Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.27, NO.6, Dec. 2017 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2017.27.6.1251

재등록이 필요 없는 암호 해시체인 기반의 일회용 패스워드 인증기법*

신 동 진,† 박 창 섭‡
단국대학교

One-Time Password Authentication Scheme Based on Cryptographic Hash Chain without Re-Registration*

Dong-jin Shin,† Chang-seop Park‡
Dankook University

요 약

고정된 패스워드 그리고 패스워드의 사전공유라는 단순 패스워드가 지니는 문제점을 해결하기 위해 해시체인 기반의 일회용 패스워드가 제안되었다. 루트 해시값을 사전에 등록시킨 후에 사용하기 때문에 고정된 패스워드의 문제점을 해결하였으나, 해시체인을 구성하는 해시값들이 소진된 이후에는 새로운 해시체인의 루트 값을 재등록 하는 단점을 가지고 있다. 재등록을 필요로 하지 않는 여러 유형의 해시체인 기반의 일회용 패스워드 기법들이 제안되었으나, 제약조건 및 효율성 측면에서 문제점들을 내포하고 있다. 본 논문에서는 재등록이 요구되지 않으면서 기존 제약조건을 만족하면서도 매 인증 시 각 2회의 암호해시함수만으로 일회용 패스워드를 생성하고 이를 검증하는 해시체인 기반의 일회용 패스워드 기법을 새로이 제안하고 기존 기법들과 보안요구사항 및 효율성 측면에서 비교 분석한다.

ABSTRACT

One-time password has been proposed for the purpose of addressing the security problems of the simple password system: fixed passwords and pre-shared passwords. Since it employs the consecutive hash values after a root hash value is registered at the server, the security weakness of the fixed passwords has been addressed. However, it has a shortcoming of re-registering a new root hash value when the previous hash chain's hash values are exhausted. Even though several one-time password systems not requiring re-registration have been proposed, they all have several problems in terms of constraint conditions and efficiency. In this paper, we propose the one - time password scheme based on a hash chain that generates one - time passwords using only two cryptographic hash functions at each authentication and satisfies the existing constraints without re-registration, Security requirements and efficiency.

Keywords: One-time Password, Hash Chain

Received(09. 01. 2017), Modified(10. 17. 2017), Accepted(10. 27. 2017)

* 이 논문은 2017년도 정부(교육부)의 재원으로 한국 연구재단의 지원을 받아 수행된 기본 연구지원사업 성과임.(NRF-2017R1D1A1B03027862)

* 본 연구는 미래창조과학부 및 한국인터넷진흥원의 "고용 계약형 정보보호 석사과정 지원사업"의 연구결과로 수행되었음 (과제번호 H2101-17-1001)

† 주저자, tls057@naver.com

‡ 교신저자, csp0@dankook.ac.kr(Corresponding author)

I. 서 론

클라이언트의 사용상의 편리성에 초점을 맞춘 "단순 패스워드"의 3가지 문제점은 첫째, 매번 사용시마다 고정된 패스워드가 사용되며 둘째, 서버측에 동일한 패스워드가 저장된다는 데에 있다. 비록, 해당 패스워드가 서버측에서 암호화 또는 해시 처리된 상태에서 저장된다고는 하지만 패스워드의 검증과정에서 해당 패스워드의 노출은 피할 수 없게 된다. 다양한 보안위협 모델 하에서의 패스워드 기반 인증방식들이 제안됨에 따라서 서버측에 클라이언트와 동일한 비밀정보를 저장하여 사용하는 방식의 문제점들은 꾸준히 지적되어 왔다. 셋째, 클라이언트가 사용자인 경우에는 어떠한 보조수단 없이도 패스워드를 사용해야 하기에 기억하기 쉬운 제한된 유형의 패스워드를 사용함으로써 패스워드에 대한 온라인 및 오프라인 추측공격이 가능하게 된다.

*Lamport*에 의해 제안된 해시체인(hash chain) 기반의 일회용 패스워드(One-Time Password)[1]는 이러한 단순 패스워드 방식의 문제점을 해결하는 클라이언트 인증방식이다. 즉, 클라이언트가 선정한 임의의 난수(random number)에 대해 암호해시함수를 연속적으로 적용하여 생성된 해시체인의 해시값들을 역순으로 사용하는 것이다. 서버측에는 항상 해시체인 상에서 이미 사용된 최신 해시값만이 저장되기 때문에, 다음 인증세션에서 사용될 해시값의 도출은 암호해시함수의 특성상 불가능하게 된다. 특히, 클라이언트가 사용자인 경우 보조수단을 활용하여 해시값들을 11비트씩 그룹화하여 단어로 대체하여 사용할 수 있는 S/Key 방식[2]도 제안되었다. 암호해시함수가 수반하는 계산부담은 타 암호 프리미티브들에 비해 낮기 때문에 사용상의 효율성도 보장된다. 해시체인 기반의 일회용 패스워드는 단순 패스워드의 문제점을 해결한 측면도 있지만 자체적인 단점도 지니고 있다. 첫째, 유한한 길이의 해시체인이 생성되기 때문에 해시체인을 구성하는 해시값들이 다 소진된 이후에는 새로운 해시체인을 생성하여 서버측에 재등록 하는 과정이 필요하며 둘째, 클라이언트측면에서는 해시체인 구성 값들을 사전에 저장하고 있거나 또는 매번 인증과정에서 초기 값을 기반으로 암호해시함수 계산을 여러 번 수행해야 하는 부담이 존재한다.

해시체인 기반의 일회용 패스워드가 지니는 이러한 단점을 보완하기 위한 다양한 기법들이 제안되있는데, 특히 재등록 과정을 우회하기 위한 기술에 초점을 맞추고 있는 기법들이다. 무한길이의 해시체인(infinite hash chain) 또는 자가 해시체인(self-renewable hash chain) 등으로 명명된 이들 기법[5, 6]들은 별도의 재등록을 우회하기 위한 방안들이지만 단순 패스워드의 문제점인 클라이언트와 서버 간에 비밀정보의 공유를 전제로 하고 있거나, 공개키 암호 및 전자서명의 활용을 전제로 하고 있기 때문에 해시체인 기반의 일회용 패스워드가 가지는 장점을 살리지 못하는 제안들이다. 본 논문에서는 암호해시함수 이외의 타 암호 프리미티브를 사용하지 않고, 해시체인 기반 일회용 패스워드가 지니는 장점을 그대로 유지하면서 자체적인 단점을 보완하는 기법을 제안한다. 본 논문의 구성은 다음과 같다. 2장에서는 해시체인 기반의 일회용 패스워드의 보안요구사항과 기존연구들을 소개하며, 3장에서는 재등록이 필요 없는 해시체인 기반 일회용 패스워드를 새로이 제안한다. 4장에서는 안전성 및 효율성을 타 기법들과 비교 분석하며, 5장에서 결론을 맺는다.

II. 해시체인 기반 일회용 패스워드의 보안요구사항 및 기존연구

2.1 가정 및 보안요구사항

Lamport 일회용 패스워드에서는 클라이언트가 임의로 생성한 난수 *seed*에 대해 암호해시함수 $h(.)$를 연속적으로 적용하여 해시체인 $\{x_{j-1}=h(x_j),\ j = 1, 2, \ldots, n \text{ and } x_n = seed\}$을 생성하여 루트값 x_0를 서버측에 사전 등록하고, 매번 인증세션에서 클라이언트는 해시값 x_j를 일회용 패스워드로 서버에게 전송하면 서버는 $h(x_j)$를 계산하여 등록된 x_{j-1}와 일치하는지를 확인함으로써 클라이언트에 대한 인증이 수행된다. 해당 인증세션이 성공한 경우에는 서버측에 저장되어 있는 해시값이 클라이언트가 전송한 새로운 해시값으로 갱신 저장된다. 클라이언트는 초기 해시체인 생성과정에서의 해시값들을 사전에 저장한 이후에 하나씩 사용할 수도 있고, 또는 매번 $seed$ 값을 기반으로 $n-j$번의 해시함수 계산을 통해 생성된 해시값을 사용할 수도 있다. 결국 n번의 인증세션이 종료된 이후에는 해시체인의 해시값들이 모두 소진되기 때문에 새로운 해시체인을 생성하여 신규 루트값을 재등록 하여야 한다. *Lamport* 일회용 패스워드의 특징은 클라이언트와 서버간에 사전

공유되는 비밀정보가 존재하지 않는다는 데에 있다.

Lamport 일회용 패스워드를 이용한 안전한 클라이언트 인증을 위해서는 다음과 같은 보안요구사항이 충족되고 있다고 가정해야 한다. 첫째, 클라이언트와 서버간에는 무결성이 보장되는 채널이 존재하며 서버측에 대한 인증은 별도방식으로 행해진다. 이 보안요구사항이 충족되지 못한다면 클라이언트와 서버간의 인증세션에 대해서 중간자(man-in-the-middle) 공격이 가능하게 된다. 이 보안요구사항은 서버 인증서 (server certificate) 보안모드 (security mode)의 SSL/TLS이 기동된다면 충족이 가능하다. 둘째, 서버측에 저장되는 클라이언트의 해시값에 대한 무결성은 보장되어야 한다. 이는 서버가 공격을 당했을 경우 또는 서버측의 내부 공격자들에 의해 서버측에 저장된 클라이언트의 해시값을 공격자의 해시값으로 변조하여 정당한 클라이언트를 가장(impersonation)하는 공격을 방지하기 위한 목적이다. 이는 서버측에서 루트 값 및 새로운 해시값을 전자서명하여 저장한다면 역시 충족이 가능하다. 서버측에는 클라이언트와의 비밀정보가 없기 때문에 저장된 정보의 기밀성은 요구되지 않는다.

2.2 기존연구 분석

본 절에서는 해시체인 소진 후에 새로운 해시체인의 루트 값에 대한 재등록을 우회하기 위한 기존연구들을 분석한다. 비록, *Challenge-Response* 방식의 OTP 인증기법[3, 4]들도 존재하기는 하지만, 이들은 양자간에 비밀정보의 공유를 전제로 하기 때문에 논의의 대상에서 제외하기로 한다. *Bicakci* 등에 의해서 제안된 무한 체인방식의 일회용 패스워드 [5]는 클라이언트가 생성한 *seed* 값에 클라이언트의 *RSA* 개인키 d로 서명한 값 $P_1 = A_1(seed, d)$을 공개키 e와 함께 서버측에 사전 등록한다. 매번 인증세션에서 클라이언트는 $P_j = A_j(P_{j-1}, d)$, $j = 2, 3,, 4 \cdots$, 을 서버측에 전송하면, 서버는 클라이언트의 공개키로 P_j을 복호화하여 등록된 P_{j-1}과 일치하는지를 확인함으로써 클라이언트에 대한 인증이 수행된다. 이 방식은 *Lamport* 일회용 패스워드의 재등록 문제점을 해결하였지만, *RSA* 방식이 적용되기 때문에 해시함수를 사용하는 *Lamport*의 일회용 패스워드보다 상대적으로 계산이 무겁다.

Zhang 등은 *Lamport* 일회용 패스워드에서 최초 해시체인의 루트 값을 등록한 이후에는 해당 해시체인이 모두 소진이 되어도 별도의 재등록이 요구되지 않는 자가갱신 (self-renewable) 해시체인 기법[6]을 제안하였다. 현재 해시체인의 해시값을 이용한 인증과정에서 새로운 해시체인의 루트 값을 구성하는 비트 값들을 사전에 서버측에 전송하는 방식이다. 즉, 해시값 x_j $\{x_{j-1} = h(x_j)$, $j = 1, 2, \cdots, n$ and $x_n = seed\}$를 일회용 패스워드로 전송하면서, 새로운 해시체인 $\{x_{j-1}' = h(x_j')$, $j = 1, 2, \cdots, n$ and $x_n' = new_seed\}$의 루트 값 $x_0' = [b_0 b_1 \cdots, b_l]$을 구성하는 비트 $b_j \in \{0, 1\}$들을 함께 미리 전송하는 방식이다. 물론 이것을 보호하기 위해 OTS (One-Time Signature) 등의 보조 암호기법이 적용된다. 결과적으로 현재 해시체인이 소진됨과 동시에 새로운 해시체인의 루트 값이 자동적으로 등록되게 된다. 하지만 이 기법들은 전송 메시지의 길이가 길어지고 특히, 클라이언트와 서버간에 해시값에 대한 불일치가 발생할 경우에는 복구가 불가능한 문제점을 가지고 있다.

Hamdy 등에 의해 제안된 무한중첩 해시체인 일회용 패스워드[4]는 양자간에 *seed* 값을 공유하면서 해시체인 생성 및 사용을 *Lamport* 방식 (역방향)과는 반대로 순방향으로 사용하는 방식이다. 즉, 순방향으로 생성된 해시체인 $\{x_j = h(x_{j-1})$, $j = 1, 2, 3, \cdots,$ and $x_0 = seed\}$을 기반으로 서버측에서 인덱스 (j, i)를 클라이언트에게 전송하면 클라이언트는 일회용 패스워드 $h_i(h_j(seed))$로 응답하게 된다. ($h_j(.)$는 해시함수를 연속적으로 j번 적용하는 것을 의미한다.) 해시함수를 순방향으로 사용하기 때문에 재등록의 필요성은 없어지지만 *seed* 값이 양자만이 공유하는 비밀정보이기 때문에 단순 패스워드의 문제점을 그대로 가지게 된다.

Kim 등이 제안한 해시체인 재활기법[9]은 *Lamport* 방식과 동일한 방식으로 해시체인을 이용해 클라이언트 인증을 하는 방식이지만, 해시체인을 구성하고 있는 해시값들을 매 번 클라이언트 인증 시 소모할 때, 다음 사이클의 해시값들을 하나씩 생성해 사용한 해시값들을 대체하여 보관한다. 클라이언트가 해시체인을 모두 소모하게 되면 다음 사이클의 해시체인을 자연스럽게 모두 보관하고 있는 상태에서 루트 해시값을 재등록 하게 되는데, 이 과정에서 서버는 양자 간 사전에 공유한 *seed* 값을 이용하여 다음

사이클의 *seed* 값을 클라이언트와 약속된 방식으로 생성하여 클라이언트에게 전송 받은 루트 해시값을 자신도 생성해 검증한다. 이 기법은 재등록의 필요성은 없어지지만 *seed* 값을 사전에 공유함으로써 단순 패스워드의 문제점을 그대로 가지고 있고, 해시체인의 길이가 길어질수록 저장공간을 많이 사용하게 되는 단점이 있다

Bittle 이 제안한 무한 해시체인 제안기법[11]은 마찬가지로 *Lamport* 가 제안한 방식의 단점인 해시체인의 해시값들을 모두 소모하게 되면 재등록을 해야한다는 점을 해결하기 위해 클라이언트와 서버간에 사전에 *seed* 값을 공유하고 클라이언트는 이 *seed* 값에 암호해시함수를 적용하여 나온 출력값에 클라이언트와 서버만 공유하고있는 암호해시함수를 적용하여 일회용 패스워드로 사용하는 방식이다. 하지만 이 방식도 사전에 비밀정보를 공유해야 한다는 단순 패스워드의 문제점을 그대로 내포하고 있다.

III. 새로운 해시체인 기반의 일회용 패스워드 제안

3.1 설계원리 및 가정

본 제안에서는 기존 *Lamport* 방식이 가지는 장점 (가변 패스워드 및 비밀정보 비 공유)을 그대로 유지하면서 단점 (인증 시마다 과다한 해시함수 계산 및 해시체인 소진 시의 재등록 과정)을 제거하는 해시함수 기반의 일회용 패스워드를 설계한다. 특히, 기존 방식들과는 달리 암호해시함수 이외의 암호 프리미티브는 사용하지 않는다. 특히, 재등록 및 인증 시마다 과다한 해시함수를 회피하기 위해서는 필연적으로 해시체인을 역방향이 아닌 순방향으로 생성해서 사용하여야 한다. 매 인증세션마다 클라이언트는 임의의 난수 x_j를 기반으로 길이가 2인 해시체인 $\{z_{j-2} = h(y_{j-1}),\ y_{j-1} = h(x_j)\}$을 생성하여 서버에 등록된 이전 해시값들과의 검증을 통해 인증하는 방식이다. 본 논문에서 세션 *Anchor*로 명명되는 x_j 값들은 상호 독립적으로 생성되어 사용되기에 순방향으로 무제한 인증이 가능하게 된다. 특히, 본 제안에서는 2.1절에서 언급한 바와 같이 *Lamport* 일회용 패스워드를 이용한 안전한 클라이언트 인증에서 가정한 동일한 보안요구사항이 충족되고 있다고 가정한다.

3.2 새로운 일회용 패스워드 인증기법

본 연구에서 제안하는 일회용 패스워드 역시, 다른 방식들과 마찬가지로 초기화 및 등록과정이 선행되어야 한다.

i) 초기화 및 등록

먼저 클라이언트는 2개의 *seed* 값 x_1, x_2를 생성한다. 생성한 x_1, x_2를 이용하여 $y_0 = h(x_1)$, $y_1 = h(x_2)$, $z_0 = h(y_1,\ x_1)$을 계산하고, 서버측에 ID_C, $\{y_0,\ z_0\}$, $j=1$을 등록한다. 특히 z_0 계산시에 매개변수가 y_1, z_1 두 개 들어가는 과정에서, 두 매개변수를 concatenation 하여 암호해시함수 $h(.)$를 적용한다. 이 과정에서 등록되는 값의 기밀성은 요구되지 않으며, 무결성만이 요구된다. 이후 일회용 패스워드 생성을 위해 클라이언트는 x_1, x_2를 보관함으로써 초기화 및 등록단계를 마친다.

ii) 해시체인 생성 및 사용

초기화 및 등록단계를 성공적으로 수행하면 서버는 ID_C, $\{y_0,\ z_0\}$, $j=1$ 클라이언트는 ID_C, $\{x_1,\ x_2\}$를 보관하게 된다. 이후 클라이언트는 첫 번째 인증시도 시 자신의 식별자인 ID_C를 서버측에 전송하게 되면 서버는 전송 받은 식별자인 ID_C에 해당하는 인덱스 j를 클라이언트에게 전송한다. 인증횟수가 증가함에 따라 인덱스 j의 값이 증가하지만, 첫 번째 인증시도라고 하면 서버에게 전송받은 인덱스 j의 값은 1이다. 서버에게 인덱스 $j=1$를 전송 받은 클라이언트는 인덱스 $j=1$에 해당하는 $\{x_1,\ y_1,\ z_1\}$를 계산한다. 클라이언트는 x_1과 x_2를 이미 보관하고 있기에 x_1, $y_1 = h(x_2)$를 계산할 수 있지만, $z_1 = h(y_2,\ x_2)$의 매개변수인 $y_2 = h(x_3)$를 계산하기에는 x_3을 가지고 있지 않으므로 새로운 임의의 난수 *seed* 값인 x_3을 생성 및 보관하고 $y_2 = h(x_3)$을 계산하여 $z_1 = h(y_2,\ x_2)$를 계산한다. 계산이 모두 끝나면 클라이언트는 $\{x_1,\ y_1,\ z_1\}$을 일회용 패스워드로 사용하여 서버측에 전송하면서 인증을 시도한다. 서버는 전송 받은 $\{x_1,\ y_1,\ z_1\}$을 본인이 보관하고 있던 y_0과 z_0을 이용해 클라이언트에 대한 검증을 하게 된다. 먼저 $h(x_1) = y_0$이 일치하는지를 확인하고, 일치하다면 $h(y_1,\ x_1) = z_0$를 비교하면서 클라이언트에 대한 검증이 수행된다. 올바르게 클라이언트에 대한 검증이 완료되면, 서버는 본인이 보관하고 있던 $\{y_0,\ z_0\}$대신

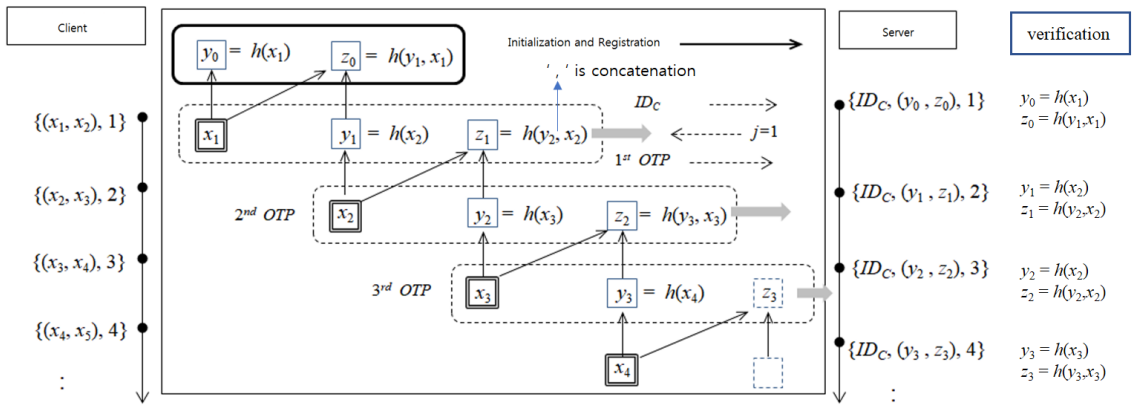

Fig. 1. Proposed Authentication Scheme

에 전송 받은 $\{y_1, z_1\}$을 보관한다. 클라이언트는 올바르게 검증이 완료되면 본인이 첫번째 일회용 패스워드로 사용한 x_1을 버리고 x_2와 x_3만 보관한다.

이를 일반화 하면, 클라이언트는 매번 인증 시도 시 자신의 식별자인 ID_C를 서버측 에게 전송하고 서버는 전송 받은 클라이언트의 식별자 ID_C에 해당하는 인덱스 $j(j=1,2,\cdots)$를 클라이언트에게 전송한다. 서버에게 인덱스 j를 전송 받은 클라이언트는 인덱스 j에 해당하는 일회용 패스워드 $\{x_j, y_j, z_j\}$를 생성하기 위해 본인이 가지고 있지 않은 임의의 난수 $seed$ 값인 x_{j+2}를 생성하고 일회용 패스워드 $\{x_j, y_j, z_j\}$를 계산한다. 특히 z_j를 계산시에 매개변수 두 개를 concatenation 하여 암호해시함수 $h(.)$를 적용하여 계산한다. 클라이언트는 계산한 일회용 패스워드 $\{x_j, y_j, z_j\}$를 서버측 에게 전송하고 서버는 전송 받은 클라이언트의 일회용 패스워드 $\{x_j, y_j, z_j\}$와 본인이 보관하고 있던 $\{y_{j-1}, z_{j-1}\}$를 이용하여 클라이언트를 검증한다. 먼저 서버는 $h(x_j)=y_{j-1}$을 일치하는지 확인하고, 일치하다면 $h(y_j, z_j)=z_{j-1}$을 계산하고 클라이언트 검증을 수행한다. 검증이 올바르게 완료되면 그 결과를 클라이언트에게 통보하고, 서버는 본인이 보관하고 있던 y_{j-1}, z_{j-1} 대신에 클라이언트에게 전송 받은 y_j, z_j를 다음 검증을 위해 보관하고 클라이언트의 IDc에 해당하는 인덱스 j의 값을 하나 증가시킨다. 검증결과가 올바르지 않다면 세션을 종료시킨다. 클라이언트는 서버에게 올바른 검증결과를 통보받으면, 본인이 생성하고 보관하고 있는 임의의 난수 $seed$ 값인 x_j, x_{j+1}, x_{j+2} 중 x_j를 제거하고 다음 검증을 위해 x_{j+1}과 x_{j+2}만 보관한다.

3.3 세션 Anchor 생성

세션 $Anchor$ $\{x_1, x_2, \cdots, x_j, \cdots\}$는 해시체인 구성에 가장 기본이 되는 클라이언트가 생성하는 값들이다. 매 세션마다 이 값들이 상호 독립적으로 생성되어야 하는 의미는 클라이언트 이외의 그 누구도 $\{x_1, x_2, \cdots, x_j\}$이 공개되었을 때, 그 다음 값 x_{j+1}을 예측할 수 없어야 한다는 것이다. 따라서, 이러한 제약조건을 만족하면서 클라이언트에 의한 세션 $Anchor$ 생성 및 유지관리를 용이하게 하기 위한 방안은 다음과 같다. $x_j = h(K, seed, j)$. K와 $seed$는 클라이언트만이 알고 있는 비밀키 및 초기값이다. 따라서, $x_{564} = h(K, seed, 564)$ 이 된다.

IV. 안전성 및 효율성 비교 분석

4.1 안전성 분석

제안된 일회용 패스워드의 안전성은 암호해시함수의 안전성, 즉 역상 저항성 (pre-image resistance)에 기반을 두고 있다. j번째 일회용 패스워드 $\{x_j, y_j, z_j\}$가 공격자에게 노출된 이후에 서버에 의해 받아 들여질 $j+1$번째 일회용 패스워드 $\{x_{j+1}, y_{j+1}, z_{j+1}\}$를 도출해 내기 위해서는 다음의 조건이 만족되어야 한다. $\{y_j=h(x_{j+1})$ and $z_j=h(x_{j+1}, y_{j+1})\}$. 즉, 서버측에 저장된 j번째 일회용 패스워드

중에서 $\{y_j, z_j\}$ 만이 $j+1$번째 일회용 패스워드의 유효성 검증에 사용이 된다. 위의 조건을 만족하는 $\{x_{j+1}, y_{j+1}\}$을 찾는다는 것은 결국 해당 암호해시함수의 역상 저항성을 깬다는 것인데, 이는 불가능하게 된다.

또한, $j+1$번째 일회용 패스워드 중에서 $\{z_{j+1}\}$는 $j+2$번째 일회용 패스워드에 대한 일종의 commitment 역할을 담당하기 때문에 $j+1$번째 일회용 패스워드의 유효성 검증에는 사용 되지 않는다. 따라서, 외부 공격자가 $j+2$번째 일회용 패스워드를 불법적으로 생성하기 위해 전송 중인 $\{z_{j+1}\}$을 다음과 같이 조작한다고 가정해 보자. 즉, 임의의 x_{j+2}' 와 y_{j+2}'를 생성해서 $z_{j+1}' = h(x_{j+2}', y_{j+2}')$을 도출한 이후에 이를 $\{z_{j+1}'\}$과 교체한다. 따라서, 서버측에 저장되는 내용은 $\{x_{j+1}, y_{j+1}, z_{j+1}'\}$이다. 공격자가 $j+2$번째 일회용 패스워드로 $\{x_{j+2}', y_{j+2}', z_{j+2}'\}$을 서버측으로 전송할 경우 서버는 다음의 계산을 수행한다. $\{y_{j+1} = h(x_{j+2}')$ and $z_{j+1}' = h(x_{j+2}', y_{j+2}')\}$. 이 경우에 둘째 식은 만족하지만 첫째 식은 만족하지 않는다. 그 이유는 (x_{j+2}, y_{j+1})에 대해서 $y_{j+1} = h(x_{j+2})$이 성립하기 때문에 $x_{j+2}' \neq x_{j+2}$일 경우에는 $y_{j+1} \neq h(x_{j+2}')$이 된다. 따라서, $\{y_{j+1} = h(x_{j+2}')\}$을 만족하는 x_{j+2}'을 찾는다는 것은 결국 역상 저항성을 깨야 한다는 의미이기에 이 역시 불가능하게 된다.

4.2 인증정보 불일치에 대한 동기화 방안

클라이언트와 서버 간에 다수의 인증세션이 진행되는 가운데 서버측이 유지 관리하는 클라이언트 인증정보와 클라이언트가 관리하는 인증 정보간에 불일치가 발생할 수가 있다. 이러한 문제점은 *Lamport* 일회용 패스워드[1]에서도 지적되었다. 서버측의 클라이언트 인증정보가 $\{ID_c, (y_m, z_m), m+1\}$라고 가정할 경우, 클라이언트가 관리하는 인증정보가 $\{(x_{m+l}, x_{m+l+1}), m+l\}$, $l \geq 2$ 일 경우이다. 클라이언트가 새로운 인증세션을 요청하면 서버는 $j = m+1$을 전송하면 클라이언트는 서버와 $l-1$ 만큼의 인증세션 오차가 발생함을 인지하게 된다. 이 경우에는 $A = \{x_{m+1}, y_{m+1}, z_{m+1}\}$와 함께 이를 검증할 수 있는 $B = (x_{m+l-1}, x_{m+l-2}, \cdots, x_{m+1})$를 서버에게 전송한다. 서버측이 $(x_{m+l}, x_{m+l-1}, \cdots, x_{m+1})$을 획득한다는 의미는 서버에게 결국 $\{x_{m+1}, y_{m+1}, z_{m+1}\}$, $\{x_{m+2}, y_{m+2}, z_{m+2}\}$, $\cdots$, $\{x_{m+l-1}, y_{m+l-1}, z_{m+l-1}\}$을 보낸 것과 동일한 효과를 가지기 때문에 클라이언트와 서버 간에 동기화가 이루어지게 되어 A를 정상적으로 처리하게 된다.

4.3 비교 분석

Fig. 2는 기존 방식들의 특징과 매 인증 시 클라이언트와 서버측의 계산량을 비교한 도표이다. 그리고 Fig. 3과 Fig. 4는 각 방식의 인증세션에서 클라이언트측과 서버측의 누적 계산량(암호해시함수 사용 횟수)을 보여주는 그래프이다.

	Fixed/variable password	Sharing secret information	Synchronization	Re-registration	Client calculation in one-time authentication	Server calculation in one-time authentication
Lamport[1]	variable	unnecessary	possible	necessary	n-j hash	1 hash
Bicakci[5]	variable	unnecessary	possible	unnecessary	1 RSA dec	1 RSA enc
Zhang[6]	variable	unnecessary	impossible	unnecessary	(n-j) + 1 hash	2 + 2 hash
Hsieh[4]	variable	necessary	possible	unnecessary	variable	variable
Kim[9]	variable	necessary	possible	unnecessary	1 hash	1 hash
Bittle[11]	variable	necessary	possible	unnecessary	2 hash	2 hash
Proposed	variable	unnecessary	possible	unnecessary	2 hash	2 hash

Fig. 2. Performance Comparison Analysis

기존 연구 분석에서 다루었던 여러 가지 방식과 본 논문에서 제안하는 방식에서의 클라이언트측과 서버측 계산량을 한 눈에 볼 수 있다. 클라이언트가 매 인증 시도 시 클라이언트와 서버측이 일회용 패스워드 생성 및 일회용 패스워드 검증을 하는 계산량을 최대 인증횟수 100회까지 누적 계산량을 측정한 것이다. 먼저 클라이언트측 계산량에서 가장 높게 측정된 *Bicakci*가 제안한 방식[5]의 일회용 패스워드는 다른 방식들과는 달리 RSA알고리즘을 사용하기 때문에 가장 높게 측정 되었다. 다른 방식들과 달리 이 방식만 RSA알고리즘을 사용하므로 클라이언트가 일회용 패스워드를 생성하는데 사용하는 RSA Encrypt, 서버측에서 클라이언트에게 전달받은 일회용 패스워드를 검증하는데 사용하는 RSA Decrypt를 암호해시함수 중 하나인 SHA 256와 속도 비교를 하여 누적 계산량을 측정 하였다. i5-6300U 2.4GHz의 CPU를 포함한 데스크탑에서 OpenSSL을 이용하여 암호해시함수 SHA256의 속도를 측정한 결과 약 5~6μs, RSA Encrypt의 속도를 측정한 결과

약 1625μs, Decrypt의 속도를 측정한 결과 약 80 μs가 측정되어 해당 방식에서 클라이언트가 일회용 패스워드를 생성하는 RSA Encrypt의 속도가 다른 방식에서 사용하는 암호해시함수 SHA256의 속도보다 약 270배 느리므로 1회 RSA Encrypt를 270번 암호해시함수를 수행하는 것으로 표현하였다. 그리고 서버측에서 클라이언트가 전송한 일회용 패스워드를 검증하는 RSA Decrypt는 암호해시함수 SHA256의 속도보다 약 13배 느리므로 1회 RSA Decrypt를 13번 암호해시함수를 수행하는 것으로 표현하였다.

*Bicakci*가 제안한 방식[5]은 매 인증세션에서 클라이언트가 생성한 임의의 *seed* 값으로 자신이 보유하고 있는 개인키를 이용해 서버가 전달한 해당 인덱스만큼 암호화를 하여 일회용 패스워드로 사용하는 것 인데, 여기서는 이 방식 그대로 표현하면 다른 방식들과 비교했을 때, 상대적으로 계산량이 너무 높아서 다른 방식들의 그래프가 보이지 않기에 가장 최근에 사용한 일회용 패스워드를 클라이언트가 보관하고 매 인증세션에서 클라이언트가 자신의 개인키로 한번 암호화 하여 일회용 패스워드로 사용하는 것으로 측정 하였다. 따라서 해당 그래프에 표현된 *Bicakci*가 제안한 방식의 클라이언트측의 누적 계산량은 실제 방식의 계산량보다 현저히 낮게 측정되었지만, 다른 방식의 클라이언트측 누적계산량과 비교했을 시, 여전히 상당히 높은 계산량을 보여준다.

Bicakci 방식 다음으로 계산량이 높게 측정된 방식은 *Hamdy* 가 제안한 방식[4]이다. *Hamdy* 가 제안한 방식은 Challenge-response 방식으로 서버가 생성한 임의의 인덱스 2개에 따라서 계산량이 변동되기에 현재 측정된 계산량보다 더 높게 측정될 수도 있고 낮게 측정 될 수도 있다. 클라이언트가 서버에게 전달받은 인덱스로 일회용 패스워드를 생성하기 때문에 계산량이 동일하게 높게 측정되었다. 여기서는 서버가 랜덤하게 생성하는 인덱스 2개의 최대치를 각 50, 총 암호해시함수 최대치를 100이라 가정하여 해당 그래프를 나타내었지만, 설계방식에 따라 해당 계산량보다 높게 측정되거나 낮게 측정될 수 있다. *Hamdy* 가 제안한 방식의 특징은 서버측이 생성한 인덱스 2개를 이용해 서버측과 클라이언트측이 동일하게 암호해시함수를 적용하므로 서버측과 클라이언트측 계산량이 동일하다.

*Lamport*가 제안한 방식[1]은 임의의 *seed* 값을 암호해시함수에 여러 번 적용하여 생성한 해시체인을 사용하기 때문에 인증횟수가 증가하면서 계산량이 점점 줄어들고 해시체인의 해시값들을 모두 소모하게 되면 재등록을 해야 한다. 따라서 해당 그래프에서 처음에는 계산량이 높게 측정되지만, 인증횟수가 누적될수록 일회용 패스워드를 생성하는 클라이언트의 계산량이 낮게 측정되는 것을 볼 수 있다. 서버측 계산량은 클라이언트측 계산과 다르게 단 한번의 암호해시함수를 적용하므로 낮은 계산량을 보여준다.

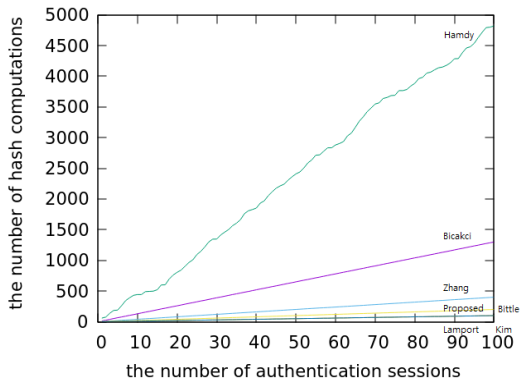

Fig. 3. Server's Calculation

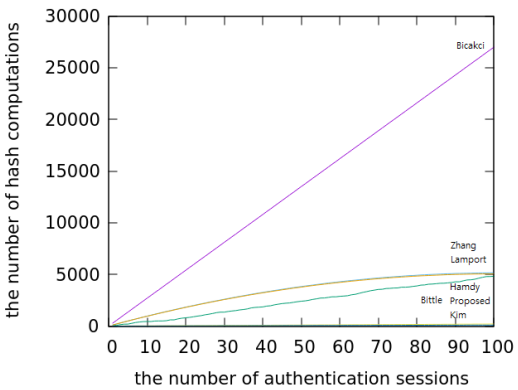

Fig. 4. Client's Calculation

*Zhang*이 제안한 방식[6]은 *Lamport* 가 제안한 방식보다 클라이언트측 계산량이 살짝 높은 이유는 클라이언트를 인증하는 방식은 동일하지만 재등록을 위한 준비를 하기 위해 계산량이 조금 더 높게 측정되었다고 생각하면 될 것 이다. *Lamport* 방식과 동일한 방식으로 클라이언트 인증이 이루어지므로 그래프의 형태는 동일한 것을 볼 수 있다. 클라이언트측이 전달한 일회용 패스워드를 검증하는 서버측 계산량도 동일하게 *Lamport* 방식보다 높다는 것을 볼

수 있다.

*Bittle*이 제안한 방식[11]은 매 인증세션에서 서버 및 클라이언트가 일회용 패스워드 생성 및 검증할 때 마다 각 2회의 암호해시함수 $h(.)$를 적용한다. 계산량만을 본다면 다른 기존 방식들과 비교했을 때 상당히 낮은 수치이다.

*Kim*이 제안한 방식[9]은 해당 그래프에서도 볼 수 있듯이, 다른 방식들과 비교했을 때 제일 낮은 클라이언트 및 서버측 계산량을 보여준다. 매 클라이언트 인증 시 클라이언트에서 일회용 패스워드를 생성하는데 사용하는 암호해시함수 한번, 클라이언트가 생성한 일회용 패스워드를 검증하는데 암호해시함수 한번을 적용하기 때문이다.

본 논문에서 제안한 방식은, 매 인증세션에서 클라이언트 인증 시 클라이언트가 일회용 패스워드를 생성하는데 총 2회의 암호해시함수를 사용하고 서버에서는 클라이언트가 생성한 일회용 패스워드를 검증하는데 총 2회의 암호해시함수를 사용하기 때문에 기존 해시체인을 사용하는 *Lamport* 방식과 다르게 인증 횟수에 따라 정비례 하는 그래프 형태를 보여준다.

결론적으로, 클라이언트가 일회용 패스워드를 생성하는데 사용하는 계산량과 서버측에서 클라이언트가 생성한 일회용 패스워드를 검증하는데 사용하는 계산량을 보면 본 논문에서 제안하는 방식은, 다른 기존 방식들과 비교했을 때 상당히 낮은 계산량으로 일회용 패스워드를 생성하고 검증하는 것을 볼 수 있다.

V. 결 론

본 논문에서는 기존 해시체인 기반의 일회용 패스워드가 지니는 문제점 그리고 이를 해결하기 위한 방안들을 다루었다. 해시체인의 해시값이 소진된 이후에 새로운 해시체인에 대한 루트 값을 재등록 하는 문제점 해결이 기존연구들의 주요 관심분야 였지만 이에 대한 기존연구들은 문제점 해결을 위해 또 다른 문제점을 야기시키고 있다. 본 연구에서는 암호해시함수만을 사용하면서 해시체인의 재등록이 필요 없는 일회용 패스워드 기법을 제안하였다. 또한 타 기존연구와의 기능별 그리고 효율성 비교를 통해 제안하는 기법이 매 인증마다 각 2회의 암호해시함수만을 사용하여 적은 계산량으로도 해시체인의 재등록이 필요 없는 일회용 패스워드 기법임을 확인할 수 있었다. 향후 이를 기반으로 다양한 응용분야에 적용하는 후속연구를 진행할 예정이다

References

[1] Leslie Lamport, "Password Authentication with Insecure Communication," Communications of the ACM, vol. 24, pp. 770-772, Nov. 1981.

[2] N. Haller Bellcore, "The S/KEY One-Time Password System," RFC 1760, Feb 1995.

[3] Hoyul Choi, Hyunsoo Kwon, and Junbeom Hur, "A Secure OTP Algorithm Using a Smartphone Application," Proceedings of the 7th International Conference on Ubiquitous and Future Networks, pp. 476-481, Aug. 2015.

[4] Mohamed Hamdy Eldefrawy, Khaled Alghathbar, and Muhammad Khurram Khan, "OTP-Based Two-Factor Authentication Using Mobile Phones," Proceedings of the 8th International Conference on Information Technology, pp. 327-331, July. 2011

[5] Kemal Bicakci, and Nazife Baykal, "Infinith Length Hash Chains and Their Applications," Proceedings of the IEEE International Workshops on Enabling Technologies, pp. 57-61, Nov. 2002.

[6] Haojun Zhang, Xiaoxue Li, and Rui Ren, "A Novel Self-Renewal Hash Chain and Its Implementation," Proceedings of the IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, pp. 144-149, Jan. 2009.

[7] Vipul Goyal, "How to Re-initialize a Hash Chain," Available from: 〈http://eprint.iacr.org/2004/097.pdf〉

[8] N. Haller, "A one-time password system," RFC 1938, June. 2000.

[9] Korea Universitv Industry-Academia Collaboration Foundation, "Method and App Aratus for infinite authentication using a hash chain," KR Application

Number. 10-2009-0016363 Feb. 2009.
〔10〕 King Saudi University, "ONE-TIME Password Authentication with infinite nested hash chains," International Application Number. PCT/US2010/057125, Nov. 2010.
〔11〕 S Bittle, "Efficient Construction of Infinite Length Hash Chains with Perfect Forward Secrecy Using Two Independent Hash Functions," Proceedings of the 11th International Conference on Security and Cryptography, pp. 1-8, Aug. 2014.

〈저자소개〉

신 동 진 (Dong-jin Shin) 학생회원
2017년 2월: 단국대학교 컴퓨터과학과 졸업
2017년 3월~현재: 단국대학교 소프트웨어 보안 석사과정
〈관심분야〉 정보보호

박 창 섭 (Chang-seop Park) 종신회원
1983년 2월: 연세대학교 경제학과 졸업
1987년 2월: Lehigh University 컴퓨터과학과 석사
1990년 2월: Lehigh University 컴퓨터과학과 박사
1990년 3월~현재: 단국대학교 소프트웨어학과 교수
〈관심분야〉 정보보호, 네트워크 보안, 무선인터넷 및 모바일 컴퓨팅 보안