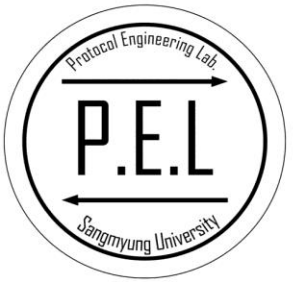


NETWORK SECURITY ESSENTIALS

보충 : 해시 체인(Hash Chain)

Boo-Hyung Lee
(boohyung@pel.smuc.ac.kr)
Protocol **E**ngineering **L**ab., **Sangmyung** University

2015. 01. 30

Content

- 해시 체인(Hash Chain)
- 해시 체인의 응용 : 일회용 패스워드(OTP)

해시 체인(Hash Chain)

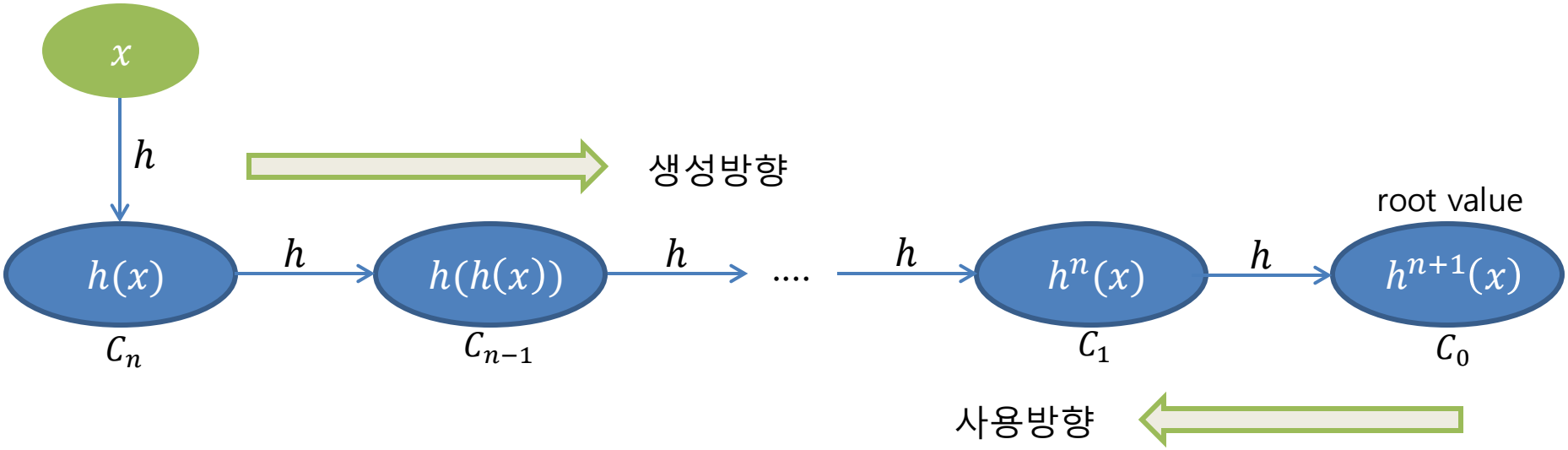
- **정의**

  - 수학자 Leslie Lamport가 처음 개발한 기법
  - 클라이언트가 정한 임의의 값(seed)을 이용하여 연속적으로 해쉬 값을 계산하는 방식을 사용

해시 체인(Hash Chain)

- **계산 과정(길이가 n인 해쉬 체인)**

  - h : Hash Function, x : seed
  - 루트 값(root value) : x로부터의 해시 연산을 통해 얻는 마지막 값; 해시 체인의 루트 값 $C_0 = h^{n+1}(x)$



해시 체인(Hash Chain)

- **특징**

  - 실제 사용할 때는 생성 순서와 반대로 $C_1, C_2 \ldots C_n$ 순으로 사용
  - 역연산 불가 : $h^n(x)$를 알고 있어도 $h^{n-1}(x)$는 계산할 수 없음; 만약 공격자가 패스워드 전송 과정을 볼 수 있어도 다음 패스워드 예측이 불가능함

$$\boldsymbol{h^n(x) = h\left(h^{n-1}(x)\right)}$$

c.f) 해쉬함수의 일방향성 : y = f(x)에서, y를 알 때 x를 계산할 수 없음

응용 : 일회용 비밀번호(OTP)

- **정의**

  - 매번 새로운 패스워드를 사용한 인증 방식 : 한 번 인증에 사용한 패스워드는 다시 사용하지 않음

- **요구사항**

  - R1. 이전에 사용된 패스워드로부터 현재 사용할 패스워드를 제3자가 계산하는 것이 계산적으로 어려워야 한다.
  - R2. 제시된 OTP 값은 사용자가 쉽게 읽을 수 있어야 하며, 사용자가 화면에 쉽게 입력할 수 있어야 한다.
  - R3. 하드웨어적으로 구현이 경제적이어야 한다.

응용 : 일회용 비밀번호(OTP)

- **과정(S/KEY방식; RFC 2289에 정의)**

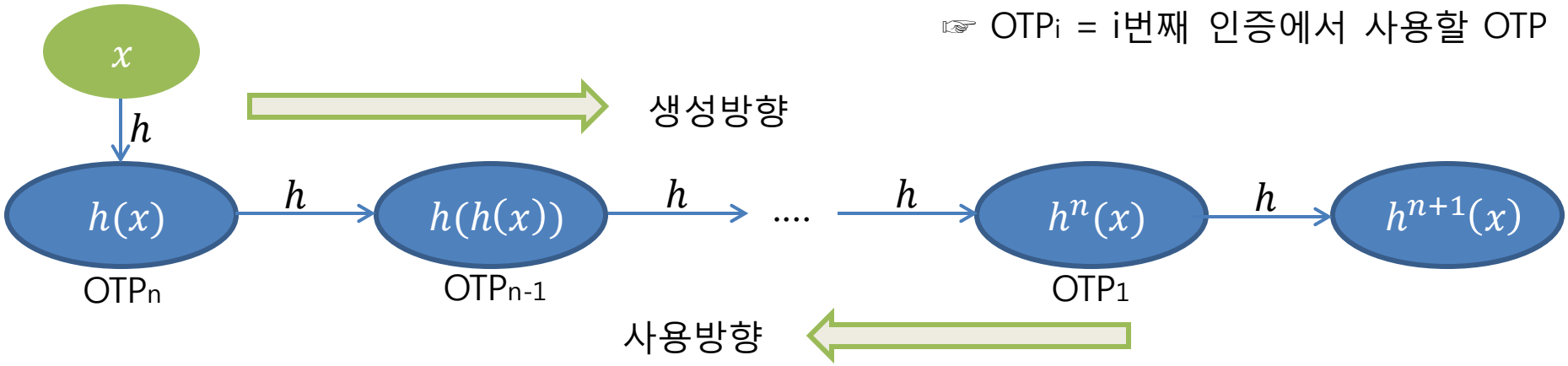
- OTP 생성(n은 클라이언트와 서버가 미리 합의; n은 인증 가능한 최대 횟수)

1) 클라이언트가 비밀 키(x)를 임의로 생성하고, x와 n을 서버의 공개키로 암호화하고 자신의 식별자와 함께 서버에게 전달; 클라이언트는 x를 이용하여 해시 체인 방식으로 n개의 OTP 생성, 저장

2) 서버는 클라이언트로 부터 비밀 키를 첫 값으로 사용하여, 해시 체인 방식으로 이전 결과 값에 대한 해시 값을 구하는 연산을 n번 수행

3) 클라이언트의 식별자와 n, $h^{n+1}(x)$을 저장

☞ OTP_i = i번째 인증에서 사용할 OTP



응용 : 일회용 비밀번호(OTP)

- **과정(S/KEY방식)**

- 인증(i번째로 서버에 인증을 요구할 때)

1) 클라이언트에서 정한 비밀 키에 해시 함수를 n-i번 중첩 적용하여 서버로 전송

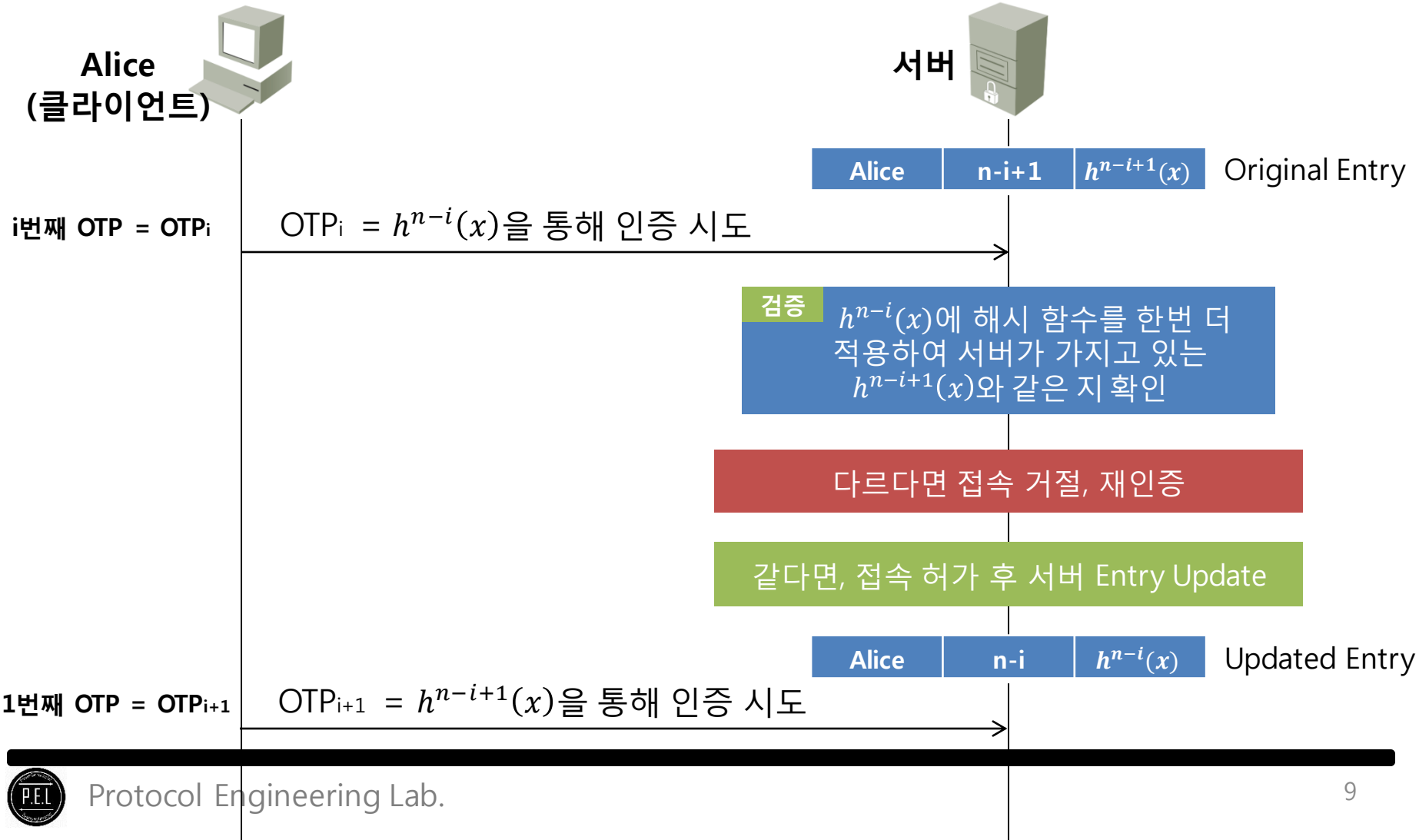
2) 서버에서는 클라이언트로부터 받은 값에 해시 함수를 적용하여, 그 결과가 서버에 저장된 값과 일치하는지 검사

3) 일치하면 인증 성공; 인증에 성공하면 서버에 저장된 n-i+1값을 n-i로 1 감소 시키고 $h^{n-i+1}(x)$를 $h^{n-i}(x)$로 바꿈

4) 불일치하면 인증 실패; 서버 저장 값 변경 없음

응용 : 일회용 비밀번호(OTP)

- 과정(S/KEY방식) : 그림(x와 n은 클라이언트와 서버가 미리 교환)



응용 : 일회용 비밀번호(OTP)

- **특징**

- 기존의 사용자 아이디/암호를 사용하는 방식보다 안전
- Two-Factor 인증 : Know(OTP) + Have(OTP Device)
- n이 유한적이므로 인증 횟수도 유한적 → 재설정(초기화)의 필요성
- 해시 체인의 특성으로 공격자가 만약 i번째 패스워드를 알고 있다고 해도, i+1번째 패스워드를 알 수 없음(패킷 스니핑을 통한 재사용 공격 불가능)

ex. n = 4, x = 사용자가 정한 비밀 키, p(i)를 i번째 패스워드라고 가정하면,

p(1) = $h(h^3(x))$, p(2) = $h\left(h^2(x)\right)$