https://doi.org/10.7840/kics.2025.50.11.1788

TOTP 기반 SSH 다요소 인증 시스템 개발

조 진 용◆, 김 동 균*, 조 부 승*

Development of TOTP-Based SSH Mutli-Factor Authentication System

Jinyong Jo◆, Dongkyun Kim*, Buseung Cho*

요 약

본 연구는 SSH 환경에서 단일 요소 인증의 보안 취약점을 해결하기 위해 TOTP 기반 다요소 인증 시스템을 설계하고 구현하였다. 전 세계 SSH 서버의 65% 이상이 패스워드 인증만을 허용하고 있어 무차별 대입 공격과 중간자 공격에 취약한 상황이다. OpenSSH 기반의 SSH 다요소 인증 방식은 패스워드와 TOTP를 하나의 문자열로 처리하여 인증 실패의 원인을 파악하기 어렵고 다양한 인증 요소를 조합하는 데 한계가 있다. 제안한 시스템은 OpenLDAP 확장 연산과 SSH 서버의 PAM 모듈을 활용하여 1차 인증과 2차 인증을 분리하는 모듈식 구조로 개발되었다. 이를 통해 각 인증 요소의 독립적 검증이 가능하며, 인증 오류의 원인을 명확히 구분할 수 있고, 다양한 인증 요소의 조합이 가능하다. 또한 TOTP 공유키를 OpenLDAP 서버에서 중앙 집중 관리하여 클라이언트 측 보안 위험을 최소화할 수 있다. 기능 및 성능 평가 결과, OATH 표준을 준수하여 범용 SSH 클라이언트와 TOTP 인증 앱과 호환되며, 동시 사용자 20명 환경에서 100% 응답률과 초당 12.6건의 처리 성능을 확인하였다.

키워드 : 다요소 인증, OpenLDAP, SSH, TOTP, PAM
Key Words : Multi-factor authentication, OpenLDAP, SSH, TOTP, PAM

ABSTRACT

This study presents the implementation of a TOTP-based multi-factor authentication system to address the security vulnerabilities of single-factor SSH authentication. More than 65% of SSH servers worldwide rely solely on password authentication, making them vulnerable to brute-force or man-in-the-middle attacks. Existing OpenLDAP-based SSH multi-factor authentication systems using TOTP process a password and a TOTP passcode as a single concatenated string, which hinders identifying the cause of authentication failures and limits the combination of various authentication factors. The proposed system separates primary and secondary authentication by implementing an OpenLDAP extended operation and SSH server PAM modules independently. The implementation enables independent verification of each authentication factor, clear identification of failures, and flexible combination of diverse authentication methods. To reduce security risks from client-side key breaches, TOTP shared keys are centrally managed in the OpenLDAP server. Evaluation results demonstrate OATH compliance and broad compatibility with SSH clients and TOTP applications. The system achieved a 100% response rate and processed 12.6 authentication requests per second in a concurrent 20-user environment.

※ 본 연구는 한국과학기술정보연구원의 지원으로 수행되었습니다(K25L5M1C1).
◆ First Author : Korea Institute of Science and Technology Information, jiny92@kisti.re.kr, 정회원
* Korea Institute of Science and Technology Information, {mirr, bscho}@kisti.re.kr, 정회원
논문번호 : 202506-131-B-RU, Received June 4, 2025; Revised June 23, 2025; Accepted June 23, 2025

Ⅰ. 서 론

SSH(Secure Shell) 이용 환경에서 단일 요소 인증(Single-factor Authentication, SFA)은 서버 시스템에 심각한 보안 위협을 야기할 수 있다. 관련 연구[1]에 따르면, 전 세계 2,000만 개 이상의 SSH 서버 중 65% 이상이 패스워드 인증만 허용하고 있는 것으로 조사되었다. 단일 요소 인증 방식은 패스워드가 중간자 공격이나 무차별 대입 공격에 의해 유출될 경우 서버 시스템 전체가 보안 위험에 노출되어 단일 보안 방어선(Single Point of Security)으로 인한 취약점을 가진다. 따라서 서버 시스템의 보안 위협을 낮추기 위해서는 SSH 환경에서도 다요소 인증(Multi-factor Authentication, MFA)의 적용이 권장된다.

시간 기반 일회용 패스워드(TOTP: Time-based One-Time Password) 방식은 구현이 용이하여 다양한 인증 시스템에서 활용되고 있다[2]. TOTP는 시간을 기반으로 일회용 패스코드를 주기적으로 생성한다. 패스워드 기반의 MFA 상황에서 공격자가 사용자의 패스워드를 획득하더라도 TOTP 패스코드 없이는 시스템에 접근할 수 없으므로 보안성을 높일 수 있다. 실제로 TOTP를 포함한 MFA는 사이버 공격 위험을 감소시키는 효과적인 방어 수단임이 입증되고 있다[3],[4].

OpenLDAP[5]은 디렉토리 서비스를 제공하는 오픈소스 소프트웨어로 사용자 인증 정보를 중앙 집중적으로 관리할 수 있게 한다. SSH 로그인을 위한 사용자 계정 정보를 OpenLDAP에 저장하고 통합 관리하면 보안 적용의 일관성, 사용자 편의성, 관리 효율성 등을 높일 수 있다. 특히 다수의 사용자가 여러 SSH 서버에 접근해야 하는 환경에서는 각 서버마다 개별적으로 계정을 생성하고 관리하는 것보다 OpenLDAP을 활용한 계정의 중앙 관리를 통해 운영 효율성을 높일 수 있다.

SSH 환경에서 OpenLDAP을 통한 인증 방식은 다수의 장점에도 불구하고 다음과 같은 문제점이 있다. 먼저, OpenLDAP 기반의 SSH 인증 환경에서 TOTP를 활용한 MFA를 구현하기 위해서는 사용자 패스워드와 OTP 패스코드를 하나의 문자열로 결합하는 방식을 취해야 한다[6]. 예를 들어, 사용자 패스워드 "userpass"와 OTP 패스코드 "123456"을 "userpass123456"과 같이 결합하여 입력해야 한다.

패스워드와 패스코드를 결합해 사용하는 방식은 SSH 서버의 PAM(Pluggable Authentication Modules) 모듈이나 OpenLDAP 소스 코드를 수정 없이 활용할 수 있다는 장점이 있다. 하지만, MFA 실패 시 어떤 요소(패스워드 또는 패스코드)가 잘못 입력되었는지 확인하기 힘들다. 또한 공개키 등 타 인증 요소와의 결합이 어려워 고정된 요소로만 조합이 가능하므로 유연성이 결여된다.

인증 요소 선택의 유연성을 높이기 위해 패스워드와 패스코드에 대한 검증을 각각 OpenLDAP과 SSH 서버로 분리하여 수행하는 방식이 활용될 수 있다[6]. 그러나 SSH 서버가 TOTP 패스코드를 검증하기 위해서는 TOTP 공유키가 SSH 서버에 저장되어야 하므로 키 유출로 인한 보안상 취약점이 발생할 수 있다.

본 논문은 OpenLDAP에서 패스워드와 TOTP 패스코드를 모두 검증하되, 사용자가 패스워드와 패스코드를 분리하여 입력할 수 있고 각각을 독립적으로 검증하는 OpenLDAP 기반의 MFA 시스템을 제안한다. 제안하는 시스템은 SSH 인증 과정에서 사용자가 인증 실패의 원인을 직관적으로 파악할 수 있고, 다양한 SSH 인증 요소와의 조합이 가능하므로 기존 OpenLDAP 기반의 MFA가 갖는 문제점을 해결한다.

본 연구의 기여점은 다음과 같다. 첫째, 사용자 패스워드와 TOTP 패스코드를 결합해 처리하는 기존 OpenLDAP의 한계를 극복하고, TOTP 공유키의 중앙 관리를 통해 SSH 서버에서 발생할 수 있는 키 노출의 위험을 해소하였다. 둘째, SSH 서버에서 모듈화된 PAM 인증 체계를 구현하여 공개키, 패스워드, TOTP 등 다양한 인증 방식을 유연하게 조합할 수 있는 MFA 프레임워크를 설계하였다. 셋째, OATH 표준과 keyboard-interactive 방식을 준수하여 기존 SSH 클라이언트나 TOTP 인증 앱과의 호환성을 확보하였다.

본 논문의 구성은 다음과 같다. 2장에서는 SSH 인증 메커니즘, PAM, TOTP 등 배경 기술을 소개한다. 3장에서는 관련 연구를 검토하고, 4장에서는 제안하는 MFA 시스템의 설계 내용을 설명한다. 5장에서는 시스템 환경 구성 및 구현 세부 사항을 다루며, 6장에서는 성능 및 기능 평가 결과를 제시하고, 7장에서 결론을 맺는다.

Ⅱ. 배 경

2.1 SSH 인증 메커니즘 및 PAM

SSH는 네트워크를 통해 원격 시스템에 접속할 수 있는 암호화된 프로토콜로서, RFC(Request For Comments) 4252에 정의된 인증 방식들을 지원한다. 패스워드 인증은 사용자 ID와 패스워드를 활용하며, 공개키 인증은 Ed25519나 ECDSA 같은 비대칭 암호화를 사용하여 개인키의 소유를 증명한다. GSSAPI (Generic Security Services API) 방식은 Kerberos 등

외부 인증 시스템과 연동해 사용자를 인증한다.

추가적으로 SSH는 유연한 인증을 위한 상호작용 메커니즘을 제공한다. Keyboard-interactive 방식은 SSH 서버가 동적으로 프롬프트를 생성하여 사용자와 상호작용할 수 있게 하며 RFC 4256에 정의되어 있다. 주로 challenge-response 형태의 MFA에 사용된다. 각 인증 방식은 환경 설정 파일(예, sshd_config)을 통해 활성화하거나 비활성화할 수 있다.

PAM은 운영체제 계층에서 인증, 계정 관리, 세션 관리, 패스워드 관리 등의 기능을 모듈화하여 제공하는 보안 인증 프레임워크이다. SSH의 인증 방식들은 PAM과 연동될 수 있으며, 특히 keyboard-interactive 인증 방식은 PAM의 동적 프롬프트를 활용하여 MFA의 구현에 적합하다. PAM은 여러 인증 모듈을 순차적으로 실행하는 스택 구조로 동작하며, 환경 설정 파일에 등록된 인증 처리 절차에 따라 각 모듈이 순서대로 실행된다. 제어 플래그를 통해 각 모듈의 성공과 실패에 따른 인증 흐름을 제어할 수 있다.

표 1은 PAM에서 사용할 수 있는 제어 플래그를 보여준다. PAM의 설정은 [service-type] [flag] [module] [options]의 형태로 사용된다. PAM 제어 플래그는 각 인증 모듈의 성공 또는 실패 여부가 전체 인증 과정에 미치는 영향을 결정한다. required 플래그는 해당 모듈이 실패해도 이후 인증 모듈을 순차적으로 실행하지만, 최종 인증은 실패로 처리한다. requisite 플래그는 required와 유사하나 인증 실패 시 즉시 전체 인증을 중단한다는 점에서 차이가 있다. sufficient 플래그는 해당 PAM 모듈이 성공하면 즉시 전체 인증을 성공으로 처리한다. optional 플래그는 해당 모듈의 결과가 최종 인증의 성공 또는 실패에 영향을 주지 않으며, 주로 로깅이나 세션 설정 등의 보조적인 기능을 수행하는 데 사용된다.

표 1. 주요 PAM 제어 플래그
Table 1. Major PAM control flags

Flags	On success	On failure	Description
required	continue	continue yet final failure	module that must succeed
requisite	continue	immediately return failure	terminates on failure
sufficient	immediately return success	continue	skip remaining modules on success
optional	continue	continue	not affect final result

각 플래그를 [success=2 default=ignore]와 같이 고급 제어 구문으로 대체함으로써 보다 복잡한 조건을 처리할 수 있다. 예시의 success=2는 해당 모듈이 성공하면 다음 2개의 모듈을 건너뛰고 계속 진행함을 의미한다. default=ignore는 성공을 제외한 모든 결과(실패, 오류 등)에 대해서는 무시하고 다음 모듈을 실행한다는 의미이다.

2.2 TOTP와 HOTP

일회용 패스코드는 한 번만 사용할 수 있는 임시 패스워드로서, 사용자 패스워드와는 달리 시간이나 카운터에 따라 동적으로 생성되므로, 재사용 공격(Replay attack)이나 패스워드 탈취로 인한 보안 위협을 효과적으로 방어할 수 있다.

HOTP(HMAC-based One-Time Password)는 카운터 기반의 일회용 패스코드 알고리즘[7]이다. 공유키(K)와 카운터(C) 값을 HMAC(Hash-based Message Authentication Code) 함수에 입력하여 패스코드 $O(K,C)$를 생성한다. d는 OTP 패스코드의 자릿수 이다. 인증 장치(또는 인증 토큰)과 서버가 동일한 카운터 값을 유지해야 한다.

$$O(K,C) = \mathrm{Truncate}(\mathrm{HMAC}(K,C)) \bmod 10^{d} \quad (1)$$

TOTP는 시간 기반의 일회용 패스코드 알고리즘[8]이다. HOTP에서 사용하는 카운터를 대신해 현재 시간을 기준으로 패스코드를 생성한다. 유닉스 시간을 정의된 시간 간격(일반적으로 30초)으로 나누어 시간 카운터(T)를 계산한 후, 공유키(K)와 T를 사용하여 패스코드 $O(K,C)$를 생성한다.

패스코드의 생성과 관리의 보안성을 보장하기 위해서는 다음과 같은 다수의 보안 권고사항을 준수해야 한다[9].

- 패스코드 무작위성(S1): 패스코드는 암호학적으로 안전한 의사난수 생성기를 사용해야 한다. 예측 가능한 패턴이나 취약한 난수 생성기로 생성된 패스코드는 공격자가 악용할 수 있어 보안상 위험하다.
- 패스코드 길이(S2): 패스코드는 최소 6자리 이상[10]이어야 무차별 대입 공격의 성공 확률을 줄일 수 있다. 짧은 길이의 패스코드는 수분 내에 해독될 수 있어 OTP 인증 프로토콜의 보안 이점을 무효화할 수 있다.
- 재시도 횟수 제한(S3): 패스코드 검증을 위한 최대 시도 횟수는 제한[10]되어야 한다. 최대 시도 횟수에 도달하면 서버는 무차별 대입 공격을 방어하기 위해

사용자 계정을 비활성화해야 한다.

- 일회성 사용(S4): 재사용 공격을 방지하기 위해 생성된 패스코드는 하나의 인증 세션에 대해서만 유효해야 한다. 동일한 패스코드를 반복적으로 사용하도록 허용하는 것은 OTP 사용의 이점을 무력화할 수 있다.
- 유효기간(S5): 패스코드는 제한된 시간 동안만 유효[7]해야 한다. 긴 유효기간은 무차별 대입 공격이나 중간자 공격에 취약할 수 있다. 패스코드의 유효기간은 시스템의 보안 수준과 사용자 편의성에 따라 결정되며 만료된 패스코드에 대해서는 반드시 인증을 거부해야 한다.
- 갱신주기(S6): 패스코드의 갱신주기는 새로운 패스코드가 생성되는 주기를 의미한다. TOTP의 경우 보안성과 사용성을 고려하여 30초를 권장[7]하고 있다. 권장 시간은 네트워크의 지연 문제를 해소하고 공격 윈도우를 최소화하기 위한 값으로 제시되었다.

2.3 OpenLDAP의 MFA 처리 방식

OpenLDAP은 인증 요소로 TOTP와 HOTP를 지원한다. 두 방식은 서로 다른 오버레이 모듈로 구현되었다. 오버레이 모듈은 플러그인 형태의 OpenLDAP 구성 요소이다. LDAP 요청을 가로채어 플러그인에 정의된 로직을 처리한 후 백엔드 데이터베이스로 요청을 전달하거나, 백엔드 데이터베이스가 전달한 응답을 수정하여 클라이언트에게 반환하는 역할을 한다.

OpenLDAP의 slapo-totp 모듈은 TOTP를 구현하며 두 가지 모드를 제공한다: (1) TOTP 전용 모드에서는 패스워드 없이 TOTP 코드만으로 사용자를 인증한다. 이 방식은 SSH 공개키 인증과 결합하여 2차 인증을 구성할 때 유용하다. (2) 결합 모드에서는 사용자 패스워드와 TOTP 패스코드를 하나의 문자열로 결합하여 MFA를 처리한다. slapo-totp는 SHA-1, SHA-256, SHA-512 해시 알고리즘을 지원하며 Google Authenticator 등 TOTP 인증 앱과 호환된다.

slapo-otp 모듈은 OATH 표준을 구현하여 HOTP와 TOTP를 모두 지원하지만, 현재는 패스워드와 OTP 패스코드를 하나의 문자열로 결합하여 사용하는 방식만 지원한다.

slapo-otp와 slapo-totp의 구현은 LDAP 바인드 프로토콜의 특성에 영향을 받는다. LDAP 바인드 프로토콜은 클라이언트 인증을 수행하는 메커니즘으로 anonymous, simple, SASL(Simple Authentication and Security Layer) 방식을 지원한다. Anonymous는 바인드 없이 요청을 보내는 경우이고, Simple은 클라이언트의 DN과 패스워드로 인증하는 방식이며, SASL은 확장 가능한 프레임워크로서 다양한 인증 요소를 지원하지만 구현의 복잡성과 SSH 클라이언트와의 호환성에 문제가 있다.

LDAP 바인드 프로토콜은 단일 요청, 단일 응답 구조로 설계되어 있어 MFA 구현 시 다음과 같은 제약사항이 존재한다: (1) 단일 패스워드 필드만 제공하므로 패스워드와 OTP 패스코드를 별도의 필드로 분리하여 입력할 수 없다. (2) 한 번의 바인드로 모든 인증을 완료해야 하므로 일반적인 다요소 간 상호작용이 불가능하다. 따라서 OpenLDAP에서 2FA를 구현할 때는 패스워드와 OTP 패스코드를 하나의 문자열로 결합하여 전송하는 방식을 사용해야 한다.

OpenLDAP의 slapo-totp는 userPassword 속성에 저장된 스키마 유형(예: {TOTP1}, {TOTP1ANDPW} 등)에 따라 처리 방식을 결정한다. {TOTP1}과 같이 TOTP 전용 모드인 경우, 입력된 문자열 전체를 TOTP 패스코드로 인식한다. 결합 모드({TOTP1ANDPW}) 인 경우에는 사용자 패스워드와 TOTP 패스코드가 하나의 문자열로 입력되며, 전체 문자열 중 마지막 6자리를 TOTP 패스코드로, 나머지를 패스워드로 분리하여 처리한다.

slapo-otp는 바인드 가로채기(Bind intercept) 방식을 통해 MFA를 처리한다. 예를 들어, 사용자의 LDAP 개체에 oathTOTPUser 또는 oathHOTPUser 객체가 설정된 경우, 입력된 문자열의 마지막 6자리 또는 8자리를 OTP 패스코드로 추출하고 나머지 부분을 패스워드로 간주하여 각 요소를 검증한다.

2.4 PAM 기반 공개 소스 TOTP 인증 모듈

본 절에서는 리눅스 상에서 OATH 표준에 따라 동작하고 TOTP와 HOTP의 처리를 지원하는 PAM 모듈을 소개한다.

OATH Toolkit은 OATH 표준의 참조 구현으로, libpam-oath PAM 모듈을 2차 인증에 사용할 수 있다. LGPL(Lesser GPL) 2.1 라이센스를 따르며, SSH 서버의 로컬 파일에 모든 사용자의 TOTP 공유키를 저장하여 관리한다. SSH 서버에 사용자가 추가될 때마다 TOTP 공유키를 수동으로 등록해야 하며, 등록된 공유키를 사용자에게 전달해야 하므로 관리 부담이 크다는 단점이 있다. 또한 윈도우 크기, 자릿수 등 다수의 매개변수를 명시적으로 설정해야 하므로 PAM 설정이 복잡하다. SHA1, SHA256, SHA512 알고리즘을 지원하며, 6-8자리 패스코드 길이를 설정할 수 있다.

Google Authenticator PAM 모듈은 Apache 2.0 라

이센스를 따르는 리눅스용 TOTP 인증 솔루션이다. 사용자가 PAM 모듈을 설치해야 하지만, 설치와 설정 과정이 간단하다는 장점이 있다. 특히 QR 코드를 통해 스마트폰에 공유키를 등록할 수 있어 사용자 편의성이 높다. 사용자 공유키가 홈 디렉토리에 평문으로 저장되며, NFS(Network File System) 환경에서는 접근 권한 문제가 발생할 수 있다. NFS는 root 권한을 제한하므로, PAM 모듈이 root 권한으로 실행될 때 사용자 ID와의 매핑 오류로 인해 공유키를 저장한 파일 접근이 불가능할 수 있다.

두 솔루션 모두 SSH 서버에서 공유키를 관리해야 하는 구조적 한계로 인해 공통된 보안 취약점을 갖는다. 첫째, SSH 서버는 다수의 일반 사용자들이 접근하는 환경이므로 동일 서버 내 일부 사용자의 악의적 행위, 공유 디렉토리 오설정, 또는 수평적 권한 상승 공격 등으로 인해 다른 사용자의 공유키가 노출될 위험이 있다. 둘째, PAM 모듈이 root 권한으로 실행되면서 공유키 파일에 접근하므로, 로컬 권한 상승 취약점이 존재하면 공격자가 모든 사용자의 공유키를 탈취할 수 있다. 셋째, 관리해야 하는 공유키의 수가 늘어나면 공유키의 주기적 갱신, 폐기된 키의 안전한 삭제, 키의 무결성 검증 등 키 생명주기 관리가 어려워지는 문제가 발생한다.

Ⅲ. 관련 연구

본 연구와 직접적으로 관련된 연구를 찾기 어려워 본 장에서는 2차 인증의 활용 사례와 1차 인증 방식 변화에 따른 2차 인증 요구사항을 다룬 논문들을 중심으로 소개한다.

PSC(Pittsburgh Supercomputing Center)는 Bridges 슈퍼컴을 위한 Linux PAM 기반의 SSH MFA 시스템을 구현하였다[11]. 해당 SSH MFA 시스템은 상용 Duo MFA 서비스와 통합되어 특정 사용자 그룹에게만 선택적으로 MFA를 적용할 수 있으며, 범용 SSH 클라이언트와의 호환성을 갖는다. 또한, Duo에서 제공하는 pam_duo 모듈을 수정하여 연합 신원 관리(Federated Identity Management)[12]를 지원하도록 개발되었다. 공개키, 패스워드, GSI(Grid Security Infrastructure) 등의 인증 방식과 조합이 가능하다.

그러나 Duo 클라우드 서비스에 의존해 공유키 등 인증 정보를 관리하므로 데이터 주권 문제가 발생할 수 있으며, 네트워크 장애 시 사용자 인증이 불가능하다는 한계가 있다. 또한 상용 서비스의 활용으로 비용 문제가 발생한다. 본 연구는 공개 소스 기반의 OpenLDAP을 사용하여 MFA를 중앙 집중식으로 처리하도록 설계함으로써 비용 및 데이터 주권 문제가 발생하지 않는다는 점에서 차별화된다.

연합 신원 관리 환경에서 기관 간 2차 인증 시스템의 공유를 목적으로 TOTP 기반의 검증자 및 대리인증자를 설계한 연구가 수행되었다[13]. 해당 연구에서는 MFA를 지원하지 않는 기관을 대신하여 대리인증자가 MFA 요청을 처리한다. 아이디제공자의 모듈 형태로 구현되어 비용 효율성이 높으며, OTP 보안 권고사항을 준수하여 보안성을 확보하였다. 해당 연구에서 개발된 TOTP 기반의 2차 인증 시스템은 OATH 표준을 준수한다는 점에서 본 연구와 유사하나, 웹 환경에 특화되어 있어 본 연구의 대상인 SSH 서버 환경과는 적용 범위 및 세부 기술에서 차이가 있다.

SSH 인증 보안과 관련된 최신 연구 동향으로, OIDC 토큰을 활용한 접근법이 최근 주목받고 있다. 본 연구에서는 TOTP를 활용한 SSH MFA 시스템을 제안하지만, 기술 동향을 파악하기 위해 OIDC 토큰 기반의 SSH 인증 연구를 소개한다.

OpenSSH 공개키는 서버에서 제거하기 전까지 반영구적으로 이용될 수 있어 사용자 간 무단 공유의 위험이 있다. 또한 키 생명 주기의 관리에 높은 운영 비용이 소요되며, 사용자의 신원 정보와 공개키 간 신원 바인딩이 불가능하여 보안 침해 시 사용자 추적이 어렵다는 문제가 있다. 반면, OIDC 토큰은 유효기간이 짧으며 사용자 신원과 바인딩이 가능하고, 세분화된 권한 관리가 가능하다는 장점을 갖는다.

SSH-OIDC는 연합 신원 관리 환경에서 OIDC 토큰을 활용해 SSH 접속을 가능하게 하는 인증 프레임워크이다[14]. SSH-OIDC는 사용자가 OIDC 제공자로부터 획득한 접근 토큰(Access token)을 SSH 인증에 활용한다. 사용자가 SSH 로그인 시 PAM 모듈을 통해 접근 토큰을 입력하면, OIDC 인증 프록시 서비스인 motley-cue가 토큰 유효성을 검증한다. 검증 결과는 PAM으로 반환되어 최종 로그인 허용 여부가 결정된다. 해당 연구는 2차 인증 요소를 직접 다루지 않았으나, 접근 토큰의 획득 과정에서 웹 기반의 2차 인증이 가능할 것으로 판단된다.

하지만 세션 하이재킹 공격 등으로 인한 토큰 탈취 시 웹 환경의 2차 인증을 우회할 수 있다는 문제가 있다. 또한 영신뢰(Zero trust) 관점에서 매 접근 시마다 사용자 또는 토큰에 대한 재검증이 요구되므로 SSH 환경에서도 독립적인 2차 인증이 필요하다. 따라서 본 연구에서 제안하는 PAM 기반 TOTP 2차 인증 모듈은 SSH-OIDC의 토큰 기반 1차 인증과 결합하여 강화된 MFA를 구현할 수 있을 것으로 판단된다.

Ⅳ. MFA 시스템의 설계

4.1 설계 목표

본 연구를 통해 구현하는 TOTP 기반 SSH MFA 시스템은 다음과 같은 설계 목표를 갖는다.

- 인증 방식 분리(R1): OpenLDAP 환경에서 패스워드 인증과 TOTP 패스코드 인증을 완전히 분리하여 각 인증 방식이 독립적으로 동작하게 한다. 모듈화된 구조를 통해 각 인증 요소의 독립적인 검증이 가능해야 한다. 또한 사용자에게 명시적이고 단계적인 인증 프로세스를 제공해야 한다.
- 클라이언트 호환성(R2): 표준 SSH 프로토콜을 준수하여 OpenSSH, PuTTY, SecureCRT 등 기존 SSH 클라이언트와의 호환성을 유지한다. 별도의 클라이언트 소프트웨어 설치나 수정 없이 기존 환경에서 MFA를 즉시 적용할 수 있어야 한다.
- 인증방식 다양화(R3): SSH 공개키 인증과 OTP, 패스워드 인증과 OTP 등 다양한 인증 방식의 조합이 가능해야 한다. 서버 관리자의 보안 정책과 요구사항에 따라 인증 방식을 유연하게 조합할 수 있으며, 서로 다른 보안 수준이 필요한 시스템이나 사용자 그룹에 맞춤형 인증 정책을 적용할 수 있어야 한다.
- 보안성 확보(R4): LDAP 접속 정보 및 사용자 패스워드, TOTP 공유키 등 인증 정보를 LDAP 서버에 중앙 집중화해야 한다. SSH 클라이언트 측에 저장되는 민감한 정보를 최소화하여 공격 표면을 최소화해야 한다. 또한 중앙화된 관리를 통해 보안 정책의 일관된 적용과 관리가 가능해야 한다.

4.2 시스템 구조 및 설계

그림 1은 제안하는 MFA 시스템의 구조와 인증 요소별 처리 절차를 보여준다. MFA 시스템은 SSH 클라이언트, SSH 서버(SSHD), PAM, OpenLDAP 서버(slapd), 백엔드 데이터베이스 엔진, TOTP 모듈로 구성된다.

SSH 클라이언트는 사용자가 원격 서버에 접속하기 위해 사용하는 소프트웨어이며, SSH 서버는 원격 접속 요청을 처리하는 서버 측 소프트웨어이다. PAM은 리눅스 시스템의 모듈식 인증 프레임워크로서 다양한 인증 방식을 지원한다.

OpenLDAP 서버(slapd)는 LDAP 요청을 처리하고 인증을 담당하는 서버 프로세스이다. 백엔드 데이터베이스 엔진은 디렉토리 데이터를 저장하고 관리하는 데이터베이스 계층이며, 사용자 엔트리와 인증 정보를 저장한다. TOTP 모듈은 slapd에 동적으로 로드되는 OpenLDAP 확장 연산(Extended Operation) 핸들러로, TOTP 알고리즘을 구현하여 일회용 패스코드를 검증한다.

제안하는 MFA 시스템의 전체적인 동작 과정은 1차 인증(그림 1의 Primary authentication)과 2차 인증(Secondary authentication)으로 구분된다.

1차 인증에서 공개키 인증 방식을 사용할 경우, SSH 클라이언트가 공개키 인증을 시도하면 SSH 서버는 사

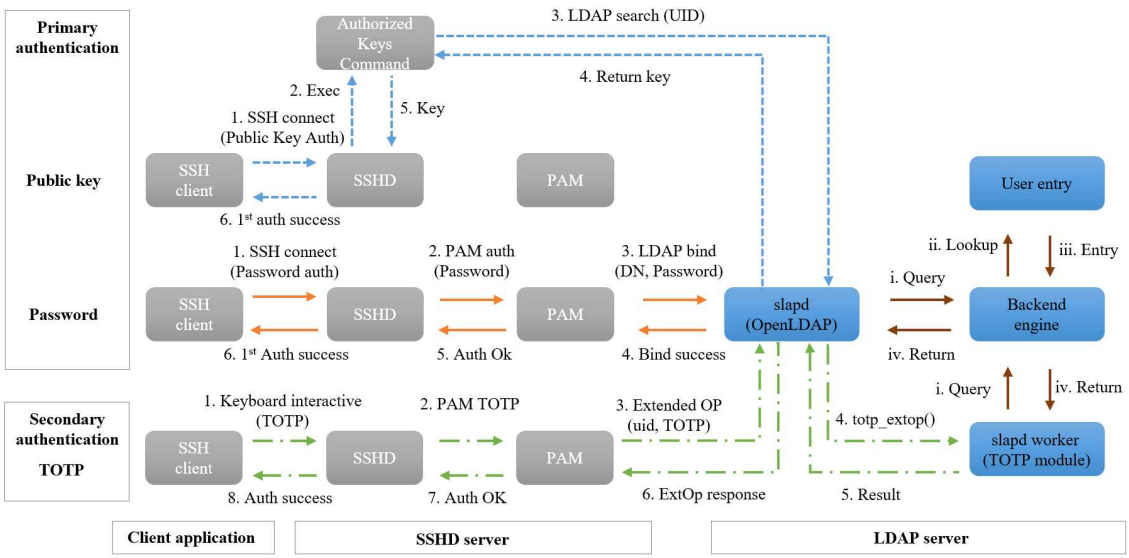

그림 1. 시스템 구조 및 인증 절차.
Fig. 1. System architecture and authentication flows.

용자의 공개키 정보를 확인하기 위해 OpenLDAP 서버에 질의한다. slapd는 백엔드 데이터베이스 엔진을 통해 사용자 정보를 조회한다. LDAP 조회 과정은 요구 속성에 대한 질의(그림 1의 i. Query), 데이터 검색(ii. Lookup), 엔트리 획득(iii. Entry), 결과 반환(iv. Return)의 단계로 진행된다. 조회된 공개키 정보는 SSH 서버로 반환되며, SSH 서버는 조회된 공개키 정보를 기반으로 클라이언트의 개인키 소유를 검증하여 1차 인증을 완료한다.

패스워드 방식의 경우, SSH 클라이언트가 패스워드 인증을 시도하면 SSH 서버는 PAM을 통해 인증 요청을 처리한다. PAM은 OpenLDAP 서버와 바인드를 수행하고, slapd는 제공된 패스워드를 확인하여 바인드 결과를 반환한다. PAM이 성공적인 바인드 결과를 받으면 1차 인증이 완료된다.

1차 인증 성공 후 2차 인증이 시작되며, SSH 서버는 keyboard-interactive 방식으로 TOTP 패스코드의 입력을 사용자에게 요청한다. 사용자가 입력한 TOTP 패스코드는 PAM 모듈을 통해 OpenLDAP의 확장 연산으로 전달되어 검증되고, 최종적으로 두 단계 인증이 모두 성공해야 사용자의 SSH 접속이 허용된다.

앞서 설명한 시스템 구조와 인증 처리 절차를 바탕으로, 4.1절에서 제시한 설계 목표 R1부터 R4까지를 달성하기 위한 구체적인 설계 방안을 다음과 같이 제시한다.

- 인증 방식 분리(R1)

MFA 시스템은 설계 목표 R1을 충족하기 위해 1차 인증과 2차 인증을 분리하여 실행하는 2단계 구조를 갖는다. 1차 인증 요소로는 SSH 로그인에 일반적으로 사용되는 공개키 또는 패스워드 방식을 채택하고, 2차 인증 요소로는 TOTP 패스코드를 사용한다.

패스워드 기반 인증 방식은 PAM에서 처리되지만, 공개키 인증 방식은 SSH 서버가 직접 처리하므로 PAM을 우회한다는 특성이 있다. 따라서 OpenSSH 공개키를 1차 인증 요소로 활용하기 위해서는 공개키 인증의 성공 여부를 확인하는 별도의 PAM 모듈이 필요하다. 이는 2차 인증 요소를 실행하기 전에 공개키 기반의 1차 인증이 성공했는지 확인하기 위함이다. 패스워드 방식의 경우에는 pam_unix나 pam_ldap 등 기존 PAM 모듈을 활용할 수 있어 추가적인 개발이 필요하지 않다.

모듈식 구조로 설계된 TOTP 인증 요청 및 검증 기능은 기존 PAM 모듈이나 OpenLDAP 모듈과 독립적으로 동작한다. 따라서 기존 OpenLDAP 시스템에 대한 영향을 최소화하면서 확장 가능한 구조를 제공하여 설계 목표 R1을 충족한다.

- 클라이언트 호환성(R2)

SSH 클라이언트와의 호환성 확보를 위해 1차 인증 완료 후 2차 인증은 keyboard-interactive 메커니즘을 활용한다. 기존 SSH 클라이언트들이 이미 keyboard-interactive 방식을 지원하고 있으므로, 사용자는 별도의 소프트웨어 설치나 소스 코드의 수정 없이도 TOTP 기반 MFA를 사용할 수 있다. 표준 메커니즘을 사용하지 않는다면, SSH 클라이언트가 TOTP 패스코드 입력을 받기 위해 소스코드를 수정하거나 전용 클라이언트를 새로 개발해야 하는 문제가 발생한다.

- 인증방식 다양화(R3)

1차와 2차 인증의 분리 구조를 통해 다양한 인증 요소를 MFA로 조합할 수 있어 설계 목표 R3를 충족한다. 개별 PAM 모듈에 대한 독립적 설정이 가능하므로, 1차 인증을 생략하고 TOTP만으로 사용자를 인증하거나 OpenSSH 공개키로만 인증하는 등의 유연성을 확보할 수 있다. 서버 관리자의 보안 정책과 요구사항에 따라 인증 방식을 다양하게 구성할 수 있으며, 서로 다른 보안 수준이 필요한 시스템이나 사용자 그룹에 맞춤형 인증 정책을 적용할 수 있다.

- 보안성 확보(R4)

TOTP 공유키를 중앙 집중식으로 관리하여 SSH 서버에 노출시키지 않도록 하기 위해, OpenLDAP 서버 내부에서 TOTP 패스코드의 검증을 수행할 소프트웨어 모듈이 필요하다. 해당 모듈은 클라이언트로부터 전달받은 TOTP 패스코드를 검증하고 결과를 반환한다.

추가적인 보안성 확보를 위해 OpenLDAP 서버에서는 클라이언트 인증서를 통한 접근 제어를 채택하여 특정 인증서를 가진 클라이언트만 TOTP 공유키 속성에 접근할 수 있도록 구성하였다. 일반적으로 LDAP 클라이언트가 LDAP 서버에 접근하기 위해서는 관리자 계정의 사용자 ID와 패스워드를 LDAP 클라이언트(SSH 서버) 측에 저장해야 하는데, 이는 인증 정보가 노출되어 공격 표면이 확대되는 문제가 발생할 수 있다.

V. 시스템 환경 구성 및 구현

5.1 SSH 서버 설정

MFA 시스템의 구현을 위해 SSH 서버는 OpenSSH를 활용하였다. 먼저, SSH 서버가 MFA 과정에 참여할 수 있도록 OpenSSH 환경 설정 파일인 sshd_config의 주요 설정을 그림 2와 같이 수정해야 한다.

패스워드 인증과 공개키 인증을 모두 활성화하기 위해 PasswordAuthentication과 PubkeyAuthentication

```
PasswordAuthentication yes
PubkeyAuthentication yes
KbdInteractiveAuthentication yes
ChallengeResponseAuthentication yes
UsePAM yes
AuthenticationMethods publickey,keyboard-interactive keyboard-interactive
```

그림 2. MFA 지원을 위한 SSH 서버 설정.
Fig. 2. SSH server configuration for MFA support.

을 yes로 설정한다. Keyboard-interactive 방식을 이용하여 MFA를 처리하기 위해 challenge-response 또는 KbdInteractiveAuthentication을 활성화해야 한다. KbdInteractiveAuthentication과 challenge-response는 실질적으로 동일한 기능을 갖지만, OpenSSH 버전 호환성을 고려하여 모두 설정하였다.

MFA 시스템에서 SSH 서버와 PAM 모듈을 연동하기 위해서는 UsePAM 옵션을 반드시 활성화해야 한다. 해당 옵션의 설정을 통해 PAM 모듈이 LDAP 서버로부터 사용자 인증을 수행하고, TOTP 패스코드 기반의 2차 인증을 수행할 수 있다. AuthenticationMethods 지시어는 SSH 클라이언트가 인증할 때 반드시 거쳐야 하는 인증 방법들을 순서대로 지정하는 역할을 한다. 쉼표(,)는 AND 관계를 의미하고 공백은 OR 관계를 나타낸다.

본 연구에서는 AuthenticationMethods에 두 가지 인증 방법을 설정하였다. publickey,keyboard-interactive 방식은 공개키 기반의 1차 인증을 먼저 수행하고, 인증 성공 시 PAM 스택에 따라 keyboard-interactive 방식의 2차 인증을 진행한다. 공개키 인증에 성공한 경우 불필요한 패스워드 인증을 생략하기 위해 PAM 모듈에서 SSH 인증 상태를 확인하여 조건부 인증을 수행한다. PAM 스택에 포함된 PAM 모듈에 따라 MFA가 가능하며, 공개키 인증에 실패하면 keyboard-interactive 방식만으로 사용자 인증을 수행한다.

5.2 PAM 설정

SSH 서버에서 MFA 인증 요소를 조합하기 위해서는 PAM 스택 구성이 중요하다. 5.1 절에서 기술했듯이 keyboard-interactive 방식은 PAM 모듈의 배치 순서에 의해 제어되므로, MFA 요구사항을 충족하는 모듈 배치가 요구된다. 그림 3은 패스워드 인증 또는 공개키 인증과 TOTP 인증을 조합하기 위한 PAM 설정의 예시이다. 사용자의 1차 인증 결과에 따라 MFA 인증 요소를 다르게 적용하는 구조를 보여준다.

그림 상단의 두 블록은 패스워드 인증과 TOTP 인증을 조합한 MFA를 구현한다. 1차 인증에서는 pam_unix나 pam_sss PAM 모듈이 실행되어 각각 로컬 시스템

```
# 1st authentication: Standard Unix or LDAP authentication (ID/PW)
auth [success=2 default=ignore] pam_unix.so nullok
auth [success=1 default=ignore] pam_sss.so use_first_pass
# here's the fallback if no module succeeds
auth   requisite                pam_deny.so
```
/ect/pam.d/common-auth

```
# 2nd authentication
# OTP with the standard Unix or LDAP authentication (ID/PW)
auth required pam_ldap_otp.so auth=remote
auth required pam_sepermit.so
```
/ect/pam.d/sshd

```
# 1st authentication: Public key authentication
#auth [success=2 default=ignore] pam_unix.so nullok
#auth [success=1 default=ignore] pam_sss.so use_first_pass
```
/ect/pam.d/common-auth

```
# 2nd authentication:  OTP with Public key authentication
auth   requisite pam_exec.so  pam_publickey_authed.so
auth   required pam_ldap_otp.so auth=remote
```
/ect/pam.d/sshd

그림 3. PAM 설정 (상) 패스워드 및 TOTP (하) 공개키 및 TOTP.
Fig. 3. PAM configuration (Top) password with TOTP (bottom) public key with TOTP.

또는 LDAP 기반의 패스워드 인증을 수행한다. 구체적으로 pam_unix 모듈은 SSH 서버에 저장된 사용자 ID와 비밀번호를 이용하여 인증하며, pam_sss 모듈은 LDAP 서버에 저장된 사용자 계정을 활용하여 인증한다.

1차 인증에 성공하면 본 연구를 통해 개발한 pam_ldap_otp 모듈이 실행되어 2차 TOTP 인증을 처리한다. pam_ldap_otp 모듈은 사용자로부터 입력받은 TOTP 패스코드를 OpenLDAP 서버의 확장 연산을 통해 검증하고, 검증 결과를 바탕으로 인증 과정을 완료한다. PAM을 통해 1차 인증과 2차 인증을 분리함으로써 다양한 인증 요소의 조합이 가능하다.

그림 하단의 두 블록은 공개키 인증과 TOTP 인증의 조합을 보여준다. 공개키 기반 구성에서는 패스워드 인증이 생략되므로 pam_unix나 pam_sss 모듈을 비활성화한다. 대신 pam_publickey_authed 모듈을 통해 OpenSSH 공개키 인증의 성공 여부를 확인한다. 공개키 인증이 성공하면 pam_ldap_otp 모듈이 실행되어 TOTP 패스코드 인증을 수행한다.

5.3 PAM 모듈 구현

본 연구에서는 SSH MFA 시스템을 구현하기 위해 pam_publickey_authed와 pam_ldap_otp 등 두 개의 PAM 모듈을 개발하였다. pam_publickey_authed 모듈은 OpenSSH 공개키 인증의 성공 여부를 확인하며, pam_ldap_otp 모듈은 사용자가 입력한 TOTP 패스코드를 OpenLDAP 서버로 전달하고, 검증 결과를 회신받아 최종 인증 결과를 결정한다.

OpenSSH에서 공개키 인증은 PAM을 우회하여 처

리되므로, 2차 인증 단계에 앞서 공개키 인증의 성공 여부가 확인되어야 한다. pam_publickey_authed 모듈을 통해 공개키 인증 결과를 검사한다. 공개 소스인 pam-ssh-auth-info[15]를 활용할 수 있으나, GPL 라이센스 제약을 회피하기 위해 자체적으로 구현하였다.

먼저 pam_sm_authenticate() 함수가 PAM 컨텍스트에서 서비스 이름, 원격 호스트, 사용자 ID 정보를 추출한다. 추출된 정보를 이용해 SSH 서비스에 대한 인증 요청임을 확인한 후, 그림 4와 같이 SSH_AUTH_INFO_0 값을 조회하여 공개키 인증의 성공 여부를 판단한다. 공개키 인증이 성공하면 PAM_SUCCESS를 반환한다. 그림 3의 공개키 기반 인증 블록에서 보듯이 pam_publickey_authed가 auth requisite로 설정되어 있으므로, 공개키 인증에 실패한 경우 인증 실패로 처리되어 로그인이 차단된다.

pam_ldap_otp 모듈은 사용자가 입력한 TOTP 패스코드를 OpenLDAP의 TOTP 모듈로 전달하고, 검증 결과를 기반으로 인증 여부를 결정한다. pam_ldap_otp 모듈은 로컬 인증과 원격 인증 등 두 가지 방식을 지원한다. 로컬 인증 방식은 OpenLDAP 서버에서 사용자의 TOTP 공유키를 조회한 후 SSH 서버에서 TOTP 패스코드를 검증한다. 원격 인증 방식은 OpenLDAP 확장 연산을 통해 LDAP 서버에서 TOTP 검증을 수행한다.

pam_ldap_otp 모듈은 설정 파일을 통해 OpenLDAP 서버 URI, TOTP 검색 기준 DN, TOTP 공유키 속성명, TLS(Transport Layer Security) 인증서 경로, TOTP 예외 사용자 목록 등의 구성 정보를 확보한다. 본 시스템의 OpenLDAP 서버에서는 클라이언트 인증서를 기반으로 접근 제어를 실시한다. 따라서 특정 인증서를 갖는 클라이언트만 TOTP 공유키 속성에 접근할 수 있다. 이를 통해 관리자 암호나 별도의 바인드 패스워드 없이도 안전하게 사용자 속성에 접근할 수 있다.

인증 과정에서는 먼저 사용자 ID를 확인하고 TOTP 예외 사용자 목록에 포함되어 있는지 검사한다. 예외 사용자인 경우 TOTP 검증을 건너뛰고 1차 인증의 성공 여부만으로 최종 인증을 결정한다.

그림 5는 LDAP 서버 연결, 사용자 계정 잠금 상태

```
PAM_EXTERN int pam_sm_authenticate(pam_handle_t *pamh, int flags,
                                   int argc, const char **argv) {
  ...,
  ssh_auth_info = pam_getenv(pamh, "SSH_AUTH_INFO_0");
  if (ssh_auth_info != NULL && strncmp(ssh_auth_info, "publickey", 9) == 0) {
    return PAM_SUCCESS;
  } ...,
}
```

그림 4. 공개키 인증 여부의 확인을 위한 PAM 코드
Fig. 4. PAM code to verify public key authentication.

```
static int perform_remote_auth(pam_handle_t *pamh, ...,
                               const struct auth_options *opts) {
  /* 1. Check account lock status */
  if (check_account_locked(pamh, username, opts)) {
     pam_syslog(pamh, LOG_WARNING, "Authentication denied for locked
account: %s", username);
     return PAM_AUTH_ERR;
  }
  /* 2. Connect OpenLDAP server */
  ...,
  /* 3. Remote TOTP verification */
  rc = r_verify_totp(pamh, ld, username, otp, user_search_base);
  if (rc != LDAP_SUCCESS) {
     increment_fail_count(pamh, username); /* count up if failure */
     return PAM_AUTH_ERR;
  }
  /* reset counter if success */
  reset_fail_count(pamh, username);
  return PAM_SUCCESS;
}

static int r_verify_totp(pam_handle_t *pamh, ...,
                         const char *user_search_base) {
  /* BER encoding: uid, passcode */
  ...,
  /* OpenLDAP Extended operation */
  rc = ldap_extended_operation_s(ld, TOTP_VERIFY_OID, reqdata,
                  NULL, NULL, &retoid, &retdata);
  return rc;
}
```

그림 5. 사용자 잠금 및 TOTP 검증을 위한 PAM 구현 코드
Fig. 5. PAM code for user account locking and TOTP verification.

확인, TOTP 검증 과정을 포함한 PAM 모듈의 구현 코드를 간략히 보여준다. perform_remote_auth() 함수는 원격 인증의 전체 흐름을 제어한다. 먼저 check_account_locked() 함수를 호출하여 계정 잠금 상태를 확인한 후 LDAP 서버 연결과 TOTP 검증을 순차적으로 수행한다. 본 연구에서는 TOTP 공유키 관리와 검증의 중앙화가 목표이므로 원격 인증 모드를 중심으로 설명한다.

원격 인증 모드에서는 r_verify_totp() 함수를 통해 OpenLDAP 서버에 전달할 데이터를 구성한다. 데이터는 사용자 ID, 패스코드, 검색 기준점을 포함하고 BER(Basic Encoding Rules) 인코딩된다. TOTP 패스코드가 입력되면 ldap_extended_operation_s() 함수를 호출한다. 호출 시 OpenLDAP 서버의 TOTP 모듈을 가리키는 객체 식별자(Object ID, OID)인 TOTP_VERIFY_OID와 함께 패스코드 검증을 요청한다. 서버의 응답 결과를 통해 PAM 모듈에서 인증 성공 여부를 판단한다.

TOTP 계정 잠금 기능을 적용하기 위해 패스코드 검증에 실패한 경우 increment_fail_count() 함수를 호출하여 실패 카운터를 증가시킨다. 성공한 경우 reset_fail_count() 함수로 실패 카운터를 초기화한다. 원격 인증 방식은 TOTP 공유키가 클라이언트로 전송되지 않아 보안성이 향상되며, 중앙 집중식 TOTP 패스코

드 검증이 가능한 장점이 있다.

무작위 대입 공격을 방어하기 위해 실패 카운터를 이용한 계정 잠금 기능을 구현하였다. check_account_locked() 함수는 계정 잠금 상태를 확인한다. 계정 잠금 메커니즘은 사용자별 인증 실패 횟수를 추적하고 설정된 임계값(max_tries)을 초과하면 특정 시간(lock_time) 동안 계정을 잠근다. 임계값과 잠금 시간은 PAM 모듈의 매개변수로 동적 설정이 가능하다.

관리자가 lock_time 이전에도 수동으로 계정을 해제할 수 있도록, 인증 실패 기록을 파일로 관리하여 운용성을 높였다. 잠긴 계정은 lock_time 경과 후 자동으로 해제된다. 인증 성공 시에는 reset_fail_count() 함수를 호출하여 실패 카운터를 초기화한다.

5.4 OpenLDAP TOTP 모듈

PAM에서 전송한 TOTP 패스코드를 검증하기 위해 OpenLDAP 서버에 TOTP 모듈을 구현하였다. 구현된 TOTP 모듈은 OpenLDAP 확장 연산을 통해 TOTP 패스코드를 검증하고 결과를 반환한다. TOTP 공유키를 OpenLDAP 서버에서 중앙 관리하여, 공유키의 네트워크 전송이나 클라이언트 노출을 방지함으로써 보안성을 향상시켰다.

개발된 TOTP 모듈은 OATH 커뮤니티의 LDAP 스키마를 활용해 구현하였다. oathTOTPToken 객체 클래스를 통해 사용자별 TOTP 정보를 관리한다. 공유키는 oathSecret 속성에 저장되며, 사용자가 여러 개의 공유키를 갖도록 다중값 속성으로 구현하였다. 프로토타입의 신속한 구현을 위해 TOTP 매개변수는 코드에 정적으로 설정하였으나, 향후 oathTOTPParams와 oathTOTPUser 객체를 활용하여 TOTP 설정 정보와 사용자별 활성화 여부를 동적으로 관리할 예정이다.

그림 6은 OpenLDAP 확장 연산을 통해 구현된 TOTP 모듈의 패스코드 검증 절차를 간략히 보여준다. 4.3절에서 설명했듯이 PAM 모듈은 사용자 ID와 TOTP 패스코드를 BER 인코딩하여 확장 연산을 요청한다. OpenLDAP은 PAM 모듈이 함께 전달한 OID 정보를 활용해 TOTP 모듈을 실행한다. 각 확장 모듈은 고유한 OID를 가지므로, OpenLDAP 서버는 OID를 통해 수행할 확장 연산을 결정할 수 있다. 확장 연산은 TOTP 모듈의 totp_extop() 함수가 처리한다. 전달받은 사용자 ID와 검색 기준점을 조합하여 사용자 DN을 구성한다. 이후 select_backend() 함수를 통해 해당 DN에 대응하는 백엔드 데이터베이스를 선택하고 사용자의 LDAP 엔트리를 조회한다.

attr_find() 함수는 사용자 엔트리에서 oathSecret 속

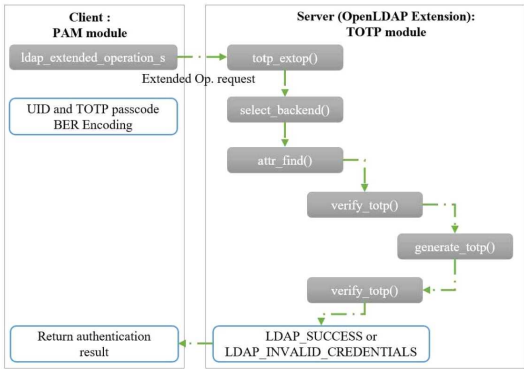

그림 6. TOTP 검증 절차.
Fig. 6. TOTP verification process.

성에 저장된 TOTP 공유키를 검색한다. 이후 verify_totp() 함수를 호출하여 TOTP 검증을 수행한다. 사용자가 복수의 공유키를 보유할 수 있으므로, verify_totp() 함수는 사용자가 가진 모든 공유키를 순차적으로 검증한다. generate_totp() 함수는 공유키와 현재 시간 윈도우를 기준으로 예상 패스코드를 생성한다. 예상 패스코드는 HMAC-SHA1 알고리즘과 동적 절삭(Dynamic truncation)[10]을 통해 6자리 숫자로 생성된다. 생성된 패스코드는 Google Authenticator 등 범용 스마트폰 OTP 앱과의 호환성을 보장한다.

생성된 예상 패스코드는 verify_totp() 함수에서 PAM 모듈이 전달한 패스코드와 비교되고, 최종적으로 두 패스코드의 일치 여부가 PAM 모듈에 반환된다. SSH 서버의 PAM 모듈과 OpenLDAP 서버의 TOTP 모듈을 표준 LDAP 프로토콜과 호환되게 구현하여 확장성 있는 SSH MFA 시스템을 구현할 수 있다.

Ⅵ. 평 가

본 장에서는 제안한 TOTP 기반 SSH MFA 시스템의 실용성과 보안성을 평가한다. 평가를 위해 범용 SSH 클라이언트를 활용한 MFA 절차와 로그인 결과를 제시하고, 2.2절에서 정의한 보안 요구 항목들에 대한 구현 결과를 검증한다. 또한 OpenLDAP에 구현한 TOTP 확장 연산의 성능을 측정하여 시스템의 효율성과 실제 운영 환경에서의 적용 가능성을 검증한다.

제안한 시스템의 성능 평가를 위해 다음과 같은 실험 환경을 구성하였다. SSH 서버는 Intel Xeon E5-2680 v1@2.70GHz CPU 4개(코어당 1개의 쓰레드)와 4GB 메모리를 갖춘 가상 머신에 설치되었다. LDAP 서버는 Intel Xeon E5-2680 v1@3.20GHz CPU 1개와 8GB

메모리를 갖춘 가상 머신을 활용하였다. 클라이언트는 Ubuntu 22.04, 서버는 Rocky 9.4에서 동작한다. SSH와 LDAP 소프트웨어로 OpenSSH 8.9p1과 OpenLDAP 2.6.8을 각각 사용하였다. TOTP 인증을 위해 Google Authenticator를 사용하였으며, SSH 클라이언트로는 Bitvise SSH를 사용하여 keyboard-interactive 방식의 MFA 결과를 평가하였다.

그림 7은 개발한 SSH MFA 시스템이 적용된 SSH 서버에 Bitvise SSH로 로그인하는 과정을 보여준다. Keyboard-interactive 방식을 사용(그림 7의 ①)하고 패스워드와 TOTP 패스코드를 이용해 인증한 결과이다. 설계 목표에서 정의한 요구사항들이 다음과 같이 구현되었음을 확인할 수 있다. 먼저, 인증 방식 분리(R1) 관점에서, 사용자는 패스워드 인증을 수행(②)한 후 TOTP 인증(③) 단계를 거쳐 최종 인증에 성공(④)함으로써, 각 인증 방식이 독립적으로 동작함을 알 수 있다. 또한, Bitvise SSH 클라이언트에서 별도의 소프트웨어 설치 없이 MFA가 동작하여 클라이언트 호환성(R2)를 충족함을 확인하였다.

결과를 간략화하기 위해 그림 7에는 포함하지 않았지만, Bitvise SSH를 통해 OpenSSH 공개키와 TOTP 패스코드의 조합으로도 사용자 인증이 가능하여 설계 목표 중 하나인 인증방식 다양화(R3)를 달성하였다. TOTP 공유키와 패스워드의 중앙 집중화를 통한 보안성 확보(R4)에 대해서는 5.4절에서 다루었다.

2장에서 정의한 OTP 보안 요구사항의 준수 여부를 검증하기 위해 시뮬레이션 기반 평가를 수행하였다. OTP 무작위성(S1) 검증을 위해 OpenLDAP에 구현한 TOTP 모듈과 동일한 방식으로 10,000개의 6자리 TOTP 패스코드를 생성하고 패스코드 무작위성(S1)을

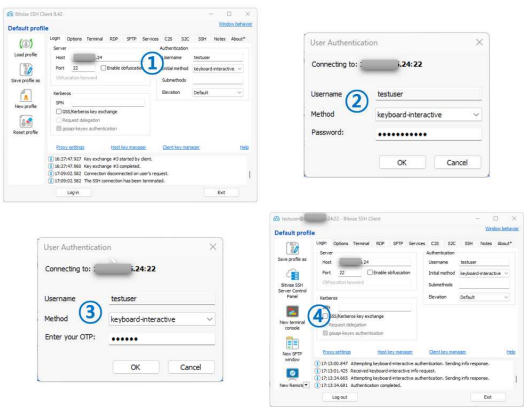

그림 7. MFA를 통한 로그인 과정.
Fig. 7. Login process through MFA.

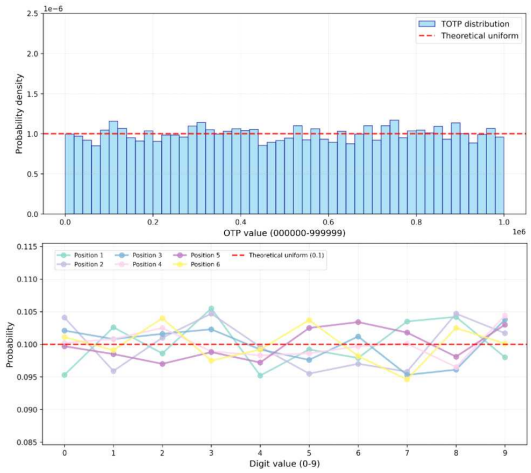

그림 8. TOTP 패스코드 발생 분포 (상) 6자리 확률 밀도 (하) 자리별 확률.
Fig. 8. TOTP passcode generation distribution (Top) six-digit probability density (bottom) position-wise probability.

분석하였다. 시스템 설정은 패스코드 길이 6자리(S2), 유효기간과 갱신 주기 30초(S5, S6)로 구성하여 TOTP 보안 요구사항을 충족하였다.

그림 8은 시뮬레이션을 통해 생성한 TOTP 패스코드의 발생 분포를 6자리 전체와 각 자리수 별로 시각화한 결과이다. 생성된 10,000개 TOTP 패스코드 중 9,942개가 고유값(중복률 0.58%)을 가져 이론적 중복률 0.50%에 매우 근접하였으며, 자릿수별 분석에서도 평균 0.1000, 표준편차 0.0021~0.0036으로 측정되어 위치별 편향성도 없음을 확인하였다. Chi-square 적합도 검사 결과, 전체 분포의 p-value가 0.055로 유의수준(α=0.05)을 초과하여 균등분포로 볼 수 있으며, 개별 자릿수의 p-value도 모두 0.16 이상으로 나타나 TOTP 알고리즘의 높은 무작위성(S1)을 입증하였다.

본 연구에서는 OpenLDAP 서버의 부하를 줄이고 시스템 구조를 단순화하기 위해 재시도 횟수 제한(S3) 기능을 SSH 서버의 PAM 모듈에만 구현하였고 OpenLDAP의 TOTP 모듈에는 구현하지 않았다. 방화벽 설정을 통해 사전에 승인된 SSH 서버만 OpenLDAP 서버에 접근하도록 제한하여 OpenLDAP 서버에 대한 무차별 대입 공격을 방지할 수 있다.

TOTP의 일회성 사용(S4) 보장을 위해서는 마지막으로 사용된 TOTP 패스코드를 LDAP 속성에 저장하여 중복 사용을 차단하는 것이 효과적인 방안이다. 하지만 본 연구에서는 TOTP 인증이 요청될 때마다 LDAP 속성을 읽고 써야 하는 부하를 고려하여 해당 기능을 구현하지 않았다. 대신 사후 분석과 대응이 가능하게

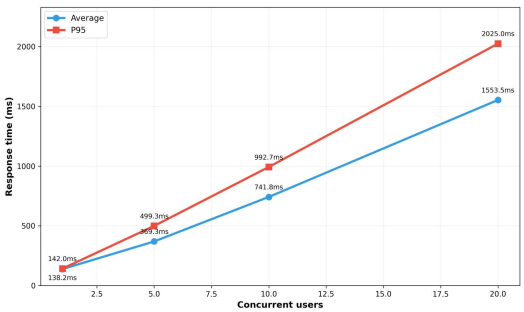

그림 9. 동시 사용자 수별 평균 및 95 백분위 응답 시간.
Fig. 9. Average and 95th percentile response time by concurrent users.

TOTP 패스코드와 관련된 정보를 로그 메시지에 기록하도록 구현하였다. 그러나 TOTP의 일회성 보장은 보안상 중요한 요소이므로 향후 연구를 통해 OpenLDAP 부하와 보안성 향상 간의 균형을 맞출 수 있는 구현 방안을 모색할 계획이다.

다음으로, 개발된 TOTP 모듈의 성능과 안정성 평가를 위해 Python 기반 부하 테스트 도구를 구현하고 SSH 서버에서 실험을 수행하였다. 실험에 사용된 가상 머신의 낮은 하드웨어 사양을 고려하여 절대적인 성능 수치보다는 TOTP 모듈의 안정적 동작과 확장 가능성에 초점을 맞춰 결과를 분석하였다.

OpenLDAP에 구현된 TOTP 모듈의 안정적 동작 여부를 검증하기 위해 동시 사용자 수를 1명에서 20명까지 증가시키며 실험을 수행하였다. 각 시나리오에서 사용자당 15회의 인증 요청을 보내고 응답 시간을 측정하여 OpenLDAP 서버의 동시 처리 성능을 평가하였다. 그림 9는 동시 사용자 수에 따른 평균 및 95 백분위수의 응답 시간을 보여주며, 표 2에는 전체 성능 지표를 요약하였다.

표 2에서 보듯이 동시 사용자 수와 관계없이 OpenLDAP 서버가 TOTP 검증 요청에 대해 100% 응답하였고, 최대 초당 12.6건의 요청을 처리하여 안정적인 성능을 확인하였다. 대규모 과학 컴퓨팅 환경인 유럽 입자물리연구소(CERN) WLCG(World LHC Computing Grid)의 토큰 발행 횟수는 일반적으로 초당 3~5건이고 최대 초당 200건[16]이다. 한 번의 로그인 이후에 다수의 토큰이 발행될 수 있는 점을 참작하면 실험에서 측정한 초당 12.6건의 처리 성능은 제한된 하드웨어 환경에서도 TOTP 모듈이 실용적으로 활용될 수 있음을 보여준다.

표 2. 성능 요약
Table 2. Performance summary

Users	Response rate	Response time	Throughput
1	100%	138.2ms	6.7
5	100%	369.3ms	12.6
10	100%	741.8ms	12.5
20	100%	1553.5ms	12.1

Ⅶ. 결 론

본 연구에서는 SSH 환경에서 패스워드와 TOTP 패스코드를 결합해 입력하는 기존 OpenLDAP 방식의 문제점을 해결하기 위해 분리형 다요소 인증 시스템을 개발하였다. OpenLDAP 표준 확장 연산을 통한 TOTP 패스코드 검증과 PAM 기반의 모듈화된 인증 방식을 적용하여 인증 요소 다양화, 클라이언트 호환성, 보안성 확보 등의 설계 목표를 달성하였으며, 동시 사용자 성능 실험을 통해 실용성을 검증하였다.

향후 연구 과제는 다음과 같다. 첫째, TOTP 패스코드의 일회성 사용을 보장하여 재사용 공격을 방지하는 메커니즘 구현이 필요하다. 둘째, 특정 LDAP 구조에 의존하는 현재 시스템을 다양한 조직의 LDAP 환경에 범용적으로 활용할 수 있도록 개선해야 한다.

References

[1] R. Andrews, D. A. Hahn, and A. G. Bardas, “Measuring the prevalence of the password authentication vulnerability in SSH,” in *Proc. IEEE ICC 2020*, pp. 1-7, Dublin, Ireland, Jun. 2020. (https://doi.org/10.1109/ICC40277.2020.9148912)

[2] L. Lumburovska, J. Dobreva, S. Andonov, H. Mihajloska Trpcheska, and V. Dimitrova, “A comparative analysis of HOTP and TOTP authentication algorithms. Which one to choose?,” *Int. Sci. J. Secur. & Future*, vol. 5, no. 4, pp. 131-136, 2021.

[3] L. A. Meyer, et al., “How effective is multifactor authentication at deterring cyberattacks?,” *arXiv preprint arXiv:2305.00945*, May 2023. Retrieved Jun. 2, 2025, from https://arxiv.org/abs/2305.00945.

[4] D. W. Woods and T. Moore, “Evidence-based cybersecurity policy? A meta-review of security control effectiveness,” *J. Cyber*

Policy, vol. 8, no. 3, pp. 365-383, 2024. (https://doi.org/10.1080/23738871.2024.2335461)

[5] The OpenLDAP Project, *OpenLDAP Software 2.6 Administrator's Guide*, OpenLDAP Foundation, 2023. Retrieved Jun. 2, 2025, from https://www.openldap.org/doc/admin26/.

[6] Google, *Google Authenticator PAM module, GitHub Repository*, 2022. Retrieved Jun. 2, 2025, from https://github.com/google/google-authenticator-libpam.

[7] D. M'Raihi, S. Machani, M. Pei, and J. Rydell, "*TOTP: Time-based one-time password algorithm*," RFC 6238, Internet Eng. Task Force, May 2011.

[8] H. Krawczyk, M. Bellare, and R. Canetti, "*HMAC: Keyed-hashing for message authentication*," RFC 2104, Internet Eng. Task Force, Feb. 1997.

[9] S. Ma, et al., "An empirical study of SMS one-time password authentication in android apps," in *Proc. 35th Annu. Comput. Secur. Appl. Conf.*, pp. 339-354, Dec. 2019. (https://doi.org/10.1145/3359789.3359828)

[10] D. M'Raihi, M. Bellare, F. Hoornaert, D. Naccache, and O. Ranen, "*HOTP: An HMAC-based one-time password algorithm*," RFC 4226, IETF, Dec. 2005.

[11] D. Simmel and S. Filus, "Flexible enforcement of multi-factor authentication with SSH via Linux-PAM for federated identity users," in *Proc. PEARC17*, pp. 1-9, New Orleans, LA, USA, Jul. 2017. (https://doi.org/10.1145/3093338.3093392)

[12] A. A. Malik, H. Anwar, and M. A. Shibli, "Federated identity management (FIM): Challenges and opportunities," in *Proc. 2015 CIACS*, pp. 75-82, Rawalpindi, Pakistan, 2015. (https://doi.org/10.1109/CIACS.2015.7395570)

[13] J. Jo, S. Kim, and B. Cho, "Development of TOTP verifier and proxied authenticator to enable strong authentication in identity federation," *J. KICS*, vol. 48, no. 10, pp. 1-12, Oct. 2023. (https://doi.org/10.7840/kics.2023.48.10.1277)

[14] D. Gudu, M. Hardt, L. Brocke, and G. Zachmann, "Enabling secure shell access with OpenID connect," *Computing and Software for Big Sci.*, vol. 9, no. 5, 2025. (https://doi.org/10.1007/s41781-025-00136-5)

[15] *PAM SSH authentication information module*, GitHub Repository, Retrieved Jun. 2, 2025, from https://github.com/eehakkin/pam-ssh-auth-info

[16] B. Balci, M. Litmaath, and A. Nappi, *IAM @ Data Challenge 24*, presented at the WLCG/HSF Workshop 2024, CERN, Geneva, Switzerland, May 2024. Retrieved Jun. 2, 2025, from https://indico.cern.ch/event/1369601/contributions/5761956/attachments/2851851/4987905/IAM_DC24_WLCG_HSF_Workshop_2024.pdf

조 진 용 (Jinyong Jo)

2013년 : 광주과학기술원 정보통신공학과 박사
2003년~현재 : 한국과학기술정보연구원
2016년~현재 : 국제인증연합(eduGAIN) 운영그룹 위원
<관심분야> 신뢰기반 신원관리, 네트워크 응용 및 서비스
[ORCID:0000-0001-6830-3604]

김 동 균 (Dongkyun Kim)

2005년 2월 : 충남대학교 컴퓨터과학 박사
2000년~현재 : 한국과학기술정보연구원 책임연구원
2022년~현재 : 과학기술연합대학원대학교 데이터 및 HPC 과학 겸임교수
<관심분야> SDN/NFV, 5G/6G, Network Intelligence, Advanced Research Network
[ORCID:0000-0001-9484-339]

조 부 승 (Buseung Cho)

2017년 : 성균관대학교 컴퓨터공학 박사

2005년~현재 : 한국과학기술정보연구원

2018년~현재 : 과학기술연합대학원대학교 데이터 및 HPC 과학 부교수

<관심분야> 소프트웨어 정의 네트워크, 네트워크 관리

[ORCID:0000-0002-4661-5700]