https://doi.org/10.7840/kics.2023.48.10.1277

강성인증의 신원연합 적용을 위한 TOTP 검증자 및 대리인증자의 개발

조 진 용◆, 김 승 해*, 조 부 승*

Development of TOTP Verifier and Proxied Authenticator to Enable Strong Authentication in Identity Federation

Jinyong Jo◆, Seung-Hae Kim*, Buseung Cho*

요 약

유출된 비밀번호나 비밀번호의 취약성으로 인한 보안 침해를 예방하기 위해 통합인증 환경에서의 다요소 인증은 필수적이다. 기관 연합형 통합인증 환경에서 다요소 인증을 지원하는 아이디제공자의 수가 적을 경우, 다요소 인증을 필수로 요구하는 온라인 서비스에 사용자의 접근이 제한된다. 또한 다요소 인증은 지원하지만 표준을 준용하지 않는 아이디제공자로 인해 온라인 서비스가 다요소 인증의 실행 여부를 확인할 수 있는 방법이 제한될 수 있다. 본 논문은 비용 효율성, 보안성, 가용성, 그리고 표준과의 호환성과 같은 소프트웨어 요구사항을 고려하여 설계한 TOTP 검증자와 아이디제공자를 대신해 다요소 인증을 수행하는 대리인증자의 개발 내용을 소개한다. 마지막으로, 개발된 TOTP 검증자와 대리인증자를 기관 연합형 통합인증 환경에 적용하여 소프트웨어 요구사항의 충족 여부와 활용 가능성을 검증한다.

키워드 : SAML, 연합인증, 일회용 비밀번호, 데이터보호, 신원 보호
Key Words : SAML, Federated Single Sign-On, One-Time Password, Data Protection, Identity Protection

ABSTRACT

Multi-factor authentication (MFA) is crucial in single sign-on environments to mitigate the risk of security breaches caused by leaked or weak passwords. In federated single sign-on, the low availability of MFA from identity providers imposes significant restrictions on user access to online services that require it. Additionally, if an identity provider supports MFA but does not comply with standards, it can be challenging for online services to verify whether the identity provider has performed MFA. This paper presents a proxied authenticator that conducts MFA as a substitute for identity providers, along with a TOTP verifier that can be integrated within identity providers. Finally, the developed TOTP verifier and proxied authenticator are implemented in the federated single sign-on environment to assess their adherence to software requirements including cost-effectiveness, security, availability, and compatibility with standards.

※ 본 연구는 한국과학기술정보연구원(K-23-L04-C01-S01)의 지원으로 수행되었습니다.
◆ First Author : Korea Institute of Science and Technology Information, jiny92@kisti.re.kr, 정회원
* Korea Institute of Science and Technology Information, shkim@kisti.re.kr, 정회원; bscho@kisti.re.kr, 정회원
논문번호 : 202306-129-B-RU, Received June 20, 2023; Revised July 7, 2023; Accepted July 12, 2023

Ⅰ. 서 론

다수의 보안침해는 아이디와 비밀번호와 같은 크리덴셜(Credential)의 관리실패로 인해 발생한다[1]. 비밀번호의 유출은 날로 증가하고 있고 약 6.8%의 비밀번호는 유출된 이후에도 계속해서 사용되고 있다[2]. 유출된 비밀번호는 크리덴셜 스터핑(Credential stuffing) 공격 - 유출된 크리덴셜을 이용해 무작위로 로그인을 시도 - 등에 악용될 가능성이 매우 높다. 미국의 국립표준기술연구소(NIST)는 온라인에서 유출된 비밀번호 등으로 인한 개인정보의 유출을 막기 위해서 다요소인증(MFA, Multi-Factor Authentication)의 사용을 권고하고 있다[3].

MFA는 사용자를 인증하기 위해 지식(Something you know), 소유(Something you have) 및 존재(Something you are)에 속하는 2개 이상의 요소를 이용한다. 예를 들어, 비밀번호(지식)와 하드웨어 토큰(소유) 등 2개의 요소를 이용해 사용자를 인증하면 MFA에 해당한다. 이메일 인증과 SMS를 포함한 전화인증 등은 장치의 소유를 증빙하기 어렵고 보안공격에 취약하므로 대역 외(Out of band) 인증요소로 인정되지 않는다[3].

하나의 클리덴셜과 한번의 로그인으로 여러 서비스에 액세스할 수 있는 통합인증(Single Sign-on) 환경에서, MFA의 중요성은 더욱 커진다. 중간자 공격(Man-in-the-middle attack) 등으로 사용자의 크리덴셜이 탈취되면, 공격자는 희생자(Victim)가 접속 권한을 갖는 모든 서비스에 접근할 수 있으므로 보안 침해가 발생한다[4]. 특히, 통합인증의 적용범위가 멀티 보안도메인으로 확장되면 보안사고 대응을 위한 시간적 비용이 크게 증가하므로, MFA의 도입을 적극적으로 고려해야 한다.

기관 연합형 통합인증(Federated single sign-on), 즉 연합인증[5]은 학연 분야에서 많이 사용되는 멀티 보안도메인 간 사용자 인증체계이다. 연합인증은 신원정보를 제공하는 아이디제공자와 신원정보를 활용하는 서비스제공자가 물리적으로 분리되어 있으며, 양자 간에 신뢰를 기반으로 동작한다는 특징이 있다. 국립표준기술연구소는 NIST SP 802-63-3[3]의 부록에서 연합인증의 구성요소와 OTP(One Time Password) 등 MFA에서 사용되는 인증자(Authenticator)의 보증등급을 규정하여, 신뢰기반의 신원관리 환경에서 연합인증과 MFA의 안전한 사용을 권고하고 있다.

또한 국제 연구교육연합(REFEDS, Research and Education Federations)은 아이디제공자와 서비스제공자가 분리된 연합인증 환경에서 서비스제공자가 MFA의 실행을 요청하고 아이디제공자가 MFA의 실행을 주장할 수 있도록 MFA 프로파일[6]을 개발하여 국제 학연기관과 서비스제공자에게 보급하고 있다. 서비스제공자는 아이디제공자의 주장을 신뢰하며, 정의된 프로파일의 구문에 따라 MFA의 실행 여부를 파악할 수 있다.

하지만, 상용 MFA 솔루션은 도입비용이 높고 운영에 추가적인 비용이 요구되므로 연합인증 환경에서 큰 규모의 학·연 기관이 MFA를 도입해야 할 경우에는 비용문제가 MFA의 활성화에 장벽으로 작용할 수 있다. 또한 MFA 프로파일을 준용하지 않는 아이디제공자와 서비스제공자는 MFA의 실행요청과 실행응답을 확인할 수 없으므로 최악의 경우에는 두 제공자가 각각 MFA를 실행해 사용자 편의성이 크게 악화될 수 있다.

본 논문은 연합인증 환경에서 아이디제공자에 탑재되는 비용 효율적이고 사용자 친화적인 MFA 검증자(Verifier)와 MFA를 지원하지 않은 아이디제공자를 대신해 MFA를 실행시킬 수 있는 중앙형 대리인증자(Proxied authenticator)의 개발 내용을 소개한다. 또한 다수의 아이디제공자가 MFA 프로파일을 준용하지 않는 연합인증 환경에서 서비스제공자가 대리인증자의 환경설정을 통해 대리인증자에게 MFA 프로파일을 위임하는 방법을 제안한다.

본 논문이 기여하는 점은 다음과 같다. 첫째, RFC 보안권고 사항[7]을 준수하고 세션관리, 사용자 잠금, 우회 토큰(Bypass token) 등의 사용자편의 기능을 갖추었으며 SAML(Security Assertion Markup Language) 표준 기반의 연합인증 환경에서 공개소프트웨어의 형태로 활용할 수 있는 아이디제공자용 OTP(One Time Password) 소프트웨어를 개발했다. 둘째, 서비스제공자를 대리해 MFA를 요청하거나 아이디제공자를 대신해 OTP 검증자를 선택적으로 실행시키고 MFA 요청에 응답할 수 있는 대리인증자를 최초로 개발하고 실제 연합인증 환경에 적용했다.

본 논문은 다음과 같이 구성되어 있다. 먼저, 제 2장에서는 본 논문에서 사용되는 용어를 정리하고 유사 연구를 살펴본다. 연합인증 환경에서 MFA 소프트웨어가 갖춰야할 특성과 상세한 설계 내용은 제 3장과 제 4장에서 각각 설명한다. 제 5장에서는 소프트웨어의 구현 결과를 평가하고, 마지막으로 제 6장에서 결론을 맺는다.

Ⅱ. 용어 정의 및 관련 연구

2.1 용어 정의

본 논문은 SAML 기반의 연합인증 환경에서 MFA

를 구동하기 위한 아이디제공자용 OTP 소프트웨어와 MFA 프로파일을 준용하고 MFA를 대리 실행할 수 있는 대리인증자의 개발 내용을 소개한다.

본 논문에서 사용되는 용어를 다음과 같이 정의한다.

- 연합인증(Federated single-sign on): 국제표준 보안 인증 규약인 SAML과 OIDC(OpenID Connect)등을 이용해 통합인증의 범위를 멀티도메인으로 확장한 통합인증 체계이다.
- 신원연합(Identity federation): 동일한 정책 프레임워크와 기술 프로파일의 사용에 동의한 아이디제공자와 서비스제공자의 집합이다. 신원연합 내의 아이디제공자와 서비스제공자 간에 연합인증이 가능하다.
- 아이디제공자(Identity provider): 신원정보를 처리해 사용자를 인증하며 서비스제공자에게 사용자의 인증정보와 신원정보(예, 전자우편 주소 등)를 전달하는 SAML 개체(Entity)이다.
- 서비스제공자(Service provider): 아이디제공자가 전달한 인증정보와 신원정보의 무결성을 검사하고 사용자에게 접근 권한을 부여하는 SAML 개체이다.
- 대리 구조(Proxied architecture): 서비스제공자와 아이디제공자가 메시지를 교환하기 위해서 반드시 대리자(Proxy)를 거쳐야 하는 신원연합의 구조이다. 대리자가 메시지를 중계하므로 Hub-and-Spoke 구조로도 불린다.
- 그물망 구조(Mesh architecture): 서비스제공자와 아이디제공자가 대리자를 거치지 않고 메시지를 교환하는 신원연합의 구조이다.
- 시간기반 일회용 비밀번호(TOTP, Time-based one-time password): TOTP는 RFC 6238에 정의되어 있으며 다음 식에 의해 OTP 코드인 $TOTP_v$이 생성된다.

$$TOTP_v = Truncate(HMAC_H(K, \frac{(T_c - T_o)}{T_x}))$$

해시함수 HMAC(Hashed Message Authentication Code)은 RFC 2014에 정의되어 있다. K는 비밀키이고 T_c는 현재의 Unix 시간이며 T_0는 기준시간이다. RFC 6238은 OTP 코드의 갱신 주기 또는 시간 간격(Time step) T_x를 30초로 권장한다.

2.2 관련 연구

다수의 RFC 문서는 OTP 인증규약의 안전한 구현을 위해 OTP의 무작위성(Randomness), 길이(Length), 재시도횟수(Retry attempts), 일회성, 유효기간(Expiration), 갱신 주기(Renewal interval) 등 6개 규칙의 준용을 권장하고 있다[7]. 본 논문은 위 6개 규칙을 고려해 OTP 소프트웨어를 개발했다. 관련 연구와의 기능을 비교하면 표 1과 같이 요약할 수 있다.

OTPaaS[8]은 OTP 계정관리의 번거로움을 해소하고 OTP의 도입과 관리에 필요한 비용을 절감시키는 것을 목표로 제안된 클라우드 기반의 OTP 검증자 서비스이다. 하지만, OTPaaS는 아이디제공자의 OTP 소프트웨어가 아니며 MFA 프로파일을 지원하지 않는다. 또한 본 연구가 MFA를 지원하지 않는 아이디제공자를 대신해 대리인증자가 MFA를 실행하는데 초점을 두고 있는 반면에 OTPaaS는 아이디제공자에 종속되지 않은 독립형 서비스라는 점에서 본 논문과 차이가 있다.

CILogon[9]은 대리 구조를 갖는 연합인증 환경에서 MFA 프로파일을 지원하는 대리자로 동작한다는 점에서 본 연구와 유사하다. 하지만 대리자에서 대리인증자 또는 MFA 검증자를 제공하지 않는 것으로 판단되며 RequestedAuthnContext도 지원하지 않는다는 점에서 본 연구와 차이가 있다. 서비스제공자가 아이디제공자에게 MFA의 실행을 요청할 때 전송하는 SAML 메시지에는 그림 1과 같은 RequestedAuthnContext 요소를 포함한다.

MFA 제공자[10]는 OTP, FIDO2[11], 푸시알림(Push notification) 등의 2차 인증요소를 제공하는 독립형(Standalone) MFA 플랫폼으로서 Shibboleth 아이디제공자와 연동해 MFA 프로파일을 지원한다는 점에서 본 논문과 유사하다. Shibboleth는 SAML 소프트웨어의 한 종류이다. 그러나 본 연구는 OTP 검증자를 아이디

표 1. 관련 연구의 비교
Table 1. Comparison with related work

Software or service	OTPaaS[8]	CILogon[9]	MFA[10]	Proposed
MFA software for identity provider	×	×	×	○
MFA software for proxy	×	×	×	○
Standalone MFA platform or cloud service	○	×	○	×
Operating as a proxy	×	○	×	○
Support for MFA profile	×	△	×	○

```
<saml2p:RequestedAuthnContext>
  <saml2:AuthnContextClassRef>
     https://refeds.org/profile/mfa
  </saml2:AuthnContextClassRef>
</saml2p:RequestedAuthnContext>
```

그림 1. SAML 요청 메시지의 표준 MFA 문맥 (https://refeds.org/mfa)
Fig. 1. Standard MFA syntax (https://refeds.org/mfa) in the SAML request message

제공자의 소프트웨어 모듈로 구현했으며, API(Application Programming Interface)를 통해 상용 MFA 플랫폼에서 제공하는 FIDO2, TOTP, SMS OTP 등을 활용할 수 있다는 점에서 MFA 제공자와 차이가 있다. 또한 MFA 제공자는 대리인증자를 지원하지 않는다.

Ⅲ. 고려 사항

본 장은 연합인증 환경에서 MFA 소프트웨어를 개발할 때 고려해야 할 사항에 대해서 기술한다. 본 연구에서 2차 인증자(Second factor authenticator)는 TOTP를 이용한다. TOTP는 다양한 형식의 인증 토큰(예, 하드웨어, 모바일 앱 등)을 지원하고 구현이 용이한 장점이 있다.

P1: 비용 효율성(Efficiency) - 일반적으로 MFA 솔루션의 도입과 운영에 필요한 비용은 기관 구성원의 수에 비례하므로 신원연합에 참여 중인 아이디제공자 기관의 규모가 클수록 비용문제가 MFA의 활용을 어렵게 할 가능성이 있다. 상용 솔루션의 도입과 별개로 공개소스를 활용해 개발된 비용 효율적인 MFA 소프트웨어가 필요하다. 개발될 소프트웨어는 신원연합에 적용할 수 있어야 한다.

P2: 보안성(Security) - MFA 소프트웨어는 OTP의 보안권고 사항[7,12]을 준용해야 한다. 보안권고 사항의 내용은 다음과 같다.

P2-1: OTP 무작위성 - 암호학적으로 강인한 유사 랜덤 발생기를 사용해 OTP 코드를 무작위 값으로 생성해야 한다.

P2-2: OTP 길이 - 추측 공격(Guess attack)이나 무작위 대입공격(Brute force attack)을 방어하기 위해 TOTP 코드는 최소 6자리 이상이어야 한다.

P2-3: 재시도 횟수 - 무작위 대입공격(Brute force attack)을 방지하기 위해 한 번의 로그인 과정에서 시도할 수 있는 OTP 코드의 입력 횟수를 제한해야 한다.

P2-4: 일회성 사용 - 로그인 세션이 유지되는 동안 탈취된 코드가 재사용되는 것을 막기 위해 생성한 OTP 코드는 한 번만 사용해야 한다.

P2-5: OTP 유효기간 - TOTP 알고리즘으로 생성한 OTP 코드는 사용 기한이 만료되면 사용할 수 없어야 한다.

P2-6: OTP 갱신주기 - TOTP 알고리즘으로 생성한 OTP 코드는 30초 이하의 시간 동안에만 유효하다.

P3: 가용성(Availability) - 신원연합에 포함된 많은 아이디제공자들이 MFA를 지원하지 않지만 관리주체가 서로 달라 모든 아이디제공자들에게 MFA를 일괄적으로 적용하기는 어렵다. MFA를 개별 서비스에서 구현하면 사용자가 여러 개의 인증 토큰(예, 스마트폰에 설치된 OTP 앱)을 보유해야 하는 문제가 있다. 아이디제공자들이 MFA를 지원하지 않는 연합인증 환경에서도 MFA를 필요로 하는 서비스제공자들을 위해 2차 인증을 강제함으로써 MFA의 가용성을 높일 수 있는 효과적인 메커니즘이 필요하다.

P4: 호환성(compatability) - MFA 프로파일을 따르는 아이디제공자와 서비스제공자는 상대 개체의 MFA 요청과 실행 여부를 확인할 수 있다. 신원연합에는 MFA 프로파일을 준용하는 개체와 준용하지 않는 개체들이 혼재되어 있다. 아이디제공자가 MFA 실행하지만 MFA 프로파일을 준용하지 않을 경우, 서비스제공자는 해당 아이디제공자가 MFA를 실행했는지 확인할 수 없으므로 MFA가 실행되지 않은 것으로 간주하고 사용자의 서비스 접근을 제한할 수 있다. MFA 프로파일을 준용하지 않는 개체들에게도 MFA 프로파일과의 논리적인 호환성을 제공해야 한다.

Ⅳ. 소프트웨어 설계

본 장에서는 비용 효율성, 보안성, 가용성 및 호환성을 고려하여 설계한 MFA 소프트웨어의 세부 내용을 자세히 살펴본다. 본 장의 D1부터 D4는 이전 장의 P1부터 P4와 상응한다.

Shibboleth와 simpleSAMLphp[13]는 연합인증 환경에서 범용으로 사용되는 SAML 소프트웨어이다. 본 연구는 대리자(Proxy)로 활용할 수 있으며 모듈 형태의 소프트웨어 구현이 용이한 simpleSAMLphp를 SAML 소프트웨어로 이용한다. 대리자는 사용자를 인증인가(Authentication and authorization)하는 과정에서 서비스제공자와 아이디제공자를 중개하는 서비스이다. 인증규약의 처리와 신원연합과의 연동 등을 대리자에게

위임함으로 서비스제공자의 기능을 간소화할 수 있다. 또한 인증인가 절차의 중앙 집중화를 통해 관리의 용이성을 향상시킬 수 있다.

D1: 비용 효율성과 사용자 편의성을 높이기 위해 독립형 MFA 플랫폼을 개발하지 않고 그림 2와 같이 사용자가 로그인하는 과정에서 TOTP 계정을 생성하도록 설계한다. 그림 2의 등록 과정은 MFA를 필요로 하는 서비스제공자의 요청이나 관리자의 설정에 의해 실행되고 TOTP 계정이 신규로 생성되는 경우에만 실행된다. 상용 MFA 플랫폼을 보유한 기관의 아이디제공자는 오프라인으로 MFA 계정을 등록한 후에 API(Application Programming Interface)를 통해 TOTP, FIDO2, SMS OTP가 작동하도록 구현함으로써 활용성을 높인다.

그림 2의 TOPT의 등록 과정은 보안성과 유용성(Usability) 간에 트레이드오프를 갖는다. 예를 들어, 아이디와 비밀번호가 유출된 사용자보다 공격자가 먼저 TOTP를 모바일 장치에 등록하면 그림 2의 등록 과정은 TOTP를 무효화시킬 수 있다. 일반적으로 OTP의 등록과정은 유용성을 악화[14]시키는 요소이므로 본 연구에서는 수동적인 방법으로 위 문제점을 회피함으로써 유용성을 높인다. 즉, 그림 2에서 사용자가 TOTP를 등록하면 아이디제공자는 해당 사용자의 전자우편 주소로 계정 잠금 링크를 발송한다. 전자우편을 수신한 사용자가 보안공격으로 판단하면 잠금 링크를 클릭해 자신의 계정을 잠글 수 있게 한다.

본 연구에서 TOTP는 Google Authenticator(GA)를 이용해 구현한다. GA는 OTP 검증자와 인증자의 소스

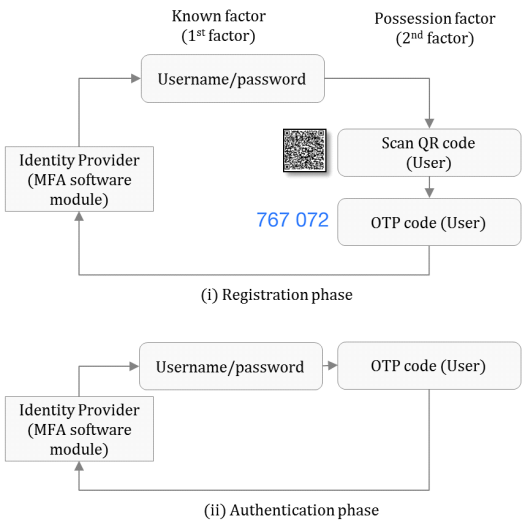

그림 2. TOTP 등록(위) 및 2차 인증(아래) 개략도
Fig. 2. TOTP registration (above) and second-factor authentication (below) schematic

코드와 소프트웨어가 모두 공개되어 있고 대다수의 모바일 장치에서 사용할 수 있다. 또한 인터넷이 연결되어 있지 않아도 실행이 가능하다는 장점이 있다.

공개소스로 제공되는 GA 검증자는 QR 코드를 생성하기 위해서 googleapis.com이나 qrserver.com 등에서 제공하는 API에 의존한다. 하지만 그림 3에서 알 수 있듯이 QR 코드를 생성하기 위해서는 사용자 아이디와 비밀키가 API 서버에 전달되어야 한다. 해당 서버가 침해(Compromised)되었다면 비밀키가 유출될 수 있는 문제점이 있다. 본 연구에서는 아이디제공자가 외부의 API 서버를 이용하지 않고 자체적으로 QR 코드를 생성할 수 있도록 개발하여 앞서 언급한 비밀키의 유출문제를 완화한다.

마지막으로, 모바일 장치를 분실하거나 일시적으로 장치를 사용할 수 없는 사용자들을 위해 일정 기간 동안만 유효한 우회 코드(Bypass code)를 제공하여 사용자 편의성을 높인다. 보안성을 높이기 위해 우회 코드의 발급과 관리는 관리자가 담당한다.

D2: 본 연구에서는 TOTP의 코드 길이와 갱신 주기를 각각 여섯 자리(P2-2)와 30초(P2-6)로 설정해 OTP의 보안권고 사항을 준수한다. 생성된 TOTP 코드는 만료기간이 지난 후에는 사용할 수 없으며(P2-5) 마지막으로 사용된 TOTP 코드는 재사용할 수 없도록 제한(P2-4)한다. 또한 로그인 세션(Session) 동안 TOTP 코드를 입력할 수 있는 최대 횟수를 n번으로 제한하고 위반 시에는 m초 간 사용자 계정을 잠글 수 있도록 소프트웨어 모듈을 개발해 P2-3을 충족시킨다.

동일한 로그인 세션 동안 2차 인증자(예: OTP)의 상태 정보를 1차 인증자와 별도로 관리하여 1차 인증자(예: 비밀번호)와 독립적으로 잠금 기능이 유지될 수 있도록 한다. simpleSAMLphp에서 1차 인증자는 authSource 필터를 상속받아 구현한다. 2차 인증자는 authSource 필터나 authProc 필터를 이용해 구현한다. 각 필터는 추상화된 PHP 클래스이다. 2차 인증자가 authSource 필터를 상속할 경우, 1차 인증자로 부터 상태 정보를 분리하기 어려우므로 본 연구에서는 2차 인증자의 구현을 위해 authProc 필터를 이용한다.

otpauth://totp/kafe_member_org_otp:**student4**?secret=**MJ2TXQPRXUAPA2ZJ**&issuer=kafe_member_org_otp

그림 3. QR 코드 및 내용의 예시
Fig. 3. An example QR code and its content

authProc 필터를 상속받으면 2차 인증자 외에도 접근 제어, 속성 값의 변환 등 다양한 기능의 소프트웨어 모듈을 구현할 수 있다.

OTP 코드의 무작위성(P2-1)은 hash_hmac() 함수의 무작위성에 의존한다. GA의 TOTP 코드가 0으로 시작하지 않아 무질서도(Entropy)가 낮다는 연구[15]가 있지만, 현재는 GA의 TOTP 코드가 0으로 시작하므로 무질서도가 낮은 문제는 해결된 상태이다. 무질서도가 감소하면 무작위 대입공격에 더 쉽게 노출될 수 있다. 무작위 대입공격은 P2-3으로 방어한다.

D3: MFA를 필요로 하는 서비스를 위해, 그림 4의 MFA 모듈인 대리인증자가 아이디제공자(그림 4의 ③)를 대신하여 2차 인증을 수행한다. 아이디제공자가가 2차 인증을 수행하면 대리인증자는 2차 인증을 생략한다. 대리자는 authProc 필터를 상속받아 구현한 다수의 소프트웨어 모듈을 탑재할 수 있다. 각 모듈은 순차적으로 실행된다.

개별 서비스가 대리자의 기능을 자율적으로 활용할 수 있도록, 대리자 제어기(Proxy controller)는 서비스제공자(그림 4의 ①)별로 테넌트(Tenant)를 구분하여 관리한다. 개별 서비스의 소유자가 테넌트 관리자가 된다. 즉, 테넌트 관리자가 대리자 제어기(Proxy controller)에서 설정한 환경변수에 따라 개별 소프트웨어 모듈이 동작한다. 테넌트 관리자는 소유한 서비스제공자에 대해 MFA의 활성화 여부, TOTP 코드의 재시도 횟수, TOTP 세션의 유효 시간, 접근제어 필터 등을 설정할 수 있다. 접근 제어 필터를 사용하면, 테넌트 관리자는 아이디제공자, IP 주소 범위, 사용자 속성 값 등을 기준으로 일반 사용자의 서비스 접근을 정밀하게 제어할 수 있다. TOTP 세션의 유효 시간 동안에는 TOTP를 재실행하지 않으므로, 사용자 편의성을 높일 수 있다.

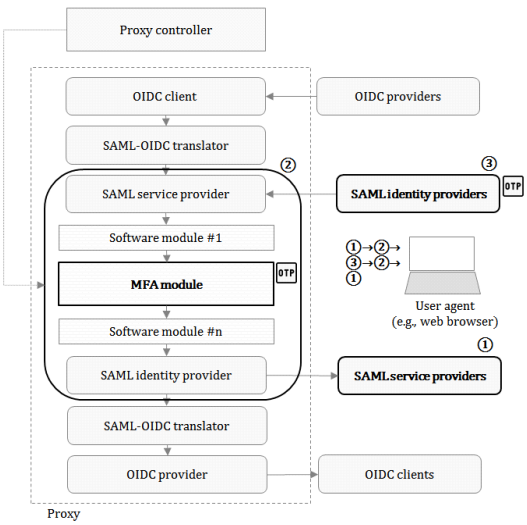

그림 4. MFA 모듈을 통합한 대리 구조의 설명
Fig. 4. Illustration of the proxied architecture incorporating the MFA module

D1에서 설계한 MFA 소프트웨어를 재사용하기 위해, 대리인증자에서도 그림 2와 동일한 방식으로 TOTP를 등록하거나 인증하도록 개발한다. 또한, 대리인증자에서 TOTP의 제공을 위해 필요한 비밀키의 등록, 재발행, 폐기에 대한 권한을 시스템 관리자에게만 부여함으로써 사용자 편의성은 희생시키지만 보안성을 높이는 방향으로 설계한다. 사용자는 해당 권한을 갖지 않는다. 추가로 보안성을 높이기 위해, 대리인증자는 TOTP 시도횟수 초과로 인한 잠금, 테넌트 관리자에 의한 잠금, 시스템 관리자에 의한 잠금 등 3가지 방식의 사용자 잠금 기능을 제공한다. 시도횟수 초과로 인한 잠금은 설정된 시간 이후에 자동으로 해제되며, 기타 잠금은 관리자에 의해서만 해제할 수 있다.

대리자는 SAML 인증규약과 OIDC 인증규약을 함께 지원하며 두 규약이 호환될 수 있도록 메시지 변환기(SAML-OIDC translator)를 제공한다. 본 논문은 SAML을 기준으로 MFA의 실행 절차를 설명한다. 그림 4의 ①, ②, ③으로 표시된 개체들은 서버 간에 직접 통신하지 않고 웹 브라우저를 통해 메시지를 교환한다. 서비스제공자로부터 인증요청을 받은 시점부터 아이디제공자로부터 응답 메시지를 수신할 때까지만(그림 4의 ②→③→②) 대리자가 로그인 세션을 유지하도록 설계한다. 대리자에서 로그인 세션을 길게 유지하는 경우, 연동된 서비스제공자들(그림 4의 ①)의 싱글로그인(Single login)과 싱글로그아웃(Single logout)을 관리하는 것이 복잡해지므로 일시적으로만 유지하도록 설계한다.

대리인증자는 MFA를 필요로 하는 서비스제공자(S, 그림 4의 ①)와 MFA를 제공하는 아이디제공자(I, 그림 4의 ③)를 사전에 인지하고 있어야 한다. 대리자가 협약을 통해 아이디제공자와 서비스제공자를 수용하기 때문에 S와 I를 알 수 있다. 아이디제공자는 신원연합의 운영자(Operator)를 통해 S에 대한 정보를 얻는다고 가정한다. D4의 MFA 프로파일이 지원되지 않는 경우, 실시간으로 S와 I에 대한 정보를 얻기 어려워 아이디제공자와 대리인증자에서 MFA 설정이 중복될 가능성이 있다.

아이디제공자(그림 4의 ③)와 대리자(그림 4의 ②)는 $\exists x, x \in S$의 인증요청에 대해서 각각 TOTP를 실행

$(f_{idp}(x) : x \rightarrow otp(x))$하거나 $\exists y, y \in I$에서의 인증응답 $y(x)$에 대해 TOTP를 실행하지 않도록 $(\tilde{f}_{proxy}(y(x)) : y(x) \rightarrow \sim otp(y(x)))$ 설정할 수 있어야 한다. 또한 $\exists x, x \in S$에 대해 $\tilde{f}_{idp}(x)$와 $\exists z, z \in \bar{I}$에 대해 $f_{proxy}(z(x))$의 구성도 가능하게 설계해 MFA의 지원 여부가 혼재된 연합인증 환경에서 MFA 프로파일을 유연하게 적용할 수 있어야 한다. $\bar{I}$는 MFA를 지원하지 않는 아이디제공자의 집합이다.

아이디제공자나 대리자 제어기에서 MFA를 수동으로 설정하는 방식은 $\exists x, x \in S$에 대해 $\tilde{f}_{idp}(x)$와 $\exists z, z \in (\bar{I} \cup I)$에 대해 $\tilde{f}_{proxy}(z(x))$가 되어 MFA를 실행하지 않거나 $\exists x, x \in S$에 대해 $f_{idp}(x)$와 $\exists z, z \in (\bar{I} \cup I)$에 대해 $f_{proxy}(z(x))$로 설정되어 MFA의 중복 실행이 문제가 된다. 또한 연합인증 환경에서 아이디제공자와 서비스제공자는 1:1로 연동되기 때문에 그림 4의 ③은 그림 4의 ②에 탑재된 서비스제공자만 식별할 수 있는 문제점이 있다. 첫 번째 문제점과 두 번째 문제점은 각각 D4에서 설명할 MFA 프로파일과 그림 5의 SAML Scoping 요소를 활용해 해결한다.

Scoping은 SAML 2.0 인증요청(AuthnRequest) 메시지에 포함되는 선택적 요소(Optional element)이다. 대리자는 아이디제공자(그림 4의 ③)에게 전달할 인증요청 메시지를 구성할 때, 서비스제공자(그림 4의 ①)의 개체식별자(entityID)와 그림 4의 ②에 탑재된 서비스제공자의 개체식별자를 RequesterID에 포함한다. 아이디제공자(그림 4의 ③)는 RequesterID 요소에 포함된 서비스제공자들의 개체식별자들을 이용하여 인증요청 메시지의 최초 발신자를 확인할 수 있다. 아이디제공자와 대리인증자가 MFA를 필요로 하는 서비스제공자를 확인할 때, RequesterID와 Issuer를 모두 활용하도록 개발한다. Issuer 요소는 인증요청 메시지를 보낸 서비스제공자의 개체 식별자를 포함한다.

```
<samlp:Scoping>
<samlp:RequesterID>
  The entityID of ①
</samlp:RequesterID>
<samlp:RequesterID>
  The entityID of the service provider in ②
</samlp:RequesterID>
</samlp:Scoping>
```

그림 5. SAML 인증요청 메시지의 Scoping 요소
Fig. 5. Scoping element within the SAML authentication request message

D4: MFA 프로파일을 준용하지 않는 아이디제공자와 서비스제공자를 대신해 MFA 프로파일을 실행할 수 있도록 대리인증자를 개발한다. 그림 6은 대리인증자가 SAML 인증요청 또는 응답 메시지를 수신했을 때 TOTP를 실행하고 MFA 구문(Syntax)을 추가하는 과정을 보여준다. 대리인증자는 인증요청(AuthnRequest) 메시지를 수신하면 MFA 구문이 인증요청 메시지에 포함되어야 하는지를 결정하고, 필요한 경우 아이디제공자에게 MFA를 요청한다. 만약 서비스제공자가 MFA 구문을 포함해서 인증요청을 했거나 테넌트 관리자가 소유한 서비스제공자에게 MFA가 필요하다고 설정한 경우(그림 6의 ⓐ에서 MFA가 필요한 경우), 대리인증자는 그림 1과 동일한 MFA 구문을 SAML 인증요청 메시지에 포함하여 아이디제공자에게 전달한다.

MFA의 실행을 요청받은 아이디제공자는 다음과 같은 방식으로 MFA를 처리할 수 있다.

i MFA를 실행하고 MFA 프로파일을 준수한다.
ii MFA를 실행하지만 MFA 프로파일을 준수하지 않는다.
iii MFA를 실행하지 않는다.

i 의 경우, 아이디제공자는 응답 메시지에 MFA 구문을 포함하며 ii와 iii은 MFA 구문을 포함하지 않는다.

그림 6의 ⓑ에서 아이디제공자가 $\exists x, x \in S$, $f_{idp}(x)$ 하면 대리인증자는 TOTP를 실행하지 않는다.

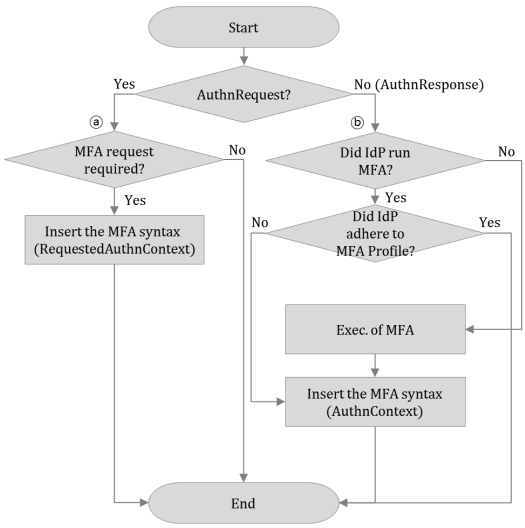

그림 6. 대리 인증자에서 REFEDS MFA 파일을 구현하기 위한 순서도
Fig. 6. Flowchart implementing REFEDS MFA profile in the proxied authenticator

즉, MFA를 실행하고 응답 메시지를 전달한 아이디제공자 i가 $i \in (I - \hat{I})$이거나 대리자에 $i \in \hat{I}$로 등록된 아이디제공자에 대해서 대리인증자는 OTP를 실행하지 않는다. $\hat{I}$는 MFA를 실행하지만 MFA 프로파일을 준용하지 않는 아이디제공자의 집합이다. 대리인증자는 $i \in \hat{I}$가 전달한 인증응답 메시지의 AuthnContextClassRef 값을 MFA 구문으로 수정한 후에 서비스제공자에 전달한다. 아이디제공자는 인증 방법을 표시하기 위해 AuthnContextClassRef 요소를 사용한다.

$i \notin I$인 아이디제공자로부터 응답 메시지를 전달받은 경우, 대리인증자는 TOTP를 실행하고 MFA 구문이 포함되도록 응답 메시지를 수정한 후에 서비스제공자에게 전달한다. MFA 프로파일을 준용하지 않고 대리자에 $i \in \hat{I}$로 등록되지 않은 아이디제공자가 $i \notin I$에 해당한다. 대리인증자는 OTP의 실행 여부를 수동으로 설정하는 방식이 갖는 문제점 중에서 $\exists x, x \in S$에 대해 $\tilde{f}_{idp}(x) \wedge \tilde{f}_{proxy}(x)$인 문제를 해결할 수 있다. 또한 아이디제공자가 MFA를 필요로 하는 $x \in S$를 등록하지 않아도 연합인증 환경에서 MFA와 MFA 프로파일이 작동하므로 아이디제공자에서 $x \in S$의 등록과 관리에 따르는 비용을 줄일 수 있다.

V. 구현 결과

앞 장에서 설계한 D1부터 D4의 구현결과를 비용 효율성, 보안성, 가용성 및 표준과의 호환성 측면에서 살펴본다.

먼저, TOTP 검증자를 simpleSAMLphp의 소프트웨어 모듈 형태로 구현했다. TOTP 비밀키를 모바일 장치에 등록하는 과정(p_r)과 2차 인증을 받는 과정(p_a)을 분리해 구현함으로써 상용 MFA 플랫폼과의 연계 가능성을 높였다. 사용자는 킷값을 등록한 후에 2차 인증(p_r;p_a)을 받거나 등록과정을 생략하고 인증(p_a)을 받을 수 있다. 관리자가 p_r;p_a을 활성화하면 1차 인증 후에 TOTP의 비밀키를 등록할 수 있도록 그림 7의 상단 화면이 가시화된다. 그림 7의 하단은 p_a를 활성화했을 때 나타나는 화면으로 TOTP나 SMS 및 FIDO2를 2차 인증자로 활용할 수 있다. 아이디제공자에 구현된 TOTP를 활성화하거나 상용 MFA 플랫폼을 이용할 경우에 각각 p_r;p_a와 p_a가 적합하다.

비밀키를 모바일 TOTP 앱에 등록하기 위해 그림 7과 같이 QR 코드를 이용한다. googleapis.com 등 API

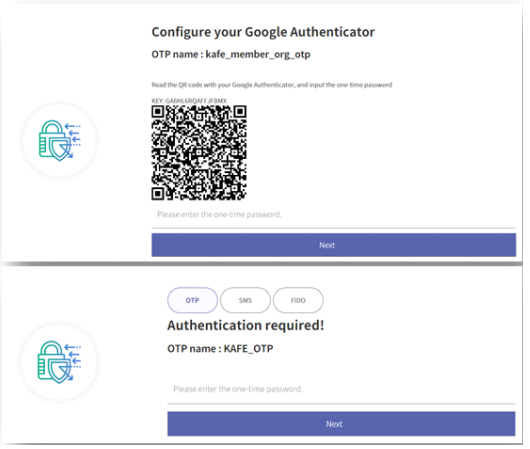

그림 7. 비밀키의 등록(위) 및 3종의 2차 인증요소(아래)
Fig. 7. Registration of a secret key (above) and 3 types of 2nd authentication factors (below)

서버의 보안 침해로 인해 비밀키가 유출되는 것을 방지하기 위해 아이디제공자가 자체적으로 QR 코드를 생성할 수 있도록 소프트웨어 모듈을 개발했다. QR 코드의 생성을 위해 공개소스인 php-otpauth[16]를 이용했다.

OTP의 보안권고 사항을 준용하기 위해 lockDuration(t_l), maxAttempts(a_m), lastCode(c_l), timeSlice(t_s), codeLength(l_c)와 같은 변수를 사용했으며 시간 간격(TOTP의 갱신주기) t_s와 코드길이 l_c는 각각 30초와 6으로 설정했다. TOTP 코드의 재사용을 막기 위해, 마지막으로 2차 인증에 성공한 TOTP 코드(c_l)을 데이터베이스에 저장한다. 대리인증자에 입력된 TOTP 코드가 c_l과 일치한다면, 2차 인증을 거부하도록 개발되었다. 2차 인증을 a_m번 실패하면 t_l 시간 동안 사용자 잠금이 실행되도록 구현해 TOTP 코드 입력의 재시도 횟수를 제한했다.

개발된 TOTP 검증자의 OTP 무작위성을 조사하기 위해 TOTP 코드를 30초 간격으로 총 7,982번 측정했다. 그림 8에서 OTP 코드들이 완벽히 균일하지는 않는 것을 알 수 있다. 여섯 자리 TOTP 코드(예, 123456)의 앞 세 자리(123)를 x축에 배치하고 뒤 세 자리(456)를 y축에 배치해 한 점으로 나타냈다. 하지만 첫 번째 자리가 0을 가질 수 없기 때문에 GA의 무질서도가 낮다는 문제는 해결되었음을 알 수 있다.

여섯 자리 TOTP 코드들이 갖는 최대 무질서도는 19.93($\because \log_2(10^6)$)이지만 개발된 TOTP 검증자의 무질서도는 약 12.96으로 측정되었다. 또한 7,982번의 측정 값 중 TOTP 코드가 총 34회 중복되는 것을 확인할 수 있었다. TOTP 코드가 중복될 확률은 4.26×10^{-3}이

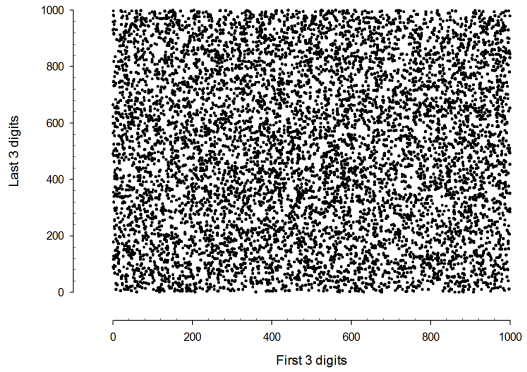

그림 8. 개발된 소프트웨어 모듈에서 측정한 OTP 코드의 분포
Fig. 8. Distribution of OTP codes obtained through the developed software module

다. 무작위성은 프로그래밍 언어의 hash_hmac() 함수(또는 이와 유사한 함수)에 의해 결정되므로 해당 함수를 사용하는 TOTP 소프트웨어는 측정된 결과와 유사한 무질서도를 가질 것으로 판단된다.

MFA의 가용성을 높이기 위해 그림 4와 같이 대리인증자를 개발했다. 시스템 관리자(a_s)와 테넌트 관리자(a_t)는 대리자 제어기에서 대리인증자의 작동방식을 설정할 수 있다. 대리인증자는 대리자(그림 4의 ②)에 속한 아이디제공자에서 2차 인증자로 동작하므로 즉, 해당 아이디제공자에 의해 관리되어야 하므로 a_s와 a_t의 권한을 표 2과 같이 분리했다. a_t는 서비스제공자를 관리한다. 보안성을 높이기 위해 사용자에게는 비밀키 등의 관리기능을 제공하지 않았다. 표 2의 $\hat{I}$는 MFA를 실행하지만 MFA 프로파일을 준용하지 않는 아이디제공자의 집합이다.

표 2에서 요약한 대로, 대리자의 시스템 관리자는

표 2. a_s와 a_t의 권한 비교
Table 2. Comparison of administrative privileges between a_s and a_t

Functions	a_s	a_t
Activation of MFA Profile	○	
Management of $\hat{I}$	○	
OTP Bypass for users	○	
Revoke or reissue of secret keys	○	
Lock or unlock users	○	○
Activation of TOTP		○
TOTP attempt limit		○
TOTP session timeout		○

MFA 프로파일을 활성화하고 $\hat{I}$를 관리하며, 특정 사용자에 대해 OTP 우회 기능을 제공한다. 또한 비밀키의 재발행과 폐기 및 사용자 잠금의 관리가 가능하게 개발되어 보안성을 높였다. 테넌트 관리자는 그림 9와 같이 TOTP의 실행 여부와 TOTP의 재시도 횟수 제한 및 TOTP 세션의 유효 시간(Timeout)을 설정할 수 있다. 유효 시간의 길이가 증가할수록 보안성은 낮아지지만 사용자의 편의성은 증가한다.

앞 장의 D1에서 설명한 바와 같이, 전자우편에 포함된 잠금 링크를 통해 사용자가 직접 계정을 잠그는 경우에는 1차 인증 수단이 유출된 것으로 볼 수 있다. 따라서 계정이 잠긴 사용자는 접근하고자 했던 서비스뿐만 아니라 대리자와 연동된 모든 서비스에 접근할 수 없도록 개발되었다. 그림 9의 필터(Filter)를 이용하여, 테넌트 관리자는 자신이 소유한 서비스제공자에 대한 사용자의 접근을 제어할 수 있다. 정밀한 접근제어를 위해 사용자가 속한 아이디제공자, 사용자의 속성 값 및 사용자의 접근 IP를 활용할 수 있도록 개발했다.

개발된 대리인증자에서 MFA 프로파일이 정상적으로 적용되는지 확인하기 위해서 그림 10과 같은 검증환경을 구성하고 SAML 인증요청과 응답 메시지를 수집했다. OIDC 클라이언트인 ai.kafe.or.kr이 사용자 인증을 요청하지만 본 논문에서는 OIDC의 인증 절차는 설명하지 않고 SAML과 관련된 내용만 기술한다. 테넌트 관리자는 ai.kafe.or.kr를 MFA가 필요한 서비스로 설정했다. ai.kafe.or.kr과 coreen-idp.kreonet.net(그림 10의 coreen-idp)는 MFA 프로파일을 지원하지 않는다.

SAML 메시지는 Chrome 브라우저의 확장 프로그램인 SAML DevTools extension을 이용해 갈무리 했으며 MFA 프로파일의 작동여부만 확인할 수 있도록 편집했다. 메시지의 내용을 살펴봄으로써, MFA의 지원 여부가 혼재한 연합인증 환경에서 대리인증자가 SAML 개체들을 대신하여 MFA 프로파일을 지원하는지 확인하고자 한다.

그림 11은 ai.kafe.or.kr의 요청에 의해 saml과

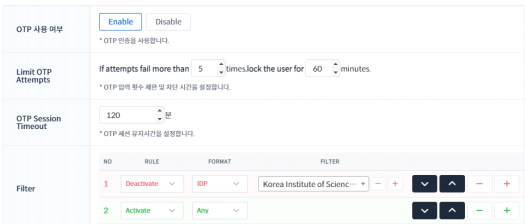

그림 9. 테넌트 관리자에 의해 설정 가능한 TOTP 선택사항
Fig. 9. Configurable TOTP options by tenant administrators

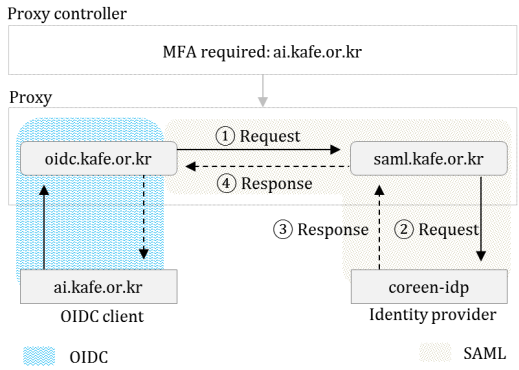

그림 10. 대리자의 SAML 인증요청 및 응답 메시지 처리
Fig. 10. Processing of SAML authentication request and response messages through the proxy

coreen-idp에 순차적으로 전달되는 SAML 인증요청 메시지의 일부를 보여준다. 그림 11의 ①을 통해 ai.kafe.or.kr이 RequestedAuthnContext 요소를 설정하지 않고 인증요청 메시지를 보낸 것을 알 수 있다. 서비스제공자는 아이디제공자에게 요청하는 인증 방법을 RequestedAuthnContext 요소에 명시한다. 그림 11의 ②에서 확인할 수 있듯이, saml.kafe.or.kr은 ai.kafe.or.kr을 대신하여 SAML 메시지에 RequestedAuthnContext 요소를 추가했다. 즉, 대리인증자가 서비스제공자를 대신해 MFA 프로파일을 적용했다.

대리 인증자가 MFA를 요청했지만 아이디제공자인 coreen-idp.kreonet.net은 1차 인증(즉, 비밀번호)만 수행했음을 그림 12의 ③, AuthnContextClassRef의 값을 통해 알 수 있다. AuthnContextClassRef 요소의 값은 아이디제공자에서 사용한 인증 방법을 나타낸다. ai.kafe.or.kr가 대리자 제어기에 MFA가 필요한 서비스로 등록되어 있으므로 대리인증자는 TOTP를 실행한 후에 그림 12의 ④와 같이 AuthnContextClassRef의 값

① Authentication request : Service provider (Proxy/OIDC)

```
<ns0:AuthnRequest
    IssueInstant="2023-06-09T06:37:07Z"
    Version="2.0">
    <ns1:Issuer
        Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
        https://oidc.kafe.or.kr/Saml2/sp
    </ns1:Issuer>
</ns0:AuthnRequest>
```

② Authentication request : Service provider (Proxy/SAML)

```
<samlp:AuthnRequest
    Version="2.0"
    IssueInstant="2023-06-09T06:37:11Z">
    <saml:Issuer>https://saml.kafe.or.kr/sp/simplesamlphp</saml:Issuer>
    <samlp:RequestedAuthnContext>
        <saml:AuthnContextClassRef>
            https://refeds.org/profile/mfa
        </saml:AuthnContextClassRef>
    </samlp:RequestedAuthnContext>
    <samlp:Scoping>
        <samlp:RequesterID>https://oidc.kafe.or.kr/Saml2/sp</samlp:RequesterID>
    </samlp:Scoping>
</samlp:AuthnRequest>
```

그림 11. 인증요청 메시지
Fig. 11. Authentication request messages

③ Authentication response : Identity provider

```
<samlp:Response
    Version="2.0"
    IssueInstant="2023-06-09T06:37:11Z">
    <saml:Issuer>
        https://coreen-idp.kreonet.net/idp/simplesamlphp
    </saml:Issuer>
    <samlp:Status>
        <samlp:StatusCode
            Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
    </samlp:Status>
    <saml:Assertion>
        <saml:AuthnStatement>
            <saml:AuthnContext>
                <saml:AuthnContextClassRef>
                    urn:oasis:names:tc:SAML:2.0:ac:classes:Password
                </saml:AuthnContextClassRef>
            </saml:AuthnContext>
        </saml:AuthnStatement>
    </saml:Assertion>
</samlp:Response>
```

④ Authentication response : Identity provider (Proxy/SAML)

```
<samlp:Response
    Version="2.0"
    IssueInstant="2023-06-09T06:37:40Z">
    <saml:Issuer>
        https://saml.kafe.or.kr/idp/simplesamlphp
    </saml:Issuer>
    <samlp:Status>
        <samlp:StatusCode
            Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
    </samlp:Status>
    <saml:Assertion>
        <saml:AuthnStatement>
            <saml:AuthnContext>
                <saml:AuthnContextClassRef>
                    https://refeds.org/profile/mfa
                </saml:AuthnContextClassRef>
                <saml:AuthenticatingAuthority>
                    https://coreen-idp.kreonet.net/idp/simplesamlphp
                </saml:AuthenticatingAuthority>
            </saml:AuthnContext>
        </saml:AuthnStatement>
    </saml:Assertion>
</samlp:Response>
```

그림 12. 인증응답 메시지
Fig. 12. Authentication response messages

을 MFA 구문으로 수정했다. 즉, 대리인증자가 아이디제공자를 대신해 TOTP를 실행하고 MFA 프로파일을 적용했다.

MFA를 실행하지만 MFA 프로파일을 준용하지 않는 아이디제공자도 그림 12의 ③과 유사한 SAML 응답 메시지를 생성한다. 대리자는 해당 아이디제공자가 MFA를 실행한다는 것을 사전에 알고 있으므로 TOTP를 실행하지 않고 AuthnContextClassRef의 값을 MFA 구문으로 변경한다.

Ⅵ. 결 론

본 논문은 연합인증 환경에서 아이디제공자에 내장되어 MFA를 실행하는 TOTP 검증자와 MFA를 지원하지 않는 아이디제공자들을 대신해 MFA를 실행할 수 있는 대리인증자의 개발 내용을 상세히 소개했다. 또한 TOTP 검증자와 대리인증자를 실증환경과 운영환경에 적용해 획득한 결과를 바탕으로 개발된 소프트웨어의 비용 효율성, 보안성, 가용성 및 호환성을 확인했다. 본 연구를 통해 연합인증 환경에서 이용되는 사용자 크리덴셜과 데이터의 보호 수준을 크게 높일 수 있을 것으로 기대한다.

References

[1] G. Bassett, C. D. Hylender, and P. Langlois,

"*Data Breach Investigation Report*," Retrieved May 2, 2022 from https://www.verizon.com/business/resources/reports/dbir/2022/master-guide/.

[2] K. Thomas, et al., "Protecting accounts from credential stuffing with password breach alerting," in *Proc. 28th USENIX Secur. Symp.*, pp. 1555-1571, 2019.

[3] P. A. Grassi, M. E. Garcia, and J. L. Fenton, "NIST special publication 800-63-3: Digital identity guidelines," *National Inst. Std. and Technol.*, U.S. Department of Commerce, Jun. 2017. (https://doi.org/10.6028/NIST.SP.800-63-3)

[4] N. M. Karie, V. R. Kebande, R. A. Ikuesan, M. Sookhak, and H. S. Venter, "Hardening SAML by integrating SSO and multi-factor authentication (MFA) in the cloud," in *Proc. 3rd Int. Conf. NISS*, pp. 1-6, Mar. 2020. (https://doi.org/10.1145/3386723.3387875)

[5] J. Jo, H. Jang, J. Kong, and Y. Chae, "Federated IAM service of KAFE identity federation," *J. KICS*, vol. 43, no. 12, pp. 2200-2214, 2018. (https://doi.org/10.7840/kics.2018.43.12.2200)

[6] *REFEDS MFA Profile*, Retrieved Mar. 2, 2023 from https://refeds.org/profile/mfa.

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

[8] E. Emir and T. S. Mehmet, "OTPaaS – one time password as a service," *J. IEEE Trans. on Inf. Forensics and Secur.*, vol. 14, no. 3, 2019. (https://doi.org/10.1109/TIFS.2018.2866025)

[9] J. Basney, H. Flanagan, T. Fleury, J. Gaynor, S. Koranda, and B. Oshrin, "CILogon: Enabling federated identity and access management for scientific collaborations," in *Proc. 2019 Int. Symp. Grids and Clouds*, vol. 351, Taiwan, Apr. 2019. (https://doi.org/10.22323/1.351.0031)

[10] R. M. Emerson, et al., "Multi-factor authentication for shibboleth identity providers," *J. Internet Serv. and Appl.*, vol. 11, no. 8, 2020. (https://doi.org/10.1186/s13174-020-00128-1)

[11] S. Srinivas, D. Balfanz, E. Tiffany, and A. Czeski, "*Universal 2nd factor (U2F) overview*," Retrieved Jun. 2, 2022 from https://fidoalliance.org/specs/fido-u2f-v1.2-ps-20170411/fido-u2f-overview-v1.2-ps-20170411.pdf.

[12] M. David, B. Mihir, H. Frank, N. David, and R. Ohad, "*Hotp: An hmac-based one-time password algorithm*," RFC 4226, 2005.

[13] L. Andronache and N. Claudiu, "Web single sign-on implementation using the simpleSAMLphp application," *J. Mobile, Embedded and Distrib. Syst.*, vol. 3, no. 1, pp. 21-29, 2011.

[14] Z. A. Claudia, et al., "2FA might be secure, but it's not usable: A summative usability assessment of Google's two-factor authentication (2FA) methods," in *Proc. Human Factors and Ergonomics Soc. Annu. Meeting*, vol. 62, no. 1, pp. 1141-1145, SAGE Publications, 2018. (https://doi.org/10.1177/154193121862126)

[15] A. Dimitrienko, et al., "Security analysis of mobile two-factor authentication schemes," *Intel Technol. J.*, vol. 18, no. 4, 2014. (https://doi.org/10.1007/978-3-662-45472-5_24)

[16] *php-otpauth*, Retrieved Jun. 7, 2023 from https://github.com/MincDev/php-2-factor-authentication.

조 진 용 (Jinyong Jo)

2013년 : 광주과학기술원 정보통신공학 박사
2003년~현재 : 한국과학기술정보연구원
2016년~현재 : eduGAIN 운영그룹 위원
<관심분야> 신원 관리, 인증 및 인가
[ORCID:0000-0001-6830-3604]

김 승 해 (Seung-Hae Kim)

2008년 : 전북대학교 정보보호공학 박사
1996년~현재 : 한국과학기술정보연구원 책임연구원
2021년~현재 : 한국과학기술정보연구원 연구망서비스팀 팀장
<관심분야> 라우팅 프로토콜 보안, 인증, 망관리, 정보보호
[ORCID:0000-0002-8403-7577]

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]