성 제공을 위해 쿠키와 세션이 널리 사용된다. 쿠키는 브라우저에 저장되고 서버에 부하를 주지 않는 장점이 있지만 보안성이 낮으며, 세션은 서버에 저장되므로 보안성이 높으나 서버 부하를 초래할 수 있다. 안전한 쿠키는 기밀성, 무결성, 인증, 재생공격 방지를 모두 충족해야 하지만, 기존 방식은 일부 조건만 만족하며 고정값 사용으로 보안 취약점이 존재한다. 본 논문에서는 세션의 보안성과 쿠키의 성능을 모두 충족하는 가변쿠키(VC, Variable Cookie)를 제안한다. 접속한 IP주소, User-Agent, 지속적으로 갱신되는 쿠키의 만료시간으로 생성되는 VC는 네 가지 보안 조건을 충족한다. VC는 쿠키가 조작되면 인증이 무효화되며, 페이지 이동 시 쿠키값을 지속적으로 변경하여 보안성을 강화하도록 설계하였다. 기존의 안전쿠키 및 세션과의 안전성과 성능을 비교하여 제안한 VC가 웹 보안 및 성능 향상에 기여할 수 있음을 확인하였다.

Abstract

HTTP is a stateless protocol, and cookies and sessions are widely used to provide statefulness. Cookies have the advantage of being stored in the browser and not placing a load on the server, but they are less secure, while sessions are stored on the server, so they are more secure, but can place a load on the server. A secure cookie must satisfy four conditions: confidentiality, integrity, authentication, and anti-replay attack. However, existing methods satisfy only some of the conditions and have security vulnerabilities because they use fixed values. In this paper, we propose a Variable Cookie (VC) that satisfies both the security of sessions and the performance of cookies. VC which is generated by the connected IP address, User-Agent, and the cookie expiration time that is continuously updated, satisfies four security conditions. VC is designed to strengthen security by continuously changing cookie values when moving pages, and authentication is invalidated if cookies are manipulated. It was confirmed that the proposed VC can contribute to improving web security and performance by comparing the safety and performance with existing secure cookies and session.

Keywords

cookie security, anti-replay, session management, authentication, web security

* 디제이패밀리 대표
- ORCID: http://orcid.org/0000-0002-2602-3062
** 국립경국대학교 컴퓨터공학과(교신저자)
- ORCID: http://orcid.org/0000-0002-9448-4899

· Received: Mar. 28, 2025, Revised: May 08, 2025, Accepted: May 11, 2025
· Corresponding Author: Young-Wook Cha
Gyeongkuk National University, Korea
Tel.: +82-54-820-5714, Email: ywcha@gknu.ac.kr

Ⅰ. 서 론

HTTP(Hypertext Transfer Protocol)는 비상태성 프로토콜로 설계되었으나[1], 로그인, 쇼핑, 장바구니 기능 등의 상태성이 요구됨에 따라 쿠키와 세션 같은 메커니즘이 도입되었다[2]. 웹 프로그램 통계제공 웹사이트인 W3Techs의 2025년 1월 통계에 따르면 쿠키는 전체 웹사이트의 약 42%, 세션은 약 26.4%에서 사용되고 있다[3]. 쿠키는 브라우저에 데이터를 저장하므로 서버의 부하를 줄일 수 있는 장점이 있으나, 데이터 변조 및 탈취에 취약한 단점이 있다[2]. 세션은 사용자 데이터를 서버에 저장하므로 쿠키에 비하여 기밀성과 무결성이 뛰어나다. 그러나 세션ID가 노출되면 공격자가 이를 이용하여 사용자의 권한을 탈취할 수 있어 재생공격에 취약성이 있다. 또한 저장되는 세션이 많을 경우 서버 부하가 증가하는 문제가 있다[4].

이러한 문제를 해결하기 위해 다양한 안전쿠키 방식이 제안되었지만[5][6], 대부분의 방식이 기밀성, 무결성, 인증, 재생공격 방지와 같은 핵심 보안 요건[7]-[11]을 모두 충족하지 못하고 있다. 기존 방식은 사용자의 가용성을 위하여 쿠키의 만료시간을 길게 설정해야하고, SSL Session ID 변수의 확인이 가능한 웹 서버 언어와 HTTPS 환경에서만 사용이 가능하여 호환성 측면에 제약이 있었다. 본 논문에서는 이러한 한계를 극복하고 보안성과 성능의 균형을 이루기 위해 가변쿠키(VC, Variable Cookie) 방식을 제안한다. 가변쿠키는 사용자 고유정보, IP주소, User-Agent, 페이지 이동 시 생성되는 쿠키만료시간, 그리고 서버 내에 숨겨져 있는 서버키를 결합하여 생성된 해쉬값을 쿠키로 사용한다. 새롭게 생성되는 쿠키만료시간으로 인하여, 지속적으로 변하는 쿠키의 해쉬값과 갱신된 쿠키만료시간을 데이터베이스에 저장한다.

가변쿠키는 쿠키를 생성하는 구성요소들을 연결한 후 해쉬함수로 암호화하기 때문에 안전쿠키의 조건인 기밀성을 만족하며, 해쉬함수의 입력값 구성요소의 길이가 길기 때문에 해쉬값 크래킹에도 안전함을 증명하였다. 또한, 쿠키값을 한 글자라도 변조하면 인증이 되지 않거나 데이터를 복호화할 수 없으므로 무결성도 만족한다. 기존 안전쿠키방식[5][6]은 쿠키 알고리즘이 유효하기만 하면 다수의 쿠키가 생성될 수 있다. 반면, 가변쿠키는 해쉬값과 쿠키의 만료시간을 데이터베이스에 저장하므로, 유효한 사용자의 단 하나의 쿠키 인증만 허용한다. 가변쿠키는 데이터베이스에 쿠키의 해쉬값이나 쿠키의 만료시간이 존재하지 않으면 재생공격을 통한 인증이 불가능하다. 또한, 해쉬값이 데이터베이스에 저장된 값과 일치하지 않거나, 데이터베이스에 저장된 쿠키의 만료시간이 지났거나, 이용 중인 사용자의 IP주소와 User-Agent가 쿠키와 일치하지 않으면 인증이 불가능하다. 이는 재생공격 방지를 만족한다.

본 논문의 2장에서는 쿠키와 세션의 보안 위협 및 기존 안전한 쿠키의 연구동향을 기술하며, 3장에서는 웹 보안 및 성능 향상에 기여할 수 있는 가변쿠키의 설계 및 안전성에 대하여 기술한다. 4장에서는 가변쿠키와 기존 안전한 쿠키 및 세션과의 안전성, 인증 및 검증 처리시간 등의 성능을 비교하며, 5장에서는 결론 및 추후 계획에 대하여 기술한다.

Ⅱ. 관련 연구

2.1 HTTP 쿠키와 세션

HTTP의 비상태성을 상태성으로 만드는 방법으로 쿠키(Cookie)와 세션(Session)이 제안되었다[2]. 표 1은 쿠키와 세션의 차이점을 나타낸다[2].

표 1. 쿠키와 세션의 차이점
Table 1. Difference between cookies and sessions

Category	Cookie	Session
Store location	Client	Server
Capacity	Limited by web browser	Limited by web server's memory
Security	Secure than session	Not secure than Cookie
Expiration	Following the cookie setting	When the browser closes

쿠키는 클라이언트, 즉 브라우저에 텍스트 파일 형태로 저장되며, 저장 가능한 쿠키의 개수와 최대 크기는 웹 브라우저마다 상이하다. 구글 크롬 브라우저는 최대 180개의 쿠키를 저장 가능하며, 쿠키당 최대 4096바이트까지 저장할 수 있다[12]. 세션은 서버에 세션파일로 저장된다. 세션파일의 최대 크기는 웹서버의 설정에 따라 다르지만, PHP의 최대 크기 기본값은 128MB 이다[13]. 저장되는 세션 데이터가 많아질수록 파일이 커지게 되며, 파일을 불러오는 속도가 저하되므로 서버에 부하를 주게 된다. 쿠키는 세션보다 저장 가능한 데이터 크기가 작으며, 사용자 브라우저에 정보를 보관하므로 서버의 부담을 상대적으로 주지 않는다.

2.2 안전한 쿠키체계

2.2.1 안전한 쿠키의 조건

서버 부하를 줄이면서도 보안성을 강화하기 위해 제안된 안전한 쿠키 체계는 쿠키의 기밀성, 무결성, 인증 및 재생 공격 방지 기능을 모두 충족하는 것을 의미한다[7]-[11]. 쿠키의 기밀성은 클라이언트에 저장된 쿠키 정보를 서버 이외의 다른 클라이언트나 공격자가 접근하여 읽을 수 없도록 보호하는 것을 의미한다[14]. 무결성은 특정 데이터가 임의로 생성, 삭제되거나 변조되지 않도록 보호하는 것을 의미한다[15]. 쿠키값에 사용자의 로그인 정보가 있다고 가정하였을 때, 공격자는 쿠키를 변조하여 다른 사용자의 정보로 로그인이 가능할 수 있다.

인증은 사용자 또는 시스템의 신원을 확인하는 과정이다[15]. 사용자가 로그인을 시도하면 서버는 아이디와 패스워드를 검증한다. 합당한 사용자로 판단되면 인증 쿠키를 만들어 사용자에게 전송하며, 사용자는 전송받은 쿠키를 저장한다. 브라우저는 서버와 통신 시 저장된 쿠키를 전달하여 해당 쿠키가 유효한지 검증받는다[16]. 공격자는 사용자가 보내는 인증 데이터를 그림 1처럼 스니핑하여 서버 측에 보내면, 공격자는 사용자의 인증과 동일하게 인증될 수 있다. 이처럼 공격자가 자신이 아닌 타 사용자의 인증 데이터를 서버 측에 보내어 인증하는 것이 재생공격이다.

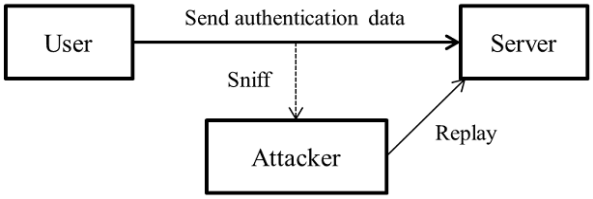


그림 1. 재생공격
Fig. 1. Replay attack

2.2.2 Alex의 쿠키

W3Techs의 2025년 1월 통계에 따르면, Wordpress는 전체 CMS(Content Management System) 시장의 61.8%를 점유하며, 전체 웹사이트의 43%를 차지하고 있다[17]. 현재 Wordpress의 로그인은 Alex가 제안한 안전한 쿠키방식을 사용하고 있다[5][18]. Alex의 안전한 쿠키체계는 그림 2와 같다.

$$\text{user name}|\text{expiration time}|(\text{data})_k$$
$$|\text{HMAC}(\text{ user name}|\text{expiration time}|\text{data}|\text{session ID},\ k)$$
$$\text{where } k=\text{HMAC}(\text{user name}|\text{expiration time},\ sk)$$

그림 2. Alex의 쿠키
Fig. 2. Alex's cookie

사용자 이름과 쿠키의 만료시간, 키값 k로 암호화한 데이터, 그리고 SSL(Secure Sockets Layer) Session ID를 붙여 k로 HMAC 해싱한 결과를 서로 붙여서 쿠키를 생성한다. 사용자 이름과 쿠키 만료시간, 그리고 데이터를 사용하여 해쉬값을 만들기 때문에 쿠키값을 조작할 수 없고, 조작된 쿠키값으로는 인증이 불가능하므로, 안전한 쿠키체계의 조건인 무결성과 인증을 만족한다. Alex 쿠키의 문제점은 사용자 이름이 암호화되지 않아, 저장된 쿠키를 통해 특정 사용자가 인증되었음을 확인할 수 있다는 것이다. 브라우저에 남아있는 쿠키정보를 통해 사용자 계정이름을 바로 알 수 있으므로 이는 쿠키의 기밀성을 만족하지 않는다고 볼 수 있다.

Alex의 쿠키는 재생공격방지를 위해 SSL Session ID를 사용하였다. W3Techs 의 2024년 3월 통계에 따르면, 가장 많이 사용하는 웹 개발언어는 PHP로 약 76.4%이며, 두 번째 언어는 ASP로 약 6.4%이다.

따라서 SSL Session ID를 지원하지 못하는 PHP나 ASP 같은 백엔드 언어는 SSL Session ID 값을 쿠키에 붙일 수 없으므로, 재생공격방지가 이루어질 수 없다. 정리하자면, Alex의 쿠키는 안전한 쿠키체계 조건인 기밀성, 무결성, 인증, 재생공격 방지 중에서 기밀성과 재생공격방지를 위반한다.

2.2.3 Lee의 쿠키

Alex 쿠키체계의 기밀성을 보완한 Lee[6]의 쿠키체계는 그림 3과 같다.

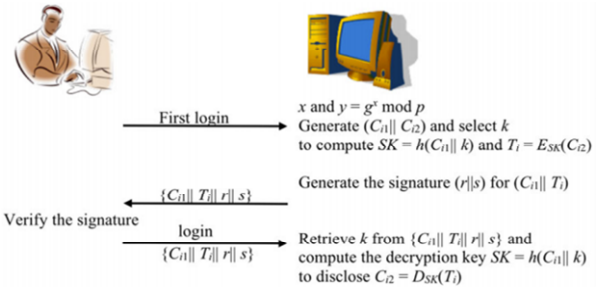


그림 3. Lee의 쿠키
Fig. 3. Lee's cookie

Lee의 쿠키는 일반적인 데이터와 개인정보에 민감한 데이터를 먼저 구분한다. 큰 소수를 선택하고 계산하여 키를 생성한 후, 이를 일반 데이터와 결합하고 해시 함수를 사용해 해시값을 생성하여 서버키를 생성한다. 민감한 데이터는 생성된 서버키를 이용해 대칭키로 암호화된 후, 일반 데이터를 암호화된 데이터와 함께 결합하여 쿠키를 생성한다. Lee의 쿠키는 붙여진 전체 데이터를 대칭키 암호화하기 때문에 쿠키값은 공격자나 사용자가 식별할 수 없는 값이 되므로 기밀성을 만족한다고 볼 수 있다. 또한, 암호화했던 데이터를 복호화하였을 때 복호화된 데이터의 식별 가능 여부와 유효성을 검증하므로 무결성과 인증 또한 만족한다.

Lee 쿠키의 문제점은 사용자가 재인증을 하지 않거나 쿠키 데이터가 갱신되지 않으면, 인증 시 생성된 쿠키가 항상 동일한 값으로 유지된다는 것이다. Lee의 쿠키는 사용자가 처음 로그인 시 사용자마다 서로 다른 서버키를 생성하므로 공격자로 하여금 쿠키값을 유추하기 어렵게 만들었으나, 처음 로그인 외에는 고정값인 쿠키를 통하여 서버와 통신하며 검증 절차를 거친다. 따라서 해당 쿠키가 공격자에게 탈취되면, 외부에서 인증을 시도할 수 있어 여전히 재생 공격에 취약할 수 있다.

2.2.4 Paypal의 Super 쿠키

2024년 2월 온라인 송금회사인 페이팔(Paypal)이 재생공격방지에 특화된 쿠키 구조를 특허로 등록하였다[19]. 현재 페이팔에 적용된 페이팔의 Super쿠키는 먼저 인증을 성공한 사용자의 기존 쿠키를 통하여, 해당 사용자가 방문했던 사용자인지 아니면 처음으로 방문한 사용자인지 확인한다. 만약 처음 방문자라면 사용 중인 디바이스 정보를 데이터베이스와 쿠키에 저장하고 인증에 성공한 페이지로 이동한다. 기존에 방문했던 사용자라면 쿠키에 저장된 디바이스와 현재 인증하려는 디바이스를 비교하여, 디바이스가 일치하면 성공페이지로 이동하고, 일치하지 않으면 페이팔에서 자체 보유한 사기 위험점수를 계산하여 처리한다.

사용자의 디바이스 정보는 HTTP 헤더에 있는 User-Agent로 가져올 수 있다[20]. User-Agent는 현재 사용 중인 브라우저, OS정보 등을 나타낸다. User-Agent의 취약점은 조작이 매우 쉽다는 것이다. 어떠한 네트워크에 접속 시 자신의 User-Agent를 조작하여 접속하면 서버는 User-Agent가 조작되었는지 판단할 수 없게 된다. 따라서 페이팔이 특허로 등록한 Super쿠키는 공격자가 User-Agent를 조작하고, 사용자의 인증뿐 아니라 디바이스 정보까지 들어있는 쿠키를 가져올 수 있다면 재생공격이 가능해진다.

III. 가변쿠키를 이용한 웹 보안 강화

3.1 가변쿠키의 생성과 유효성 확인

3.1.1 가변쿠키의 생성

본 논문에서 제안하는 가변쿠키(VC)는 안전한 쿠키체계의 조건인 기밀성, 무결성, 인증, 그리고 재생공격방지를 모두 만족하기 위하여 그림 4와 같이 생성한다.

VC1 = Hash(Username|IP Address|User-Agent|Cookie's new Expiration|Secret Key)

(a) Without data to be encrypted

VC2 = Hash(Username|IP Address|User-Agent|Cookie's new Expiration|Secret Key)|(Data|Secret Key)k
where k = Hash(Username|IP Address|User-Agent|Cookie's new Expiration|Secret Key)||Secret Key

(b) With data to be encrypted

* | is separator * || is merging string

그림 4. 암호화되는 데이터의 유무에 따른 가변쿠키
Fig. 4. Variable Cookie depending on whether data is encrypted or not

가변쿠키는 기밀성과 무결성을 만족하기 위하여 쿠키를 해쉬함수와 대칭키 암호화로 암호화하였다. 또한, 인증을 만족하기 위하여 생성된 쿠키를 데이터베이스에 저장된 쿠키값과 비교하고, 쿠키 생성 시에 필요한 값들을 페이지 이동 시마다 비교한다. 마지막으로 재생공격을 방지하기 위하여 데이터베이스에 저장된 쿠키값과 실시간으로 비교하고, 페이지 이동 시마다 쿠키 만료시간의 갱신 및 쿠키를 새로 생성하여 데이터베이스에 저장하였다.

가변쿠키는 암호화가 필요한 데이터의 존재 유무에 따라 VC1과 VC2로 나뉘게 된다. 먼저 회원의 아이디 혹은 고유정보, 현재 접속한 IP주소와 User-Agent, 현재 시간에 짧은 시간을 더한 쿠키의 만료시간, 그리고 서버 비밀키(Secret Key)를 구분자 | 로 연결하여 해쉬값 VC1을 생성한다. 암호화가 필요한 데이터가 없는 경우에 사용하는 VC1은 만들어진 해쉬값을 쿠키로 사용하여 사용자를 인증할 수 있다. 암호화가 필요한 데이터가 있는 경우에 사용되는 암호화키 k는 VC1의 뒤에 서버 비밀키를 붙여 생성한다. Data의 앞이나 뒤에 구분자와 함께 서버 비밀키를 추가한 후 k로 대칭키 암호화된 값과 VC1을 구분자 | 로 연결하여 생성된 결과가 암호화가 필요한 데이터가 있는 경우의 쿠키값 VC2가 된다. Alex의 쿠키와 같이 HMAC의 사용도 가능하지만, 구현이 더 간단한 SHA-256 해쉬함수만 사용하였다. 가변쿠키의 해쉬 입력값 길이가 충분히 길기 때문에 크래킹으로부터 안전하며, SHA-256이 HMAC-SHA256에 비하여 약 40% 속도가 빠르다[21].

3.1.2 가변쿠키의 유효성 확인

브라우저에 저장된 쿠키는 공격자에 의해 조작되어 해당 웹 애플리케이션의 사용자 인증에 사용될 수 있으므로 쿠키의 유효성 확인이 필요하다. 먼저 쿠키에 저장된 해쉬값을 가지고 데이터베이스의 사용자 테이블을 검색하여 같은 해쉬값이 존재하는지 파악한다. 만약 사용자 테이블에 같은 해쉬값이 존재한다면, 해당 사용자에 저장된 쿠키 만료시간이 현재 시간보다 앞인지 뒤인지 확인한다. 만약 현재 시간보다 앞이라면 쿠키 만료시간이 경과한 것이므로 가변쿠키가 더 이상 유효하지 않은 것으로 처리한다. 쿠키가 유효하지 않으면 사용자 브라우저에 저장된 쿠키를 삭제하며, 데이터베이스의 사용자 테이블에 저장된 해쉬값과 쿠키 만료시간도 모두 삭제한다.

쿠키 만료시간이 현재 시간보다 뒤라면 쿠키를 생성할 때와 같이 사용자 이름(Username), 현재 사용자 브라우저에서 사용 중인 IP주소와 User-Agent, 쿠키 만료시간, 서버 비밀키를 |로 붙여 해쉬값을 생성한다. 생성된 해쉬값과 데이터베이스에 저장된 해쉬값, 그리고 현재 사용자 브라우저에 저장된 해쉬값이 일치하는지 확인한다. 암호화된 데이터가 존재하지 않고, 3개의 해쉬값이 모두 일치한다면 쿠키는 유효한 것으로 판단한다. 그러나 암호화된 데이터가 존재한다면 암호화된 데이터 부분을 대칭키로 복호화한다. 복호화하였을 때, "|Secret Key" 즉, 데이터를 암호화 시 오른쪽에 붙였던 |와 서버 비밀키가 존재한다면 유효한 것으로 판단하며, 존재하지 않는다면 쿠키가 조작된 것으로 판단한다.

3.2 가변쿠키의 안전성

3.2.1 가변쿠키의 기밀성

가변쿠키를 생성하는 데 사용되는 서버 비밀키는 서버 내부에 하드코딩되어 비밀키로 활용되는 값이다. 그림 4에서 가변쿠키 VC1의 생성에 사용된 회원의 고유정보와 IP주소, User-Agent, 그리고 쿠키의 만료시간은 공격자가 알 수도 있는 정보이다. 따라서 공격자는 이 정보를 조합하여 해시 값을 생성하고, 이를 통해 크래킹을 시도할 수 있다.

하지만 VC에 서버 비밀키를 포함함으로써 서버 비밀키를 알 수 없는 공격자는 크래킹이 불가능하다. 암호화가 필요한 데이터가 있는 경우 공격자는 암호화된 Data값의 암호화 키값이 구분자 | 의 앞에 있는 해쉬값이라고 추측할 수도 있다. 해쉬값에 비밀키를 한 번 더 포함하여 Data의 암호화 키값을 예측할 수 없게 하였다. 안전한 해쉬함수는 일방향 암호의 특징을 가지고 있어 복호화할 수 없다[22]. 이에 따라 이 해쉬값과 서버 비밀키를 키로 한 Data의 대칭키 암호화 결과값도 키 값을 알 수 없으므로 가변쿠키의 기밀성이 보장된다.

3.2.2 가변쿠키의 무결성

가변쿠키의 구성 값을 알 수 없는 공격자는 해시함수의 충돌 저항성에 따라 임의의 값으로 동일한 쿠키를 생성하거나 인증할 수 없게 된다. 쿠키는 조작이 가능하므로 공격자가 인증을 위한 해쉬값을 변조한다면, 서버는 브라우저에 저장된 해쉬값과 데이터베이스에 저장된 해쉬값을 비교하여 일치 여부를 판단한다. 만약 일치하지 않다면 쿠키가 변조된 것으로 판단하여 인증할 수 없게 된다. 만약 일치하더라도 서버 측에서 현재 사용자가 브라우저에서 사용 중인 IP주소, User-Agent를 가져와서 다시 해쉬값을 생성하여 일치 여부를 확인한다.

Alex와 Lee의 쿠키는 Data 부분의 쿠키를 조작할 시, 시스템에 따라 부하를 야기하거나 Data가 변조될 위험이 있다. 가변쿠키는 완성된 쿠키값에 | 와 같은 구분자가 존재하는지 확인하여 쿠키가 변조되었는지 바로 확인할 수 있다. 또한, Data를 복호화할 때 서버 비밀키가 Data 앞이나 뒤에 붙여져 있기 때문에 이를 확인하여 복호화된 값을 검증할 수 있어, 쿠키 조작으로부터 안전성을 유지할 수 있다. 서버 비밀키를 “ANU13579” 라고 가정할 시, 공격자가 암호화된 Data 값을 조작하면, 시스템은 조작된 암호값을 키로 복호화한 후, Data의 앞이나 뒤에 | 를 포함한 “ANU13579”가 있는지 확인하여 검증한다. “ANU13579”가 있다면 올바른 Data 값으로 처리하고, 없다면 쿠키를 초기화시켜 공격으로부터 안전할 수 있다.

3.2.3 가변쿠키 기반의 로그인 인증

가변쿠키 기반의 전체적인 로그인 인증절차는 그림 5와 같다. 사용자가 아이디와 패스워드를 입력하면 해당 사용자 정보의 패스워드가 일치하는지 서버가 판단한다. 정보가 일치하면 쿠키를 생성하고, 생성된 해시 값과 쿠키의 만료시간을 데이터베이스에 저장한다. 가변쿠키에는 사용자를 식별하는 사용자 아이디가 있으므로, 쿠키의 해쉬값은 타 사용자와 겹칠 수 없는 고유값이다. 브라우저에 저장된 쿠키를 사용하여 데이터베이스 내 회원 테이블의 해쉬컬럼을 조회하고, 이를 통해 사용자 정보를 가져올 수 있다.

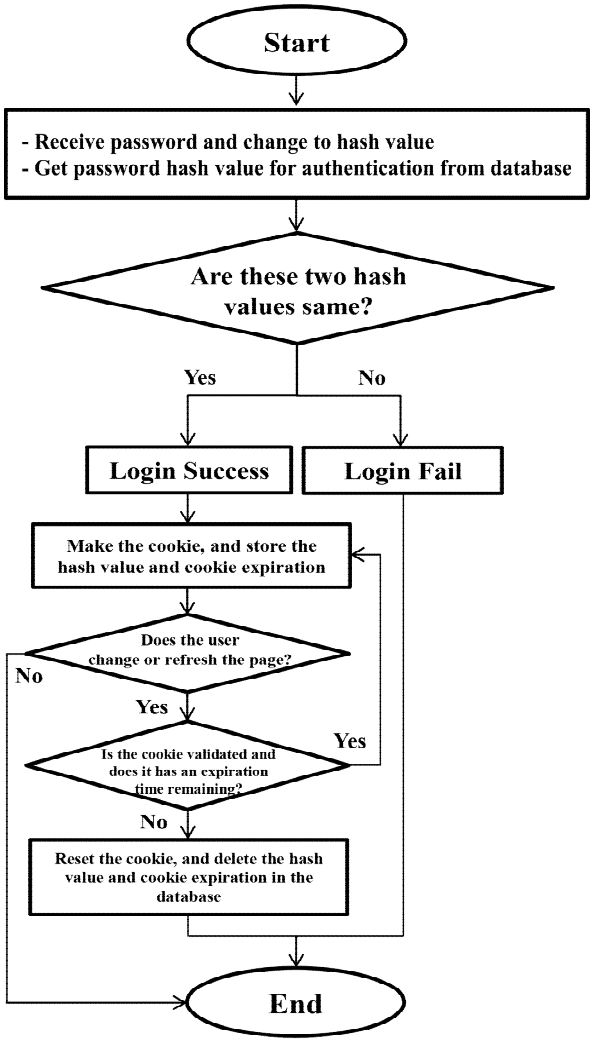


그림 5. 가변쿠키 기반의 로그인 인증 절차
Fig. 5. Login authentication procedure based on variable cookie

사용자가 페이지를 이동하거나 새로고침하면, 서버에서 쿠키의 유효성과 만료시간을 확인한다. 쿠키가 조작되지 않았고 만료시간이 남아 있다면, 브라우저와 데이터베이스에 저장된 해쉬값과 쿠키의 만료시간을 갱신한다. 만약 쿠키 검증에 실패하거나 사용자가 로그아웃 버튼을 누르면, 브라우저에 저장된 쿠키와 데이터베이스에 저장된 해쉬값 및 쿠키의 만료시간을 삭제한다. 데이터베이스에 해쉬값과 쿠키의 만료시간이 없으면 사용자가 가진 쿠키를 검증할 수 없기 때문에, 가변쿠키 방식에서는 쿠키를 조작하여 로그인을 시도하는 것이 불가능하다. 웹사이트 관리자는 데이터베이스에 저장된 사용자의 쿠키와 만료시간을 강제로 삭제하여 로그인된 사용자를 강제로 로그아웃할 수 있어 회원 인증을 강화할 수 있다.

3.2.4 가변쿠키의 재생공격방지

가변쿠키의 구성요소 중 쿠키의 해쉬값(VC1)과 만료시간은 사용자 로그인 시 데이터베이스에 저장된다. 쿠키의 만료시간은 사용자가 페이지를 이동하거나 새로고침 할 때마다 현재 시간에 짧은 시간을 추가하여 갱신되므로 쿠키도 지속적으로 변하게 된다. 쿠키의 재생공격은 공격자가 만료시간 내에 해당 쿠키를 가져와 자신의 브라우저에서 실행할 수 있어야 발생한다. 공격자가 짧은 쿠키 만료시간 내에 사용자의 쿠키를 확보하지 못하면, 재생공격은 불가능해 진다. 공격자가 해당 쿠키를 가져온다고 해도 쿠키에 포함된 사용자의 IP주소와 User-Agent가 동일하지 않으면, 쿠키 재생을 통한 사용자 인증이 불가능하다. 또한, 사용자가 페이지를 이동하거나 새로고침을 한다면 쿠키가 변하게 되어 이전에 탈취한 쿠키값으로 재생공격을 할 수 없게 된다.

해당 웹사이트에서 사용자가 한 페이지에 머무는 시간을 예측하여, 만료시간에 적용되는 짧은 시간을 정할 수 있다. 로그인한 사용자가 해당 쿠키의 만료시간 전에 자동으로 새로고침 시키거나, Javascript와 같은 프론트엔드 언어를 통하여 사용자 브라우저가 켜져 있음을 감지하고 만료시간을 갱신한다면, 페이지를 켜놓고 다른 일을 하다가 만료시간이 지나서 로그아웃 되는 것을 막을 수 있을 것이다. PHP.net에 따르면 세션 만료시간(session.gc_maxlifetime)의 기본값은 1440으로 24분이다[23]. 쿠키 만료시간을 세션의 만료시간보다 짧게 설정하면, 브라우저를 강제로 닫은 후의 인증 값 유지 측면에서 세션보다 보안적으로 더 안전한 효과를 얻을 수 있다.

3.2.5 가변쿠키와 기존 쿠키의 안전성 및 호환성 비교

본 논문에서 제안하는 가변쿠키는 안전한 쿠키체계 요소인 기밀성, 무결성, 인증, 그리고 재생공격방지를 모두 지원한다. 반면, Alex의 쿠키[5]는 기밀성과 재생공격방지를 만족하지 못하고, Lee의 쿠키[6]는 재생공격방지를 만족하지 못한다. Alex 쿠키의 SSL Session ID 값은 SSL이 지원하는 환경과 SSL Session ID 값을 가져올 수 있는 일부 서버 개발언어에서만 지원하므로 호환성에 문제가 있다. W3Techs의 통계에 따르면, PHP를 사용하는 76.4%의 웹사이트와 ASP를 사용하는 6.4%의 웹사이트는 Alex의 쿠키를 사용하지 못한다[24]. 또한, 여전히 HTTP를 사용하는 약 16.7%의 웹사이트에서도 Alex 쿠키를 사용하지 못한다[25]. 이에 따라 PHP로 개발된 Wordpress도 SSL Session ID를 제외하여 지원하고 있는 상태이다[18]. Wordpress는 오픈 소스이기 때문에 서버 비밀키가 노출되면 특정 사용자의 쿠키를 생성하여 인증할 수 있게 된다. 이는 보안적으로 위협이 된다. Lee의 쿠키와 본 논문에서 제안한 가변쿠키는 모든 환경에서 사용이 가능하다.

Ⅳ. 가변쿠키와 기존 쿠키의 성능비교

성능 비교를 위한 시험 코드는 PHP 7.3으로 구현하였으며, Windows Server 2012 R2, Apache 2.4.17, MySQL 8버전, Intel(R) Xeon(R) CPU E3-1231 v3 @ 3.40GHz (4 cores) CPU, 그리고 DDR3 16GB 메모리 환경에서 시험하였다.

4.1 쿠키의 생성, 로그인 인증 및 쿠키 검증 처리시간

표 2는 본 논문에서 제시한 가변쿠키(VC) 방식의 쿠키 생성시간과 로그인 인증시간, 페이지 이동 시의 쿠키 검증시간을 세션방식, 그리고 Alex[5]와 Lee[6]의 쿠키방식과 비교한 결과이다. 각 시험의 측정시간은 시험당 100회 측정하여 평균한 값이다. 세션은 데이터를 웹서버에 저장하므로 암호화하지 않고 데이터를 세션에 바로 삽입하였다. 또한, 세션이 지속적으로 생성되는 것을 방지하기 위하여 매 시험 완료 후 세션을 모두 삭제하였다. 쿠키방식의 해쉬 알고리즘은 SHA-256, 그리고 대칭키 알고리즘은 AES-256을 사용하였다.

표 2. 쿠키의 생성, 인증 및 검증시간 비교
Table 2. Comparison of generating, authentication and verification times (unit: ms)

Category	Session	Alex	Lee	VC
Generating cookie value	0.182	0.025	0.026	0.024
Login authentication	15.167	12.786	12.668	12.596
Cookie verification	15.218	12.724	12.659	21.435

4.1.1 쿠키값 생성시간

쿠키값 생성시간은 백엔드 코드에서 각 쿠키방식에 대한 쿠키값 생성시간의 차이를 확인하기 위하여 측정한다. 측정한 시간은 데이터베이스 사용 없이 클라이언트로 전송할 쿠키값을 임의의 값으로 생성하는 데 걸리는 시간이다. 이때 세션은 세션파일을 생성한 후 사용자 이름을 세션파일에 입력하는 데 걸리는 시간을 측정하였다. 임의의 데이터를 통한 쿠키값의 생성에는 세션이 0.182ms, Alex의 쿠키방식이 약 0.025ms, Lee의 쿠키방식이 약 0.026ms, 가변쿠키 방식이 약 0.024ms로 측정되었다. 세션은 세션파일을 생성하는 시간이 쿠키를 생성하는 시간보다 크게 나왔다. 3가지 쿠키 방식의 쿠키 생성시간은 매우 근소한 차이를 보이고 있다.

4.1.2 인증시간

로그인 인증시간은 데이터베이스를 사용하고, 사용자의 아이디와 패스워드를 입력 받아 로그인 인증 및 생성된 쿠키를 처리하는 시간이다. 쿠키의 해쉬값과 만료시간을 데이터베이스에 유일하게 저장하는 가변쿠키 방식과 타 쿠키와의 처리시간 차이를 비교하기 위하여 로그인 인증시간을 측정하였다. 세션의 경우에는 세션파일을 생성하고 로그인 인증시간 및 세션에 사용자 이름을 추가하는 시간을 측정하였다. 로그인 인증에서는 세션, Alex와 Lee의 쿠키, 그리고 가변쿠키가 각각 15.167ms, 12.786ms, 12.668ms, 12.596ms로 측정되었다. 로그인 인증에서는 실제 사용자가 로그인을 위하여 입력한 아이디와 패스워드를 데이터베이스에서 확인한 후, 확인된 사용자 정보에 기반하여 쿠키값을 생성한다. Alex와 Lee의 쿠키는 사용자의 접속기록을 데이터베이스에 저장할 때 IP주소와 User-Agent만 저장하지만, 가변쿠키에서는 쿠키 생성에 사용되는 해쉬값과 쿠키의 만료시간 변화를 감지하기 위하여 쿠키 만료시간도 함께 저장한다. Alex의 쿠키방식에서는 쿠키 생성을 위하여 HMAC-SHA256을 사용한 반면, 가변쿠키에서는 HMAC-SHA256에 비해 약 40% 빠른 SHA256을 사용하였다[21]. Lee의 쿠키방식에서는 큰 소수를 선택하고 해당 소수에 -1을 한 수의 서로 소의 개수를 찾는 등 복잡한 계산과정이 포함되어 있지만, 가변쿠키는 단순하며 계산이 필요 없다. 또한, 데이터베이스에 한 쿼리문으로 접속기록뿐만 아니라 해쉬값과 쿠키 만료시간을 함께 저장하더라도, 다른 쿠키 방식과 비교했을 때 시간 차이가 거의 발생하지 않았다.

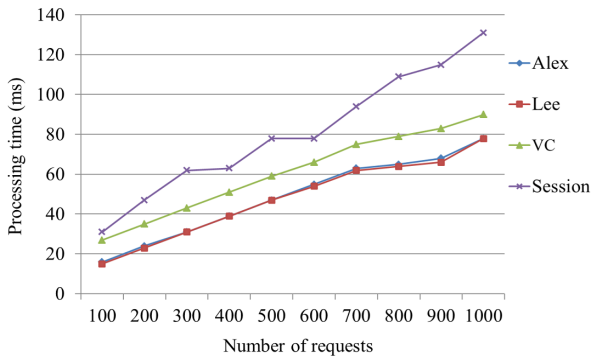
4.1.3 쿠키 검증시간

페이지 이동이나 새로고침 시의 쿠키검증 시간은 쿠키의 유효성을 확인하고 로그인된 사용자의 정보를 가져오는 데 걸리는 시간이다. 세션은 브라우저에 저장된 세션ID를 통하여 생성된 세션파일을 지정하고, 세션파일에 저장된 사용자 이름을 가져와서 로그인된 사용자의 정보를 가져오는 시간을 측정하였다. 세션과 Alex, Lee의 쿠키 방식에서의 검증시간은 각각 15.218ms, 12.724ms, 12.659ms 걸렸고, 가변쿠키 방식에서는 약 21.435ms로 측정되었다.

가변쿠키에서는 페이지 이동 시마다 쿠키를 생성 후 갱신된 해쉬값과 쿠키의 만료시간을 다시 데이터베이스에 저장하므로 약 9ms의 시간이 추가되었다. 가변쿠키에서 갱신된 쿠키를 데이터베이스에 저장하면 쿠키가 지속적으로 변경되어 재생공격을 차단할 수 있으며, 짧은 시간이 추가되어 갱신된 만료시간은 재생공격을 방지할 뿐만 아니라 쿠키를 세션보다 더 안전하게 만든다.

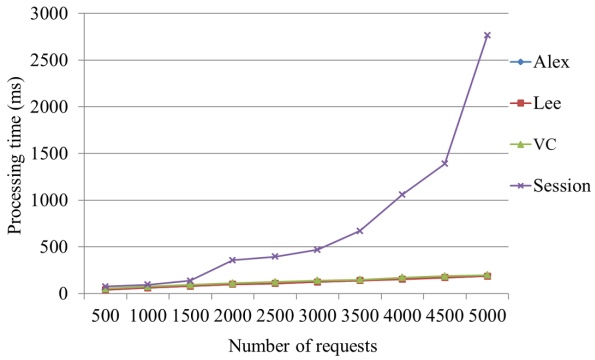
4.2 가변쿠키와 세션 및 기존 쿠키의 처리시간

그림 6은 가변쿠키(VC)와 세션 및 기존 쿠키에 대하여 쿠키를 생성하고 페이지를 이동할 때의 요청에 따른 처리시간을 비교하여 나타낸다. 부하에 따른 처리시간을 비교하기 위하여 웹서버 성능검사 도구인 ApacheBench[26]를 사용하였다. 동시요청 개수는 100으로 지정하고 전체 요청은 각각 1000, 5000으로 시험하였다.



(a) 1000회 요청

(a) 1000 requests



(b) 5000회 요청

(b) 5000 requests

그림 6. 가변쿠키와 세션 및 기존 쿠키의 처리시간 비교

Fig. 6. Comparison of processing times between variable cookie, session, and existing cookies

동시요청 100회, 전체요청 1000회의 시험에서 Alex와 Lee의 쿠키는 처리시간 차이가 거의 없었고, 가변쿠키는 Alex와 Lee의 쿠키 처리시간에 비하여 약 12ms 정도의 차이를 유지하였다. 세션의 경우 처음 100회 요청 시에는 가변쿠키에 비해 약 4ms 정도 처리시간이 증가한 것으로 측정되었다. 그러나 800회의 요청부터 차이가 발생하기 시작하여, 마지막 1000회 요청에서의 세션 처리시간은 131ms로 측정되었다. 이는 가변쿠키의 90ms에 비해 세션의 처리시간이 약 45% 증가한 것이다. 동시요청 100회, 전체요청 5000회의 시험에서도 가변쿠키는 Alex와 Lee의 쿠키에 비하여 1000회 시험과 같이 약 12ms 정도의 차이를 유지하였다. 그러나 세션의 경우 처음 500회 요청에서는 가변쿠키에 비해 약 22ms 정도의 처리시간이 증가하였으며, 1500회 요청부터 세션의 처리시간이 급격히 증가하고 있다. 마지막 5000회 요청에서는 2766ms로 측정되었으며, 이는 가변쿠키의 처리시간 199ms에 비하여 약 1289%의 처리시간이 증가하였다.

세션의 경우 앞서 세션파일 1개를 가지고 쿠키 생성 및 인증속도를 다른 쿠키방식과 비교하였을 때에는 다른 쿠키 방식들과 시간차이가 크게 나지 않았다. 그러나 부하가 많이 발생할수록 세션의 처리시간이 기하급수적으로 증가하는 것을 시험으로 확인하였다. 그러므로 동시 사용자가 많을수록 세션은 서버에 부하를 주어 처리속도가 쿠키에 비해 매우 증가하므로, 다수 사용자가 이용하는 웹 애플리케이션에는 세션 방식을 사용하는 것이 장기적으로는 적합하지 않다고 볼 수 있다. 기밀성과 재생공격 방지를 제공하지 못하는 Alex의 쿠키와 재생공격방지를 만족하지 못하는 Lee의 쿠키는, 페이지 이동시에 쿠키를 데이터베이스에 갱신하지 않으므로 요청에 따른 처리시간이 거의 비슷하였다. 기밀성과 재생공격방지를 모두 제공하는 가변쿠키는 페이지 이동시마다 사용자 테이블에 쿠키값과 쿠키 만료시간을 갱신하므로, Alex와 Lee의 쿠키에 비하여 최종요청에서 처리시간이 약 12ms 증가하였다.

V. 결 론

기존의 안전쿠키는 쿠키 생성에 사용자의 이름,

쿠키의 만료시간 등 고정값을 이용하므로, 한번 생성된 쿠키는 변경되지 않고 고정값으로 저장된다. 본 논문에서 제안하는 가변쿠키는 사용자의 이름과 접속 IP주소, User-Agent, 새롭게 갱신되는 쿠키의 만료시간, 서버 비밀키를 결합하여 생성되는 해쉬값을 이용한다. 이는 페이지 이동 시에 쿠키값을 지속적으로 변하게 하여 보안성과 가용성을 강화하였다.

분석 결과, 가변쿠키는 기존 쿠키와 비교하여 보안 측면에서 기밀성, 무결성, 재생공격 방지를 모두 만족했으며, 데이터베이스를 통해 갱신된 쿠키값을 확인함으로써 단 하나의 쿠키값만 인증이 가능하도록 설계되었다. 또한, HTTPS와 SSL Session ID 변수 환경에 제한되지 않고 모든 환경에서 사용이 가능하므로 호환성 측면에서도 뛰어나다. 성능 측면에서는 로그인 인증과 쿠키 생성에서 기존 쿠키와 유사한 처리시간을 보였으며, 페이지 이동 시 약 9ms의 추가 시간이 발생하였다. 100회의 동시 요청에서 가변쿠키의 처리시간은 기존 쿠키에 비해 약 12ms 증가하였다. 이는 재생공격 방지를 위한 쿠키 갱신 과정에서 데이터베이스 갱신에 필연적으로 요구되는 처리시간이다. 전체 5000회 요청 시험에서 가변쿠키는 세션 대비 1289% 향상된 처리 성능을 보여주었다.

공격자가 쿠키생성에 관한 소스코드를 볼 수 있고 데이터베이스까지 제어할 수 있다면, 기존 쿠키 및 가변쿠키 방식에서 공격자는 쿠키값을 생성하여 사용자 인증 및 데이터를 복호화할 수 있게 된다. 공격자가 쿠키 생성방법을 알고, 데이터베이스 제어까지 가능한 경우에도 사용자 인증 및 데이터를 복호화할 수 없는 방법을 추후 연구할 계획이다.

References

[1] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "RFC2616: Hypertext Transfer Protocol — HTTP/1.1", RFC Editor, Jun. 1999. https://doi.org/10.17487/RFC2616.

[2] D. Kristol and L. Montulli, "RFC2109: HTTP State Management Mechanism", RFC Editor, Feb. 1997. https://doi.org/10.17487/RFC2109.

[3] W3Techs, https://w3techs.com/technologies/details/ce-sessioncookies. [accessed: Mar. 20, 2024]

[4] J. S. Won, J. Park, and J.-G. Son, "A Defense Mechanism Based on Session Status against Cookie Replay Attack in Web Applications", KIPS Transactions on Computer and Communication Systems, Vol. 4, No. 1, pp. 31-36, Jan. 2015. https://doi.org/10.3745/KTCCS.2015.4.1.31.

[5] A. X. Liu, J. M. Kovacs, and M. G. Gouda, "A secure cookie scheme", Computer Networks, Vol. 56, No. 6, pp. 1723-1730, Jun. 2012. https://doi.org/10.1016/j.comnet.2012.01.013.

[6] W.-B. Lee, H.-B. Chen, S. S. Chang, and T.-H. Chen, "Secure and efficient protection for HTTP cookies with self-verification", International Journal of Communication Systems, Vol. 32, No. 2, Jan. 2019. https://doi.org/10.1002/dac.3857.

[7] ISO/IEC 27001:2022, "Information security, cybersecurity and privacy protection — Information security management systems — Requirements", 2022.

[8] ITU-T, "Security Architecture for Open Systems Interconnection for CCITT Application", X.800, 1991.

[9] National Institute of Standards and Technology(NIST), "An Introduction to Information Security", NIST SP 800-12 Rev. 1, Jun. 2017. https://doi.org/10.6028/NIST.SP.800-12r1.

[10] OWASP, https://owasp.org/Top10/A01_2021-Broken_Access_Control/. [accessed: Mar. 20, 2024]

[11] K. Fu, E. Sit, K. Smith, and N. Feamster, "Dos and don'ts of client authentication on the web", Proc. of the 10th conference on USENIX Security Symposium, Washington, D.C., Vol. 10, pp. 19, Aug. 2001. https://dl.acm.org/doi/abs/10.5555/1267612.1267631.

[12] Tutorials Point, https://www.tutorialspoint.com/What-is-the-maximum-size-of-a-web-browser-s-cookies-value. [accessed: Mar. 20, 2024]

[13] PHP.net, https://www.php.net/manual/en/ini.core.php. [accessed: Mar. 20, 2024]

[14] J. S. Park and R. S. Sandhu, "Secure cookies on the Web", IEEE Internet Computing, Vol. 4, No. 1, pp. 36-44, Jan. 2000. https://doi.org/10.1109/4236.865085.

[15] ISO/IEC 27000:2014, "Information technology — Security techniques — Information security management systems — Overview and vocabulary", 2014.

[16] S. Barbato, S. Dorigotti, and T. Fossati, "SCS: KoanLogic's Secure Cookie Sessions for HTTP", RFC 6896, 2013. https://doi.org/10.17487/RFC6896.

[17] W3Techs, https://w3techs.com/technologies/details/cm-wordpress. [accessed: Mar. 20, 2024]

[18] Wordpress, https://developer.wordpress.org/reference/functions/wp_generate_auth_cookie/. [accessed: Mar. 20, 2024]

[19] PayPal Inc, "Super-cookie identification for stolen cookie detection", US Patent No. US20240037279A1, Feb. 2024.

[20] T. Berners-Lee, R. Fielding, and H. Frystyk, "RFC1945: Hypertext Transfer Protocol — HTTP/1.0", 1996. https://www.ietf.org/rfc/rfc1945.txt.

[21] Hash Algorithm Performance, https://www.jacksondunstan.com/articles/3206. [accessed: Mar. 20, 2024]

[22] R. C. Merkle, "Secrecy, Authentication and Public Key Systems", Ph.D. dissertation, Dept. of Electrical Engineering, Stanford Univ., 1979. https://dl.acm.org/doi/10.5555/909000.

[23] PHP.net, https://www.php.net/manual/en/session.configuration.php. [accessed: Mar. 20, 2024]

[24] W3Techs, https://w3techs.com/technologies/overview/programming_language. [accessed: Mar. 20, 2024]

[25] W3Techs, https://w3techs.com/technologies/details/ce-httpsdefault. [accessed: Mar. 20, 2024]

[26] Apache, https://httpd.apache.org/docs/2.4/programs/ab.html. [accessed: Mar. 20, 2024]

저자소개

정 진 호 (Jin-Ho Jeong)

2019년 2월 : 안동대학교 컴퓨터공학과(공학사)
2021년 2월 : 안동대학교 컴퓨터공학과(공학석사)
2024년 8월 : 안동대학교 컴퓨터공학과(공학박사)
2015년 7월 ~ 현재 : 디제이패밀리 대표
관심분야 : 웹보안

차 영 욱 (Young-Wook Cha)

1987년 2월 : 경북대학교 전자공학과(공학사)
1992년 2월 : 충남대학교 전자통계학과(공학석사)
1998년 8월 : 경북대학교 컴퓨터공학과(공학박사)
1987년 3월 ~ 1999년 2월 : 한국전자통신연구원 선임연구원
2025년 6월 ~ 현재 : 국립경국대학교 컴퓨터공학과 교수
관심분야 : 정보보안, 블록체인, 개방형통신망