# 사전공격 방지를 위한 S/KEY의 정형 명세 및 검증
## (Formal Specification and Verification for S/KEY Against Dictionary Attack)

김 일 곤 † 최 진 영 ††
(Il-Gon Kim) (Jin-Young Choi)

**요 약** S/KEY 시스템은 공격자의 패스워드 재공격을 방지하기 위해 제안되었다. 하지만 S/KEY 시스템은 공격자가 자신이 가지고 있는 사전에서 패스프레이즈(passphrase)를 유추해 낼 경우, 결국 인증을 하는데 필요한 일회용 패스워드를 알아낼 수 있는 취약점을 가지고 있다. 이 논문에서는 passphrase에 대한 사전공격을 방지하기 위해 *EKE(Encrypted Key Exchange)* 프로토콜을 적용한 새로운 S/KEY 시스템을 제시한다. 그리고 새로 제안된 S/KEY 시스템의 안전성을 검증하기 위해 Casper와 CSP로 프로토콜을 명세하고, FDR 모델 체커를 이용하여 그 안전성을 검증하였다.

**키워드** : S/KEY, 일회용 패스워드, 패스프레이즈, 사전 공격, EKE 프로토콜

***Abstract*** S/KEY system was proposed to guard against intruder's password replay attack. But S/KEY system has vulnerability that if an attacker derive passphrase from his dictionary file, he can acquire one-time password required for user authentication. In this paper, we propose a correct S/KEY system mixed with EKE to solve the problem. Also, we specify a new S/KEY system with Casper and CSP, verify its secrecy and authentication requirements using FDR model checking tool.

**Key words** : S/KEY, one-time password, passphrase, dictionary attack, EKE protocol

## 1. 서 론

최근 인터넷의 급격한 확산과 더불어 전세계 어디에서나 손쉽게 인터넷에 연결된 기관의 전산망에 접근하여 필요한 정보를 획득, 이용하는 것이 가능해지게 되었으나 이에 대한 역기능으로 전산망에 대한 크래커들의 불법적인 침입과 위협이 날로 증대하고 있다. 특히 인터넷의 주요 구성요소인 유닉스 운영체제와 TCP/IP가 정보 보호 측면에서 많은 취약점을 가지고 있어 인터넷에 연결된 모든 전산망이 크게 공격자로부터 위협을 받고 있는 상태이다. 이러한 공격으로부터 시스템을 안전하기 보호하기 위한 많은 방법들이 연구되고 있으며 이러한 방법들에는 암호 알고리즘, 즉 대칭키 및 비대칭키를 이용한 여러 가지의 보안 프로토콜(예, Kerberos[1], IPsec[1], SSH[2,3]등) 및 보안장치(예, 방화벽 및 침입탐지 시스템)들이 있다. 이 논문에서 설명하고자 하는 S/KEY[2,4,5,6] 시스템은 공격자가 네트워크상에서 암호를 가로챈다 하여도, 그 이후에 로그인할 때는 그 암호를 재사용할 수 없도록 하는 방식을 사용하여 시스템의 안정성을 보호하고 있다. 하지만 점차 컴퓨터 시스템 연산능력의 향상으로 인해, 강력한 CPU를 갖춘 컴퓨터들이 등장함에 따라, 기존의 암호 프로토콜을 파괴하는데 예전보다 연산시간이 점차 짧아지고 있음을 알 수 있다. 그리고 사용자가 입력한 패스워드의 길이가 그다지 길지 않다는 부주의성을 이용하여 행해지는 이른바 사전공격(Dictionary Attack)으로 인해 발생하는 피해는 매우 크다. S/KEY 시스템 또한 공격자가 사용자의 passphrase를 유추할 경우, 인증 시 요구되는 일회용 패스워드를 알아 낼 수 있다는 위험성을 갖고 있다. 이에 따라, 이 논문에서는 S/KEY에 대한 사전공격 위험성을 사전에 제거하기 위해, 패스워드 기반 암호화 방식에서 사용되는 EKE(Encrypted Key Exchange) 프로토콜[4-6]을 사용하여 S/KEY 시스템의 안전성을 향상시키도록 설계하였다. 또한 제안된 S/KEY 시스템의 안전성을 검증하기 위해, Casper(A Compiler for Analyis of Security Protocols)[7]로 명세하고, 자동으로 CSP(Communicating Sequential Processes)[8-13] 코드를 생성한 후, FDR(Failure & Divergence Refine-

† 학생회원 : 고려대학교 컴퓨터학과
igkim@formal.korea.ac.kr
†† 종신회원 : 고려대학교 컴퓨터학과 교수
choi@formal.korea.ac.kr
논문접수 : 2003년 6월 2일
심사완료 : 2004년 6월 17일

ment)[14,15] 모델 체킹 도구를 사용하여 비밀성 및 인증 속성을 검증하였다.

이 논문의 주요 내용은 다음과 같이 요약할 수 있다.

- 사전공격 방지를 위한 새로운 S/KEY 시스템 제안
- EKE[16-18] 프로토콜의 안전성 검증 및 수정
- Casper와 CSP/FDR을 이용한 새로운 S/KEY 시스템 정형명세 및 검증

본 논문의 구성은 다음과 같이 이루어졌다. 2장에서는 정형검증 방법을 이용하여 보안 프로토콜의 안전성을 검증한 관련연구를 소개한다. 제3장 Casper와 CSP 그리고 FDR 도구에 대해 간략히 소개하고,제4장에서는 S/KEY의 간략한 인증절차 및 시스템의 보안상 문제점을 지적하고, 제5장에서는 수정된 S/KEY 시스템의 보안사항과 이 논문에서 제안하고 있는 변형된 EKE 프로토콜에 대해 설명하도록 한다. 그리고 제6장에서는 새로 제안한 S/KEY 시스템을 명세하고 검증한 결과를 보여주고, 마지막으로 제7장에서는 결론 및 향후 연구방향을 제시하고자 한다.

## 2. 관련연구

다양한 정형 명세 및 검증 도구를 이용하여 보안 프로토콜의 안전성을 명세하고 검증하기 위한 연구가 진행되어 오고 있다. 보안 프로토콜의 안전성을 검증하는 방법은 크게 두 가지로 나누어 진다. 첫번째는, 보안 로직을 이용하여 상호 호스트간의 신뢰성을 정리증명하는 방법이다. 대표적인 보안 로직으로는 BAN, GNY, SVO 로직 등을 들 수 있다. 이러한 정리증명 방법은 무한 상태 시스템을 명세 및 분석 할 수 있다는 장점이 있지만 분석방법에 자동화가 어렵고 만일 분석 결과가 거짓일 경우, 정확히 어떤 부분이 잘 못 되었는지 추론하기 어렵다는 단점이 있다. 두 번째는 모델체킹 방법이다. 이 방법은 보안 프로토콜의 동작과 속성을 정형명세 언어로 기술하고 공격자 모델을 첨가시킨 후 해당 보안 속성을 만족시키는지 검증하는 형태이다. 장점은 자동화된 정형검증도구가 지원되며 만일 해당 보안 속성을 만족하지 않는 경우는 반례 이벤트들을 보여주어 어떤 문제로 인해 보안상의 취약점이 발생하는지 명확히 알 수 있다는 것이다. 단점은 보안 프로토콜의 동작을 상세히 모델링 하였을 경우, 상태 폭발 문제가 발생 한다는 점이다. Murphi[19], NRL Protocol Analyzer[20], FDR 등과 같은 모델체킹 도구를 이용하여 보안 프로토콜의 안전성을 정형적으로 검증하기위한 연구가 진행되어 오고 있다. 그 중에서도, 특히 FDR 모델체킹 도구는 현재는 보안 프로토콜의 안전성을 검증하는 도구로 널리 인정받고 있다. 정형검증도구를 이용해서는 일반적으로 보안 프로토콜이 비밀성(secrecy)과 인증(authentication)을 만족시키는지를 검증하게 된다.

## 3. Casper, CSP와 FDR 도구

### 3.1 Casper(A Compile for the Analysis of Security Protocols) 도구

Casper는 보안 프로토콜의 명세를 보다 쉽고 간단하게 하기위해 개발되어졌다. 기존의 CSP를 이용하여 보안 프로토콜의 동작을 명세하고 분석하는 방식은 매우 복잡하여 CSP 명세 전문가 조차도 사소한 실수나 에러를 야기시킴으로써 보다 정확한 분석을 어렵게 만들었었다. 이에 따라 보안 프로토콜을 보다 쉽게 명세할 수 있고 자동으로 CSP 명세 소스를 생성할 수 있도록 개발된 도구가 바로 Casper이다. Casper를 이용하여 보안 프로토콜에서 사용되는 각종 키 타입, 동작 절차, 보안 속성, 공격자 모델 등을 8개 영역으로 나누어 명세할 수 있다.

### 3.2 CSP(Communicating Sequential Processes) 언어

CSP는 프로세스 알제브라 언어로서 병렬성을 갖는 통신 프로토콜의 동작을 효율적으로 명세하기 위해 제작되어졌다. 처음에는 일반적인 통신 프로토콜과 제어 시스템을 명세하기 위해 사용되어졌지만 점차 보안 프로토콜을 명세하기 위한 영역으로도 확대되어 오고 있다. CSP에서 제공하는 pure synchronization(|||)과 Interleaving parallelism(||) 개념을 사용하여 분산 시스템 환경에서 동작하는 클라이언트 서버 및 공격자 모델을 정형적으로 표현할 수 있다는 장점을 갖고 있다. 예를 들면, 분산시스템 환경에서 동작하는 보안 시스템은 다음과 같이 간략히 표현될 수 있다.

$$SYSTEM = CLIENT1 \;|||\; CLIENT2 \;|||\; SERVER \;||\; INTRUDER$$

### 3.3 FDR(Failure & Divergence Refinement) 모델 체킹 도구

FDR은 모델체킹 도구로서 CSP 언어로 구현된 보안 모델이 safety, authentication과 같은 보안 속성을 만족시키는지 체크하게 되며 만일 해당 속성을 만족시키지 않을 경우 반례를 보여주어 어떤 공격 시나리오가 가능한지 분석하도록 도와준다. 기본적으로 trace, failure, failure and divergence 이 세 가지 동치성 검사 방법을 제공하며 trace는 시스템의 safety, failure는 deadlock, failure and divergence는 livelock을 검사하기 위해 각각 사용되어진다. 보안 프로토콜의 안전성을 검증하기 위해서는 trace 방법을 사용하게 된다.

## 4. S/KEY

### 4.1 개요

네트워크 컴퓨팅 시스템에서 생길 수 공격유형중의 하나는 로그인 아이디, 패스워드와 같은 인증정보를 네

트워크상에서 가로채는 것이다. 이러한 정보를 일단 획득하면 후에 언제든지 이용할 수 있게 되는 것이다. S/KEY 시스템은 재공격을 방어하기 위해 Bellcore사에 의해 고안되어졌다. 일회용 패스워드 인증 시스템은 일련의 일회용 패스워드를 생성하기 위해 passphrase라는 비밀키를 입력하게 된다. 이 passphrase는 사용자가 인증을 기다리는 동안 네트워크상으로 전달될 필요가 없다. 따라서 공격자의 재공격으로부터 보호될 수 있다. S/KEY 시스템의 동작에는 두개의 중요한 동작절차가 있다. 일회용 패스워드 생성기는 서버로부터 받은 챌린지 정보와 사용자의 비밀 passphrase로부터 적절한 일회용 패스워드를 생성해야 한다. 서버는 생성기에게 적절한 생성 파라미터를 포함한 챌린지 정보를 전달해야 하며 수신한 일회용 패스워드를 확인하고 일회용 패스워드 일련 숫자를 저장해야만 한다. 일회용 패스워드 생성기는 서버로부터 수신한 seed, sequence number와 더불어 사용자의 passphrase를 입력받아 해쉬함수를 일정량 반복하여 일회용 패스워드를 생성하게 되는 것이다. 인증이 성공적으로 이루어지면 서버는 해쉬함수의 반복계수를 하나만큼 감소시키게 된다.

### 4.2 일회용 패스워드의 생성

일련의 일회용 패스워드 Pi를 해쉬함수에 반복 적용하여 생성된다. 우선, 단방향 패스워드는 사용자의 비밀 패스워드 s를 해쉬함수 F에 N번 적용하여 만들어지게 된다.

$$P_0 = F^N(s)$$

다음번에 사용되어지는 일회용 패스워드는 해쉬함수를 N-1번만큼 적용하여 만들게 되는 것이다.

$$P_1 = F^{N-1}(s)$$

이러한 과정을 통해 악의적인 공격자가 패스워드를 도청한다 하여도 시스템을 안전하게 유지할 수 있게 되는 것이다. 비밀 passphrase는 일회용 패스워드 생성기에만 보여진다. seed는 오로지 알파벳 문자들로만 구성되며 공백을 포함해서도 안 된다. sequence number와 seed는 challenge라는 하나의 큰 데이타를 형성하게 된다. challenge 정보의 구성형태는 다음과 같다.

otp-<algorithm identifier><sequence number><seed>

3개의 토큰들은 공백으로 구분되며 알고리즘 식별자는 대소문자를 구별하지만 seed는 구별하지 않는다.

### 4.3 인증 절차

S/KEY 시스템은 다음과 같은 인증 절차를 거치게 된다.

1. 사용자 A는 자신의 로그인 이름을 입력한다.
2. 시스템은 seed와 sequence를 포함한 challenge 정보를 반환한다. (예를 들어 seed는 latour1이고 sequence는 55라고 가정)
3. 사용자 A가 55와 latour1을 생성기에 입력하면, 프롬프트는 사용자의 passphrase를 입력하도록 요구한다.
4. 사용자 A는 passphrase를 입력하게 되고 생성기는 55번째 일회용 패스워드를 생성 화면에 보여준다.
5. 사용자 A는 생성된 일회용 패스워드를 해당 시스템에 입력하여 정상적으로 접속하게 된다. 다음 번 사용자 A가 시스템에 접속하고자 하면, 사용자 A는 54번째 일회용 패스워드를 생성하여 시스템에 접속해야 한다.

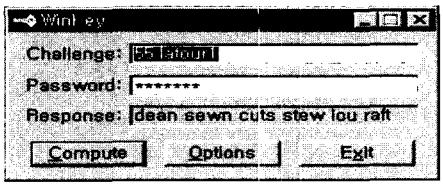


그림 1 일회용 패스워드 생성기

위의 그림 1은 사용자가 S/KEY 시스템을 이용하여 인증을 받는데 사용하는 윈도우용 일회용 패스워드 생성기를 보여주고 있다.

### 4.4 S/KEY 일회용 패스워드의 문제점

S/KEY 시스템에서 생길 수 있는 문제점으로는 클라이언트와 서버간의 challenge-response 프로토콜[5]상에서 challenge 정보를 추측해 낼 수 있는 가능성이 존재한다는 것이다. S/KEY에서 사용하는 challenge-response 스킴은 ISO/IEC 9728-4의 5.1.2항에서 기술하고 있으며 이 프로토콜의 기본적인 방식은 B가 A를 확인하기 위해 다음과 같은 메시지 교환을 갖는다.

1. B는 A에게 challenge R을 보내게 된다.
2. A는 B에게 fK(R) 에 해당하는 response를 보내게 된다. (K: 비밀키, f: 암호화 체크 함수)

이러한 인증 프로토콜의 요구사항으로서 ISO/IEC 9798-4에서는 challenge R이 제3자에 의해 예측할 수 없어야함을 명시하고 있다. 그러나 S/KEY 시스템에서 전달되어지는 일련의 seed와 count값은 동일한 값을 취해 이러한 값은 예측할 수 있다. 이러한 사항들은 다음과 같은 문제점을 야기시킬 수 있다. 처음 이러한 사실을 인지한 사람은 Gong[21]이며, 그에 따르면 만일 challenge-response프로토콜에서 challenge를 예측할 수 있고, challenge가 암호화 되어 있지 않다면 'suppress-replay attack'과 같은 종류의 공격 가능성이 있

다. 공격이 행해지는 일반적인 시나리오는 다음과 같다. 사용자 C는 B에게 사용자 A인 것처럼 위장한다. 만일 C가 B에 의해 A에 전달되어지는 challenge R을 가로채거나 예측할 수 있다면 C는 A로 위장하여 B에게 인증요청을 하게 된다. 즉, A가 B에게 전달하는 response까지 가로챈다면 그 위험성은 더욱 커지게 된다. 이러한 시나리오를 S/KEY 시스템에 적용해 보면 공격자는 서버에서 전달되는 seed와 sequence number를 포함하는 challenge를 유추하거나 가로챈 후, passphrase를 사전에 있는 단어들과 비교하여 결국에는 $f_K(R)$인, response까지 알아낼 수 있다는 것이다. 기존의 S/KEY 시스템은 다음과 같이 간단히 표현할 수 있다.

Message 1. A -> B : A

Message 2. B -> A : challenge

Message 3. A -> B : response

($challenge = seed + sequencenumber$,

$response = f_{Hash}(seed, sequence\ number, passphrase)$)

기본적으로 해쉬 함수의 특성상, $f_{Hash}$(seed, sequence number, passphrase)를 알게 된다고 해도, 각각의 아규먼트인 seed, sequence number, passphrase의 값을 도출해 낼 수는 없다. 그러나 만일 3개의 아규먼트중에서 2개의 아규먼트 값을 알게 된다면, 나머지 한 개의 값을 유추해 낼 수는 있다. 즉, 공격자는 B 호스트로부터 A 호스트로 전달되는 seed와 sequence number를 가로채고 난 후 공격자의 지식정보에 A와 B의 호스트 식별자와 더불어 이 정보들을 추가시키게 된다. 공격자가 갖고 있는 지식정보는 IntruderKnowledge = {A, B, M, $N_m$}으로 나타낼 수 있다. A와 B는 정상적인 호스트의 식별자를 의미한다. 좀더 쉽게 말하면, 호스트의 IP 주소를 나타낸다는 의미이다. M은 악의적인 공격자 호스트의 식별자를 나타낸다. 그리고 $N_m$은 M호스트가 생성하는 임의의 난수를 가리킨다. 각 호스트의 주소는 외부에 공개되어 있기 때문에 공격자가 기본적으로 알고 있다고 가정하였다. Message 2를 수행하고 나게 되면, 공격자는 IntruderKnowledge′ = {A, B, M, $N_m$, seed, sequence number}의 지식정보를 갖게 된다. 이제 공격자는 적합한 사용자로 인증받기 위해, passphrase 정보만 유추해 내면 되는 것이다. $f_{Hash}$(seed, sequence number, passphrase)에서 passphrase의 추측을 가능하게 하는 원인은 크게 세 가지로 나누어 볼 수 있다.

첫째, 다양한 플랫폼 환경에서 동작하는 사전 공격 도구들을 손쉽게 인터넷상에서 구할 수 있다. 예를 들어, John the Ripper와 Crack 5.0[22]은 일반적으로 많이 사용되는 사전공격 도구이다.

둘째, 패스워드 추측이 쉽다. 다시 말해서, 일반 사용자들은 처음 패스워드를 설정할 때, 자신들이 기억하기 용이한 이름, 일반 단어명, 생일 등 복잡도가 낮은 패스워드를 선택하는 경향이 많다. 한 대학의 실험결과에 따르면 학생들의 패스워드중 4분의 1이상이 실제 사전공격을 통해 깨지게 되었다고 한다.

셋째, 인터넷상에서 무수히 많은 양의 사전단어 파일들을 다운로드 할 수 있다. 사전 단어가 많으면 많을 수록 패스워드를 크래킹하기는 쉽다. [7,9,10]에서 사용하는 표현식을 이용하여 사전공격 가능성을 아래와 같이 간단히 나타낼 수 있다.

$$IntruderKnowledge = \{Anne, Bob, Mallory, N_m, seed, sequence\ number\} \cup \{passphrase\} \vdash f_{Hash}(seed, sequence\ number, passphrase)$$

## 5. 수정된 S/KEY 시스템

S/KEY 시스템의 보안성을 향상시키기 위해서는 두 가지 측면의 위험 요소 모두를 고려해야만 한다.

첫 번째는 온라인상에서 발생하는 공격이다. 클라이언트와 서버간의 안전한 데이타 통신을 하기 위해서는, 무엇보다 안전한 통신 채널이 요구된다. IPSec, SSL, SSH, VPN[1]등과 같은 암호화 통신 지원 장비나 애플리케이션을 사용하여 상호간의 안전한 키 전달을 하는 것은 매우 기본적이면서도 필수적인 사항이라 할 수 있겠다. 그리고 만일 사용자의 패스워드 입력이 3회 이상 잘 못 되었을 시 접속을 차단시키는 것도 간단한 수준의 대비책이라 할 수 있다.

두 번째는 오프라인상에서 발생하는 공격이다. 사전공격의 경우 암호화된 패스워드를 공격자가 가로채어 오프라인상에서 자신의 컴퓨터에서 사전공격용 도구를 사용하여 본래 패스워드를 추측하게 된다. 따라서 본 논문에서는 앞으로 오프라인 상에서의 공격방지에 초점을 맞추어 설명하도록 하겠다. passphrase에 대한 사전공격을 방지하기 위해서는 우선 passphrase가 추측할 수 없는 난수이어야 한다. 새로 제안된 S/KEY 시스템에서 passphrase는 서버에 의해 난수로 클라이언트에 전달되게 된다. 난수 passphrase의 안전한 전송을 위해 수정된 EKE 프로토콜을 사용하였다. EKE는 본래 Belovin과 Merritt에 의해 개발되었으며 Diffie Hellman 키 교환 방식과 유사한 형태로 이루어져 있다. EKE는 패스워드 기반 인증체계에서 패스워드의 안전한 전송을 보장하기 위해 제안된 방법이다. 처음 제안된 이후, SPEKE, EKE, A-EKE, DH-EKE[18]와 같은 변형 형태들이 등장하였다. 본 논문에서는 EKE과 S/KEY 시스템의 challenge-response 프로토콜을 혼합 하여, 새

로운 S/KEY 시스템을 제안하였다. 다른 확장된 형태의 EKE 프로토콜은 매우 복잡하여 보안성을 만족시킬 수 있는 있지만, 그 만큼 네트워크 트래픽 속도 및 성능을 감속시킬 수 있기 때문에 challenge-response 프로토콜과의 변형생성 과정에서 배제하였다. 또한 기존에 제안된 EKE 프로토콜은 취약점을 갖고 있기 때문에 새롭게 변형한 EKE 프로토콜을 사용하였다. 다음은 기존의 EKE 프로토콜에 대한 수행동작을 보여주고 있다.

- EKE(Encrypted Key Exchange ) 프로토콜
Message 1: A -> B : A, $\{PK(a)\}K_{ab}$
Message 2: B -> A : $\{\{RK_{ab}\}PK(a)\}K_{ab}$
Message 3: A -> B : $\{N_a\}RK_{ab}$
Message 4: B -> A : $\{N_a, N_b\}RK_{ab}$
Message 5: A -> B : $\{N_b\}RK_{ab}$

PK(a)는 A 호스트의 공개키이며 $K_{ab}$는 A와 B 호스트가 함께 공유하고 있는 세션키이고 $RK_{ab}$는 A와 B호스트간에 임의적으로 생성되는 난수 세션키를 나타낸다. $N_a$는 A 호스트가 생성한 난수이며 $N_b$는 B 호스트가 생성한 난수를 의미한다. Message 3를 수행하고 나면 위의 프로토콜은 Needham Schroeder 프로토콜에서 발생한 것과 유사한 형태의 man-in-the-middle 공격이 발생된다. FDR 모델체킹 도구를 사용하여 보안성을 검증한 결과, 그림 2와 같은 공격 시나리오를 발견할 수 있었다. 그림 2에서 공격자 모델은 A와 B호스트 사이에서 A가 B에 인증을 받기 위해 전달하는 메시지를 가로채거나 혹은 자신이 정당한 사용자 B처럼 위장하는 과정을 나타내고 있다. 각 메시지 앞에 나타난 번호는 인증을 받기 위한 인증 절차 순서를 의미한다. 예를 들어 설명하면 A는 우선 B에게 인증을 받기 위해 1번 메시지인 A, $\{PK(a)\}K_{ab}$를 B에게 전달한다. 그런데, 공격자는 이 메시지를 스니핑한 후, 동일한 메시지를 B에게 전달한다. B의 입장에서는 A가 보낸 메시지인줄 알고 3번째 메시지 $\{RK_{ab}\}PK(a)K_{ab}$를 공격자에게 전송한다. 공격자는 자신이 B인 것처럼 가장하여 A에게 3번째 메시지를 전달하게 된다. 그런 다음 B가 자신이 보낸 메시지를 잘 받은 줄 인식하고 5번째 메시지 $\{N_a\}RK_{ab}$를 B에게 보내려고 한다. 하지만, 이 메시지는 공격자가 중간에서 가로채게 되고, 공격자는 5번째 메시지를 B에게 자신이 A것처럼 가장하여 전달한다. 이제 B는 A가 보낸 $\{N_a\}RK_{ab}$ 정보를 바탕으로 A를 신뢰하게 되며 자신의 중요 정보 $N_b$를 담아 7번째 메시지 $\{N_a, N_b\}RK_{ab}$를 공격자에게 보내게 되고 공격자는 다시 이 메시지를 A에게 전달하게 되면 결국 B에게 인증 받는데 필요한 메시지 $\{N_b\}RK_{ab}$를 B인줄 알고 공격자에게 보내게 되면, 공격자는 10번 메시지를 B에게 보내게 되고, B는 정상적인 사용자 A 인줄 알고 인증을 허가한다.

EKE 프로토콜의 이런 취약점을 막기 위해서 Message 2와 Message 4를 아래와 같이 변경하였다.

Message 2: B -> A : $\{\{RK_{ab}\}PK(a), B\}K_{ab}$

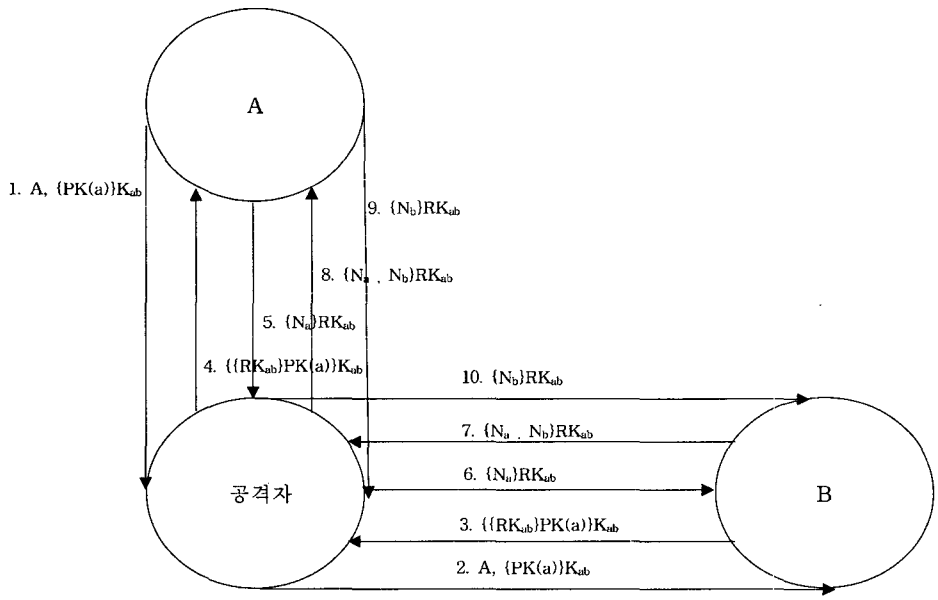


그림 2 EKE 프로토콜 공격 시나리오

Message 4: B -> A : {$N_a$, $N_b$, B}$RK_{ab}$

B 호스트의 식별자를 암호화된 메시지안에 첨가시킴으로써, 공격자의 스푸핑 공격을 방지하도록 하였다. 이 방법은 매우 간단하면서도 효율적인 보안 대처방안이라 할 수 있다. 다음은 이 논문에서 새로 제안하고 있는 수정된 S/KEY 시스템의 메시지 교환 절차를 보여주고 있다.

- 수정된 S/KEY 시스템
Message 1: A -> B : A, {PK(a)}$K_{ab}$
Message 2: B -> A : {B, Challenge}PK(a)
Message 3: A -> B : {$N_a$}$K_{ab}$
Message 4: B -> A : {B, $N_a$, $Passphrase_{ab}$}$K_{ab}$
Message 5: A -> B : Response

A 호스트는 인증을 요청하는 사용자 시스템을, B 호스트는 서비스를 제공하는 S/KEY 시스템을 가리킨다. PK(a)는 A 호스트의 공개키이며 $K_{ab}$는 A와 B 호스트가 공유하고 있는 비밀 패스워드이고 Challenge는 seed와 sequence number를 포함하는 임의 난수를 의미한다. $N_a$는 A 호스트가 생성한 임의 난수이다. 그리고, Response는 A 호스트에서 생성한 일회용 패스워드를 나타낸다. 기존의 S/KEY 시스템과의 차이점은 우선 기본적으로 A의 공개키 PK(a)와 A와 B의 비밀 패스워드 $K_{ab}$가 첨가되었고 passphrase가 A호스트에서만 사용되던 방식과는 달리, B 호스트로 전달되어 진다는 점이다. 수정된 EKE 프로토콜에서는 $RK_{ab}$가 임의의 패스워드였듯이 새로 제안된 S/KEY 시스템에서의 $Passphrase_{ab}$는 사전공격에 강한 임의의 패스워드를 사용하고 있다. 즉, $RK_{ab}$와 $Passphrase_{ab}$는 동일한 역할을 하는 키라고 볼 수 있다. 수정된 EKE 프로토콜에서는 Message 3, 4, 5를 통해 상호 인증단계를 수행하였다. 빈번한 메시지 암호화는 네크워크 전송 속도를 저하시키는 요인으로 작용할 수 있기 때문에. 새로운 S/KEY 시스템에서는 이 세 가지 메시지 전달 단계를 Message 4와 Message 5 단계를 이용해 간소화 시켰다. Message 1에서 A 호스트는 B 호스트에 인증을 요청하기 위해서 우선 자신의 공개키를 $K_{ab}$로 암호화 하여 전송하게 된다. Message 2에서, B 호스트는 자신의 호스트 식별자 B와 Challenge를 A 호스트의 공개키로 암호화 하여 전송하게 된다. Message 3에서는 A 호스트에 대한 1차 인증 과정으로써, A 호스트가 생성한 임의 난수 $N_a$를 비밀키 $K_{ab}$로 암호화하여 B 호스트에 전송한다. Message 4에서 B 호스트는 A 호스트로부터 수신한 임의 난수 $N_a$와 더불어 자신의 식별자 B와 난수 $Passphraseab$를 $K_{ab}$로 암호화 하여 보내게 된다. 난수를 사용하는 이유는 공격자의 사전공격을 방지하기 위함이고 서버가 난수를 송신하는 이유는 사용자의 부주의한 유추 가능한 passphrase의 생성을 막기 위해서이다. 사전공격을 방지하기 위해서, 난수의 길이는 최소한 8개의 문자보다 커야만 한다. Message 5에서, A 호스트는 일회용 패스워드를 생성하여 B 호스트에 사용자 인증을 받게 된다.

## 6. 수정된 S/KEY 시스템 명세 및 검증

4장에서 설명한 메시지 교환을 토대로 Casper를 이용하여 8개의 영역으로 나누어 각각 명세하였다. 본 논문에서는 페이지 사정상, Casper 명세중에서 #Free variables, #Protocol description, #Specification 3개 부분만 설명하도록 하겠다. 다음은 각종 키 타입을 정의하고 있는 #Free variables 영역을 보여주고 있다.

#Free variables
a, b : Agent
challenge, na, passphrase : Nonce
passwd, otp: SessionKey
ServerKey : Agent -> ServerKeys
F : HashFunction
InverseKeys = (passwd, passwd),(otp, otp), (ServerKey, ServerKey), (F, F)

a, b는 Agent 타입을 갖게 되며, 호스트를 의미한다. challenge, na와 passphrase는 난수 타입을 갖고 paswd, otp는 각각 패스워드와 일회용 패스워드를 의미하며 세션키 타입을 갖는다고 정의하였다. ServerKey는 공개키 및 개인키를 표현하기 위해 사용되는 타입이며 F는 해쉬함수를 나타낸다. InverseKeys는 암호화하는데 사용되는 키와 복호화 하는데 사용되는 키 쌍을 나타내주고 있다. 다음은 프로토콜의 동작을 명세하기 위해 사용되는 #Protocol description 영역을 보여주고 있다.

#Protocol description
0. -> a : b
1. a -> b : a, {ServerKey(a)}{passwd}
2. b -> a : {b, challenge}{ServerKey(a)}
3. a -> b : {na}{passwd}
4. b -> a : {b,passphrase}{passwd}
5. a -> b : F(otp)

첫 번째, 0번 메시지에서는 a 호스트가 b 호스트와 통신해야 함을 명시적으로 지정해 주었다. 나머지 부분의 동작은 이미 4장에서 언급하였기 때문에 생략하도록 하겠다. 다음은 보안 프로토콜의 안전성을 검증하기 위해 사용되는 보안 속성을 기술하는 #Specification 영역이다.

#Specification
Secret(a, na, [b])
Secret(b, challenge, [a])
Agreement(a, b, [na, challenge, otp])
Agreement(b, a, [na, challenge, otp])

Secret는 비밀성을 나타내며 Agreement는 인증속성을 표현한다. Secret(a, na, [b])는 'a 호스트는 a가 생성한 난수 na가 b 호스트 하고만 공유하고 있다고 믿는다'고 해석된다. Secret(b, challenge, [a])는 'b 호스트는 challenge가 a 호스트 하고만 공유하고 있다고 믿는다. Agreement(a, b, [na, challennge, otp])는 a 호스트는 b 호스트에 na, challenge와 otp 정보에 의해 인증을 받는다'는 의미이다. Agreement(b, a, [na, challennge, otp])는 'b 호스트는 a 호스트에 na, challenge와 otp 정보에 의해 인증을 받는다'는 의미이다. 다시 말해서, 만일 공격자가 중간에 위치해서 중요한 *na*, *challenge*와 *otp* 정보를 모두 가로채게 되면 위의 두 가지 보안 속성은 모두 깨지게 된다.

공격자 모델을 형성하기 위해서는 우선, 공격자가 가지고 있는 기본지식을 단순히 정의해 주기만 하면 된다. 3장의 마지막 부분에서 언급한 공격자 정보지식 IntruderKnowledge를 바탕으로 Casper에 명세하면 된다. Casper의 컴파일러 기능을 이용하여 CSP 코드를 생성하면, A와 B 그리고 공격자 모델은 다음과 같이 간단히 변형되게 된다.

```
-- The intruder
INTRUDER = (INTRUDER_1 [|{|intercept.Mallory|}
|] STOP) ||| SAY_KNOWN
-- Initiator A
INITIATOR_Alice = INITIATOR(Alice, Na, Passwd,
OTP)
```

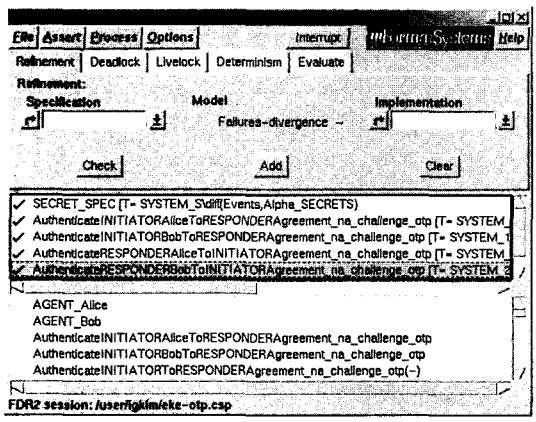


그림 3 FDR 도구를 이용한 수정된 EKE 프로토콜 검증 결과

```
-- Initiator B
RESPONDER_Bob = RESPONDER(Bob, Challenge,
Passphrase, Passwd, OTP)
```

Casper 명세로부터 자동으로 생성된 CSP 코드를 FDR 모델체킹 도구에 입력한 후, 앞에서 기술한 비밀성 및 인증속성을 만족하는지 체킹하였다. FDR에서 사용한 검증 방법은 명세 모델과 구현 모델의 trace 동치성을 검사하였다. 명세 모델은 보다 상위의 추상적 모델로, 보안 속성을 나타내는 모델로 표현되었고, 구현 모델은 정상적인 사용자 모델에 공격자 모델이 첨가된 상세 모델을 의미한다. 만일 이 두 모델의 traces 이벤트들이 모두 같다면, 공격자는 정상적인 호스트들의 메시지 교환에 어떠한 악의적인 영향도 미치지 않음을 의미하게 된다. 앞에서 언급한 4가지 보안 속성을 FDR 도구를 이용하여 모두 체킹해 본 결과, 본 논문에서 제안하고 있는 EKE 프로토콜이 공격자의 man-in-the-middle 공격으로부터 안전함을 확인할 수 있었다. 그림 3은 FDR 도구를 이용하여 수정된 EKE 프로토콜을 검증한 화면을 보여주고 있다.

## 7. 결 론

S/KEY 시스템은 악의적인 공격자로부터의 재공격을 방지하기 위해, 일회용 패스워드를 사용함으로써 사용자 인증의 보안성을 향상시켰다. 하지만 컴퓨터의 연산속도가 점차 크게 향상되어짐에 따라, 암호화 알고리즘의 안전도는 완벽성을 장담할 수 없는 실정이다. 이에 따라, S/KEY 시스템 몇 가지 취약점들이 하나씩 밝혀졌다. 본 논문에서는 S/KEY 시스템의 보안성을 향상시키고자 수정된 EKE 프로토콜을 사용하여, 보다 안전한 새로운 S/KEY 시스템을 제안하였으며, 보안 프로토콜 설계단계에서부터 정형적으로 모델을 명세하고, 해당 보안 속성을 만족시키는지 검사한 결과, 그 안전성을 확인할 수 있었다. 정형적 방법론의 사용은 보안 프로토콜의 안전성 검증 뿐만 아니라, 고등급의 보안 시스템을 개발하는데 중요한 핵심 기술이다. 이에 따라 본 논문은 이론적인 모델이 아닌 실제 사용되는 보안 프로토콜을 정형적으로 명세하고 검증함으로써, 정형적 개발 방법론에 익숙치 않은 개발자들에게는 활용가이드의 역할을 수행하였다. 향후 연구방향으로는 보안 프로토콜을 명세하고 검증하는 범위를 벗어나, 전체 보안 시스템을 검증하기 위해 보안 시스템 모델과 공격자 모델을 정형적으로 명세하고 검증하는 연구를 진행하고자 한다.

## 참 고 문 헌

[1] W. Stallings, Cryptography and Network Security: Principles and Practice, Prentice-Hall, 1998.

[2] S. Mann and E. L. Mitchell, Linux System Security: The Administrator's Guide to Open Source Security Tools, Prentce-Hall, 2000.
[3] D. Song, D. Wagner and X. Tian, "Timing Analysis of Keystrokes and SSH Timing Attacks," 10th USENIX Security Symposium, pp. 337-352, Washington, 2001.
[4] N. M. Haller, "The S/KeyTM One-Time Password System," Proceedings of the Symposium on Network and Distributed System Security, pp. 151-157, San Diego CA, 1994.
[5] N. Haller, "The S/Key One-Time Password System," 1995.
[6] L. Chen and C. J. Mitchell, "Comments on the S/KEY User Authentication Scheme," ACM SIGOPS Operating Systems Review, Vol. 30, Issue 4. pp. 12-16, 1996.
[7] G. Lowe, "Casper: A Compiler for the Analysis of Security Protocols," 10th IEEE Computer Security Foundations Workshop, pp. 18-30, Massachusetts, 1997.
[8] C. A. R. Hoare, Communicating Sequential Processes. Prentice-Hall, 1985.
[9] G. Lowe and A. W. Roscoe, "Using CSP to Detect Errors in the TMN Protocol," IEEE Transactions in Software Engineering, Vol. 23, No. 10, pp. 659-669, 1997.
[10] G. Lowe, "Analysing Protocols Subject to Guessing Attacks," Proceedings of the Workshop on Issues in the Theory of Security (WITS '02), pp. 53-84, 2002.
[11] P. Y. A. Ryan and S. A. Schneider, Modelling and Analysis of Security Protocols: the CSP Approach, Addison-Wesley, 2001.
[12] S. Schneider, "Security Properties and CSP," IEEE Symp. Security and Privacy, pp. 147-187, Oakland, 1996.
[13] S. Schnedier, "Verifying Authentication Protocols with CSP," 10th IEEE Computer Security Foundations Workshop, pp. 3-17, Massachusetts, 1997.
[14] Formal Systems(Europe) Ltd, Failure Divergence Refinement-FDR2 User Manual, 1999.
[15] G. Lowe, "Breaking and Fixing the Needham-Schroeder Public-Key Protocol using FDR," Proceedings of TACAS, pp. 147-166, Germany, 1996.
[16] M. Bellovin and M. Merritt, "Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks," AT&T Bell Laboratories, Proceedings of the IEEE Computer Society Conference on Research in Security and Privacy, pp. 72-84, Oakland, 1992.
[17] J. Clark and J. Jacob, "A Survey of Authentication Protocol Literature: Version 1.0," Available via http://www.win.tue.nl/~ecss/downloads/clarkjacob.pdf, 1997.
[18] D. Jablon, "Strong Password-Only Authenticated Key Exchange," ACM Computer Communications Review, pp. 5-26, 1996.
[19] C. Mitchell, "Automated Analysis of Cryptographic Protocols Using Murphi," IEEE Symposium on Security and Privacy, pp. 141-153, Oakland, 1997.
[20] C. Meadows, "The NRL Protocol Analyzer : An Overview," Journal of Logic Programming, Vol. 26, No. 2, pp. 113-131, 1994.
[19] L. Gong, "Java Security: Present and Near Future," IEEE Micro, Vol. 17 No. 3, pp. 14-19, 1997.
[20] B. Hatch, J. Lee and G. Kurtz, Hacking Linux Exposed, McGraw-Hill, 2001.

김 일 곤
정보과학회논문지 : 소프트웨어 및 응용
제 31 권 제 3 호 참조

최 진 영
1982년 서울대학교 컴퓨터공학과 졸업 1986년 Drexel University Dept. of Mathematics and Computer Science 석사. 1993년 University of Pennsylvania Dept. of Computer and Information Science 박사. 1993년~1996년 Research Associate, University of Pennsylvania. 1996년~1999년 고려대학교 컴퓨터학과 조교수. 1999년~2003년 고려대학교 컴퓨터학과 부교수. 2003년~현재 고려대학교 컴퓨터학과 교수. 관심분야는 컴퓨터이론, 정형기법(정형 명세, Formal verification), 실시간 시스템, 프로세스 알 제브라, 소프트웨어 공학