情 報 保 護 學 會 誌
第20卷 第6號, 2010. 12

# TOTP와 패스워드를 이용한 Two-Factor 사용자 인증 방식 및 구현

# A Two-Factor User Authorization Method and Its Implementation using TOTP and Password

제 주 환* , 유 승 록**, 임 학 창**, 배 동 환**, 이 윤 호**, 양 형 규**

요 약

인터넷 및 통신 기술의 발전은 사용자로 하여금 금융, 방송, 게임 등의 온라인 서비스 제공에 대한 시간 및 공간적 제한을 없애 주었지만, 다른 한편으로는 해커 등의 악의적 사용자로 인한 피해 가능성도 높이고 있다. 이를 해결하기 위한 다양한 보안 기법 가운데 하나가 OTP를 이용한 사용자 인증 방법이다. OTP는 재사용하지 않는 패스워드로서 기존 패스워드 인증 방식이 갖는 취약점을 해결할 수 있는 방식이다. 하지만 OTP 생성 단말의 도난이나 서버 해킹으로 인한 패스워드 추측공격 또는 Stolen verifier 공격 등에 취약할 수 있다. 본 논문에서는 위와 같은 문제점을 해결하기 위해서 두 가지 인증 정보 즉, 시간 기반 OTP 생성방식인 TOTP 및 패스워드를 이용하는 새로운 Two-Factor 인증 프로토콜인 POTP(Password embedded OTP)를 제안한다. 제안한 방식은 재전송 공격에 안전하며, 공격자가 OTP 생성용 디바이스를 획득하더라도 패스워드를 유추할 수 없고 서버의 인증 정보 데이터베이스를 획득하더라도 정상적인 사용자로 위장할 수 없는 장점과 함께, 서버에서 인증 정보 보관시 연산 속도가 빠른 해쉬 함수를 이용할 수 있어 보다 효율적이다.

## 1. 서 론

최근 금융권과 온라인 게임 사이트를 중심으로 OTP(OneTime Password)를 이용한 사용자 인증 방식이 빠르게 확산되고 있다. OTP란 로그인할 때마다 그 세션에만 사용할 수 있는 일회성 패스워드를 이용하여 사용자를 인증하는 보안 시스템이다. 기존 패스워드 방식과는 달리 OTP는 재사용이 불가능하기 때문에 이미 사용된 패스워드를 이용한 재전송 공격(Replay attack)에 안전한 방식이다. 현재 OTP 제품들은 RSA Security사의 SecureID, Vasco사의 Digpass, CryptoCard사의 CRYPTO 등 다양한 제품들이 상용화되어 있는데, 토큰 형태나 카드 형태, 그리고 휴대전화를 이용하는 모바일 방식 등 다양한 제품이 출시되어 있다 [1, 6, 7].

하지만 많은 OTP 제품들은 각각 독자적인 OTP 생성 알고리즘을 사용하고 있고, 서로 호환되지 않기 때문에 여러 OTP 서비스를 이용할 경우 회사에 따라 다른 토큰을 소지해야 하는 불편함이 있다. 이런 이유로 개방형 인증시스템을 위한 비영리 산업체 연합 모임인 OATH(Open AuTHentication)에서는 누구나 이용할 수 있도록 OTP 알고리즘을 공개함과 동시에 표준화를 추진하여 HMAC 기반 OTP 알고리즘인 HOTP를 인터넷 표준인 RFC4226으로 등록하였고, HOTP를 기반으로 하는 TOTP도 표준화를 추진하고 있다 [3, 4, 5].

일반적으로 OTP는 다른 인증 정보와 함께 Two-factor 사용자 인증 방식에 사용되는데, OTP 생성용 디바이스의 소유 여부와 패스워드를 동시에 확인하는 것이 대표적인 예이다. OTP와 기존 패스워드를 동시에

* 강남대학교 컴퓨터미디어정보공학부(chejoohwan@lycos.co.kr)
** 강남대학교컴퓨터미디어정보공학부(yabi01@naver.com hakchangs@naver.com shadowbug@nate.com sleeyh@security.re.kr, hkyang@kangnam.ac.kr)

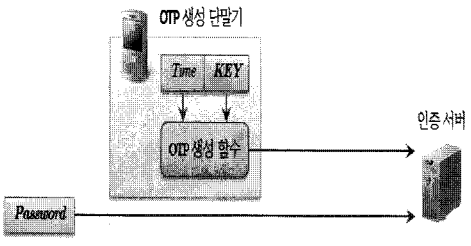

〔그림 1〕 미래테크놀로지 의 다양한 형태의 OTP 생성기

사용하는 경우, 공격자가 디바이스를 획득했더라도 패스워드를 알아내기 위한 공격, 즉 오프라인 패스워드 추측 공격이 불가능해야 한다. 또한, 서버의 경우 공격자가 사용자의 인증 정보를 보관하고 있는 데이터베이스를 획득한 후 이를 이용하여 정상적인사용자로 위장하는 공격, 즉 Stolen-verifier 공격에 안전해야 한다 [2].

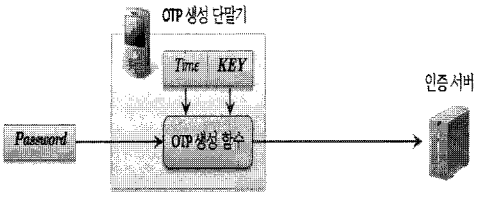
본 논문에서는 TOTP와 패스워드를 이용한 새로운 Two-Factor 인증 방식인 POTP를 제안하고자 한다. 제안한 방식은 재전송 공격에 안전하며, 공격자가 OTP 생성용 디바이스를 획득하더라도 패스워드를 유추할 수 없으며, 서버의 인증 정보 데이터베이스를 획득하더라도 정상적인 사용자로 위장할 수 없다.

TOTP와 패스워드를 이용한 인증방법은 크게 다음 세 가지로 구분할 수 있다.

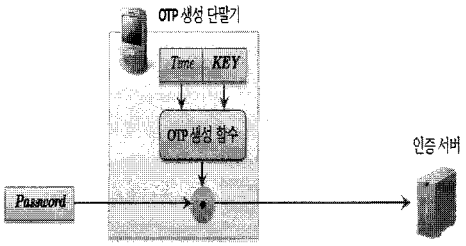
첫 번째로 OTP와 패스워드를 개별 검증하는 방법이다 ([그림 2] 참조). 인증 서버는 인증요청을 받았을 때 사용자의 아이디와 패스워드, 그리고 OTP를 입력받게 되는데, 이때 패스워드와 OTP가 모두 일치할 경우에만 사용자를 인증하게 된다. 하지만, 이 방법은 기존 패스워드가 갖는 취약점을 그대로 가지며, 전송량도 많다는




〔그림 2〕 첫 번째 Two-Factor 인증방법




〔그림 3〕 두 번째 Two-Factor 인증방법




〔그림 4〕 세 번째 Two-Factor 인증방법

단점이 있다.

두 번째 방법은 패스워드를 이용하여 OTP를 생성하는 방법이다 ([그림 3] 참조). OTP 생성함수에 패스워드를 추가 입력하여 OTP를 생성하게 되는데, 인증서버에서 OTP를 확인하기 위해서는 같은 패스워드를 입력해야 하기 때문에 패스워드를 안전하게 저장해야 하는 문제가 있다.

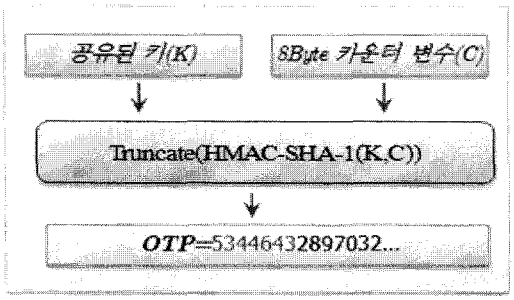
세 번째는 제안 프로토콜이 이용하는 방식으로서 OTP와 패스워드를 결합하여 전송하는 방식이다 ([그림 4] 참조). 이 방식은 OTP를 생성한 후 패스워드를 결합하여 인증서버에 전송하게 되고, 인증서버에서는 결합된 OTP를 제거한 후 전송된 패스워드를 확인하는 방법이다. 이 방식의 장점은 서버에서 패스워드를 OTP 연산의 입력 값으로 사용하지 않기 때문에 패스워드를 저장할 때 연산 속도가 빠른 해쉬함수를 이용할 수 있다는 점이다 [8].

## Ⅱ. 기존 방식

### 2.1. HOTP와 TOTP

#### 2.1.1. HOTP

HOTP는 인증 값을 생성할 때 카운터 C와 비밀키 K

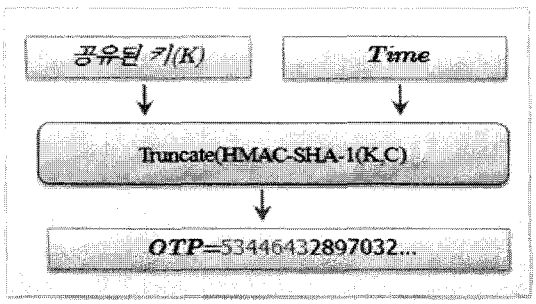



〔그림 5〕 HTOP 생성방법

를 이용한다. C는 카운터로서 8바이트이며, 매 인증시마다 변하게 되지만 검증자와 증명자는 같은 값을 유지해야 한다. K는 검증자와 증명자가 비밀리에 공유하고 있는 비밀키로서, 각 증명자의 K는 서로 달라야 한다. HOTP는 C와 K를 입력받아 HMAC 연산을 통해 32비트 값을 얻게 되는데, 부호비트 처리로 인한 잘못된 결과 출력을 방지하기 위해 31비트 값만 사용하게 된다. 31비트 OTP를 숫자로 변환하게 되면 $0 \leq OTP \leq 2^{31} = 2,147,483,647$ 의 범위가 나오게 되므로, 최대 9자리의 OTP를 생성할 수 있다. 일반적으로 6~8자리를 이용하며, 7자리 이상을 권고하고 있다 [3].

### 2.1.2. TOTP

카운터 방식은 인증이 성공할 때 마다 카운터 값이 증가하기 때문에, OTP를 생성한 시점과 검증시점의 차가 크게 날 수도 있다. 이에 비해 TOTP는 카운터 대신 시간 정보를 동기유지에 사용하는 방식으로서, OTP 생성 시간과 검증시간의 차이가 정해진 범위를 초과할 경우 해당 OTP는 사용할 수 없게 된다 ([그림 6] 참조). 따라서 증명자와 검증자의 시간 값이 일정 오차 내에서 동기화되어 있어야 한다 [4].




〔그림 6〕 TOTP 생성 방법

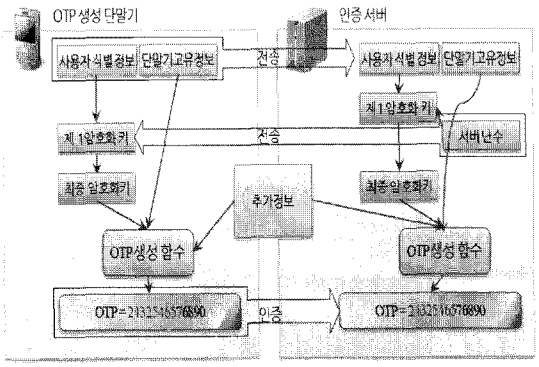
## 2.2. 국내 모바일 OTP 제품방식

OTP를 위한 전용 H/W 토큰을 이용할 경우 다수의 토큰을 소지해야 하는 불편함으로 인해 최근에는 모바일 단말에 OTP 생성 S/W를 이용하는 경우가 많다. 본 절에서는 국내 모바일 기반 OTP 가운데 에이티솔루션의 U-OTP와 이니텍의 MOTP를 설명하고, 취약점을 살펴보도록 한다.

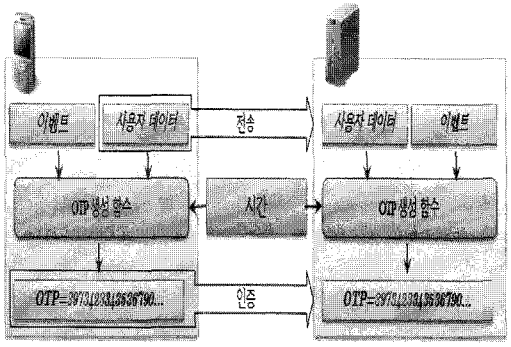
U-OTP의 경우 한게임, 엠게임, 넥슨, 싸이월드, T월드, 11번가 등에서 서비스되고 있으며, MOTP의 경우 십이지천, 던전앤파이터, 파천일검, NC소프트 등에서 서비스 되고 있다 [9].

U-OTP는 특허 제1020070001141호로 출원되어 있으며, 청구항을 그림으로 표현하면 <그림 7>과 같다. 사용자 식별정보는 ID 또는 주민등록번호가 쓰이고, 단말기 고유정보에는 전화번호, 단말기 일련번호, 플랫폼 버전정보, 단말기 모델명 등이 사용된다. 추가정보는 현재시간, 단말기 위치정보, 카운터 등이 될 수 있다. 암호화키 생성에는 사용자 식별정보와 서버에서 생성되는 서버난수를 이용하고, OTP 생성에는 최종 암호화키와 단말기 고유정보, 현재시간을 넣어서 OTP를 생성한다 [8]. 이 방식의 경우, 서버는 OTP 생성에 있어 중요한 정보인 사용자 식별정보 또는 해쉬된 사용자 식별정보를 안전하게 보관해야 하기 때문에 암호화가 필수적이며, 만약 암호화하지 않고 보관한다면 Stolen-verifier 공격에 취약한 문제가 있다.

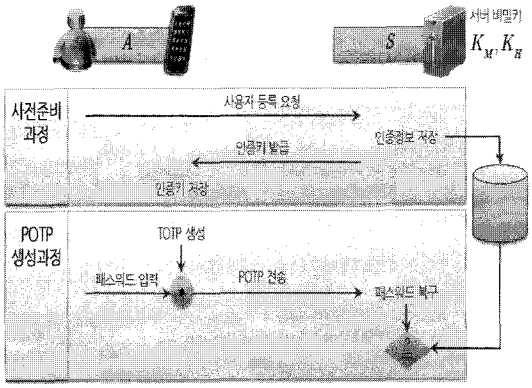
이니텍의 MOTP는 특허 제 100412986호로 등록되어 있으며, 인증방식을 그림으로 표현하면 [그림 8]과




〔그림 7〕 U-OTP 생성방법




〔그림 8〕 MOTP 생성방법




〔그림 9〕 전체 과정 요약

같다. MOTP는 패스워드와 같은 사용자 데이터를 입력받은 후 이를 해쉬하여 일회용 비밀번호 생성용 Seed 값으로 사용한다. MOTP는 시간/카운터 동기화를 모두 사용하는데, 일정 범위의 시간내에서 카운터에 따라 다른 OTP를 사용하는 방식이다 [11].

MOTP 역시 U-OTP와 마찬가지로 OTP 생성 시 사용자 식별정보를 입력하고 있기 때문에 U-OTP의 경우와 마찬가지로 사용자 식별정보의 안전한 보관이 문제가 될 수 있다. MOTP 방식과는 별도로 서비스 운영에 있어서 몇 가지 문제점이 발견되었는데, MOTP를 이용하는 '던전앤파이터'라는 게임 사이트의 경우, 다음과 같은 세 가지 문제점이 발견되었다. 첫 번째 문제점은 단말기에서 이벤트 값을 초기화할 수 없기 때문에, MOTP 단말기와 서버 간 카운터 동기가 어긋나게 되면 다음 동기화 유지시까지 인증 오류를 발생시킨다. 두 번째 문제점은 인증 시 오류 횟수 제한을 두지 않는다는 점이다. 보통 3회 연속 인증 실패 시 일정 시간동안 인증을 제한하는 등의 조치를 취하지만, 해당 서비스는 그러한 기능을 제공하지 않는다. 마지막으로 패스워드와 같은 사용자 식별정보를 단말기에서 인증하기 때문에 단말기 분실 시 오프라인 패스워드 추측 공격에 노출될 수 있는 문제가 있었다. 물론 단말기 내에서 패스워드 연속 검증 횟수를 제한하고 있지만, 모바일 소프트웨어의 경우 역공학 등의 방법으로 얼마든지 이러한 제한을 풀 수 있기 때문에 단말기에서의 패스워드 인증은 피하는 것이 바람직하다.

## Ⅲ. 제안 방식

본 장에서는 제안 방식인 POTP에 대해 설명한다. 제안 방식은 OTP와 패스워드를 이용하여 사용자를 인증하는 Two-factor 인증 방식이지만, 패스워드를 OTP 생성에 사용하는 것이 아니라, 생성된 OTP와 결합시켜 전송하는 방식이다. 전체 과정을 요약하면 [그림 9]와 같다.

### 3.1. 용어 정의

제안하는 방식에서 사용하는 기호와 정의는 다음과 같다.

- $ID_A$ : 사용자 $A$의 ID
- $PW_A$ : 사용자 $A$의 패스워드
- $PW_A'$, $PW_A''$ : 사용자 $A$의 패스워드를 반으로 나눈 형태로 3.2.2 절에서 자세히 설명
- $A$ : 사용자 (클라이언트)
- $S$ : 인증 서버
- $K_M$, $K_H$ : 인증 서버 $S$가 선택한 고유의 비밀 키
- $H(\cdot)$ : 암호학적으로 안전한 해쉬 함수 (SHA-1 또는 SHA-256)
- $HPW_A'$, $HPW_A''$ : $PW_A'$, $PW_A''$를 해쉬한값 ($HPW_A' = H(K_H \| PW_A')$, $HPW_A'' = H(K_H \| PW_A'')$)
- $t$ : OTP 생성 시간
- $TOTP(\cdot, \cdot)$ : 시간 기반 OTP 생성 함수
- $K_A$ : 사용자 $A$가 OTP를 생성하는데 필요한 키 ($K_A = H(K_M \| ID_A)$)
- $OTP_A$ : 사용자 $A$가 $t$ 시간에 생성한 OTP

($OTP_A = TOTP(K_A, t)$)

- $POTP_A$ : 사용자 $A$의 패스워드 $PW_A$가 포함된 OTP ($POTP_A = OTP_A \odot PW_A$)
- $\odot$, $\odot^{-1}$ : (mod 100) 상에서의 덧셈과 뺄셈 연산

### 3.2. TOTP와 패스워드를 이용한 Two-Factor 인증 방식

#### 3.2.1. 사전 준비 과정

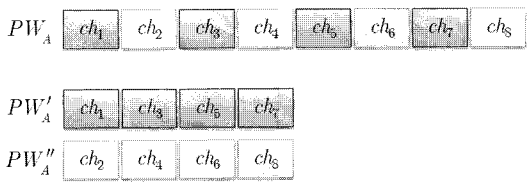
서버 $S$는 비밀키 $K_M$과 $K_H$를 정하여 안전하게 보관한다. $K_M$은 사용자마다 서로 다른 OTP 생성용 키를 발급하기 위한 용도이고, $K_H$는 패스워드를 안전하게 저장하기 위해 사용된다. $K_M$과 $K_H$는 추측 공격에 안전하도록 최소한 128비트 이상의 길이를 가져야 한다.

사용자 $A$는 자신의 아이디 $ID_A$와 패스워드 $PW_A$를 전송하여 사용자 등록과정을 시작한다. $ID_A$와 $PW_A$는 $POTP_A$ 검증을 위해 $S$가 보관하고 있어야 하며, $PW_A$는 사전 준비 과정에서만 전송하여 노출 위험성을 낮춘다.

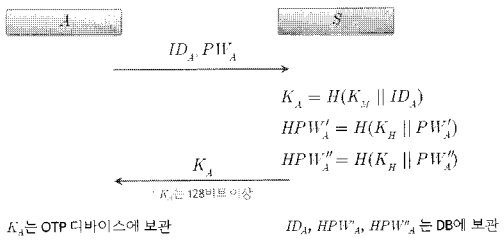
$S$는 $K_M$과 $ID_A$를 이용하여 OTP 생성키인 $K_A = H(K_M \| ID_A)$를 계산한다. 그리고 $A$에 대한 인증 정보인 $PW_A$를 안전하게 저장하기 위해 다음과 정을 거친다.

1. $PW_A$가 8자리 미만인 경우 0x00으로 패딩하고, 초과하는 경우에는 해당 문자들을 제거하여 길이를 8자리로 맞춘다.
2. 자리수 변경 과정을 마친 $PW_A$는 각각 1, 3, 5, 7 번째 문자로 구성된 $PW_A'$와 2, 4, 6, 8 번째 문자를 갖는 $PW_A''$로 분할된다 ([그림 10] 참조).
3. $S$는 $PW_A$를 $PW_A'$, $PW_A''$ 두 개로 나누고, $K_H$와 연접하여 해쉬한 후 데이터베이스에 각각 저장한다.
4. 서버 $S$는 사용자 $A$의 OTP 생성용 비밀키인 $K_A$를 $A$에게 전송한다.

위의 과정을 마친 후 $A$는 $S$로부터 제공받은 $K_A$를




〔그림 10〕 사용자 패스워드의 분할

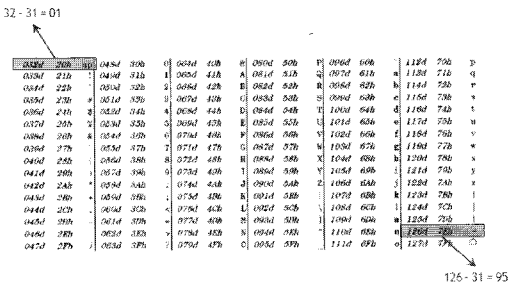



〔그림 11〕 사전 준비 과정

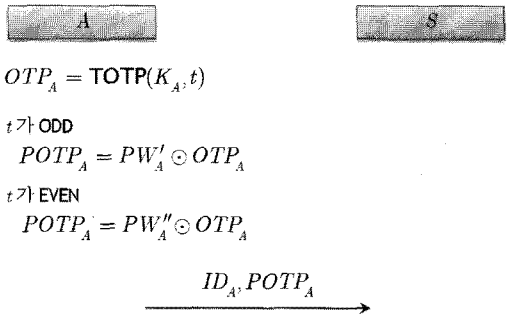
디바이스에 보관하는데, 이상의 내용을 그림으로 표현하면 [그림 11]과 같다.

#### 3.2.2. $POTP_A$ 생성 과정

$A$는 로그인을 위해, OTP 생성키 $K_A$와 OTP 생성시간 $t$를 TOTP 알고리즘에 입력하여 $OTP_A$를 생성한다. $OTP_A$는 $POTP_A$를 생성하기 위해 사용되는데, 이 때 $t$가 홀수이면 $PW_A'$, 짝수이면 $PW_A''$을 선택한 후 해당 값과 함께 $\odot$연산되어 $POTP_A$를 생성한다. $\odot$연산은 (mod 100)상의 덧셈연산으로 정의되는데, 8자리 숫자로 구성된 $OTP_A$를 앞에서부터 두 자리씩 나누어 4개의 두자리숫자 $N_{11}$, $N_{12}$, $N_{13}$, $N_{14}$ ($00 \le N_{1i} \le 99$, $1 \le i \le 4$) 를 생성하고, 4 자리 문자로




〔그림 12〕 아스키코드 값의 두 자리수 변형



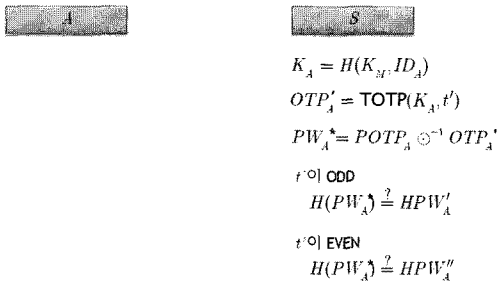

〔그림 13〕 $POTP_A$ 생성 과정

구성된 $PW_A'$ 또는 $PW_A''$로부터 각 문자의 ASCII 코드 -31의 연산을 거쳐 $N_{21}$, $N_{22}$, $N_{23}$, $N_{24}$ ($01 \le N_{2i} \le 95$, $1 \le i \le 4$)를 생성한다 ([그림 12] 참조). 그리고 $i = 1, 2, 3, 4$에 대해, $N_{1i} \odot N_{2i} = N_{1i} + N_{2i} \pmod{100}$을 통해 8자리 숫자로 구성된 $POTP_A$를 생성한다.

위의 과정을 마친 후 $A$는 $S$에게 $ID_A$와 $POTP_A$를 전달하며, 이상의 내용을 그림으로 표현하면 [그림 13]과 같다.

### 3.2.3. $POTP_A$ 검증 과정

$S$는 $K_M$과 전달된 $ID_A$를 이용하여 $K_A = H(K_M, ID_A)$를 생성하고, 검증시간 $t'$을 이용하여 $OTP_A'$을 계산한다. $A$가 전송한 $POTP_A$와 $OTP_A'$을 이용하여 $POTP_A \odot^{-1} OTP_A'$ 연산을 통해 $PW_A^*$를 얻는다. 이 값을 $K_H$와 연접하여 해쉬한 후 $t'$이 홀수이면 $HPW_A'$, 짝수이면 $HPW_A''$을 데




〔그림 14〕 $POTP_A$ 검증 과정

이터베이스로부터 가져와 비교하여 인증 절차를 진행한다. 이상의 내용을 그림으로 표현하면[그림14]와 같다.

만약, $t = t'$이고, $PW_A$와 $K_A$가 정확하다면 서버의 인증도 문제없이 진행될 수 있지만, $t \neq t'$일 수도 있기 때문에 서버는 $t'-1$과 $t'+1$의 경우도 고려하여 인증 절차를 진행해야 한다 [6].

## Ⅳ. 안전성 및 효율성 분석

### 4.1. 안전성 분석

제안한 방식의 안전성을 분석하기에 앞서, 본 절에서는 다음 두가지를 가정한다.

(가정 1) 서버의 키 $K_M$과 $K_H$는 안전하게 보관되고 있으며 공격자에게 노출되지 않는다.

(가정 2) $OTP_A = TOTP(K_A, t)$가 주어졌을 때, $OTP_A$와 $t$를 이용하여 $K_A$를 찾는 것은 계산상 불가능하며, $t' \neq t$인 $TOTP(K_A, t')$을 예측하는 것도 계산상 불가능하다.

(가정 2)의 경우 $K_A$를 찾을 수 있다면 해쉬 알고리즘을 역변환할 수 있다는 의미가 되기 때문에 합리적인 가정으로 생각할 수 있으며 사실상 $TOTP(\cdot, \cdot)$ 연산 결과는 무작위 난수와 구별할 수 없음을 의미한다. 본 절에서는 이러한 가정 사항에 근거하여 제안한 POTP 방식이 패스워드 추측 공격과 stolen verifier 공격에 안전함을 증명한다.

#### 4.1.1 Password Guessing Attack

패스워드 추측 공격(password guessing attack)이란, 정당한 사용자의 패스워드를 모르는 상태에서 무작위로 패스워드를 추측하여 대입해 보는 공격 방법을 말한다. 일반적으로 $k$ 비트 길이의 난수를 추측하기 위해서는 평균적으로 $2^{k-1}$ 회를 시도해야 하지만, 패스워드의 경우 사람이 기억하기 쉬운 공간에서 선택된다는 특성 때문에 이보다 훨씬 적은 시도만으로도 성공할 확률이 높다. 따라서, 사용자 패스워드를 이용한 인증 시

스템은 패스워드 추측 공격을 고려하여 안전하게 설계해야 한다.

(정리 1) $OTP_A \odot PW'$과 $OTP_A \odot PW''$로 부터 $PW'$과 $PW''$을 찾는 것은 계산상 불가능하다.

(증명) $R$을 무작위 난수라고 가정했을 때, 공격자는 $S_1 = R \odot PW$와 $S_2 = OTP_A \odot PW$를 구별할 수 없다. 만약 $S_1$과 $S_2$를 구별할 수 있다면 공격자는 $R$과 $OTP_A$를 구별할 수 있다는 의미이고, 이는 (가정 2)에 어긋나게 된다. 따라서, 공격자는 $S_1 = OTP_A$와 $S_2 = OTP_A \odot PW$ 역시 구별할 수 없기 때문에 $OTP_A \odot PW'$과 $OTP_A \odot PW''$을 알더라도 $PW'$이나 $PW''$을 찾아낼 수 없다. □

제안한 방식의 경우 사용자 $A$가 인증을 위해 전송하는 정보는 $POTP_A$이며, 이는 $OTP_A \odot PW'$ 또는 $OTP_A \odot PW''$이다. 공격자는 무작위로 추측한 패스워드 $PW^*$를 이용하여 $OTP^*_A = OTP_A \odot PW' \odot {}^{-1}PW^*$를 계산할 수 있다. 하지만, $OTP^*_A$와 $OTP_A$를 구별할 수 없기 때문에 이러한 공격은 무의미하다.

#### 4.1.2 Stolen Verifier Attack

사용자 인증을 위한 서버 시스템은 모든 사용자의 인증 정보를 데이터베이스에 보관하기 때문에 공격자들의 주요 공격 대상이 되곤 한다. 이런 이유로 대부분의 인증 서버 시스템은 패스워드와 같은 중요 인증 정보를 평문 형태로 보관하지 않고 해쉬 결과와 같은 암호학적 변형 결과를 보관하는 것이 일반적이다. Stolen-verifier 공격이란 서버에서 획득한 인증 정보를 이용하여 합법적인 사용자로 인증 절차 통과를 시도하거나 획득한 인증 정보로부터 패스워드 추측 공격을 시도하는 것을 말한다 [2]. 따라서, 인증 시스템이 Stolen-verifier 공격에 안전하기 위해서는 공격자가 서버의 인증 정보를 획득하더라도 그로부터 어떠한 정보도 얻을 수 없어야 한다.

제안한 방식의 경우 서버는 사용자 $A$에 대한 인증 정보로 $HPW_A' = H(K_H \| PW_A')$와 $HPW_A'' = H(K_H \| PW_A'')$을 보관하고 있으며, $K_H$는 서버의 키로 안전하게 보관되고 있다. 따라서, $K_H$를 모른다면 $HPW_A'$로부터 $PW_A'$를 찾을 수는 없으며, 마찬가지로 $PW_A''$를 찾는 것도 계산상 불가능하다고 할 수 있다. 또한 서버는 클라이언트로부터 $HPW_A'$이나 $HPW_A''$을 입력받지 않기 때문에 공격자가 $HPW_A'$와 $HPW_A''$을 획득하더라도 사용자 $A$로 위장하는 것은 불가능하다.

### 4.2. 효율성 분석

제안한 방식의 인증 과정은 크게 $OTP_A$를 생성하는 단계와 $POTP_A$를 생성하는 단계로 구분할 수 있다. 이 때, $POTP_A$는 $OTP_A \odot PW_A$ 연산을 거쳐 생성하는데, 다른 방식으로 $OTP_A$를 생성하는 단계에서 패스워드를 입력하는 것을 생각할 수 있다. 즉, $OTP_A = TOTP(PW_A, K_A, t)$를 이용하는 것인데, 이 경우 고려해야 할 것은 서버가 사용자 $A$의 패스워드인 $PW_A$를 암호화하여 데이터베이스에 보관해야 한다는 점이다. 즉, 사용자가 인증 요청을 할 때마다 데이터베이스에 보관된 사용자 패스워드를 복호화하여 검증해야 하는데, 복호화(또는 암호화) 연산은 해쉬 연산보다 느리기 때문에 효율성이 떨어지게 된다. [8] 에 따르면, SHA-1 연산의 경우 153MiB/s인데 비해 128비트 키를 사용한 AES/ECB 연산은 109MiB/s로 SHA-1 대비 연산 속도가 약 71% 수준이며 DES-EDE3는 13MiB/s로 약 8.5%에 불과하다. 따라서, 사용자의 인증 정보를 보관함에 있어 암호화 연산을 적용하는 것보다 해쉬 연산을 적용하는 것이 훨씬 효율적임을 알 수 있다.

제안한 POTP와 U-OTP 및 MOTP의 안전성 및 효율성을 비교한 결과는 다음 [표 1]과 같다.

1) 모바일 단말에서 패스워드를 확인하기 때문에 오프라인 패스워드 추측 공격 가능

〔표 1〕 안전성 및 효율성 비교

| 비교 항목 | POTP | U-OTP | MOTP |
|---|---|---|---|
| 재전송 공격 | 안전 | 안전 | 안전 |
| 패스워드 추측공격 | 안전 | 안전 | 취약[1)] |
| Stolen-verifier 공격 | 안전 | 취약[2)] | 취약[2)] |
| 서버의 인증정보 보관 | 해쉬 | 암호화[2)] | 암호화[2)] |

〔표 2〕 구현 환경

| | | |
|---|---|---|
| 클라이언트 | 플랫폼 | 모토로이 / 갤럭시 S / AVD + ecliplse |
| | OS 버젼 | Android v2.1 / v2.2 |
| 서버 | OS | WINDOWS XP |
| | 웹서버 | Apache Tomcat + JSP |
| | 데이터베이스 | mySQL |

2) U-OTP와 MOTP 모두 패스워드를 입력받아 OTP를 생성하는 방식이기 때문에 서버에서는 OTP 생성을 위해 인증 정보(패스워드)를 평문으로 복원할 수 있어야 함. 따라서, 해쉬 함수는 이용할 수 없기 때문에 연산 효율성이 저하되며, 암호화 하지 않고 보관하는경우 Stolen-verifier 공격에 취약함.

## V. 제안 방식의 구현

### 5.1. 구현 환경

POTP 생성을 위한 클라이언트는 Android v2.1 및 v2.2 상에서 eclipse를 이용하여 구현하였고, 모토로이, 갤럭시 S, 그리고 AVD를 통해 테스트하였다. 검증용 서버는 Apache Tomcat과 JSP를 사용하였으며, 데이터베이스로는 mySQL을 이용하였다.

### 5.2. 구현 세부사항

#### 5.2.1. 변수 입력방식

제안한 방식을 구현하기 위해 설정한 주요 파라미터 값의 길이는 다음과 같다.

- $K_A$ : 128 비트
- $PW_A$ : 자릿수 제한은 없으나 구현상 문제로 최대 8자리 (3.2.2절 참조)
- $HPW_A'$, $HPW_A''$:160 비트(SHA-1 이용)

#### 5.2.2. 데이터베이스 설계

클라이언트는 서버에서 제공한 인증키 $K_A$를 반드시 저장하여야 하며, 패스워드는 저장하지 않기 때문에 사용자가 입력한 패스워드를 검증할 수는 없다 ([표 3] 참조).

〔표 3〕 클라이언트의 데이터베이스 테이블

| 필드명 | 타입 |
|---|---|
| 서버 이름 | 문자열 |
| 인증키 | 바이트열(128-bit) |

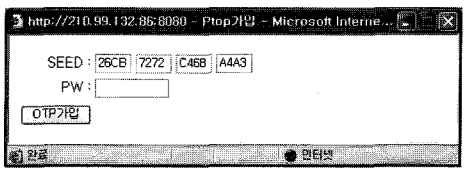
〔표 4〕 서버의 데이터베이스 테이블

| 필드명 | 타입 |
|---|---|
| ID | 문자열 |
| HPW1 | 바이트열(160-bit) |
| HPW2 | 바이트열(160-bit) |
| time | time |

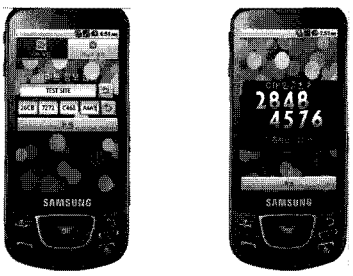
서버의 데이터베이스는 해쉬된 패스워드 $HPW_A'$와 $HPW_A''$을 보관하기 위한 HPW1과 HPW2, 그리고 ID 및 최종 인증 시간 등을 보관하기 위한 ID와 time 필드 등으로 구성된다 ([표 4] 참조).

### 5.3. 구현 결과

[그림 15-(a)]은 테스트용 페이지로부터 OTP 생성용 비밀키 $K_A$를 획득한 화면이고, [그림 15-(b)]는 디바이스에 등록할 사이트 이름과 획득한 비밀키 $K_A$를 저장하는 화면이며, [그림 15-(c)]는 모바일 단말에서 8자리 OTP를 생성하는 화면이다




(a) 서버로부터 인증키 획득




(b) 인증키 저장 (c) OTP 생성

〔그림 15〕 구현 결과

## Ⅵ. 결론

다른 암호 알고리즘과 마찬가지로, OTP의 경우도 SHA-1이나 SHA-256 등을 기반으로 하기 때문에 매우 안전하지만, 그렇다고 해서 이를 이용하는 서비스가 항상 안전한 것은 아니다. OTP 생성에 패스워드를 입력 값으로 사용하는 경우 단말에서 패스워드를 검증하게 된다면 단말기 분실시 오프라인 패스워드 추측 공격에 취약할 수 있으며, 서버에서 연속적인 인증 실패를 검사하지 않는다면 온라인 패스워드 추측 공격에 취약할 수 있다. 또한, 패스워드를 입력 값으로 사용하는 경우 서버는 암호화 등을 이용하여 패스워드를 안전하게 보관해야 Stolen-verifier 공격에 안전할 수 있다.

위와 같은 문제점을 해결하시 위해서, 두 가지 인증 정보 즉, 시간 기반 OTP 생성방식인 TOTP 및 패스워드를 이용하는 새로운 Two-Factor 인증 프로토콜인 POTP를 제안하였다. 제안한 방식은 재전송 공격에 안전하며, 공격자가 OTP 생성용 디바이스를 획득하더라도 패스워드를 유추할 수 없고 서버의 인증 정보 데이터베이스를 획득하더라도 정상적인 사용자로 위장할 수 없는 장점이 있다. 또한, 제안한 방식은 사용자 단말과 패스워드를 모두 이용하는 Two-factor 사용자 인증 방식이지만 패스워드를 입력 값으로 사용하지 않기 때문에 패스워드를 암호화해서 보관하지 않아도 되는 장점이 있다. 제안한 방식을 이용한다면 금융 보안이나 기타 OTP를 필요로 하는 온라인 서비스 상에서 보다 안전한 사용자 인증 서비스를 제공할 수 있다.

## 참고 문헌


[1] CryptoCard,"CRYPTO," http://www.cryptocard.com/index.php?option= com_content&view=article&id=253&Itemid=31/

[2] C.M.ChenandW.C.Ku, "Stolen-Verifier Attack on Two New Strong-Password Authentication Protocols," IEICE Trans. on Communications, Vol. E85-B, No. 11, pp. 2519-2521, 2002.

[3] IETF, "HOTP: An HMAC-Based One-Time Password Algorithm," RFC 4226, 2005.

[4] IETF, "TOTP: Time-based One-time Password Algorithm," http://tools.ietf.org/html/draft-mraihi-totp-timebased-05, 2010.

[5] OATH, http://www.openauthentication.org/

[6] RSA Security, "SecureID," http://www.rsa.com/

[7] Vasco, "Digpass," http://www.vasco.com/products/digipass/digipass_ index.aspx

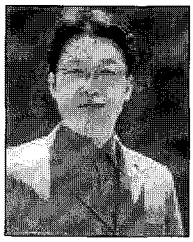
[8] Wei Dai, "Crypto++ 5.6.0 Benchmarks," http://www.cryptopp.com/ benchmarks.html, 2009. 03.

[9] 에이티솔루션, "U-OTP의 원리 및 서비스 방식(1)," http://www.u-otp.co.kr/blog/42/

[10] 에이티솔루션, "일회용 비밀번호 생성방법과 키 발급 시스템 및 일회용 비밀번호 인증 시스템," 국내특허 제1020070001141호(출원), 2007. 01.

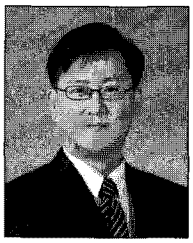
[11] 이니텍, "동기식 일회용 비밀 번호 생성 및 인증 방법 및 그러한 일회용 비밀 번호를 생성하는 프로그램이 기록된 컴퓨터 판독 가능 기록 매체," 국내특허 제100412986호(등록), 2003. 12.

## 〈著者紹介〉

**양 형 규(Hyung Gyu Yang)**

1995년 2월 : 성균관대학교 정보공학과 공학박사
1995년~현재 : 강남대학교 컴퓨터미디어공학부 교수
<관심분야> 암호이론, 정보이론, 정보보호, 디지털 워터마킹



**이 윤 호(Yunho Lee)**

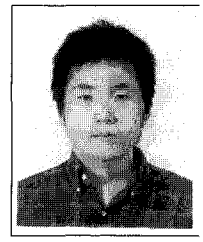
정회원

2008년 2월 : 성균관대학교 컴퓨터공학과(공학박사)
1993년 3월-2000년 4월 : 한국통신 연구개발본부 전임연구원
2000년 5월-2005년 1월 : KBS인터넷(주) 기술지원팀장
2006년 6월-2008년 2월 : (주)애니온소프트 기술이사
2008년 3월-현재 : 성균관대학교 정보통신공학부 연구교수
<관심분야> 암호이론, 정보보호 응용, 전자투표, 워터마킹

**제 주 환(Juhwan Jae)**

정회원

2004년 3월~2010년 9월 : 강남대학교 컴퓨터공학부 학생
2010년 8월-2010년 11월 : 오라클 실무자 양성반
2011년 9월~현재 :(주)신시웨이
<관심분야> 오라클, 서버



**임 학 창(Hakchang Lim)**

정회원

2005년 3월~현재 : 강남대학교 컴퓨터공학부 학생
<관심분야> 안드로이드

**유 승 록(Seunglok Yoo)**

정회원

2005년 3월~현재 : 강남대학교 컴퓨터공학부 학생
관심분야 : 정보보안

**배 동 환(Donghwan Bae)**

정회원

2007년 3월~현재 : 강남대학교 컴퓨터공학부 학생
관심분야 : 안드로이드, 윈도우개발, 서버