새로운 패스워드 강화 기법을 이용한 키 로밍 프로토콜

(A Key Roaming Protocol with a New Password Hardening Scheme)

정 현 철 † 김 승 호 ††
(Hyon Cheol Chung) (Sung Ho Kim)

요 약 본 논문은 Ford와 Kaliski가 제안한 로밍 프로토콜에 RSA 알고리즘을 이용하는 프로토콜을 제안하고 제안된 프로토콜을 이용하여 중요 정보를 위탁 시킬 수 있는 크레덴셜 서버와 로밍 서버로 구성된 시스템의 안전성과 기존의 다중 로밍 서버를 이용한 패스워드 기반의 로밍 프로토콜과의 성능을 분석하여 제안된 프로토콜의 우수성을 검증한다. 그리고 기존 프로토콜이 채택하고 있는 SSL과 같은 안전한 보안 채널의 사용을 최소화 할 수 있는 방법을 제안한다.

키워드 : 공개키 암호, 키로밍, 로밍서버, 크레덴셜서버, 패스워드강화, RSA

Abstract In this paper, we present more efficient protocol than that of Ford and Kaliski. We use RSA in the roaming protocol and it plays decisive role to reduce the total time cost. We show that our protocol is safe from various attacks. We present the performance of our protocol and verify that. This protocol needs fewer secure channel like SSL than other protocols.

Key words : Public Key Crypto System, Key Roaming, Roaming Server, Credential Server, Password Hardening, RSA

1. 서 론

네트워크 환경의 발전과 암호기술의 발전은 수많은 사용자가 인터넷 뱅킹 또는 보안 메일 등과 같은 암호 서비스를 언제 어디서나 사용할 수 있도록 해주었다. 하지만 이런 암호 서비스는 사용자가 가진 개인키를 이용해서 사용자를 인증하기 때문에 사용자는 자신의 개인키를 하드 디스크나 플로피 디스켓 또는 스마트카드 등과 같은 저장 매체에 저장해 두고 암호 서비스를 사용해야 한다는 단점이 있다. 그러나 하드 디스크와 같은 저장 매체는 사용자의 이동성을 제약하고, 플로피 디스켓의 경우 분실의 위험과 파손의 위험을 가진다. 스마트카드를 이용한 방법은 충분한 안전성을 제공해 주기는 하지만 추가적인 비용의 부담과 스마트카드 리더가 없는 경우 사용이 불가능하다는 단점을 가지고 있다. 따라서 사용자는 추가적인 장치 없이 자신의 개인키를 접근할 수 있고, 자신의 이동성까지 보장되는 방법이 필요하게 된다.

패스워드 기반 로밍 시스템은 위와 같은 사용자의 요구를 충족할 수 있는 방법이다. 이 방법은 중앙의 서버에 사용자의 개인키를 저장해 두고, 사용자는 기억이 가능한 패스워드를 이용해서 중앙의 서버로부터 인증을 받아 자신의 개인키를 언제 어디서나 가져올 수 있도록 하는 시스템이다. 가장 단순한 패스워드 기반 로밍 시스템은 사용자의 패스워드에 대한 해쉬 값으로 사용자를 인증하는 방법이다. 즉, 사용자는 자신의 패스워드로 중앙의 서버에 인증을 하고 자신의 비밀 데이타를 가져오는 방식이다. 이런 방식은 현재 유닉스의 로그인과 같이 대부분의 시스템에서 사용하는 방식이다. 하지만 이런 방식의 가장 큰 문제점은 사전검색 공격과 같은 패스워드 추측 공격에 시스템이 취약하다는 것과 서버에 공격자가 침입하는 경우 사용자 비밀 데이타가 모두 노출된

· 이 논문은 2002년도 경북대학교 특성화사업팀(KNURT) 연구비에 의하여 연구되었음
† 비 회 원 : 알파로직스(주) 대표이사
hcchung@softforum.com
†† 종신회원 : 경북대학교 컴퓨터공학과 교수
shkim@knu.ac.kr
논문접수 : 2003년 1월 8일
심사완료 : 2003년 3월 3일

다는 것이다.

Ford와 Kaliski는 위와 같은 문제점을 해결하기 위해서 다중 로밍 서버를 이용하는 패스워드 기반의 로밍 프로토콜을 제안하였다[1]. 이 프로토콜은 여러 개의 로밍 서버를 이용해서 사용자의 약한 패스워드를 강한 패스워드로 변환하는 패스워드 강화 프로토콜을 이용하는 방법이다. 이 방법은 패스워드 추측 공격을 무력화 시킬 수 있고, 로밍 서버 자체도 사용자의 패스워드에 대한 어떠한 정보도 알 수 없기 때문에 로밍 서버가 공격을 당해도 사용자의 데이타가 노출될 위험이 없다는 장점을 가진다[2]. 하지만 Ford와 Kaliski가 제안한 방법은 SSL과 같은 안전한 보안 채널을 필요로 하는 단점이 있다. 최근 SSL과 같은 안전한 보안 채널 없이도 안전한 프로토콜이 제안되었다[3,4].

이 논문에서는 Ford와 Kaliski가 제안한 다중 서버를 이용한 패스워드 기반의 로밍 프로토콜에 RSA 알고리즘을 이용하여 RSA 기반의 패스워드 강화 프로토콜을 제안한다. 그리고 이 방법의 안전성을 검증하도록 한다. 또한 SSL과 같은 안전한 보안 채널의 사용을 최소화할 수 있는 방법을 기술한다.

본 논문의 구성은 다음과 같다. 2절에서는 다중 서버를 이용한 로밍 시스템 모델에 대해서 기술한다. 3절은 RSA 기반의 로밍 프로토콜을 기술하고 4절과 5절에서는 제안된 프로토콜의 안정성을 검증하고 성능을 분석한다. 마지막으로 6절에서 결론을 맺는다.

2. 다중 서버 로밍 시스템

이 절에서는 다중 로밍 서버를 이용한 패스워드 기반의 로밍 시스템 모델을 정의하고 이 시스템이 만족해야 하는 안전성 조건이 무엇인지 알아 보도록 한다. 또한, 기존에 발표된 다중 서버 로밍 시스템을 소개하고 각 시스템에서 사용된 프로토콜을 정리해 본다.

2.1 시스템 구성요소 및 정의

다중 로밍 서버를 이용한 패스워드 기반의 로밍 시스템은 다수의 로밍 서버와 크레덴셜 서버를 이용해서 서로 다른 위치에서 서버에 접근하는 로밍 사용자의 비밀 데이타 등을 안전하게 보관하고 사용자가 필요로 하는 경우 비밀 데이타를 안전하게 제공해주는 시스템이다[1]. 이 시스템은 다수의 로밍 서버들과 크레덴셜 서버 그리고 사용자로 구성된다. 구성 요소에 대한 더욱 자세한 내용은 다음과 같다.

2.1.1 로밍 서버

로밍 서버는 사용자의 약한 패스워드를 강한 패스워드로 변환시켜주는 패스워드 강화 프로토콜을 수행하는 기능과 사용자 인증을 통해서 저장된 사용자의 암호키를 전달하는 역할을 수행한다.

2.1.2 크레덴셜 서버

크레덴셜 서버는 사용자의 비밀 데이타를 저장하는 기능을 수행하는 서버이다. 이 서버에 저장되는 비밀 데이타는 안전하게 암호화되거나 또는 공개되어도 되는 정보이기 때문에 최소한의 안전성만이 요구된다.

2.1.3 사용자

로밍 시스템을 사용하는 사용자로 다수의 로밍 서버와 패스워드 강화 프로토콜을 수행하고, 크레덴셜 서버에 자신의 비밀 데이타를 저장하거나 가져오는 역할을 수행한다. 이렇게 가져온 자신의 비밀 데이타는 인터넷 뱅킹과 같은 서비스에 바로 사용될 수 있다.

2.2 시스템 공격 유형 및 안전성 요구조건

다중 로밍 시스템이 안전성을 갖기 위해서는 패스워드 추측 공격으로부터 안전해야 한다. 그리고 소수의 다중 로밍 서버가 공격자에 침입을 당하는 경우에도 사용자의 패스워드를 유추하거나 사용자의 개인키를 복구하는 것이 어려워야 한다. 안전한 다중 로밍 시스템은 다음에 기술한 공격방법에 대해 안전한 시스템이다. 다중 로밍 시스템에 대한 공격 방법은 다음과 같이 분류된다.

2.2.1 오프라인 공격

오프라인 공격은 공격자가 로밍 프로토콜에 참여하지 않고 이미 알려진 정보만을 이용해서 사용자의 패스워드를 유추하거나 또는 사용자의 개인키를 복호화하는 공격 방법이다.

2.2.2 온라인 공격

온라인 공격은 공격자가 로밍 시스템에 온라인으로 접근해서 사용자의 패스워드 또는 개인키를 유추하는 방법으로 오프라인 공격까지 포함하는 공격 방법이다. 이 방법은 공격자의 위치에 따라서 사용자를 위장한 공격과 로밍 서버를 위장한 공격 두 가지로 구분된다. 각 공격에 대한 더욱 자세한 내용은 다음과 같다.

2.2.2.1 사용자 위장 공격

사용자를 위장한 공격 방법은 공격자가 사용자의 위치에서 로밍 서버, 크레덴셜 서버와 온라인 통신을 통해서 사용자의 패스워드를 유추하거나 사용자의 개인키를 복구하는 공격 방법이다. 즉, 공격자는 일반 사용자와 동일하게 로밍 서버, 크레덴셜 서버 간의 프로토콜에 참여할 수 있다.

2.2.2.2 로밍 서버 위장 공격

로밍 서버 위장 공격 방법은 공격자가 소수의 로밍 서버에 침투해서 사용자의 패스워드 또는 개인키를 유추하는 방법이다. 즉, 공격자는 소수의 로밍 서버에 저장된

모든 데이타와 사용자간의 프로토콜에 참여할 수 있다.

따라서 다중 로밍 서버를 이용한 패스워드 기반의 로밍 시스템이 안전하기 위해서는 사용자를 위장한 온라인 공격과 로밍 서버를 위장한 온라인 공격에 안전해야 한다.

2.3 Ford와 Kaliski 시스템

Ford와 Kaliski가 제안한 시스템은 다중 로밍 서버를 이용하는 패스워드 기반의 로밍 프로토콜에 기반한 시스템이다. 이 로밍 프로토콜은 여러 개의 로밍 서버를 이용해서 사용자의 약한 패스워드를 강한 패스워드로 변환하는 패스워드 강화 프로토콜을 사용하는 방법이다. 패스워드 강화 프로토콜을 사용해서 로밍 프로토콜을 구성하는 것은 대부분의 시스템이 유사하므로 여기서는 이들이 제안한 패스워드 강화 프로토콜을 기술하도록 한다.

먼저 프로토콜에서 수행하기 전에 로밍서버는 임의의 소수 q에 대한 2q+1 형태의 소수 p를 선택한다. 그리고 각 사용자를 위해서 1과 q-1 사이 값의 비밀값 d를 선택한다[1,3].

① 사용자는 다음과 같이 r 값을 계산해서 로밍서버에게 전달한다. 이때 PWD는 사용자의 패스워드, k 값은 랜덤값이다.

r ← f(PWD)^k mod p;

② 로밍서버는 사용자가 전달한 r 값을 이용해서 s값을 계산해서 사용자에게 전달한다.

s ← r^d mod p;

③ 사용자는 로밍서버로부터 받은 s 값을 이용해서 강한 패스워드 R 값을 다음과 같이 생성한다.

R ← s^{1/k} mod p;

Ford와 Kalisk가 제안한 이 프로토콜은 사용자와 로밍서버 간에 주고받는 메시지가 변조되는 것을 막기 위해서 SSL과 같은 채널로 보호되어야 한다.

2.4 Jablon 시스템

Jablon이 제안한 시스템은 Ford와 Kaliski가 제안한 시스템을 개선한 방법이다. 이 시스템은 기존 Ford와 Kaliski 시스템과 달리 SSL과 같은 보안 채널이 필요하지 않다. 또한 기존 시스템에 비해서 클라이언트 측의 수행속도를 개선한 방법이다. 이 시스템이 사용하는 패스워드 강화 프로토콜은 다음과 같이 기술된다.

먼저 프로토콜을 수행하기 전에 로밍서버는 임의의 소수 q에 대한 2rq+1 형태의 소수 p를 선택한다. 그리고 각 사용자를 위해서 1과 q-1 사이 값의 비밀값 d를 선택한다. 사용자는 자신의 패스워드를 이용해서 gp = h(PWD)^{2r} mod p 값을 설정한다. 그런뒤 사용자는 전자서명에 사용될 서명키와 공개키 {U, V}를 생성한다. 또한 마스터키 Km을 이용해서 사용자는 자신의 전자서명 서명키를 암호화한 Uk 값을 생성하고, 마스터키에 대한 증명 값으로 사용될 ProofKm 값을 생성해서 로밍서버에 등록시켜 둔다[3].

① 사용자는 다음과 같이 r 값을 계산해서 로밍서버에게 전달한다. 이때 k 값은 랜덤값이다.

r ← gp^k mod p;

② 로밍서버는 사용자가 전달한 r 값을 이용해서 s 값을 계산한다. 그리고 저장되었던 Uk, ProofKm 값과 같이 s 값을 사용자에게 전달한다.

s ← r^d mod p;

③ 사용자는 로밍서버로부터 받은 {s, Uk, ProofKm} 값을 이용해서 강한 패스워드 R 값을 다음과 같이 생성한다.

R ← s^{1/k} mod p;

이들 강한 패스워드로부터 마스터 키를 유도해서 ProofKm 값을 이용해서 이 값의 유효성을 검증한다. 만일 마스터 키 값이 유효하지 않은 경우 프로토콜을 중단한다. 그런뒤 사용자는 s 값에 대한 전자서명을 생성해서 로밍서버에게 전달한다.

④ 로밍서버는 사용자가 전달한 전자서명을 검증한다. 만일 전자서명이 유효하지 않은 경우 공격자의 공격을 막기 위해서 이 사용자에 대한 접근을 차단시켜둔다.

Jablon이 제안한 프로토콜은 SSL과 같은 채널이 필요하지 않지만 사용자가 부가적으로 전자서명을 수행해야 하는 단점을 가진다.

3. RSA 기반의 다중 서버 로밍 프로토콜

이 절에서는 RSA 기반의 로밍 프로토콜을 제안한다. 이 프로토콜은 Ford와 Kaliski가 제안한 시스템에 RSA 알고리즘을 적용한 프로토콜이다. 이 프로토콜의 장점은 기존 Ford와 Kaliski가 제안한 시스템과 달리 패스워드 강화 과정에 RSA 연산을 사용해 연산량을 줄이고 속도를 개선시켰으며 별도로 SSL과 같은 안전한 보안 채널의 사용을 최소화 할 수 있어 로밍 프로토콜에서 채널 암호화 및 보안 모듈에 대한 오버헤드를 줄였다는 것이다.

3.1 기술 용어

프로토콜의 기술에 사용되는 기술적인 용어는 다음과 같다.

- *Client*: 사용자(*Client*)
- RS_i: 다수의 로밍 서버 중에서 i 번째 로밍 서버

(Roaming Server)
- CS: 크레덴셜 서버(Credential Server)
- PHP: 패스워드 강화 프로토콜(Password Hardening Protocol)
- KDF: 키 유도 함수(Key Derivation Function)
- SSS: 비밀 공유 기법(Secret Sharing Scheme)
- XOR: XOR 암호 연산
- MGF: 마스크 생성 함수(Mask Generation Function)
- (ID, PWD): 사용자 아이디와 패스워드
- (PRI, CERT): 사용자의 개인키와 인증서
- (PHK, PHB, PHC): 패스워드 강화 프로토콜에 사용되는 사용자의 PHP 개인키와 PHP 공개키, 그리고 사용자의 PHP 인증서
- (ENC, DEC): 대칭키 암호화와 복호화 함수

3.2 등록 프로토콜

등록 프로토콜이란 사용자 등록 및 사용자의 개인키와 인증서 등의 정보를 등록하는 프로토콜이다. 이 프로토콜은 사용자가 로밍 서버를 통해서 강한 패스워드를 얻어오는 과정, 사용자가 자신의 아이디와 패스워드 관련 값을 로밍 서버에 저장하는 과정, 그리고 사용자가 자신의 비밀 데이타를 암호화해서 크레덴셜 서버에 저장해 두는 세 가지 과정으로 나누어진다. 각 과정의 자세한 내용은 다음과 같다.

3.2.1 강한 패스워드 생성 과정

① 사용자 : 먼저 사용자는 등록 과정을 수행하기 전에 자신이 사용할 비밀 데이타를 가지고 있다. 즉, 인증센터를 통해서 자신의 개인키와 인증서를 발급 받은 후에 로밍 서비스에 사용될 아이디와 패스워드를 생성한다.

(PRI, CERT) ← CA;

사용자가 임의대로 (ID, PWD)를 생성;

② 사용자 ↔ 로밍서버 : 사용자는 패스워드 강화 프로토콜을 수행하기 위해서 자신의 ID를 각각의 로밍 서버에 등록한다.

RS_i ⇐ Client: ("register", ID);

③ 로밍서버 : 각각의 로밍 서버는 사용자를 위해서 사용자가 패스워드 강화 프로토콜에서 사용하게 될 사용자의 PHP 개인키와 사용자의 PHP 인증서를 생성해서 데이타베이스에 안전하게 저장한다. 이때 PHP 인증서는 로밍 서버의 개인키로 서명되어서 패스워드 강화 프로토콜 수행 시 사용자가 자신의 PHP 인증서가 유효한지 검증할 수 있어야 한다.

사용자의 ID에 대해 개인키 및 인증서,

(PHK_ID, PHB_ID)를 생성;

PHC_ID ← GenPHC(ID, PHB_ID);

④ 사용자 ↔ 로밍서버 : 사용자는 각 로밍 서버들과 패스워드 강화 프로토콜을 수행해서 그 결과로 각 로밍 서버로부터 강한 패스워드 값들을 얻어온다. 이 전 과정에서 생성한 PHK, PHB, PHC는 아래 PHP(RSi, ID, PWD)에서 사용되는데 이 과정에 대한 설명은 3.4절의 패스워드 강화 프로토콜에서 설명된다.

for i=1 to i=n do

R_i ← PHP(RS_i, ID, PWD);

3.2.2 암호키 생성 및 로밍 서버에 저장 과정

① 사용자 : 사용자는 키 생성 함수를 사용해서 강한 패스워드 값들로부터 강한 키 값들을 생성한다. 이때 생성된 강한 키 값들은 각각의 로밍 서버와 사용자간의 인증, 크레덴셜 서버와의 사용자의 인증 그리고 사용자의 비밀 데이타 암호에 사용된다. 사용자는 자신의 개인키 암호화에 사용될 암호키를 생성하고 이 값을 비밀 분산 기법을 사용해서 각각의 암호키 조각으로 분산한다.

for i=1 to i=n+2 do

K_i ← KDF(R_1,R_2, ⋯, R_n, i);

S ← Random;

S_1,S_2,⋯,S_n ← SSS(S);

② 사용자 ↔ 로밍서버 : 사용자는 SSL과 같은 안전한 보안 채널을 통해서 자신의 아이디와 사용자 인증 K_i 값 그리고 암호키 조각 S_i 값을 각각의 로밍 서버에게 전달한다. 이때 로밍 서버는 사용자가 안전하게 전달해준 사용자 아이디, 인증 값, 그리고 암호키 조각 값을 자신의 데이타베이스에 안전하게 저장해둔다.

for i=1 to i=n do

RS_i ⇐ Client: ("register", ID, K_i, S_i);

3.2.3 개인키를 크레덴셜 서버에 저장하는 과정

① 사용자 : 사용자는 강한 키 값과 암호키를 조합해서 데이타 암호용 키를 생성한 뒤, 이 키를 사용해서 자신의 개인키를 암호화한다.

EK ← XOR(K_{n+2}, S);

EPRI ← ENC(EK, PRI);

② 사용자 ↔ 크레덴셜서버 : 사용자는 자신의 아이디, 사용자 인증에 사용될 K_{n+1} 값, 그리고 암호화된 개인키와 자신의 인증서 등을 안전한 보안 채널로 크레덴셜 서버에게 전달한다. 이때 크레덴셜

서버는 사용자가 전달한 값들을 자신의 데이타베이스에 안전하게 저장한다.

CS ⇐ Client: ("register", ID, K_{n+1}, (EPRI, CERT))

이 과정에서 위탁되는 정보 자체가 이미 암호화되어 있는 경우 공개되어도 산술적으로 정보에 대한 조작이나 복호를 통한 위변조 위험이 없고 적합한 취득자에게만 유용하므로 보안 채널을 사용하지 않아도 되며 아래와 같이 부가적인 인증을 위해 제공되는 K_{n+1} 값을 추가로 등록하지 않아도 된다.

CS ⇐ Client: ("register", ID, (EPRI, CERT));

3.3 로밍 프로토콜

로밍 프로토콜이란 사용자가 자신의 등록된 개인키와 인증서 등을 가져오는 프로토콜이다. 이 프로토콜의 과정은 사용자가 각각의 로밍 서버로부터 사용자 인증에 필요한 강한 패스워드를 얻어오는 과정, 사용자가 로밍 서버와의 인증을 통해서 저장된 암호키 조각을 얻어오는 과정, 그리고 크레덴셜로부터 가져온 자신의 암호화된 개인키를 복호화하는 과정으로 나누어진다. 각 과정의 더욱 자세한 내용은 다음과 같다.

3.3.1 강한 패스워드 획득 과정

① 사용자 ↔ 로밍서버 : 사용자는 각 로밍 서버들과 패스워드 강화 프로토콜을 수행해서 그 결과로 각 로밍 서버로부터 강한 패스워드 값들을 얻어온다. 이때 로밍 서버는 각 사용자가 패스워드 강화 프로토콜을 수행한 횟수를 기록한다. 참고로 로밍 서버는 패스워드 강화 프로토콜을 수행시 공격자의 사전탐색 공격을 막기 위해서 프로토콜의 실행 횟수를 기록한다. 그리고 만일 패스워드 강화 프로토콜의 수행 횟수와 인증 성공 횟수의 차이가 임계값보다 큰 경우, 공격자가 그 사용자를 공격하고 있다고 추측하여 그 사용자에 대한 패스워드 강화 프로토콜 수행을 중지한다.

for i=1 to i=n do

R_i ← PHP(RS_i, ID, PWD);

② 사용자 : 사용자는 키 생성 함수를 사용해서 강한 패스워드 값들로부터 강한 키 값들을 생성한다.

for i=1 to i=n+2 do

K_i ← KDF(R_1,R_2, ⋯, R_n, i);

3.3.2 사용자 인증 및 암호키 획득 과정

① 사용자 ↔ 로밍서버 : 사용자는 각 로밍 서버들에게 사용자의 아이디와 인증에 사용할 강한 키 값을 전달해서 사용자 자신을 인증한다. 이때 로밍 서버는 자신의 데이타베이스에 저장된 K_i 값과 사용자가 전달한 K_i 값을 비교해서 사용자를 인증하게 된다. 인증이 성공하는 경우 로밍 서버는 공격자의 사전탐색 공격을 막기 위해서 인증 성공 횟수를 기록한다. 사용자 인증이 성공하고 패스워드 강화 프로토콜 수행 횟수와 인증 성공 횟수의 차이가 임계 값보다 작은 경우 로밍 서버는 사용자에게 저장된 암호키 조각 값을 전달한다. 이때 사용자와 로밍 서버간의 통신은 안전한 채널로 보호할 경우의 과정이다.

for i=1 to i=n do

RS_i ⇐ Client: ("retrieve", ID, K_i);

RS_i ⇒ Client: (S_i);

이 과정은 다음과 같이 안전한 채널을 사용하지 않도록 수정될 수 있다. 먼저 로밍 서버는 랜덤 넌스 값을 생성해서 사용자에게 전달한다. 사용자는 인증에 사용될 값과 넌스 값의 해쉬를 생성해서 자신의 아이디와 같이 로밍 서버에게 전달한다. 로밍 서버는 데이타베이스에서 사용자의 인증 값을 가져와서 넌스 값과 같이 해쉬를 한다. 이때 사용자가 보내온 해쉬 값과 계산한 해쉬 값이 동일한 경우 사용자와 로밍 서버간의 인증이 성공하게 된다. 인증이 성공한 경우 로밍 서버는 사용자의 인증 성공 횟수를 기록한다. 이때 사용자의 인증이 성공하고 패스워드 강화 프로토콜 수행 횟수와 인증 성공 횟수의 차이가 임계 값보다 작은 경우 로밍 서버는 데이타베이스에 저장된 사용자의 암호키 조각을 사용자와 로밍 서버 모두 알고 있는 K_i 값으로 암호화해서 사용자에게 전달한다.

for i=1 to i=n do

RS_i ⇐ Client: ("retrieve");

RS_i ⇒ Client: (RN_i);

RS_i ⇐ Client: (ID, HASH(K_i, RN_i));

RS_i ⇒ Client: (ENC(K_i, S_i));

② 사용자 : 사용자는 각 로밍 서버로부터 받은 암호키 조각을 비밀 분산 기법을 이용해서 다시 원본 암호키로 복구한다.

S ← SSS(S_1,S_2,⋯,S_n);

3.3.3 개인키 획득 및 복호 과정

① 사용자 ↔ 크레덴셜서버 : 사용자는 자신의 아이디, 사용자 인증에 사용될 K_{n+1} 값을 안전한 채널로 크레덴셜 서버에게 전달해서 사용자 인증을 수행한다.

CS ⇐ Client: ("retrieve", ID, K_{n+1});

CS ⇒ Client: ((EPRI, CERT));

이 과정에서 위탁된 정보 자체가 이미 암호화 되어 있는 경우 공개되어도 산술적으로 정보에 대한 조작이나 복호를 통한 위변조 위험이 없고 적합한 취득자에게만 유용하므로 보안 채널을 사용하지 않아도 되며 다음과 같은 절차로 수행할 수 있다. 먼저 사용자는 자신의 아이디를 크레덴셜 서버에게 전달한다. 크레덴셜 서버는 사용자의 아이디에 해당하는 암호화된 개인키와 인증서를 가져와서 사용자에게 전달한다.

CS ⇐ Client: ("retrieve", ID);

CS ⇒ Client: (EPRI, CERT);

② 사용자 : 사용자는 강한 패스워드 값과 복구된 암호용 키를 조합한 뒤, 이 키 값을 사용해서 자신의 암호화된 개인키를 복호화 한다.

EK ← XOR(K_{n+2}, S);

PRI ← DEC(EK, EPRI);

3.4 패스워드 강화 프로토콜

패스워드 강화 프로토콜은 로밍 서버와 연계해서 사용자의 약한 패스워드를 강한 패스워드로 변환시켜 주는 프로토콜이다. 여기서는 RSA 기반의 패스워드 강화 프로토콜을 기술하도록 한다. RSA 알고리즘 기반 프로토콜의 장점은 Ford와 Kaliski가 제안한 Discrete-Logarithm 방식과 달리 SSL과 같은 보안 채널이 필요하지 않다는 장점이 있다.

먼저 프로토콜을 시작하기 전에 사용자는 등록과정을 통해서 각각의 로밍 서버에 자신의 PHP 개인키와 PHP 인증서가 발급되어 있어야 한다. 이때 PHP 개인키는 RSA 개인키 (n,d), PHP 인증서는 RSA 공개키 (n,e) 값을 저장하고 있다. 로밍 서버 (RSi)와 사용자간의 패스워드 강화 프로토콜은 다음과 같이 기술된다.

① 사용자 ↔ 로밍서버 : 사용자는 자신의 아이디를 이용해서 로밍 서버에 저장된 자신의 PHP 인증서를 가져온다. 참고로 사용자가 이미 자신의 PHP 인증서를 가지고 있는 경우 이 과정은 생략 될 수 있다. PHP 인증서를 획득한 다음 사용자는 PHP 인증서 검증을 수행한다. 인증서 검증이 실패하는 경우 사용자는 프로토콜 수행을 중지한다. 참고로 사용자의 PHP 인증서를 검증하는 이유는 공격자가 로밍 서버로 위장해서 사용자를 공격하는 것을 막기 위해서이다.

RS_i ⇐ Client: ("retrieve", ID);

RS_i ⇒ Client: (PHC_ID);

② 사용자 : 사용자는 자신의 PHP 인증서에서 RSA 공개키를 추출하고, 프로토콜에 사용할 랜덤 k 값을 생성한 다음 패스워드에 MGF 함수를 적용한 값과 k 값의 RSA 공개키 연산 값을 곱해서 r 값을 생성한다.

(n,e) ← PHC_ID;

k ← Random;

r ← MGF(PWD) * k^e mod n;

③ 사용자 ↔ 로밍서버 : 위에서 구한 r 값을 로밍 서버에게 전달한다.

RS_i ⇐ Client: (r);

④ 로밍서버 : 먼저 로밍서버는 패스워드 강화 프로토콜의 수행 횟수와 인증 성공 회수의 차이가 임계값 보다 큰 경우 사용자에게 주어진 사용자 아이디에 해당하는 패스워드 강화 프로토콜을 수행 할 수 없다고 알려주고 프로토콜 수행을 중지한다. 그렇지 않은 경우 로밍 서버는 사용자 아이디에 해당하는 PHP 개인키로부터 RSA 개인키 값을 추출하고 사용자가 보내온 r 값에 대한 RSA 개인키 연산을 수행해서 s 값을 생성한다. 그리고 로밍 서버는 사용자의 패스워드 강화 프로토콜 수행 횟수를 기록한다.

(n,d) ← PHK_ID;

s ← r^d mod n;

⑤ 사용자 ↔ 로밍서버 : 위에서 구한 s 값을 사용자에게 전달한다.

RS_i ⇒ Client: (s);

⑥ 사용자 : 사용자는 로밍 서버가 보내준 s 값에 대한 RSA 공개키 연산을 수행해서 로밍 서버가 사용자 자신의 PHP 개인키에 해당하는 값으로 연산을 수행했는지 검증한다. 검증에 실패하는 경우 프로토콜 수행을 중지한다. 검증에 성공한 경우 연산을 통해서 MGF (PWD)^d mod n 값에 해당하는 강한 패스워드 R_i 값을 생성한다.

```
if (r = s^e mod n)
    R_i ← s/k mod n;
else
    reject;
```

4. 안전성 분석

이 절에서는 앞에서 기술한 로밍 프로토콜의 안전성을 분석하도록 한다. 안전성 분석은 안전한 패스워드 기반 다중 로밍 시스템이 만족해야 하는 조건을 기술된 시스템이 만족하는지 검증하도록 한다. 즉, 사용자 위장 온라인 공격과 로밍 서버 위장 온라인 공격에 대해서

기술된 로밍 프로토콜이 안전한지 검증한다. 또한, 앞서 기술된 로밍 프로토콜 중에서 SSL과 같은 보안 채널을 사용하지 않는 프로토콜도 안전하다는 것을 증명하도록 한다.

정리 1) 앞에서 기술된 다중 서버를 이용한 패스워드 기반 로밍 프로토콜은 암호 모듈이 안전하고 패스워드 강화 프로토콜이 안전한 경우, 사용자 위장 온라인 공격에 대해서 안전하다.

증명) 사용자를 위장한 온라인 공격 방법은 공격자가 사용자의 위치에서 로밍 서버, 크레덴셜 서버와 온라인 통신을 통해서 사용자의 패스워드를 유추하거나 사용자의 개인키를 복구하는 공격 방법이다. 즉, 공격자는 일반 사용자와 동일하게 로밍 서버, 크레덴셜 서버 간의 프로토콜에 참여해서 공격하는 방법이다.

먼저 공격자는 사용자 A와 로밍 서버들 간의 통신 그리고 사용자 A와 크레덴셜 서버간의 통신을 통해서 다음과 같은 정보를 얻을 수 있다.

(A, PHC_A_1, r_1, s_1), ⋯, (A, PHC_A_n, r_n, s_n),
(RN_1, HASH(K_1, RN_1), ENC(K_1, S_1)), ⋯,
(RN_n, HASH(K_n, RNs), ENC(K_n, S_n)),
(EPRI_A, CERT_A);

경우 1) 공격자가 사용자 A의 암호화된 개인키를 복호화 하는 경우를 가정해 보자. 프로토콜 정의에 의해서 암호화된 사용자 A의 개인키는 다음과 같은 과정을 통해서 암호화 되었다.

K_{n+2} ← KDF(R_1,R_2,⋯,R_n, n+2);
S ← SSS(S_1,S_2,⋯,S_n);
EK ← XOR(K_{n+2}, S);
EPRI_A ← ENC(EK, PRI_A);

따라서 대칭키 암호 알고리즘이 안전한 경우 공격자가 암호화된 개인키를 복호화 하기 위해서는 K_{n+2} 값과 S 값을 유추 할 수 있어야 한다. 비밀 분산 방법이 안전한 경우 S 값을 유추하기 위해서는 암호키 조각 S_1,S_2,⋯,S_n 값들을 유추해야 한다. 이들 암호키 조각 값들은 각 로밍 서버가 사용자 인증을 통해서 사용자에게 전달한 사용자의 암호키 조각 값들이다. 그런데 공격자는 사용자와 i 번째 로밍 서버와의 통신에서 다음과 같은 정보를 알 수 있다. 따라서 공격자는 이 정보로부터 S_i 값들을 유추하던지 또는 K_i 값들을 유추해야 한다.

(RN_i, HASH(K_i, RN_i), ENC(K_i, S_i))

그런데 대칭키 알고리즘 ENC가 안전한 하고 K_i 값을 모르는 경우 ENC(K_i, S_i) 암호문으로부터 S_i 값을 유추하는 것은 어렵다. 또한 HASH(K_i, RN_i) 값으로부터 K_i 값을 유추하는 것도 어렵다. 따라서 공격자는 S 값을 유추하기 위해서 K_1,⋯,K_n 값을 유추할 수 있어야 한다. 그러므로 공격자가 사용자 A의 암호화된 개인키를 복호화 하기 위해서는 K_1,K_2,⋯,K_n, K_{n+2} 값을 유추할 수 있어야 한다. 이 값들을 유추하는 것은 패스워드 강화 프로토콜에 대한 공격이 성공하는 것과 동일하다. 따라서 패스워드 강화 프로토콜이 안전한 경우 공격자가 암호화된 개인키 데이타로부터 사용자의 개인키 정보를 유추하는 것은 어렵다.

경우 2) 공격자가 사용자 A의 패스워드를 유추하는 경우를 가정해 보자. 즉, 공격자는 사전탐색과 같이 사용자가 사용 가능한 모든 패스워드를 입력해서 검증하는 방식으로 시스템을 공격할 수 있다. 공격자가 임의의 패스워드를 이용해서 이것이 사용자의 패스워드인지 검증하는 방법은 다음과 같다.

for i=1 to i=n do
    R_i' ← PHP(RS_i, A, PWD');
K_1' ← KDF(R_1',R_2',⋯,R_n', 1);
if HASH(K_1,RN_1) = HASH(K_1',RN_1) return
        success;
else return fail;

즉, 먼저 공격자는 가능한 패스워드 PWD' 을 유추하여 각 로밍 서버와 패스워드 강화 프로토콜을 수행해서 강한 패스워드 R_i' 값들을 구한다. 이 값들을 키 생성 함수에 입력하면 K_1' 값을 구할 수 있는데, 공격자는 이전 사용자와 로밍 서버와의 통신을 통해서 RN_1 값과 HASH(K_1, RN_1) 값을 알고 있으므로 이 값을 사용해서 패스워드가 올바른지 확인할 수 있다. 물론 공격자는 K_1' 값으로 로밍 서버에 인증을 수행해서 패스워드가 올바른지 검증할 수도 있다. 이때 공격자가 여러 개의 패스워드를 시도함에 따라 패스워드 강화 프로토콜의 수행 횟수와 로밍 서버의 사용자 인증 횟수는 차이가 나게 된다. 이 차이가 특정 임계 값을 넘어서면 로밍 서버는 패스워드 강화 프로토콜의 수행을 중지한다. 따라서 공격자의 패스워드 추측 공격을 막을 수 있다.

정리 2) 앞에서 기술된 다중 서버 이용한 패스워드 기반 로밍 프로토콜은 암호 모듈이 안전하고 패스워드 강화 프로토콜이 안전한 경우, 로밍 서버 위장 온라인 공격에 대해서 안전하다.

증명) 로밍 서버를 위장한 온라인 공격은 공격자가 n-1개 보다 작거나 같은 수의 로밍 서버를 공격해서 공격자 마음대로 로밍 서버를 동작 시킬 수 있는 공격이다. 이때 공격자가 사용자를 공격해서 패스워드를 유추하는 경우와 다른 로밍 서버를 공격해서 사용자의 패스워드를 유추하는 두 가지 경우에 대해서 안전성을 검증

하도록 한다.

공격자가 n-1개보다 작거나 같은 로밍 서버들을 공격해서 얻을 수 있는 정보는 다음과 같다. 즉, 공격자는 네트웍을 통해서 얻을 수 있는 정보 이외에 1,2,…,n-1 번째 로밍 서버의 내부에 저장된 사용자의 비밀 정보들을 얻게 된다.

(A, PHC_A_1, r_1, s_1), …, (A, PHC_A_n, r_n, s_n),
$(RN_1, HASH(K_1, RN_1), ENC(K_1, S_1))$, …,
$(RN_n, HASH(K_n, RN_n), ENC(K_n, S_n))$,
(EPRI_A, CERT_A);
(PHK_A_1, PHC_A_1), …, $(PHK\_A_{n-1}$,
$PHC\_A_{n-1})$, (K_1, S_1), …, (K_{n-1}, S_{n-1}),

경우 1) 공격자가 사용자를 공격하는 경우를 가정해 보자. 공격자는 사용자의 암호화된 개인키를 복호화 하거나 또는 사용자의 패스워드를 유추하는 경우 성공하게 된다. 공격자가 사용자의 암호화된 개인키를 복호화 하기 위해서는 K_n, K_{n+2} 값을 유추해야 한다. 이 값들은 강한 패스워드로부터 생성된 값이므로 공격자는 $R_1,R_2,\cdots,R_n$ 값을 유추해야 한다. 이때 최악의 경우 $R_1,R_2,\cdots,R_{n-1}$ 값들은 로밍 서버 RS_1, $RS_2,\cdots,RS_{n-1}$이 모두 담합을 하였다는 가정을 해보자. 하지만 R_n 값을 유추해낼 수가 없기 때문에 n 번째 로밍 서버로부터 R_i 값을 알아낼 수 없고 사용자의 패스워드도 알아낼 수 없다. 따라서 패스워드 강화 프로토콜이 안전한 경우 공격자가 사용자를 공격하는 경우에도 프로토콜은 안전하다.

경우 2) 공격자가 n 번째 로밍 서버를 공격하는 경우를 가정해 보자. 즉, 공격자는 1,2,…,n-1 로밍 서버들과 사용자를 위장해서 n 번째 로밍 서버를 공격해서 사용자의 패스워드를 유추하거나 암호화된 개인키를 복호화 하는 경우 성공하게 된다. 이 경우 역시 K_{n+2} 값을 알기 위해서는 강한 패스워드 값이 $R_1,R_2,\cdots,R_n$ 값들을 공격자가 유추해야 한다. 이 것 역시 앞의 경우와 마찬 가지로 R_n 값을 얻기 위해서 패스워드 추측 공격을 수행하는 경우 n 번째 로밍 서버가 이 공격을 감지 할 수 있다. 따라서 패스워드 강화 프로토콜이 안전한 경우 프로토콜은 안전하다.

정리 3) 패스워드 강화 프로토콜에 사용된 암호 모듈이 안전한 경우, 패스워드 강화 프로토콜은 안전하다.

증명) 안전한 패스워드 강화 프로토콜은 공격자의 패스워드 추측 공격을 감지 할 수 있어야 하고, 공격자가 로밍 서버를 공격하는 경우 사용자의 PHP 개인키에 관한 정보를 얻지 못하도록 해야 하고, 공격자가 로밍 서버로 위장해서 사용자를 공격하는 경우 사용자 패스워드 또는 강한 패스워드를 유추하는 것이 어려워야 하고, 사용자는 로밍 서버가 자신의 PHP 개인키로 값을 계산했다는 것을 확인 할 수 있어야 한다.

경우 1) 공격자가 패스워드 추측 공격을 수행하는 경우를 가정해 보자. 공격자는 온라인 또는 오프라인으로 패스워드 추측을 수행할 수 있다. 먼저 오프라인으로 추측하는 경우 강한 패스워드는 로밍 서버에 저장된 사용자의 PHP 개인키와 연관되는 값으로 계산이 된다. 따라서 공격자가 사용자의 PHP 개인키를 모르는 경우 오프라인 공격은 성공하지 못한다. 공격자가 온라인으로 패스워드 추측공격을 수행하는 경우 로밍 서버는 공격자의 패스워드 강화 프로토콜 수행 횟수와 사용자 인증 성공 횟수의 차이로 인해서 공격자의 패스워드 추측 공격을 감지 할 수 있다. 따라서 패스워드 강화 프로토콜은 패스워드 추측 공격에 대해서 안전하다.

경우 2) 공격자가 i 번째 로밍 서버를 공격해서 사용자 A의 PHP 개인키에 관한 정보를 얻는 경우를 가정해 보자. 공격자는 패스워드 강화 프로토콜의 통신 메시지로부터 (A, (n,e), r, s) 값을 알 수 있다. 즉, 공격자는 (n,e), (r^d mod n) 값으로부터 d 값을 유추해야 한다. 이 것은 RSA 서명 알고리즘의 안전성과 같다. 따라서 RSA 알고리즘이 안전한 경우 공격자는 사용자의 PHP 개인키에 관한 정보를 알 수 없다.

경우 3) 공격자가 i 번째 로밍 서버를 위장해서 사용자 A로부터 패스워드 또는 강한 패스워드 R_i 값을 유추하는 경우를 가정해 보자. 공격자는 (A, (n,d), (n,e), r, s) 값을 알고 있다. 패스워드 강화 프로토콜의 정의에 의해서 다음 식이 성립한다.

k ← Random;
r ← MGF(PWD) * k^e mod n;
s ← r^d mod n;
R_i ← s/k mod n;

이때 공격자가 사용자의 패스워드를 유추하기 위해서는 MGF(PWD) 값을 알아내면 되고, MGF(PWD) 값을 알아내기 위해서는 k 값을 알아 내어야 한다. 그리고 R_i 값을 유추하기 위해서 k 값을 알아내면 된다. 그런데 k 값은 사용자가 랜덤 하게 생성한 값이기 때문에 유추하기 어렵다. 따라서 공격자가 사용자의 패스워드를 유추하거나 강한 패스워드를 유추하는 것이 어렵다.

경우 4) 공격자가 로밍 서버를 위장해서 사용자의 PHP 개인키 대신 다른 값으로 s 값을 계산하는 경우를 가정해 보자. 이 경우 사용자는 자신이 보낸 r 값과 s^e mod n 값이 동일한지 비교해서 로밍 서버가 자신의 PHP 개인키로 계산했는지를 검증할 수 있다. 따라서 로밍 서버가 사용자의 PHP 개인키로 계산했다는 것을

보장한다.

정리 4) 암호 모듈이 안전한 경우 앞에서 기술된 다중 서버를 이용한 패스워드 기반 로밍 프로토콜은 안전하다.

증명) 정리 3에 의해서 패스워드 강화 프로토콜은 안전하므로 정리 1, 2에 의해서 사용자 위장 공격과 로밍 서버 위장 공격에 대해서 로밍 프로토콜은 안전하다.

5. 성능분석

이 절에서는 다중 로밍 서버를 이용하는 패스워드 기반의 로밍 프로토콜들의 성능을 비교 분석하도록 한다. 비교 대상 프로토콜은 ① Ford와 Kaliski가 제안한 Discrete-Logarithm을 이용한 로밍 프로토콜 ② Jablon이 제안한 로밍 프로토콜 그리고 ③ 이 논문에 기술된 RSA 기반의 로밍 프로토콜이다. 이들 각각의 로밍 프로토콜에 관한 더욱 자세한 내용은 각 논문을 참조하기 바란다[1,3].

로밍 프로토콜의 성능을 비교하기 위해서 프로토콜이 수행하는 모듈러 지수승 연산을 비교하도록 한다. 모듈러 지수승 연산은 프로토콜 수행시 가장 시간이 많이 걸리는 작업으로 프로토콜 수행 시간의 대부분을 차지한다. 1024 비트 키를 이용하는 경우 각 프로토콜이 수행해야 하는 모듈러 지수승 연산은 표 1과 같다. 이때 MEXP(k, t)는 지수의 길이가 k 비트이고 모듈러 길이가 t 비트인 경우 모듈러 지수승 연산이다. 그리고 RSAD(t), RSAE(t)는 모듈러 길이가 t 비트인 RSA 알고리즘의 개인키 및 공개키 연산의 모듈러 지수승 연산을 나타낸다.

표 1 1024 비트 키 이용하는 경우 각 로밍 프로토콜 연산량

	로밍 서버 연산량	클라이언트 연산량
①	MEXP(1023, 1024) + SSL_SERVER	2*MEXP(1023, 1024) + SSL_CLIENT
②	MEXP(160, 1024) + SIGN_VERIFY	MEXP(864, 1024) + 2*MEXP(160, 1024) + SIGN_GEN
③	RSAD(1024)	3*RSAE(1024)

일반적인 모듈러 지수승 연산을 수행하는 경우 MEXP(k, t) 연산은 3/2 * k * t^2 번의 비트 연산을 수행해야 한다[5]. 모듈러 길이가 t 비트인 RSA 개인키 연산의 경우 CRT (Chinese Remainder Theorem) 이용하면 3/8 * t^3 비트 연산을 수행해야 하고, 공개키 지수로 65537을 사용하는 경우 공개키 연산은 18 * t^2 비트 연산이 필요하다. 또한 FK 로밍 프로토콜 수행시 1024 비트 RSA SSL 을 사용하고, Jablon 로밍 프로토콜 수행시 1024 비트 RSA 서명 알고리즘을 사용한다고 가정하면 각 연산 수행에 필요한 비트 연산은 다음과 같다.

MEXP(k, t)의 연산량: 3/2 * k * t^2,
RSAD(t)의 연산량: 3/8 * t^3,
RSAE(t)의 연산량: 18 * t^2,
SSL_SERVER의 연산량: RSAD(t),
SSL_CLIENT의 연산량: RSAE(t),
SIGN_GEN의 연산량: RSAD(t),
SIGN_VERIFY의 연산량: RSAE(t)

이 값을 이용해서 각 로밍 프로토콜이 수행해야 하는 연산량을 상대적으로 표시한 값은 표 2와 같다. 이때 사용한 단위는 1024^3 비트 연산이다.

표 2의 결과를 토대로 로밍 서버의 연산량, 클라이언트 연산량 그리고 로밍 프로토콜 전체 연산량을 비교해 보도록 하자.

표 2 1024 비트 키 이용하는 경우 각 로밍 프로토콜 연산량의 상대적 비교

	로밍 서버 연산량	클라이언트 연산량	전체 연산량
①	1.8	3.0	4.8
②	0.2	1.3	1.5
③	0.4	0.1	0.5

로밍 서버 연산량은 Jablon 로밍 프로토콜이 가장 작은 값을 가진다. 클라이언트가 하나의 로밍 서버와 통신하는 경우 클라이언트 연산량은 RSA 기반의 로밍 프로토콜이 가장 작은 값을 가진다. 로밍 프로토콜 전체 작업은 서버의 작업과 클라이언트의 작업의 합이므로 클라이언트가 하나의 로밍 서버와 작업을 수행하는 경우 전체 연산량은 RSA 기반의 로밍 프로토콜이 가장 작은 값을 가진다. 또한 로밍 프로토콜의 수행 속도는 전체 연산량이 작을수록 빠르다. 따라서 RSA 기반의 로밍 프로토콜은 FK 로밍 프로토콜에 비해서 9.6배, Jablon 로밍 프로토콜에 비해서 3배 정도의 빠른 속도를 나타낸다.

RSA 기반의 로밍 프로토콜의 경우 프로토콜의 수행 속도가 다른 프로토콜에 비해서 빠르지만 로밍 서버의 연산량이 Jablon 로밍 프로토콜 보다 많다는 단점이 있다. 그러나 RSA 변형 알고리즘을 사용하면 RSA 기반의 로밍 프로토콜의 경우 로밍 서버 연산량을 줄일 수 있다. 기본 아이디어는 RSA 개인키 연산을 빠르게 할

수 있는 RSA 변형 알고리즘을 사용하는 것이다[6]. 각 알고리즘에 대해서 개인키 연산의 속도 향상은 표 3과 같다. 따라서 RSA 개인키 연산의 경우 최대 3배의 속도 향상이 가능하므로 다른 로밍 프로토콜에 비해서 더욱 빠른 로밍 서버 수행 속도를 얻을 수 있다.

표 3 1024 비트 키를 사용하는 RSA 변형 알고리즘의 속도 향상 비교

	소요시간 (unit)	속도향상 (倍)
RSA	10 (기준치)	1
Multi-Prime RSA	5.78	1.73
Multi-Power RSA	4.35	2.30
Rebalance RSA	3.27	3.06

6. 결 론

이 논문은 Ford와 Kaliski가 제안한 다중 서버를 이용한 패스워드 기반 로밍 프로토콜에 RSA 알고리즘을 적용한 효율적인 프로토콜을 제안하고 이 프로토콜이 안전 하다는 것을 검증했다. 즉, 이 프로토콜은 n개의 로밍 서버 중에 최소 하나의 로밍 서버라도 공격자에게 침입 당하지 않는 경우 패스워드 강화 프로토콜의 안전성에 의해서 패스워드 추측 공격을 막을 수 있기 때문에 안전하다. 또한 이 프로토콜은 Ford와 Kaliski가 제안한 프로토콜과 비교해 채널 보안을 위한 SSL과 같은 오버헤드가 요구되지 않으며 프로토콜 자체의 연산량 비교에서 나타나듯 저비용으로 최대 9배의 처리속도 향상을 기대할 수 있다.

참 고 문 헌

[1] W. Ford and B. Kaliski, "Server-Assisted Generation of a Strong Secret from a Password," IEEE 9th International Workshop on Enabling Technologies (WET ICE'00), pp. 176-180, 2000.

[2] D. Chaum, "Security without Identification: Transaction Systems to Make Big Brother Obsolete," Communications of the ACM, Vol. 28, pp. 1030-1044, 1985.

[3] D. Jablon, "Password Authentication Using Multiple Servers," Proceeding of CT-RSA 2001, pp. 344-360, 2001.

[4] 박해룡, 권현조, 김지연, 김승주, "국외 키로밍 제품 개발 현황 분석", 정보 보호 학회지, pp. 104-117, 2002.

[5] A. Menezes, P. Van. Oorschot, and S. Vanston, Handbook of Applied Cryptography, CRC Press, pp. 591-634, 1997.

[6] D. Boneh and H. Shacham, "Fast Variants of RSA," CryptoBytes, Vol. 5, No. 1, pp. 1-9, 2002.

정 현 철

1989년 계명대 컴퓨터 공학과 졸업(공학사) 1991년 경북대 대학원 컴퓨터공학과 졸업(공학석사). 1991.2.~1998.10 한국전자통신연구원. 1998.10~현재 소프트포럼(주) 부사장. 2002.12~현재 알파로직스(주) 대표이사

김 승 호

1981년 경북대학교 전자공학과 졸업(공학사). 1983년 한국과학기술원 전산학과 졸업(공학석사). 1994년 한국과학기술원 전산학과 졸업(공학박사). 1985년~현재 경북대학교 컴퓨터공학과 교수