공개키를 적용한 S/KEY 기반의 안전한 사용자 인증 프로토콜

유 일 선[†] · 조 경 산[††]

요 약

본 논문에서는 S/KEY 인증 프로토콜과 YEH와 SHEN, HWANG이 제안한 인증 프로토콜의 문제점을 개선하기 위하여 스마트 카드를 적용한 S/KEY 기반의 인증 프로토콜을 제안하였다. 제안 프로토콜은 SEED를 공유 비밀키로 적용하는 이들의 인증 프로토콜과 달리 공개키를 통해 S/KEY 인증 프로토콜을 개선하였다. 따라서 제안 프로토콜은 공유 비밀키 없이 서버를 인증하고 세션키를 분배 할 수 있다. 또한, 사용자의 암호 구문대신 임의로 생성된 강력한 비밀키를 적용하기 때문에 오프라인 사전 공격을 방지할 수 있다. 특히, 제안 프로토콜은 사용자의 비밀키 혹은 기타 비밀정보를 서버에 저장하지 않는 S/KEY 인증 프로토콜의 장점을 충실하게 만족할 수 있기 때문에 서버의 붕괴로 인해 사용자 로그인 정보가 유출되는 최악의 경우에도 유출된 정보를 통한 각종 공격에 대응할 수 있다.

A S/KEY Based Secure Authentication Protocol Using Public Key Cryptography

Il-Sun You[†] · Kyungsan Cho[††]

ABSTRACT

In this paper, we propose a S/KEY based authentication protocol using smart cards to address the vulnerabilities of both the S/KEY authentication protocol and the secure one-time password protocol which YEH, SHEN and HWANG proposed [1]. Because our protocol is based on public key, it can authenticate the server and distribute a session key without any pre-shared secret. Also, it can prevent off-line dictionary attacks by using the randomly generated user secret that is stored in the users smart card. More importantly, it can truly achieve the strength of the S/KEY scheme that no secret information need be stored on the server.

키워드 : 사용자 인증(Authentication), 일회용 비밀번호(One Time Password), S/KEY, 스마트카드(Smart Card)

1. 서 론

일회용 비밀번호 인증 프로토콜은 동일한 비밀번호를 여러 번 반복함으로써 발생할 수 있는 보안 문제점을 해결하기 위해 제안되었으며, S/KEY 인증 프로토콜, 질의/응답 인증 프로토콜, 시간 동기화 인증 프로토콜로 분류될 수 있다[2, 3, 8].

특히, S/KEY 인증 프로토콜은 하드웨어 토큰을 적용하는 다른 인증 프로토콜들과 달리 소프트웨어 기반의 구조로서 인증서버에 사용자의 비밀키(사용자가 직접 입력한 암호구문으로부터 파생된 값)가 저장되지 않는 장점을 갖는다[2, 3]. 그러나 S/KEY 인증 프로토콜이 비밀번호 유출로 인한 수동공격(passive attack)에 안전함에도 불구하고 서버위장(server spoofing) 공격, Preplay 공격 그리고 오프라인 사전(off-line dictionary) 공격에 취약한 문제점을 갖는다[1-4].

최근에 YEH와 SHEN, HWANG은 이러한 공격들에 대응할 수 있도록 스마트 카드를 적용하여 S/KEY를 개선하였다[1]. 이들의 인증 프로토콜(ysh-S/KEY)은 SEED를 서버와 클라이언트 사이의 공유된 비밀키(pre-shared secret)로 적용하여 앞서 언급된 취약점들을 해결하였으며 동시에 세션키의 제공을 통해 안전한 메시지 전송이 가능하도록 하였다. 이처럼 ysh-S/KEY 인증 프로토콜이 S/KEY 인증 프로토콜의 문제점을 개선하였으나 여전히 서비스 거부(denial of service) 공격 및 Stolen-Verifier 공격, Denning-Sacco 공격에 취약하고 사용자에게 불편함을 초래하는 문제점을 갖는다[5-7]. 이러한 문제점은 ysh-S/KEY 인증 프로토콜이 SEED를 공유 비밀키로 사용하고 사용자의 취약한 비밀키를 그대로 적용하기 때문에 발생한다. 따라서 SEED 보다는 사용자의 비밀키를 강화시키는 것이 바람직하다.

본 논문에서는 앞서 언급된 S/KEY 인증 프로토콜과 ysh-S/KEY 인증 프로토콜의 문제점을 개선한 S/KEY 기반의 안전한 사용자 인증 프로토콜을 제안한다. 제안 프로토콜은 ysh-S/KEY 인증 프로토콜과 달리 공개키를 적용

† 정 회 원 : 단국대학교 대학원 전산통계학과
†† 종신회원 : 단국대학교 정보컴퓨터학부 교수
논문접수 : 2003년 7월 22일, 심사완료 : 2003년 9월 25일

하여 서버를 인증하며 세션키를 분배한다. 또한, 사용자가 직접 입력한 암호 구문(pass phrase)으로부터 파생되는 비밀키 대신 임의로 생성된 강력한 비밀키를 스마트 카드에 저장하여 사용하기 때문에 오프라인 사전 공격에 취약하지 않은 장점이 있다.

본 논문의 구성은 다음과 같다. 2장에서는 S/KEY 인증 프로토콜과 ysh-S/KEY 인증 프로토콜을 기술한 후, 각 인증 프로토콜들의 취약점을 논의한다. 3장에서는 스마트 카드를 적용하는 S/KEY 기반의 안전한 사용자 인증 프로토콜을 제안한다. 4장에서는 제안 프로토콜을 분석하고 5장에서 결론을 맺는다.

2. 관련 연구

본 장에서는 S/KEY 인증 프로토콜과 ysh-S/KEY 인증 프로토콜을 기술하고 각 인증 프로토콜의 취약점을 논의한다. <표 1>은 각 인증 프로토콜을 기술하기 위해 사용되는 기호를 나타낸다.

〈표 1〉 기 호

기 호	의 미
K	사용자의 비밀키
C	클라이언트 컴퓨터
S	인증서버
H( )	일방향 해쉬함수
$H^X(m)$	메시지 m이 X번 해쉬됨
KR_X	X의 개인키
KU_X	X의 공개키
$P_X(m)$	메시지 m이 X의 공개키로 암호화됨
$P^{-1}_X(m)$	메시지 m이 X의 개인키로 암호화됨
p_t	t번째 일회용 비밀번호
⊕	배타적 논리합 연산
|	concatenation 연산

2.1 S/KEY 인증 프로토콜

S/KEY 인증 프로토콜은 등록 단계와 로그인 단계, 인증 단계로 구성된다[2, 3].

2.1.1 등록 단계

① C → S : id

② C ← S : N, SEED

③ C → S : p_0

$$p_0 = H^N(K \oplus SEED) \quad (1)$$

사용자를 등록하기 위하여 서버는 사용자에게 허용된 인증횟수 N을 선택하고 SEED를 임의로 생성한 후, 클라이언트에게 전송한다. 클라이언트는 사용자가 입력한 비밀키 K를 통해 초기 비밀번호 p_0를 계산하고 서버에게 전송한다. 서버는 초기 비밀번호 p_0와 N, SEED 값을 데이터베이스에 저장함으로써 사용자 등록을 종료한다.

2.1.2 로그인 단계

① C → S : id

② C ← S : CN, SEED

③ C → S : p_t

$$p_t = H^{CN}(K \oplus SEED) \quad (2)$$

사용자가 t번째 로그인 요청을 하면 서버는 로그인을 위해 CN(= N - t)과 SEED를 질의값으로서 클라이언트에게 전송한다. 질의값을 받으면 클라이언트는 t번째 비밀번호 p_t를 계산하여 서버에게 전송한다.

2.1.3 인증 단계

p_t를 받으면 서버는 p_t의 해쉬값과 데이터베이스에 있는 p_{t-1}과 비교한다. 만일 두 값이 일치하면, 클라이언트가 서버에게 성공적으로 인증되었음을 의미하며, 이 경우 서버는 사용자의 로그인 정보를 p_t와 CN으로 갱신한다.

2.1.4 취약점

S/KEY 인증 프로토콜은 다음과 같이 오프라인 사전 공격과 서버위장 공격, Preplay 공격에 취약한 문제점을 갖는다.

1) 오프라인 사전 공격

S/KEY 인증 프로토콜은 사용자의 암호 구문으로부터 파생된 비밀키 K를 사용하기 때문에 오프라인 사전 공격에 취약하다. 공격자는 네트워크에서 가로챈 $p_t = H^{CN}(K \oplus SEED)$을 통해 추측된 K의 값을 검증할 수 있다.

2) 서버위장 공격과 Preplay 공격

예측이 용이한 CN과 SEED가 질의값으로 사용되기 때문에 공격자는 쉽게 서버를 위장하면서 유효한 비밀번호를 획득한 후, 서버에 성공적으로 인증 받을 수 있다. 이처럼 S/KEY 인증 프로토콜은 서버위장 공격과 Preplay 공격에 취약하다. Michell과 Chen은 이러한 공격에 대응하기 위해 다음과 같은 두 가지 해법을 제안하였다[4].

첫 번째는 예측 가능한 질의값 CN과 SEED를 서버뿐만 아니라 클라이언트도 자체적으로 보관하는 방법이다. 즉 클라이언트는 서버로부터 질의값을 받지 않고 직접 t번째 일회용 비밀번호 p_t를 생성할 수 있다. 서버는 클라이언트의 CN이 데이터베이스에 있는 값보다 작은 경우에 클라이언트를 인증한다. 그러나 이 방법은 공격자가 서버를 가장하여 클라이언트가 제시하는 p_t를 획득한 후 이를 통해 서버에 인증받는 공격에 취약하다.

두 번째는 예측 가능한 질의값을 전자서명하는 방법이다. 클라이언트는 서버가 생성한 전자서명을 통해 서버가 질의값을 생성하였다는 것을 신뢰할 수 있으며 이와 같은 방법으로 서버위장 공격과 Preplay 공격을 회피할 수 있다. 그러나 Michell과 Chen은 서버의 전자서명 생성절차나 서버의 공개키 검증과정과 같은 구체적인 메카니즘을 제시하지 않았다.

2.2 ysh-S/KEY 인증 프로토콜

ysh-S/KEY 인증 프로토콜은 S/KEY 인증 프로토콜과 같이 등록 단계와 로그인 단계, 인증 단계로 구성된다[1]. 이 인증 프로토콜은 SEED를 공유 비밀키로 사용하여 서버 위장 공격, Preplay 공격과 오프라인 사전 공격에 대응하며, 스마트 카드를 통해 SEED를 안전하게 저장하는 동시에 로그인 과정을 단순화하였다. 또한, 세션키를 제공함으로써 네트워크 상에서 안전한 통신이 가능하게 하였다.

2.2.1 등록 단계

① $C \rightarrow S : id$
② $C \leftarrow S : N, SEED \oplus D, H(D)$
③ $C \rightarrow S : p_0 \oplus D$

초기에 스마트 카드가 사용자에게 발급될 때 서버에 의해 임의로 생성된 공유 비밀키 SEED를 포함하고 있는 것을 가정한다.

사용자의 등록 요청이 발생하면 서버는 사용자에게 허용된 인증횟수 N을 선택하고 임의의 값 D를 생성한 후, SEED⊕D와 H(D)를 N과 함께 클라이언트에게 전송한다. 클라이언트는 스마트 카드에 저장된 SEED를 통해 D를 추출하고 H(D)를 통해 D의 값을 검증한다. D가 유효할 경우, 클라이언트는 사용자의 암호 구문으로부터 파생된 비밀키 K를 통해 초기 비밀번호 $p_0 = H^N(K \oplus SEED)$를 계산하고 서버에게 전송한다.

2.2.2 로그인 단계

① $C \rightarrow S : id$
② $C \leftarrow S : CN, SEED \oplus D, H(D) \oplus p_{t-1}$
③ $C \rightarrow S : p_t \oplus D$

사용자가 t번째 로그인을 요청하면 서버는 임의로 D를 생성하고 $SEED \oplus D$와 $H(D) \oplus p_{t-1}$를 계산한 후, $CN(= N - t)$과 함께 클라이언트에게 전송한다. 클라이언트는 사용자의 스마트 카드에 저장된 SEED를 통해 D를 추출하고 $H(D) \oplus p_{t-1}$의 유효성을 검증한다. 만일 $H(D) \oplus p_{t-1}$이 유효하면 클라이언트는 서버와 D값을 신뢰할 수 있다. 클라이언트가 서버를 성공적으로 인증하면 $p_t = H^{CN}(K \oplus SEED)$를 계산한 후에 $p_t \oplus D$를 서버에게 전송한다.

2.2.3 인증 단계

서버가 클라이언트로부터 $p_t \oplus D$를 받으면 D를 통해 p_t를 추출하고 p_t의 해쉬값과 데이터베이스에 저장된 p_{t-1}을 비교한다. 만일 두 값이 일치하면, 클라이언트가 서버에게 성공적으로 인증되었음을 의미하며, 이 경우 서버는 사용자의 로그인 정보를 p_t와 CN으로 갱신한다.

과정 ②에서 임의로 생성된 D는 서버와 클라이언트 사이에 안전한 메시지 교환을 위해 세션키로 사용될 수 있다.

2.2.4 취약점

본 절에서는 ysh-S/KEY 인증 프로토콜이 서비스 거부 공격 및 Stolen-Verifier 공격, Denning-Sacco 공격에 취약하고 사용자 불편을 초래하는 문제점을 보인다.

1) 서비스 거부 공격

사용자 등록요청이 발생할 경우, 서버는 등록 단계의 과정 ② 메시지를 생성하여 클라이언트에게 보낸다. 이때 공격자는 과정 ② 메시지의 N값을 N′값으로 변경할 수 있고 변경된 N′는 서버와 클라이언트가 비밀번호 동기화(synchronization)에 실패하도록 한다. 결국, 클라이언트는 초기 비밀번호 $p_0' = H^{N'}(K \oplus SEED)$를 생성하게 되고, p_0'를 갖는 서버는 클라이언트가 정확한 일회용 비밀번호를 제시한다 할지라도 접근을 거절한다.

이와 유사하게, 공격자는 등록 단계의 과정 ③ 메시지에서 $p_t \oplus D$ 값을 동일한 크기의 임의값 r로 변경할 수 있다. 임의값 r은 서버가 잘못된 초기 비밀번호 p_0'를 추출하게 하며 앞의 경우와 같이 서버와 클라이언트가 비밀번호 동기화에 실패하도록 한다.

2) Stolen-Verifier 공격

ysh-S/KEY 인증 프로토콜은 S/KEY 인증 프로토콜과 달리 SEED를 서버와 클라이언트 사이의 공유 비밀키로 사용함으로써 서버를 인증하고 Preplay 공격 및 오프라인 사전 공격을 예방한다. 그러나 만일 공격자가 서버로부터 SEED를 훔치는데 성공한다면 훔친 SEED를 이용하여 사용자 비밀키 K에 대해 모든 가능한 값을 적용하면서 다음과 같은 오프라인 사전 공격을 시도할 수 있다.

① K의 후보 K′를 선택한다.
② $p_t' = H^{CN}(K' \oplus SEED)$ 계산한다.
③ p_t와 p_t'를 비교한다.

만일 p_t와 p_t'가 같다면 사용자 비밀키 K를 구하는데 성공함을 의미한다.

t번째 비밀번호 p_t는 서버의 데이터베이스에서 가져오지 못할 경우 다음과 같이 구할 수 있다. 공격자는 로그인 단계의 과정 ② 메시지 중 $(t+1)$번째 메시지를 가로채고 가로챈 메시지의 $SEED \oplus D$와 서버로부터 훔친 SEED의 배타적 논리합 연산을 통해 D를 구한다. 마지막으로 D의 해쉬값과 $H(D) \oplus p_t$값의 배타적 논리합 연산을 통해 p_t를 구한다.

또한, 공격자는 t번째 비밀번호 p_t와 훔친 SEED를 통해 로그인 단계의 과정 ② 메시지를 위조함으로써 서버를 위장할 수 있고 클라이언트로부터 유효한 비밀번호를 얻는데 성공할 수 있다.

이와 같이, ysh-S/KEY 인증 프로토콜은 SEED의 유출에 의한 Stolen-Verifier 공격에 취약하며 이는 이 인증 프로토콜이 여전히 오프라인 사전 공격과 Preplay 공격에 취약하도록 하는 원인이 된다.

3) Denning-Sacco 공격

ysh-S/KEY 인증 프로토콜에서 만일 공격자가 서버와 클

라이언트 사이에 교환되었던 메시지를 통해 이전에 사용된 세션키 D를 얻을 수 있다면 공격자는 D를 이용하여 SEED ⊕ D로부터 SEED를 추출할 수 있고 추출된 SEED를 갖고 2)에서 언급된 오프라인 사전 공격이나 Preplay 공격을 시도할 수 있다. 이처럼 ysh-S/KEY 인증 프로토콜은 유출된 세션키 D에 근거한 Denning-Sacco 공격에 취약하다[5, 6].

4) 낮은 사용자 편의성

ysh-S/KEY 인증 프로토콜은 사용자에게 다음과 같은 불편함을 유발한다.

첫째로 이 인증 프로토콜은 공유 비밀키 SEED를 원격으로 안전하게 분배할 수 있는 방법을 지원하지 않기 때문에 사용자는 로그인 정보(N과 SEED)를 재설정하기 위해서 직접 관리자에게 가야한다.

둘째로 이 인증 프로토콜은 사용자가 직접 입력한 암호 구문에서 파생된 값을 사용자 비밀키로 사용하기 때문에 등록 단계나 로그인 단계에 접근하기 위해서 사용자는 암호 구문과 함께 스마트 카드의 PIN(Personal Identification Number)을 입력해야 한다.

앞서 언급된 Stolen-Verifier 공격과 Denning-Sacco 공격 그리고 낮은 사용자 편의성은 공유 비밀키 SEED와 사용자의 취약한 비밀키 K로 인해 발생한다. 그러므로 ysh-S/KEY 인증 프로토콜처럼 SEED를 강화하기보다는 사용자 비밀키 K를 강화시키는 것이 바람직하다. 사용자 비밀키 K는 임의로 생성된 후, 스마트 카드에 저장되는 방법을 통해 강화될 수 있으며 이 경우 K는 오프라인 사전 공격에 취약하지 않다.

3. 제안 프로토콜

본 장에서는 S/KEY 인증 프로토콜과 ysh-S/KEY 인증 프로토콜의 문제점을 개선하는 공개키 기반의 S/KEY 인증 프로토콜이 제안된다. 제안 프로토콜은 등록 단계와 로그인 단계, 인증 단계로 구성된다.

3.1 가 정

사용자가 서버로부터 발급 받는 스마트 카드는 서버의 공개키 KU_S를 포함한다고 가정한다. 사용자의 스마트 카드는 KU_S 이외에 임의로 생성된 D와 D의 생명주기 lf, $P_S^{-1}(H(id \oplus lf \oplus D))$를 포함한다. D와 lf, $P_S^{-1}(H(id \oplus lf \oplus D))$는 사용자 등록 단계에서 사용자를 인증하기 위해 사용되는 값들로서 초기에 사용자 등록을 위해 단 한번 적용된다. 초기 사용자 등록이후, 로그인 정보의 재설정은 로그인 단계에서 클라이언트와 서버사이에 공유된 임의값 D를 통해 적용된다. 한편, 클라이언트는 사용자 등록 단계를 시작하기 전에 사용자의 비밀키 K를 임의로 생성하고 스마트 카드에 저장해야 한다. 이처럼 제안 프로토콜은 기존 S/KEY 인증 프로토콜들과 달리 임의의 생성값 K를 사용자 비밀키로 사용하기 때문에 오프라인 사전 공격에 취약하지 않다.

3.2 등록 단계

① $C \rightarrow S$: id, $H(KU_S \oplus IP_C)$, lf, $P_S(D)$, $P_S^{-1}(H(id \oplus lf \oplus D))$

② $C \leftarrow S$: N, SEED, $H(N \oplus SEED \oplus D)$

③ $C \rightarrow S$: p_0, $H(p_0 | D)$

사용자가 처음으로 등록될 경우, 클라이언트는 스마트 카드에 저장된 D를 통해 사용자의 유효성을 증명한다. 여기서 D와 D의 생명주기 lf는 $P_S^{-1}(H(id \oplus lf \oplus D))$를 통해 검증 받을 수 있다.

클라이언트는 스마트 카드에 저장된 서버의 공개키 KU_S와 자신의 주소 IP_C를 통해 $H(KU_S \oplus IP_C)$를 계산하고 KU_S를 이용하여 D를 암호화한다. 그리고 나서 과정 ① 메시지를 생성한 후, 서버에게 전송한다. 서버가 과정 ① 메시지를 받으면 $H(KU_S \oplus IP_C)$ 값을 검증한다. 만일 $H(KU_S \oplus IP_C)$가 유효하지 않다면 클라이언트의 요청이 거부되며 서버는 이러한 방식으로 공개키 연산 $P_S(D)$와 $P_S^{-1}(H(id \oplus lf \oplus D))$에 대한 서비스 거부 공격을 회피할 수 있다.

$H(KU_S \oplus IP_C)$가 유효할 경우, 서버는 $P_S(D)$를 복호화하여 D를 추출하고 $P_S^{-1}(H(id \oplus lf \oplus D))$를 검증한다. $P_S^{-1}(H(id \oplus lf \oplus D))$가 유효하고 현재의 시간이 D의 생명주기 lf에 속한다면 서버는 D를 신뢰할 수 있다. 이와 같이 서버는 사용자가 처음으로 등록될 경우에 사용자의 스마트 카드에 저장된 D와 $P_S^{-1}(H(id \oplus lf \oplus D))$, lf를 통해서 사용자를 인증할 수 있다.

사용자가 유효할 경우, 서버는 사용자를 위하여 인증허용 횟수 N을 선택하고 임의의 값 SEED를 생성한다. 그리고 나서 $H(N \oplus SEED \oplus D)$를 SEED와 N과 함께 클라이언트에게 전송한다. 만일 $H(N \oplus SEED \oplus D)$가 유효하면 클라이언트는 서버의 개인키 KR_S 소유여부를 검증할 수 있기 때문에 서버를 성공적으로 인증할 수 있다. 또한 N과 SEED의 무결성을 검증함으로써 N과 SEED 값의 변경을 통한 서비스 거부 공격을 방지할 수 있다.

서버가 성공적으로 인증되면 클라이언트는 초기 비밀번호 $p_0 = H^N(K \oplus SEED)$를 생성하고 $H(p_0 | D)$와 함께 서버에게 전송한다. 서버는 $H(p_0 | D)$를 통해 p_0를 검증하고 유효할 경우에 사용자를 등록하고 승인 메시지를 클라이언트에게 전송하며 유효하지 않을 경우 거부 메시지를 전송한다.

승인 메시지를 받으면 클라이언트는 과정 ②에서 서버가 전송한 SEED를 스마트 카드에 저장한다. SEED는 로그인 단계에서 서비스 거부 공격에 대응하기 위해 사용된다.

초기 사용자 등록이후 SEED와 N, p_0 같은 로그인 정보를 재초기화 하기 위해서 클라이언트는 로그인 단계와 인증 단계를 통과한 후 로그인 단계에서 서버와 공유한 D를 통해 과정 ②와 ③을 진행한다. 이 경우, 과정 ①은 생략된다.

3.3 로그인 단계

① $C \rightarrow S$: id, $IP_C \oplus SEED$, $P_S(D)$

② $C \leftarrow S$: CN, SEED, $H(CN \oplus SEED \oplus D)$

③ $C \rightarrow S : p_t$

t번째 로그인을 위해 클라이언트는 임의의 값 D를 생성하고 스마트 카드에 저장된 공개키 KU_S를 통해 D를 암호화한다. 또한, $P_S(D)$를 악용한 서비스 거부 공격을 방지하기 위해 $IP_C \oplus SEED$을 계산한다. 서버가 과정 ① 메시지를 받으면 먼저 $IP_C \oplus SEED$를 검증한 후 $P_S(D)$를 복호화하여 D를 추출한다. 만일 $IP_C \oplus SEED$가 유효하지 않으면 클라이언트의 요청이 거부되며 서버는 이러한 방식으로 공개키 연산 $P_S(D)$에 대한 서비스 거부 공격을 회피할 수 있다. $IP_C \oplus SEED$를 검증한 후, 서버는 $H(CN \oplus SEED \oplus D)$를 CN(= N - t)과 SEED와 함께 클라이언트에게 전송한다. 만일 $H(CN \oplus SEED \oplus D)$가 유효하면 클라이언트는 서버의 개인키 KR_S 소유여부를 검증할 수 있기 때문에 서버를 성공적으로 인증할 수 있게 된다. 또한 CN과 SEED의 무결성을 신뢰할 수 있다. 서버가 유효할 경우, 클라이언트는 t번째 비밀번호 $p_t = H^{CN}(K \oplus SEED)$를 생성하고 서버에게 전송한다.

3.4 인증 단계

로그인 단계의 과정 ③ 메시지를 받으면 서버는 p_t의 해쉬값과 데이터베이스에 저장된 p_{t-1}을 비교한다. 만일 두 값이 동일하다면 서버가 클라이언트 인증에 성공하였음을 의미하므로 이 경우 서버는 CN과 p_t를 통해 사용자의 로그인 정보를 갱신한다.

로그인 단계에서 생성된 D는 서버와 클라이언트 사이의 안전한 메시지 전송을 위해서 세션키로 사용될 수 있으며 또한 3.2에서 언급된 바와 같이 사용자 로그인 정보의 재설정을 위해 적용될 수 있다.

4. 제안 프로토콜 분석

4.1 서버 위장 공격

오직 사용자의 스마트 카드에 저장된 KU_S에 대응하는 KR_s를 가진 서버만이 클라이언트가 암호화해서 전송한 D를 얻을 수 있으며 등록 단계나 로그인 단계의 과정 ② 메시지를 생성할 수 있다. 또한, 클라이언트는 각 세션마다 D를 임의로 생성하기 때문에 공격자는 $H(N \oplus SEED \oplus D)$ 혹은 $H(CN \oplus SEED \oplus D)$를 위조하거나 재전송할 수 없다. 이와 같이 제안 프로토콜에서 서버 위장 공격이 방지 될 수 있으며 서버의 인증은 어떠한 공유 비밀키 없이 가능하다.

4.2 Preplay 공격

비밀번호를 얻기 전에 서버는 앞서 언급한 것처럼 개인키 KRs의 소유를 증명해야 한다. 따라서 공격자는 서버의 개인키 KRs 없이 서버를 사칭할 수 없고 클라이언트를 속여서 유효한 비밀번호를 얻을 수 없다.

4.3 오프라인 사전 공격

ysh-S/KEY 인증 프로토콜은 SEED에 의존하여 사용자의 암호 구문으로부터 파생된 비밀키 K를 보호한다. 그러나 ysh-S/KEY 인증 프로토콜은 SEED를 공유 비밀키로 사용하기 때문에 SEED가 유출될 경우, 오프라인 사전 공격과 Stolen-Verifier 공격, Denning-Sacco 공격에 취약하다. 또한, 사용자는 암호 구문이외에 스마트 카드의 PIN을 입력해야 하는 불편을 겪는다.

제안 프로토콜은 ysh-S/KEY 인증 프로토콜과 달리 임의로 생성된 비밀키를 스마트 카드에 저장하여 사용하기 때문에 오프라인 사전 공격을 방지할 수 있고 사용자에게는 단지 스마트 카드를 위한 PIN만을 요구한다.

4.4 서비스 거부 공격

제안 프로토콜은 계산비용이 큰 공개키 연산을 수행하기 이전에 $H(KU_S \oplus IP_C)$ 혹은 $SEED \oplus IP_C$를 검증함으로써 서비스 거부 공격에 대응한다. 또한, 서버가 등록 단계의 과정 ②에서 $H(N \oplus SEED \oplus D)$를 생성하기 때문에 클라이언트는 $H(N \oplus SEED \oplus D)$의 검증을 통해 N과 SEED가 변경되지 않았음을 확신할 수 있으며 등록 단계의 과정 ③에서 $H(p_0 | D)$를 통해 공격자가 초기 비밀번호 p_0를 p_0'로 변경하는 것을 방지 할 수 있다. 이와 같이 제안 프로토콜은 서버와 클라이언트 사이의 비밀번호 동기화 실패를 이용한 서비스 거부 공격에 취약하지 않다.

4.5 Stolen-Verifier 공격과 Denning-Sacco 공격

제안 프로토콜은 스마트 카드에 저장된 임의로 생성된 비밀키를 사용하기 때문에 오프라인 사전 공격에 취약하지 않다. 따라서 공격자가 서버의 데이터베이스로부터 CN과 SEED, p_t 같은 사용자 로그인 정보를 훔쳤다 할지라도 훔친 정보를 통해서 오프라인 사전 공격을 하는 것이 어려우며 프로토콜 도중에 유출된 세션키 D값을 이용한 Denning-Sacco 공격도 쉽지 않다.

이처럼 제안 프로토콜은 사용자의 비밀키나 혹은 기타 비밀정보가 서버에 저장되지 않는 S/KEY 프로토콜의 장점을 충실하게 만족한다.

4.6 Man-in-the-middle 공격

제안 프로토콜은 클라이언트가 사용자의 스마트 카드에 저장된 서버의 공개키 KUs를 사용하도록 하기 때문에 거짓 공개키를 통한 Man-in-the-middle 공격을 방지할 수 있다.

마지막으로 제안 프로토콜에서 임의로 생성된 D는 서버와 클라이언트 사이의 통신을 보호하기 위한 세션키로 사용될 수 있기 때문에 이를 통해 능동공격(active attack)에 대응할 수 있다.

<표 2>는 제안 프로토콜과 S/KEY 인증 프로토콜 및 ysh-S/KEY 인증 프로토콜의 보안성과 로그인 단계에서의 성능을 비교하고 있다. 비록 제안 프로토콜이 ysh-S/KEY 인증 프로토콜에 비해 2회의 공개키 연산부담을 갖지만 Stolen-Verifier 공격과 Denning-Sacco 공격에 취약하지 않으며

이러한 특성은 제안 프로토콜을 적용한 응용 시스템들이 인터넷 대란의 발생시 서버의 붕괴로 인해 혹은 악의적인 내부 사용자에 의해 사용자 로그인 정보가 유출되는 최악의 경우에도 유출된 정보를 통한 각종 공격에 대응할 수 있도록 한다.

〈표 2〉 제안 프로토콜과 기존 연구의 비교

비교항목	S/KEY	ysh-S/KEY	제안 프로토콜
도청/재전송 공격	대응가능	대응가능	대응가능
서버위장 공격	취 약	대응가능	대응가능
Preplay 공격	취 약	대응가능	대응가능
오프라인 사전공격	취 약	대응가능	대응가능
Stolen-Verifier 공격	해당사항 없음	취 약	대응가능
Denning-Sacco 공격	취 약	취 약	대응가능
세션키 설정	×	○	○
스마트 카드 적용	×	○	○
로그인 단계에서 배타적 논리합 연산수	1	7	7
로그인 단계에서 해쉬 연산수	CN + 1	CN + 3	CN + 3
로그인 단계에서 임의변수 생성수	0	1	1
로그인 단계에서 공개키 연산수	0	0	2

5. 결 론

본 논문에서는 S/KEY 인증 프로토콜과 ysh-S/KEY 인증 프로토콜의 문제점을 개선하기 위하여 스마트 카드를 적용한 S/KEY 기반의 인증 프로토콜을 제안하였다. 제안 프로토콜은 SEED를 공유 비밀키로 적용하는 ysh-S/KEY 인증 프로토콜과 달리 공개키를 통해 S/KEY 인증 프로토콜을 개선하였다. 따라서 제안 프로토콜은 공유 비밀키 없이 서버를 인증하고 세션키를 분배 할 수 있으며 원격으로 안전하게 N과 p_0, SEED 같은 사용자 로그인 정보를 재설정 할 수 있다. 또한, 제안 프로토콜은 사용자의 암호 구문 대신 임의로 생성되어 스마트 카드에 저장된 비밀키를 적용하기 때문에 오프라인 사전 공격을 방지할 수 있고 따라서 Stolen-Verifier 공격과 Denning-Sacco 공격에 취약하지 않다. 부가적으로 제안 프로토콜은 서버와 클라이언트 사이의 메시지를 보호하기 위한 세션키를 제공하기 때문에 능동공격에 대응할 수 있다.

사용자의 편의성 측면에서 제안 프로토콜은 스마트 카드를 통해 등록과정이나 로그인 과정을 단순화하였기 때문에 사용자는 인증을 위해서 단지 스마트 카드의 PIN을 입력하기만 하면 된다.

특히, 제안 프로토콜은 사용자의 비밀키 혹은 기타 비밀정보를 서버에 저장하지 않는 S/KEY 인증 프로토콜의 장점을 충실하게 만족할 수 있기 때문에 서버의 붕괴로 인해 사용자 로그인 정보가 유출되는 최악의 경우에도 유출된 정보를 통한 각종 공격에 대응할 수 있다.

참 고 문 헌

[1] T. C. Yeh, H. Y. Shen and J. J. Hwang, "A Secure One-Time Password Authentication Scheme Using Smart Cards," IEICE Trans. Commun., Vol.E85-B, No.11, pp.2515-2518, Nov., 2002.

[2] N. Haller, C. Metz, P. Nesser and M. Straw, "A one-time password system," RFC 2289, Feb., 1998.

[3] N. Haller, "The S/KEY one-time password," RFC 1760, Feb., 1995.

[4] C. J. Mitchell and L. Chen, "Comments on the S/KEY user authentication scheme," ACM Operating Systems Review, Vol.30, No.4, pp.12-16, Oct., 1996.

[5] D. Denning and G. Sacco, "Timestamps in Key Distribution Systems," Commun. ACM, Vol.24, No.8, pp.533-536, August, 1981.

[6] S. Kim, B. Kim, S. Park and S. Yen, "Comments on Password-Based Private Key Download Protocol of NDSS '99," Electron. Lett., Vol.35, No.22, pp.1937-1938, 1999.

[7] W. C. Ku, C. M. Chen and H. L. Lee, "Cryptoanalysis of a Variant of Peyravian-Zunic's Password Authentication Scheme," IEICE Trans. Commun., Vol.E86-B, No.5, pp. 1682-1684, May, 2003.

[8] 한국정보보호진흥원, "일회용 패스워드 기술", http://www.kisa.or.kr/technology/sub4/password.htm, 1998.

유 일 선

e-mail : qjemfahr@security.co.kr

1995년 단국대학교 전산통계학과(이학사)

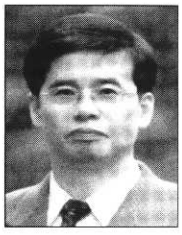
1997년 단국대학교 일반대학원 전산통계학과(이학석사)

2002년 단국대학교 일반대학원 전산통계학과(이학박사)

1997년~2000년 (주)한조엔지니어링 연구원

2000년~현재 (주)인터넷시큐리티 선임연구원

관심분야 : 침입탐지, 네트워크보안, 사용자 인증 및 접근통제



조 경 산

e-mail : kscho@dankook.ac.kr

1979년 서울대학교 전자공학과(학사)

1981년 한국과학원 전기 및 전자공학과(공학석사)

1988년 텍사스 대학교(오스틴) 전기 전산공학과(Ph.D.)

1988년~1990년 삼성전자 컴퓨터부문 책임연구원

1990년~현재 단국대학교 정보컴퓨터학부 교수

관심분야 : 컴퓨터 시스템, 컴퓨터 네트워크, 성능 분석