Smart Media Journal / Vol.9, No.1 / ISSN:2287-1322
http://dx.doi.org/10.30693/SMJ.2020.9.1.30

소유권 확인된 스마트폰을 이용한 강력한 패스워드 생성 및 관리

(Generation and Management of Strong Passwords using an Ownership Verified Smartphone)

박준철*

(Jun-Cheol Park)

요 약

패스워드 자체의 보안성을 높이는 시도와 함께, 패스워드 기반의 인증에 추가적 인증을 강제하는 것은 패스워드 인증 방식의 보안성을 향상하는데 도움이 된다. 본 논문은 강력한 패스워드를 사이트마다 달리 사용하려는 잘 알려진 문제에 대하여, 스마트폰을 이용한 추가적 인증을 내재한 패스워드의 생성 및 관리 기법을 제안한다. 제안 기법은 패스워드 매니저라 알려진 기술과 마찬가지로, 사용자에게 다수의 패스워드를 기억할 것을 요구하지 않으면서 요구가 있을 때마다, 강력한 사이트별 패스워드를 생성해서 사용자에게 제시한다. 기존 기법들과 다른 점은, 제안 기법은 패스워드 추출을 위해 스마트폰 소유권 검증을 통해 인증된 사용자 스마트폰을 통해서만 패스워드 추출 과정 진행을 허용한다는 것이다. 따라서 이중 인증을 강제하지 않거나 지원하지 않는 사이트에 대해서도, 제안 기법 적용 시 이중 인증을 적용해 로그온을 하는 것과 같은 보안성 향상 효과를 낼 수 있다. 또한 제안 기법은 패스워드 추출 서비스를 제공하는 서버나 사용자 스마트폰의 저장 정보 유출 시에도, 공격자가 사용자로 위장하거나 비밀 정보를 탈취하는 것을 막을 수 있다.

■ 중심어 : 패스워드 ; 이중 인증 ; 스마트폰 ; 보안

Abstract

Enforcing additional authentication to password-based authentication, in addition to attempting to increase the security of the password itself, helps to improve the security of the password authentication scheme. For a well-known problem of using strong passwords that differ from site to site, we propose a scheme for password generation and management with an inherent supplementary authentication. Like the so-called password manager, the scheme retrieves and presents a strong site-specific password whenever requested without requiring the user to remember multiple passwords. Unlike the existing methods, however, the scheme permits the password retrieval process to proceed only through the authenticated user's ownership verified smartphone. Hence, even for sites not enforcing or supporting two-factor authentication, the logon process can benefit from the scheme's assurance of enhanced security with its two-factor equivalent authentication. The scheme can also prevent an attacker from impersonating a user or stealing secrets even when the stored information of the server for password retrieval service or the user's smartphone is leaked.

■ keywords : password ; two-factor authentication ; smartphone ; security

Ⅰ. 서론

안전한 패스워드는 랜덤에 가까운 값이어야 하는데 반해, 사용자들은 기억하기 쉽도록 어떤 의미나 패턴을 내재한 문자열을 패스워드로 사용하기 원한다. 그 결과 사전(dictionary) 공격이나 수집된 피해자의 개인 정보들의 조합을 차례로 시도해 보는 공격의 성공 가능성을 무시할 수 없게 된다. 더욱 위험한 것은, 일반 사용자들은 여러 사이트들에 같은 패스워드를 쓰는 경향이 있어, 보안이 취약한 한 사이트의 패스워드가 노출되면 이를 다른 사이트들에 대입해보는 크리덴셜 스터핑(credential stuffing)이 공격자에게 매우 유용하다는 점이다[1-3].

* 정회원, 홍익대학교 컴퓨터공학과 교수
이 논문은 2019학년도 홍익대학교 학술연구진흥비에 의하여 지원되었음.

접수일자 : 2020년 02월 05일
수정일자 : 2020년 02월 28일
게재확정일 : 2020년 03월 03일
교신저자 : 박준철 e-mail: jcpark@hongik.ac.kr

최근 휴대폰 SMS(Short Message Service) 기반의 추가 인증을 채택하는 이중 인증(Two-Factor Authentication, 이하 2FA) 방식이 패스워드 기반 단일 인증의 보안성을 높이는 수단으로 널리 쓰이고 있다. 하지만 사용자의 불편함이 증가한다는 이유로 2FA를 강제하지 않거나 또는 아예 지원하지 않는 사이트들이 많기에, 2FA의 향상된 보안성을 누리지 못하는 사용자들이 다수 존재한다. 본 논문은 널리 알려진, 강력한 패스워드를 사이트마다 다르게 쓰면서도 사용자가 이들을 기억할 필요가 없게 하려는 문제를 다룬다. 이 문제 해결을 위해, 스마트폰 소유권 인증이 사이트별 패스워드 추출 과정에 선행되도록 하여, 사이트의 2FA 제공이나 사용자의 2FA 채택 여부에 상관없이 로그온 과정에서 2FA를 거친 것과 같은 보안성을, 감내할만한 수준의 사용 편의성을 유지하면서 제공하는 기법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서 기존 패스워드 보안성 향상 연구와 주요 2FA 방식 및 문제점들을 제안 기법과 비교하여 분석한다. 3장에서는 스마트폰의 소유권 인증을 이용해 어떻게 강력한 패스워드를 요구 시마다 생성하고, 이를 안전하게 관리하는지를 설명한다. 이어서 4장에서는 제안 기법의 보안성을 가능한 공격 유형별로 분석하고, 5장에서 결론을 맺는다.

Ⅱ. 관련연구

1. 패스워드 인증의 보안성 및 사용 편의성 개선

어떤 사이트의 (해시 함수 적용된)패스워드 데이터베이스가 유출된 후 사전 공격으로 일부 패스워드들을 확보하거나 피싱(phishing) 등의 방법으로 사용자를 속여서 패스워드를 빼내는 등의 경로로 취득한 사용자의 패스워드를 가지고, 공격자는 다른 사이트들에 로그온이 가능한지 확인한다.

이 문제에 대처하기 위해, 기억할 필요 없고 깨기 어려운 패스워드를 사이트마다 다르게 사용하도록 돕는, 패스워드 매니저(Password Manager, 이하 PM)라 부르는 부류의 여러 방식들이 제안되었다[4-13]. PM 방식들은 마스터(master) 패스워드 하나만을 사용자가 기억하면서, 사이트용 실제 패스워드는 마스터 패스워드(또는 마스터 패스워드와 그 밖의 비밀 값)를 활용해 저장되어 있던 내용을 복원하여 얻거나, 또는 로그온 시마다 계산해낸다는 공통점을 지닌다. PM은 주로 브라우저 탑재 프로그램의 형태로 제공되는데, 별도 설치되는 프로그램의 형태로 개발되기도 한다. 브라우저 탑재 형태의 경우 XSS(cross site scripting)이나 CSRF(cross site request forgery) 같은 웹 관련 공격에 대비되어야 한다[13]. Horsch 등[7]은 사용자 비밀 값을 클라이언트, 사이트별 고유 값을 서버가 각각 나누어 관리하고, 실제 사이트별 패스워드는 둘을 모두 이용하여 생성한다. Maqbali 등[8]은 마스터 패스워드에 사이트 주소를 포함시킴으로써 사이트별 다른 패스워드를 사용하도록 하는데, 사이트별로 패스워드 규정(허용하는 특수문자 등)을 준수하도록 조정하는 과정이 포함된다. Shirvanian 등[12]은 마스터 패스워드와 사이트 주소에 추가로 스마트폰의 저장 값을 포함시켜 사이트별 주소를 계산한다. 이들 방식은 마스터 패스워드나 비밀 값의 노출을 막기 위한 방어책을 제시하지 않으며, 사용자 장치 소유권을 확인하는 이중 인증 요소를 가지지 않는다는 점에서 제안 기법과 차이가 있다. 한편 여러 장치가 협조하여 패스워드를 생성하되 어떤 장치도 단독으로는 패스워드를 생성하지 못하게 하는 기법들[9-10]이 제시되었는데, Liu 등[9]은 클라이언트와 서버가 각각 패스워드의 절반씩을 만들고 인증 서버가 이 둘을 서로 연결한 후 패스워드로 취급하며, Lee[10]는 패스워드 데이터베이스를 분할 보관함으로써 임계값 이상 개수의 분할이 서로 합쳐져야만 패스워드를 구할 수 있도록 한다. 따라서 다수의 장치나 에이전트를 공격해서 데이터를 탈취하지 않는 한, 공격자는 패스워드를 알아낼 수 없다. 다만 이들은 일부 장치나 에이전트의 고장 시 아예 패스워드 추출이 불가능해진다는 한계를 가지며, 패스워드 추출을 위해 사용자 장치의 소유권 확인과 같은 추가 인증을 거치지 않는다는 점에서 제안 기법과 다르다.

그밖에 어떤 사이트의 사용자의 패스워드가 다른 사이트의 패스워드와 같거나 유사한지를 사이트 상호간 패스워드 기밀성을 유지하면서 판별하여, 필요시 사용자에게 수정을 권고하는 기법[11]이 발표되었고, 사용하는 패스워드의 안전성 강도를 측정하고 그 결과 및 수정 방향을 사용자에게 알림으로써 취약한 패스워드의 개선을 독려하려는 연구[14-15]도 진행되었다. 이들은 강력한 패스워드를 사용하도록 한다는 점에서 제안 기법과 목적이 같지만, 패스워드를 사용자에게 직접 제공하지는 않는다는 점에서 제안 기법과는 해결 대상인 문제 영역이 다르다.

2. 2FA 적용 방식 및 해킹 사례로 본 보안성

2FA[16-21] 중 가장 널리 쓰이는 것은 사용자가 ID와 패스워드와 함께 휴대폰에 도착한 SMS 일회용 인증코드 또한 사이트에 입력하도록 하는 것이다. 그럼 인증 서버는 ID와 패스워드의 일치와 함께 입력된 인증코드가 전송했던 SMS의 내용과 일치하는지도 확인한다. 따라서 2FA를 채택하면 그렇지 않은 경우에 비해 패스워드 재사용으로 가능해진 공격을 더 견고하게 방어한다. 하지만 사이트가 2FA를 아예 제공하지 않거나, 제공하더라도 사용자에게 2FA를 쓸지 선택하게 한다면, 패스워드 재사용으로 인한 공격의 피해를 입을 수 있다. 제안 기법은 패스워드 생성 시 스마트폰 소유권 확인을 거치기에, 어떤 사이트에서든지 패스워드 인증만으로 2FA의 보안성을 누리도록 한다.

SMS 인증코드는 크게 세 가지의 경로로 공격자에게 유출될 수 있다. 첫째, 스마트폰에 악성코드나 관리목적의 스파이웨어가

설치된 경우 수신한 SMS 문자 내용이 공격자의 손에 들어갈 수 있다[17-19]. 둘째, 사회공학(social engineering)이라 불리는 방법으로, 스팸문자[22], 이메일, 위조된 사이트 등으로 속여서, 피해자가 수신한 SMS 문자 내용을 공격자에게 보내는 식(예로, 관리자로 위장해서 보내는 문자의 답장으로)으로 진행된다[20]. 셋째, SIM Swapping이라는 기법[23,25]인데, 피해자의 폰 번호가 공격자가 소유한 USIM에 등록되게 하여 피해자 번호로 보낸 SMS가 공격자의 폰으로 전달되게 하는 것이다. 이는 피해자가 전혀 인지하지 못하는 상황에서도 발생할 수 있다. 예로, 통신사 고객센터에 공격자가 피해자인척 위장하여 전화하고, USIM 카드를 분실했으니 새로운 USIM 카드로 정보를 이전해 달라고 요구하는 식이다. 물론 통신사를 속이려면 공격자가 피해자의 개인 정보를 아는 것이 필요하다. 실제로 트위터의 CEO가 이 공격의 피해를 당했으며[23], 가상화폐가 이 방식 공격에 의해 도난당한 사례도 등장했다[25]. 제안 기법은 폰 번호로 목적지를 결정하는 SMS 인증코드 대신에, USIM의 비밀 값 및 사용자 입력 값을 활용한 인증을 진행하기에 SIM Swapping 공격에 영향을 받지 않는다. SMS 인증코드로 분실 패스워드 재설정이 가능한 사이트들에서는 아예 제안 기법과 무관하게 피해자의 패스워드를 공격자가 재설정할 수도 있지만, 이는 해당 사이트들의 내재된 취약점이지 사이트에 어떤 제한도 가할 필요가 없는 제안 기법 자체의 문제가 아니다. 감지가 어려운 악성코드[24]로 USIM 비밀 값과 사용자 입력 값, 추가로 보조서버의 비밀 값까지 탈취된 극단 상황이면 제안 기법에서도 패스워드 외부 유출이 가능하나 그 가능성은 매우 낮다.

Ⅲ. 랜덤 패스워드 생성 및 관리

1. 스마트폰 앱 설치 및 APGS에의 등록

먼저 스마트폰에 앱 appPG를 설치하고 패스워드 생성을 보조하는 서버 Assisting Password Generation Server(이하 APGS)에 그림 1과 같이 스마트폰 등록을 한다. 등록 과정은 패스워드 유출이나 스마트폰 분실이 발생하지 않는 한 반복되지 않는다. 이하 다음의 표기법을 쓴다. SHA-256 해시 함수를 M에 적용한 결과를 $h(M)$으로, 256 비트의 키 K로 M을 AES-256 대칭키 암호화한 결과를 $E(M,K)$로, RSA-2048 알고리즘으로 M을 공개키 PUB_i로 암호화한 결과를 $\{M\}_{\mathrm{PUBi}}$로, 연접(concatenation) 연산자를 ‖로 각각 표기한다.

APGS는 폰 번호로 SMS 인증문자를 보내고 사용자에 의해 회신된 문자의 일치를 확인한 후, 이후의 등록 과정을 이어간다. 그 목적은 스마트폰 사용자를 위한 고유의 비밀 값 S_i 의 USIM 저장 및 이 사용자의 로그온 시 필요한 $h(PP_i\|S_i)$과 $h(S_i\|PP_i)$를 서버 APGS에 등록하는 것이다. 여기서 PP_i는 패스프레이즈(passphrase)로서, 사용자는 여러 사이트들의 패스워드 전부를 기억하는 대신, 이 패스프레이즈 하나만을 기억하고 입력한다. APGS는 $h(PP_i\|S_i)$를 N_i 필드에 저장하여 사용자 식별자로 쓰며, $h(S_i\|PP_i)$를 V_i 필드에 저장하여 상대방을 인증하려 할 때 이 값을 알고 있는지 확인하는데 쓴다.

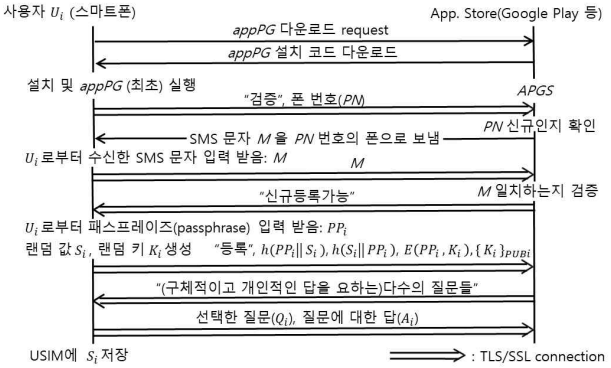

그림 1. 스마트폰 appPG 설치 및 서버 APGS에 등록

사용자(스마트폰)별 가입된 다수 사이트 각각의 비밀 값이 패스워드 계산에 필요한데, 이 값들은 그림 2와 같이 APGS의 Credentials for Password Generation(이하 CPG) 테이블에 저장된다. 인증 받은 사용자가 요청 시, APGS는 사용자가 지정한 사이트에 제시할 패스워드를 추출하는데 이 비밀 값을 활용하도록 제공한다. 추가로, 패스프레이즈의 분실이나 스마트폰 도난/분실 등의 긴급 상황에 대비한 정보 또한 CPG 테이블에 저장한다. 긴급 상황에서 사용자를 인증하기 위한 수단으로, 사용자 구분을 위한 폰 번호, 인증을 위해 사용자가 선택한 질문 및 그 질문에 대한 구체적인 답이 포함된다. 특히 패스프레이즈의 복원 요청에 대비, 랜덤한 키 K_i로 패스프레이즈 PP_i를 대칭키 암호화한 $E(PP_i,K_i)$와 키 K_i를 사용자의 공개키 PUB_i로 암호화한 $\{K_i\}_{\mathrm{PUB_i}}$도 역시 저장한다.

N_i (h(PPi ‖Si))	V_i (h(Si ‖PPi))	사이트 Xj	사이트 Yj	. . . .	사이트 Zj	폰 번호	선택된 질문	입력된 답변	E(PPi,Ki)	{Ki}PUB_i
:	:	:	:	:	:	:	:	:	:	:
ab...cd	*ef...gh*	*Rx_ij*	*Ry_ij*	. . . .	*Rz_ij*	*010... ...*	*xy는 ... ?*	*uvw원*	*mn...op*	*qr...st*
:	:	:	:	:	:	:	:	:	:	:

사용자 식별 기본 키 — 긴급 상황 식별 기본 키

그림 2. 서버 APGS에서 관리하는 CPG 테이블

2. 랜덤 패스워드 생성 및 사용자에 제시

어떤 사이트용 패스워드를 생성할 때, 사용자는 서버의 매번 바뀌는 챌린지에 대해 등록된 스마트폰의 소유권을 확인시키는 응답을 검증받게 함으로써 패스워드 인증이 실질적으로 이중 인증이 되게 하는 것이 제안 기법의 기본 아이디어이다.

가. 스마트폰 소유권 확인을 통한 인증

웹사이트 URI X_j에 사용자 U_i가 패스워드 설정 또는 설정된 패스워드를 추출하려면 우선 그림 3의 인증 과정을 거친다. 인증은 사용자의 인증 요청에 대해 APGS가 챌린지(challenge)로 C_K를 전송함으로 시작되는데, C_K는 $D_k \| T_k$로 구성되어 현재의 타임스탬프인 T_k로 인해 서버 시각에 따른 매번 새로운 값이 사용된다. 응답(response)으로 사용자가 입력한 패스프레이즈 PP_i와 스마트폰의 USIM으로부터 읽은 비밀 값 S_i로 계산한 $h(PP_i \| S_i)$는 APGS가 사용자(스마트폰)를 식별하는데 사용되며, $h(h(S_i \| PP_i) \| C_k)$는 S_i와 PP_i 외에 추가로 챌린지를 해시 계산에 포함시켜 매 인증 시마다 달라지게 함으로써, 이전의 정보가 노출된 경우라도 재사용의 위험이 없도록 한다. APGS는 수신한 $h(PP_i \| S_i)$ 값으로 CPG 테이블의 N_i 필드를 검색해 해당 사용자 튜플을 추출하고, 전송한 C_k 및 튜플의 V_i 필드의 값 $\hat{V}_i$을 이용, $h(\hat{V}_i \| C_k)$ ==수신한 $h(h(S_i \| PP_i) \| C_k)$인지 확인함으로써 인증을 완료한다.

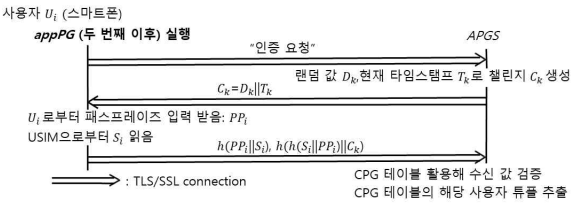

그림 3. 챌린지-응답 형식의 인증 과정

나. 패스워드 생성 및 사용자에게 제시

인증 완료 이후, 어떤 사이트의 패스워드 추출을 원하면 사용자는 그림 4와 같이 "질의/등록"을 선택하고 APGS에게 원하는 사이트 주소를 보낸다. APGS는 이 사이트를 위한 값이 이미 저장된 상태라면 그 값을 추출하고, 아니면 새로운 값을 만들어 그 값을 해당 사이트를 위한 값으로 저장한다. 저장 값이 R_{ij}(사용자 U_i의 사이트 X_j를 위한 값)이라면, APGS는 이를 챌린지의 T_k와 연접 후 V_i 필드의 값, 즉 $h(S_i \| PP_i)$,를 키로 대칭키 암호화한 결과를 사용자 스마트폰으로 회신한다. 암호화 결과는 매번 변하는 T_k를 포함하므로, R_{ij}와 키 $h(S_i \| PP_i)$ 값이 고정임에도 매번 바뀌어 재사용의 위험이 없다. 스마트폰은 인증 요청 시 사용한 $h(S_i \| PP_i)$ 값을 키로 수신한 $E(T_k \| R_{ij}, h(S_i \| PP_i))$를 복호화하여, R_{ij}를 추출한 후, 실제 패스워드 생성에 사용될 $W_{ij} = h(S_i \| X_j \| PP_i \| R_{ij})$를 계산한다.

이제 W_{ij}로부터 패스워드를 만드는데, W_{ij}는 SHA-256 결과로 32 바이트의 크기이며 $W_{ij} : B_0 \| B_1 \| \cdots \| B_{31}$ (각 B_k는 하나의 바이트)와 같이 표기한다. 이로부터 영대문자(uppercase letters, 이하 UL), 영소문자(lowercase letters, 이하 LL), 숫

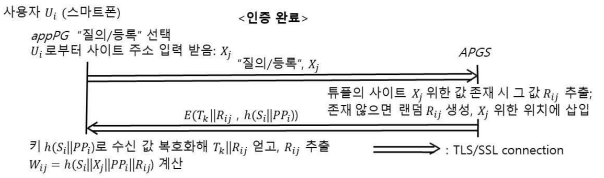

그림 4. 패스워드의 질의/등록 및 응답 처리

자(digits, 이하 DG) 및 특수문자(special characters, 이하 SC)의 조합으로 구성된, 길이 24의 문자열 패스워드를 만들 것이다. 특수문자는 !, $, #, %, &, *, +, -, ,(콤마), /, :, ;, <, =, >, ?, @, [,], ", ', ^, _, (,), \ 의 26자 중에서 선택된다. 일부 사이트들이 패스워드에 사용 금지하는 특수문자가 있다면 그런 몇몇은 제외할 수 있다. 이하 각 바이트 B_k의 l번째 비트에서 m번째 비트까지를 $B_k^{(l:m)}$으로 쓰며(단, $0 \le l \le m \le 7$), 이를 0부터 $2^{m-l+1}-1$까지 범위내의 한 정수로 취급한다.

Algorithm: 인덱스 0,1,...,23의 24개의 문자 생성

```
for i=0 to 5 do: /* 각 i: 4개의 문자 생성 */
    for j=0 to 3 do: /* 각 j: 2비트씩 추출 */
      if B_{24+i}^{(2j:2j+1)} mod 4 == 0:     /* UL */
        C_{4i+j} ← chr(ord('A')+ B_{4i+j} mod 26);
      elif B_{24+i}^{(2j:2j+1)} mod 4 == 1:   /* LL */
        C_{4i+j} ← chr(ord('a')+ B_{4i+j} mod 26);
      elif B_{24+i}^{(2j:2j+1)} mod 4 == 2:  /* DG */
        C_{4i+j} ← chr(ord('0')+ B_{4i+j} mod 10);
      elif B_{24+i}^{(2j:2j+1)} mod 4 == 3:  /* SC */
        C_{4i+j} ← '!' if B_{4i+j} mod 26 == 0;
        C_{4i+j} ← '$' elif B_{4i+j} mod 26 == 1;
          . . .
        C_{4i+j} ← '\' elif B_{4i+j} mod 26 == 25;
```

결과인 $C_x (0 \le x \le 23)$ 문자는 각각 B_x로부터 구하는데, UL, LL, DG, 또는 SC 중 무엇에 해당되는지 각각 B_{24}의 첫 두 비트 $B_{24}^{(0:1)}$부터 B_{29}의 마지막 2비트 $B_{29}^{(6:7)}$까지 두 비트 값을 차례로 검사해 결정한다. 다음으로 패스워드에 숫자와 특수문자를 각 1개 이상씩 포함시키고자 아래 과정을 진행한다.

Algorithm: 숫자(1개 이상)와 특수문자(1개 이상) 조정

```
n_d ← max(1, B_{30}^{(0:3)} mod 5); /* 1,2,3,4 중 하나 */
n_s ← max(1, B_{30}^{(4:7)} mod 5); /* 1,2,3,4 중 하나 */
/* 실제 DG 및 SC 개수를 각각 n_d 및 n_s 이상이 되게 함*/
r_d ← C_0, ... ,C_23 중 숫자(DG)의 개수;
r_s ← C_0, ... ,C_23 중 특수문자(SC)의 개수;
k_d ← n_d − r_d;    k_s ← n_s − r_s; /* 추가될 개수들 */
p ← B_31 mod 24; /* 강제 변경 작업 시작 위치 */
while (k_d > 0) do: /* 강제 변경 개수만큼 */
```

```
    if C_p is in (UL or LL): /* 변경 가능 */
        C_p ← chr(ord('0')+ B_p mod 10);
        k_d ← k_d − 1;    p ← (p+1) mod 24;
while (k_s > 0) do: /* 강제 변경 개수만큼 */
    if C_p is in (UL or LL): /* 변경 가능 */
        C_p ← '!' if B_p mod 26 == 0;
        C_p ← '$' elif B_p mod 26 == 1;
            . . .
        C_p ← '\' elif B_p mod 26 == 25;
        k_s ← k_s − 1;    p ← (p+1) mod 24;
```

마지막으로, 사용자가 패스워드의 길이를 최소 9문자에서 최대 24문자 사이에서 선택할 수 있도록, 숫자 및 특수문자가 앞부분의 9문자 이내에 적어도 한 개 이상씩 포함되도록 조정한 후 사용자에게 제시한다. 그 과정은 다음과 같다.

Algorithm: 처음 9문자(패스워드 최소 길이)에 숫자 1개와 특수문자 1개 반드시 포함되도록 순서 조정

```
/* 최종: D_0,D_1, ... ,D_23와 같이 이 순서대로 제시됨 */
mlen ← 9;  /* 지정된 패스워드 최소 길이: 9 */
/* C_0,...,C_23에서 DG인 문자와 SC인 문자 사이의
   간격이 최소인 쌍의 인덱스를 i, j로 지정 */
i ← C_i가 DG에 속하는 인덱스로 |i−j|가 최소;
j ← C_j가 SC에 속하는 인덱스로 |i−j|가 최소;
/* i와 j사이(i,j 포함)에 mlen 문자 초과면 순서 바꿈 */
if (max(i,j)-min(i,j)+1) > mlen:
    saveC ← C_max(i,j);
    for k = min(i,j)+mlen to max(i,j) do:
        C_k ← C_k−1;  /* 한 칸씩 뒤로 이동 */
    /* min(i,j)+mlen−1 위치에 DG/SC 오도록 */
    C_min(i,j)+mlen−1 ← saveC
for k=0 to 23 do: D_k ← C_(min(i,j)+k) mod 24;
```

다. 이후의 로그온 과정

사용자는 표시된 $D_0,D_1, \ldots ,D_{23}$를 보고, 자신만의 방식으로 패스워드를 정한다. 예로, 단순히 처음 9문자($D_0,\ldots,D_8$에는 숫자와 특수문자가 한 개 이상씩 반드시 존재함)를 패스워드로 정하거나, 임의의 문자열(예: prFx)에 이어서 D_0부터 D_8까지 연접한 것을 패스워드로 정할 수 있다. 결정된 패스워드를 목적지 사이트에 ID와 함께 입력하여 로그온을 진행한다.

3. 패스워드 변경

패스워드 변경 시, 먼저 그림 3의 인증 과정을 완료한 후 사용자는 그림 5의 과정을 스마트폰을 통해 APGS 서버와 진행한다. 사용자 U_i는 패스워드 변경하려는 사이트 X_j를 APGS에 알리고, APGS는 U_i의 사이트 X_j 필드 저장 값 R_{ij}와 새로 만든 값 R_{ij}^{new}를 가지고, 패스워드 질의 과정의 응답과 같은 형태의 응답 메시지를 만들어 회신한다. 이때 기존의 R_{ij} 값을 포함한 정보를 보내는 이유는 패스워드 변경 시 기존의 패스워드를 알고 있는지 확인하는 사이트도 있기 때문이다. 스마트폰은 수신 값으로부터 R_{ij} 및 R_{ij}^{new}를 추출하고, 이로부터 각각 W_{ij}와 W_{ij}^{new}를 만들어, 2장 2절에 서술한 방법대로 패스워드를 생성한다. 표시된 패스워드를 이용해 사용자는 사이트 X_j의 패스워드를 W_{ij}^{new}에서 계산한 것으로 변경한다. 변경 완료 수신 후, APGS는 사이트 X_j 필드의 기존 R_{ij} 값을 R_{ij}^{new} 값으로 바꿔, 이후 변경된 패스워드가 추출되도록 한다.

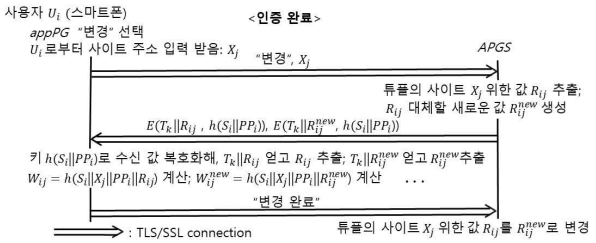

그림 5. 사이트에 등록된 패스워드 새로운 것으로 변경

4. 긴급 상황의 대처

패스프레이즈는 APGS에 인증을 받거나 인증 후 패스워드를 추출해 낼 때 필요하므로, 사용자가 패스프레이즈를 잊은 경우 이를 복원할 방법이 필요하다. 복원은 PC 등의 웹브라우저로 사용자가 APGS 사이트에 접속하여 '패스프레이즈 복원' 메뉴를 선택함으로써 시작된다(그림 6 참조). 우선 사용자 인증을 받는데, 인증은 등록 시 선택했던 질문과 그 질문의 답은 무엇인지를 사용자가 APGS에 정확히 제시함으로써 완료된다. 이 때 사용자를 구분하는 식별자는 폰 번호이며, APGS는 폰 번호로 CPG 테이블에서 해당 사용자의 튜플을 추출한다. 튜플의 저장된 질문을 포함한 다수의 무작위 추출된 질문들이 사용자에게 제시되는데, 사용자는 자신이 선택했던 질문을 골라

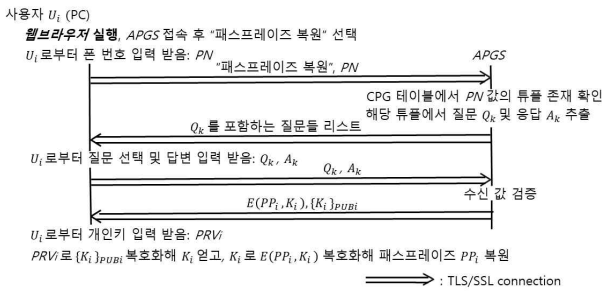

그림 6. 잊어버린 패스프레이즈의 복원

내고 질문에 대해 자신이 등록했던 정확한 답을 APGS에 제시해야 한다. 검증에 통과되면, APGS는 사용자에게 튜플의 $E(PP_i,K_i)$ 및 $\{K_i\}_{\mathrm{PUBi}}$를 회신하며, 사용자는 개인키(private key)로 $\{K_i\}_{\mathrm{PUBi}}$를 복호화하여 K_i를 얻고, 이어서 $E(PP_i,K_i)$를 K_i로 복호화하여 패스프레이즈 PP_i를 복원할 수 있다. 이를 위해 사용자는 자신의 개인키를 알아야 한다.

관련하여 자신의 패스프레이즈나 스마트폰 저장 비밀 값의 유출이 의심되는 긴급 상황에서, 대상자는 APGS의 CPG 테이블의 자신 관련 내용 전체의 폐기를 요청하여, 공격자가 APGS를 통해 자신의 패스워드를 탈취하려는 시도를 무산시킬 수 있다. 이를 위해, 사용자가 패스프레이즈 복원 시와 동일한 인증 과정을 통과하면 APGS는 그림 7과 같이 해당자의 CPG 테이블 내용을 모두 삭제하고 폐기완료를 통보한다.

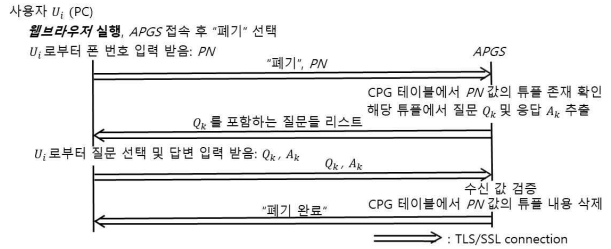

그림 7. APGS의 특정 사용자 관련 내용 폐기

IV. 보안성 분석

제안 기법은 수동 및 능동의 여러 가능한 공격 유형에 대하여 아래 서술하는 것과 같이 강력한 보안성을 제공한다.

1. 사전 공격 대비 패스워드 저항성

패스워드는 해시 함수 결과인 W_{ij}로부터 만들어지므로, 패스워드의 강력함은 SHA-256를 포함한 해시 함수 SHA 패밀리의 랜덤성에 의존한다. SHA 패밀리는 랜덤과 구분하기 어려운 비트열을 생성하는데, 실제 ANSI X9.31의 RNG(random number generator)나 NIST SP 800-90 Hash DRBG (deterministic random bit generator) 같은 랜덤 숫자 생성기는 SHA 패밀리의 해시 함수를 사용하고 있다. 따라서 랜덤 값들이 포함된 입력에 대한 SHA-256의 출력 비트열로부터 생성되는 제안 기법의 패스워드는 해커들이 사용하는 사전에 포함될 가능성이 없다. 그러므로 제안 기법에 따라 생성된 패스워드는 공격자의 사전 공격에 강한 저항성을 가진다.

2. 도청을 통한 사용자 위장 및 정보 유출 대비

제안 기법에서 사용자(스마트폰)와 APGS 사이 모든 통신 내용은 TLS/SSL으로 보호되어 통신 중의 정보 유출에 대비한다. 다만, APGS의 서버 인증서의 유효 기간 만료나 사용자 스마트폰 브라우저의 HTTPS 미지원 등 매우 예외적인 경우라면 통신 내용이 도청당할 수 있다. 그러나 이런 경우에도, 사용자의 비밀 값인 PP_i 및 S_i는 SHA-256에 입력으로만 적용되기에, SHA-256의 단방향성에 의해 공격자가 알아낼 수 없다. APGS가 관리하는 사용자의 R_{ij} 값은 AES 대칭키 암호화의 메시지 부분에 포함되어 그 결과가 전송되므로, 암호화키 값을 알지 못하는 공격자가 구할 수 없다. 한편 PP_i는 등록 및 저장 과정에서 $E(PP_i,K_i)$의 형태로 전송되는데, 이 역시 암호화키 K_i를 모르므로 공격자가 구할 수 없고, K_i는 충분한 키 길이의 RSA-2048 알고리즘을 사용해 $\{K_i\}_{\mathrm{PUBi}}$와 같이 공개키 암호화된 결과로 전송되므로 개인키가 없는 한 구할 수 없다.

그밖에 어떤 사용자의 폰 번호, 선택한 질문 및 답이 TLS/SSL에 의해 보호받지 못한 채 공격자에게 유출된 예외적 상황이 발생했다면, 공격자는 이 사용자로 위장해 그림 6의 절차를 진행할 수 있다. 하지만 공격자는 이 사용자의 개인키를 모르기 때문에 제공된 정보만으로는 원하는 패스프레이즈를 알아내지 못한다. 따라서 공격자가 이 사용자에게 피해를 주는 유일한 방법은 그림 7의 절차를 따라 APGS의 이 사용자 관련 내용의 폐기를 진행하는 것이다. 이 경우에 물론 해당 사용자는 패스워드를 추출하지 못해 사이트 접속이 불가능하지만, 공격자가 이 사용자로 위장하여 사이트에 접속한 후에 실행하는 악의적 행동에 의한 직접적인 피해를 당하지는 않는다.

3. APGS 또는 스마트폰의 저장 정보 유출 대비

서버의 보안 수준은 대체로 클라이언트 기기에 대한 보안 수준보다 월등하게 높지만, 그럼에도 서버의 저장 내용이 공격자의 손에 들어갈 가능성이 존재한다. 이에 APGS의 CPG 테이블 내용이 공격자에게 유출된 경우에 대비하는 것이 필요하다. APGS의 CPG 테이블을 획득한 공격자는 사용자 U_i의 사이트 X_j에 대한 패스워드를 구하기 위해 저장된 R_{ij}를 이용하려 할 것이다. 하지만 패스워드는 $W_{ij}=h(S_i\|X_j\|PP_i\|R_{ij})$로부터 나오고, 이 계산을 위해서는 S_i, X_j, PP_i, R_{ij}가 전부 필요하지만 공격자는 R_{ij}와 사이트 주소 X_j만 알 뿐이다. CPG 테이블의 N_i 필드나 V_i 필드의 저장 값을 통해서 S_i나 PP_i를 구하는 것은 해시 함수의 단방향성에 의해 가능하지 않다. 공격자가 획득한 정보를 이용해 APGS에 자신이 사용자 U_i인 것처럼 인증 요청을 하고 질의를 보낼 수도 있는데, 이 때 인증 받은 후 최종적으로 공격자가 수신하는 정보를 통해서는 오직 R_{ij}(획득한 CPG

테이블을 통해 이미 알고 있는)만 얻을 수 있을 뿐이다. 결국 어떻게 하든지, 공격자는 공격대상자의 패스워드를 추출하기에 충분한 정보를 얻을 수 없다. 한편 공격자가 CPG 테이블로부터 얻은 폰 번호와 긴급 상황 시 인증을 위한 질문 및 답의 내용을 이용해, CPG 테이블의 사용자 U_i의 튜플 내용을 폐기시키려 한다면 (그림 7) 이 시도는 성공할 것이다. 이러한 일종의 서비스 거부 공격은 피해자에게 어떤 사이트에도 접속하지 못하게 하지만, 공격자가 피해자로 위장 로그온한 후 시행하는 행위에 따른 피해에 비교하면 그 영향이 간접적이다.

사용자 U_i의 스마트폰을 공격자가 획득한 상황에서, 공격자는 S_i에 접근할 수 있어도 패스프레이즈 PP_i는 알 수 없기에, $h(PP_i \| S_i)$ 및 $h(S_i \| PP_i)$를 계산할 수 없다. 만약 이전의 인증 시도 과정에서 메시지를 도청했거나 다른 경로로 $h(PP_i \| S_i)$와 $h(h(S_i \| PP_i) \| C_k^{old})$를 획득했다 하더라도, 새로운 챌린지 C_k에 대한 $h(h(S_i \| PP_i) \| C_k)$는 $h(S_i \| PP_i)$를 모르면 계산할 수 없다. 따라서 공격자는 인증 과정에서 APGS를 속여 인증을 받을 수 없고, 인증에 실패하면 피해자의 패스워드 추출을 위한 이후의 과정을 이어나갈 수 없다. 한편 피해자는 스마트폰 분실을 인지한 즉시 가입된 통신사를 통하여 USIM에 락(lock) 설정을 요청할 수 있다. 공격자가 S_i를 획득하기 전에 USIM에 락이 걸리면, 공격자가 설사 다른 경로로 피해자의 패스프레이즈 PP_i를 알았다 하더라도 패스워드 추출을 막을 수 있다.

4. 악성코드 통한 사용자 위장 및 정보 유출 대비

사용자 스마트폰이 키로거(keylogger) 악성코드에 감염되면 공격자는 원격지에서 사용자가 입력하는 내용을 획득할 수 있는데, 사용자가 입력하는 패스프레이즈가 이런 경로로 유출될 수 있다. 하지만 USIM 저장 비밀 값 S_i는 사용자가 입력하는 대상이 아니라서 키로거로는 빼낼 수 없다. 다른 유형의 악성코드가 사용자를 속여 폰에 설치되고, 또한 USIM 접근 권한을 부여받았다면, S_i가 유출될 가능성이 있다. 따라서 두 개의 악성코드나 두 가지 역할을 하는 악성코드에 스마트폰이 감염되고, 악성코드가 USIM 접근 가능하도록 권한 설정된다면, 피해자의 패스워드 추출 목적으로 APGS를 속이는데 필요한 모든 정보를 공격자가 가질 수 있다. 공격자는 획득한 정보를 이용하여 APGS에 인증 및 패스워드 생성을 위한 정보를 요청할 것이다. 공격자는 표시된 길이 24의 문자열에서 사용자의 실제 패스워드의 길이를 알아내되, 이를 반복적 로그인 실패로 사이트 서버가 추가 로그인 시도를 일정 시간동안 막기 전에 완료해야 한다. 만약 피해자가 어떤 문자열을 접두어로 패스워드에 포함시켰다면, 공격자의 시도는 설사 성공하더라도 성공하기까지 상당히 오랜 시간이 걸릴 것이다. 그러므로 사용자는 스마트폰의 악성코드 감염이 의심 또는 확인되었을 때(백신 프로그램의 감지 등을 통해) 즉각 그림 7의 사용자 정보 폐기 절차를 진행할 필요가 있다. 공격자가 패스워드 길이나 사용자가 설정한 문자열 접두어를 찾는 동안 폐기 절차가 완료된다면, 공격자의 로그온 시도는 폐기 완료 전에 R_{ij} 값이 공격자에게 전달된 사이트 X_j에서만 성공가능성이 있다. 사이트별로 비밀 값 R_{ij}는 각기 다르므로 공격자가 이 피해자로 위장해 다른 사이트에 로그온 하려는 시도는 피해자의 정보가 폐기된 후에는 성공할 수 없다.

V. 결론

패스워드 추출 시 사용자가 스마트폰을 통한 인증을 받도록 하려는 아이디어를 실현하는 기법을 제안하였다. 제안 기법에 따라 생성된 패스워드는 사용자가 기억할 필요가 없으며, 공격대상자의 개인정보의 조합을 시도하거나 해커들이 사용하는 사전에 나온 단어들을 대입하는 공격에 매우 강한 내성을 가진다. 제안 기법은 서버나 공격 대상자의 스마트폰 저장 정보의 유출 시에도 공격자가 사용자로 위장하지 못하게 하며, 스마트폰의 비밀 값 및 사용자가 입력하는 비밀 패스프레이즈도 악성코드 등에 의해 노출된 예외 상황에도 피해의 범위를 APGS 질의 후 응답이 완료된 사이트로만 제한하는 방법을 제공한다. SMS 인증코드 등을 통한 이중 인증을 제공하지 않거나 강제하지 않는 다수 사이트들에 대해, 사이트 인증 체계의 수정 없이도 패스워드 인증만으로 이중 인증 수준의 향상된 보안성을 달성할 수 있음을 보임으로써, 제안 기법은 새로운 접근 방식을 제시하였다. 다만 사이트 접속을 위해 기억한 패스워드를 사용자가 바로 입력하는 기존 방식 대비, 제안 기법은 스마트폰을 통한 패스워드 추출 과정이 추가적으로 필요하여 로그온을 위한 소요시간 및 입력 양이 증가한다. 또한 스마트폰을 휴대하지 않은 상태에서는 패스워드를 구할 수 없어 사이트 접속이 불가능하다는 한계를 가진다. 이에 제안 기법의 보안성을 유지하면서 사용 편의성을 최대한 높이기 위한 추가 연구가 필요하다고 판단된다.

REFERENCES

[1] B. Ives, K.R. Walsh and H. Schneider, “The Domino Effect of Password Reuse,” *Comm. of the ACM*, vol. 47, no. 4, pp. 75-78, April 2004.

[2] C. Wang, S.T.K. Jan, H. Hu, D. Bossart, and G. Wang, “The Next Domino to Fall: Empirical Analysis of User Passwords across Online Services,” *Proc. of ACM Conf. in Data and Application Security and Privacy*, pp. 196-203, Tempe, USA, Mar. 2018.

[3] S. Gunaratna, “Why reusing your passwords is riskier than ever,” https://www.cbsnews.

com/news/reusing-passwords-is-riskier-than-ever-botnet-cyber-attacks-security/, May 2016. (accessed Jan. 30, 2020).
[4] S. Oesch and S. Ruoti, "That Was Then, This Is Now: A Security Evaluation of Password Generation, Storage, and Autofill in Browser-Based Password Managers," *Proc. of USENIX Security Symp.*, 2020(to appear).
[5] D. Silver, S. Jana, D. Boneh, E. Chen, and C. Jackson, "Password managers: Attacks and defenses," *Proc. of USENIX Security Symp.*, pp. 449-464, San Diego, USA, Aug. 2014.
[6] S. G. Lyastani, M. Schilling, S. Fahl, M. Backes, and S. Bugiel, "Better managed than memorized? Studying the impact of managers on password strength and reuse," *Proc. of USENIX Security Symp.*, pp.203-220, Baltimore, USA, Aug. 2018.
[7] M. Horsch, A. Hulsing, and J. Buchmann, "PALPAS PAsswordLess PAssword Synchronization," *Proc. on Int'l Conf. on Availability, Reliability and Security*, France, August 2015.
[8] F.A. Maqbali and C.J. Mitchell, "AutoPass: An Automatic Password Generator," *Proc. of Int'l Carnahan Conf. on Security Technology*, Spain, Oct. 2017.
[9] Y.T. Liu, Y.B. Xia, H.B. Chen, B.Y. Zang, and Z. Liang, "SplitPass: A Mutually Distrusting Two-Party Password Manager," *Jr. of Computer Science and Technology*, vol. 30, no. 1, pp. 98-115, Jan. 2018.
[10] B. Lee, "Multi-factor secure password manager using a secret sharing scheme," *Australian National University Technical Report*, May 2019.
[11] K.C. Wang and M.K. Reiter, "How to End Password Reuse on the Web," *Proc. of Network and Distributed System Security Symp.*, San Diego, USA, Feb. 2019.
[12] M. Shirvanian, S. Jarecki, H. Krawczyk, and N. Saxena, "SPHINX: A Password Store that Perfectly Hides Passwords from Itself," *Proc. of IEEE Int'l Conf. on Distributed Computing Systems*, pp. 1094-1104, 2017.
[13] Z.W. Li, W. He, D. Akhawe, and D. Song, "The emperor's new password manager: Security analysis of web-based password managers," *Proc. of USENIX Security Symp.*, pp. 465-479, San Diego, USA, Aug. 2014.
[14] X. de C. de Carnavalet and M. Mannan, "From very weak to very strong: Analyzing password-strength meters," *Proc. of Network and Distributed System Security Symp.*, San Diego, USA, Feb. 2014.
[15] D.L. Wheeler, "zxcvbn: Low-budget password strength estimation," *Proc. USENIX Security Symp.*, pp. 157-174, Austin, USA, Aug. 2016.
[16] R. Grimes, "The many ways to hack 2FA," *Network Security*, vol. 2019, issue 9, pp. 8-13, Sept. 2019.
[17] Mazar Bot Android Malware Distributed via SMS Spoofing Campaign(2016), https://www.tripwire.com/state-of-security/featured/mazarbot-android-malware-distributed-via-sms-spoofing-campaign/ (accessed Jan. 31, 2020).
[18] BankBot - Mazain(2017), https://github.com/bemre/bankbot-mazain (accessed Jan. 31, 2020).
[19] How to Read Someone's Text Messages Without Having Their Phone?(2020), https://nexspy.com/read-someones-text-messages/ (accessed Jan. 31, 2020).
[20] H. Siadati, T. Nguyen, P. Gupta, M. Jakobsson, and N. Memon, "Mind your SMSes: Mitigating social engineering in second-factor authentication," *Computers & Security*, vol. 65, pp. 14-28, Mar. 2017.
[21] 차병래, 최명수, 박선, 김종원, "보안 강화를 위한 NFC 기반 전자결제 시스템의 2 팩터 인증 기술의 초안 설계," *스마트미디어저널*, 제5권, 제2호, 77-83쪽, 2016년 6월
[22] 이현영, 강승식, "워드 임베딩과 딥러닝 기법을 이용한 SMS 문자 메시지 필터링," *스마트미디어저널*, 제7권, 제4호, 24-29쪽, 2018년 12월
[23] Hackers Hit Twitter CEO Jack Dorsey in a 'SIM Swap.' You're at Risk, Too(2019), https://nytimes.com/2019/09/05/technology/sim-swap-jack-dorsey-hack.html (accessed Jan. 31, 2020).
[24] 조우진, 김형식, "행위 유사도 기반 변종 악성코드 탐지 방법," *스마트미디어저널*, 제8권 제4호, 25-32쪽, 2019년 12월
[25] Busting SIM Swappers and SIM Swap Myths(2018), https://krebsonsecurity.com/2018/11/busting-sim-swappers-and-sim-swap-myths/ (accessed Jan 30, 2020).

저 자 소 개

박준철(정회원)

1986년 서울대 계산통계학과 학사.
1988년 KAIST 전산학과 석사.
1998년 U. of Maryland, College Park 전산학 박사.
2001년 9월~현재 홍익대학교 컴퓨터공학과 교수.

<주 관심분야 : 네트워크 보안, 소프트웨어 보안>