情報保護學會論文誌
第15券 第1號, 2005. 2

보안 프로토콜의 안전성 분석을 위한 정형적 방법론

김 일 곤,[1†] 전 철 욱,[1] 김 현 석,[1] 최 진 영,[1‡] 강 인 혜[2]
[1]고려대학교, [2]서울시립대학교

Formal Methodology for Safety Analysis of Security Protocols

Il-Gon Kim,[1†] Chul-Wuk Jeon,[1] Hyun-Seok Kim,[1] Jin-Young Choi,[1‡] In-Hye Kang[2]
[1]Korea University, [2]University of Seoul

요 약

유·무선 네트워크의 활성화와 더불어 중요 자원 혹은 사용자 정보 보호를 위해 다양한 보안프로토콜들이 개발되고 있다. 하지만, 대부분의 많은 보안 프로토콜들은 개발된 후, 시간이 지남에 따라 점차 보안 취약점들이 하나둘 발견되고 있다. 본 논문에서는 안전한 보안프로토콜을 개발하기 위해, EKE 프로토콜 분석 예제를 통해, 설계단계에서 보안 프로토콜의 안전성을 검증하는 정형적 방법론에 대해 소개하고, 정형적 검증 방법론의 실효성을 보이기 위해, Casper 및 FDR 도구를 이용하여 BCY 프로토콜의 안전성을 분석한 후, 보안성을 향상시킨 새로운 BCY 프로토콜을 제안한다.

ABSTRACT

With the development of wire and wireless based networks, a various security protocols have been proposed to protect important resources and user information against attackers. However, many security protocols have found only to be later vulnerable to attacks. In this paper, we introduce the formal methodology to verify the safety of security protocols in the design phase, and we take advantage of the formal methodology which uses Casper/CSP and FDR tools by introducing the verification example of EKE protocol and BCY protocol. Lastly, we propose a new BCY protocol after verifying it's safety.

Keywords : *security protocol, CSP, Casper, FDR, EKE, BCY*

Ⅰ. 서 론

유·무선 네트워크의 활성화와 더불어 중요 자원 혹은 사용자 정보 보호를 위해 다양한 보안프로토콜들이 개발되고 있다. 하지만, 대부분의 많은 보안 프로토콜들은 개발된 후, 시간이 지남에 따라 점차 보안 취약점들이 하나둘 발견되고 있다[1-2]. 예를 들어, Needham-Schroeder 프로토콜은 제안된지, 17년이 지나고서야 보안 취약점을 발견할 수 있었다[3]. 그 만큼 안전한 보안프로토콜을 설계하는 것은 매우 어려우면서도 중요한 연구 분야이다.

보안 프로토콜의 안전성은 크게 두 가지 방면에서 생각할 수 있다. 첫 번째는 보안 프로토콜 설계상의 안전성이다. 정형기법을 이용하여, 보안 프로토콜 설계상에서 생길 수 있는 보안 취약점을 검증하기 위해 다양한 방법론과 도구들이 개발되고 있다[4]. 둘째는 보안 프로토콜 구현상의 안정성이다. 보안 프로토콜의 설계상에 아무런 문제가 없더라고, 구현자의 실수나 혹은 프로그래밍 언어 자체의 보안 취약점으로 인해 보안상 위험이 발생할 수 있게 된다. 예를 들어,

접수일 : 2004년 9월 22일 ; 채택일 : 2005년 2월 11일
* 본 연구는 고려대학교 특별연구비에 의해 수행되었음
† 주저자, igkim@formal.korea.ac.kr
‡ 교신저자, choi@formal.korea.ac.kr

OpenSSH 프로그램에서 발견된 버퍼 오버플로우 공격[5]은 보안 프로토콜 설계상의 문제가 아닌, 프로그래밍 언어의 문제로 인식할 수 있다. 두 번째 문제를 해결하기 위해서는 궁극적으로 보다 보안상 안전한 프로그래밍 언어를 개발하는 것이 중요한 연구과제로 남아 있다.

본 논문에서는 안전한 보안프로토콜을 설계하기 위한 연구내용에 초점을 맞추어, 지금까지 진행되어 오고 있는 보안 프로토콜 정형적 설계 및 검증 방법론에 대해 소개하도록 하겠다. 특히, 그중에서도 Casper/CSP 및 FDR 도구를 이용한 검증 방법론에 대해 상세히 언급하고자 한다

본 논문의 구성은 다음과 같다. 제 2 장에서는 보안 프로토콜의 안전성을 검증하기 위한 정형적 방법론에 대해 소개한다. 그리고 Casper/CSP 및 FDR 도구를 이용한 EKE 프로토콜 안전성 분석 예제를 통해, 정형적 설계 및 검증 방법론에 대한 이해를 높이고자 한다. 또한 무선 이동 네트워크 환경에서 사용되는 BCY 프로토콜을 명세하고 검증한 사항을 언급한다. 마지막으로 결론을 맺고자 한다.

II. 본 론

1. 보안 프로토콜 검증 방법

보안 프로토콜의 안전성을 검증하기 위한 방법은 크게 모델체킹과 정리증명 방법으로 나누어진다. 아래 부분에서는 모델체킹 및 정리증명 방법에 대한 장·단점과 더불어, 대표적인 보안 프로토콜 검증도구들을 소개하고자 한다.

1.1 모델체킹

모델체킹은 시스템에 대한 유한 상태 모델과 검사하고자 하는 속성이 논리적 정형성을 가지고 만들어져 주어졌을 때, 이 시스템이 해당 속성을 만족하는지를 자동적으로 검사하는 정형 검증 기법의 일종이다.

즉, 모델체킹은 상태 탐색(state exploration)을 기반으로 하며, 상태 전이 시스템(state transition system)과 속성이 주어지면 , 모델 체킹 알고리즘은 주어진 시스템이 검증하고자 하는 속성을 만족하는지를 알아보기 위해서 전체 상태공간(state space)을 검사한다.

모델체킹의 장점은 자동화 검증도구가 지원된다는 사실이다. 즉, 사용자가 시스템의 모델을 입력하고 요구 사항 명세를 나타내는 속성들을 입력하면 도구는 자동적으로 모델의 상태를 검사하여, 속성을 만족하지 못하는 경우, 반례를 보여주어 모델의 어느 부분이 잘못되었는지를 쉽게 알 수 있게 해 준다는 것이다.

하지만 모델의 상태가 커질 경우에는 상태폭발문제로 인해, 검증할 수 없게 되는 단점이 있다. 더구나, 유한 상태 시스템에 국한된 보안 프로토콜 모델상에서 보안 속성을 위반하는 반례를 찾지 못하였을 경우에는 보안 프로토콜의 안전성 여부를 논할 수 없게 된다. 하지만, 대부분의 보안 프로토콜의 경우, 모델의 상태가 그다지 크지 않기 때문에, 보안 프로토콜의 안전성 분석 방법론으로 매우 활발히 연구되고 있다.

대표적인 도구로는 Oxford 대학의 FDR[6], CMU 대학의 SMV[7]과 Bell 연구소에서 개발한 SPIN[8] 등이 있다.

1.2 정리증명

정리 증명은 수학적인 논리식을 사용하여 시스템과 시스템 상에서 요구되는 특성들을 표현하는 기술이며, 주어진 시스템의 공리로부터 특성에 대한 증명을 사용하여 논리식의 진위를 발견해 내는 과정이다. 증명의 각 단계들은 공리(axiom)와 규칙(rule)에 의존하고 파생된 정의들은 중간 단계의 보조 정리(lemma)로 이용한다.

정리 증명의 장점은 모델 검사의 단점인 무한 상태에 대해서 직접적으로 다룰 수 있다는 점이다. 무한 공간 집합에 대한 증명을 하기 위해 구조적 귀납법(Structural Induction)등과 같은 기술을 사용하기 때문에 모델 검사에서 발생하는 상태 폭발의 문제를 해결할 수 있다. 반면에 단점은 규칙을 적용하는 부분에서는 기계의 도움을 받을 수 있어도 정의 부분에서 상호 작용이 필요하기 때문에 숙련된 논리 전문가의 도움이 필요하다는 문제점이 있다.

보안프로토콜의 안전성을 분석하기 위한 언어로는 BAN[9], GNY[10] 등이 개발되어졌다. 보안 프로토콜 상호관계 신뢰관계를 모델링 하고, 이를 증명하는 방식의 형태를 취하고 있다. 일반적으로 'Belief Logic' 으로 일컬어진다. Belief Logic'을 이용한 보안 프로토콜 설계 및 검증 방법은 다음과 같은 절차로 이루어진다.

첫째, 보안 프로토콜의 메시지를 정형화 한다.
둘째, 최초의 가정을 명세한다.
셋째, 보안프로토콜의 목적(goal)을 명세한다.
넷째, 논리 규칙을 적용한다.

'Belief Logic'을 이용한 보안 프로토콜 설계 및 검증 방법에 대한 가정 설정, 표현 방법 및 규칙 적용에 대해서는 BAN[9], GNY[10] 등 에 대한 연구 논문을 참조하기 바란다.

1.3 검증 도구

표 1은 대표적인 보안 프로토콜 검증도구의 입력 언어 및 검증방법을 보여주고 있다. 검증방법에서 'T'는 정리증명(Theorem Proving) 방식을 의미하고, 'M'은 모델체킹(Model Checking) 방식을 나타낸다. 수 많은 보안 프로토콜 검증 도구 중에서도 FDR 도구를 이용한 안전성 검증 방법이 널리 알려져 있다.

2. Casper/FDR 도구를 이용한 보안 프로토콜 검증

본 장에서는 간단한 EKE 프로토콜[9] 명세 및 분석 과정을 통해, 모델체킹 방법 중에서도, Casper/CSP 및 FDR을 이용한 보안 프로토콜 명세 및 검증방법의 이해를 돕고자 한다.

2.1 CSP, Casper 및 FDR 도구

CSP(Communication Sequential Processes)는 통신 프로토콜의 행위를 정형적으로 명세하기 위해 개발된 프로세스 알제브라 언어의 일종이다.[12] CSP 언어는 프로세스의 동시성(Concurrency) 행위를 표현할 수 있기 때문에, CSP를 이용하여 보안 프로토콜의 행위를 명세하고, FDR 모델체킹 도구를 이용하여 취약성을 분석하기 위한 많은 연구가 진행되었다[3]. 하지만 CSP 언어 명세의 복잡성으로 인해 정형적 명세 표현에 많은 시간과 노력을 필요로 한다. 이에 따라, 보안 프로토콜의 행위를 보다 간단히 표현하기 위해 Casper 도구가 개발되었다[13-14].

Casper를 이용하여 보안 프로토콜의 행위와 검증하고자 하는 속성을 명세한 후, Casper 컴파일 기능을 이용하여 자동으로 CSP 언어로 변환할 수 있다. 마지막으로 자동 생성된 CSP 모델을 FDR 도구에 입력한 후, 비밀성, 인증 등과 같은 보안속성을 만족하는지 검사하게 된다. 만일 해당 보안속성을 위반하는 이벤트를 CSP 모델에서 찾게 되면, 반례를 보여주기 때문에 보안 취약점을 분석하고 개선하는데 도움을 준다. Casper/CSP 및 FDR 도구를 이용한 전체적인 보안 프로토콜 설계 및 검증 방법은 그림 1에 나타나 있다.

우선 정형적으로 명세하고 검증하고자 하는 보안 프로토콜을 Casper로 명세해 주어야 한다. 중요한 사실은 보안 프로토콜의 안전성을 검사하기 위해서는 보안 프로토콜에 위험요소로 작용되는 공격자 모델도 함께 모델링 해주어야 한다는 사실이다. Casper를 이용하여 보안 프로토콜과 공격자를 함께 모델링 한 후, Casper의 컴파일 기능을 이용하여 자동적으로 CSP 코드를 생성하게 된다. 생성된 CSP 코드는 FDR의 입력언어로 작동하게 된다. FDR은 모델체킹 도구로서, 대략 10^7만큼의 상태 전이를 관찰할 수 있다. FDR 도구를 이용해서, CSP 언어로 구현

표 1. 보안프로토콜 검증 도구

검증도구	검증방법	입력언어
SPEAR	T	BAN, GNY
PVS	T	술어논리
FDR	M	CSP
Murphi	M	Murphi
CADP	M	LOTOS
SMV	M	SMV
SPIN	M	ProMeLa
NRL	M	NPATRL

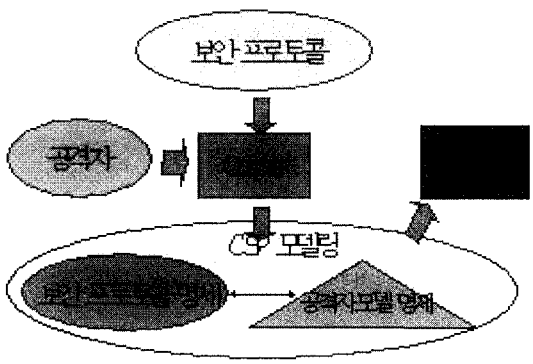


그림 1. Casper/CSP 및 FDR을 이용한 보안프로토콜 설계 및 검증 방법

된 보안 프로토콜 모델이 비밀성(confidentiality), 인증(authentication)과 같은 보안 속성을 만족시키는지 체크하게 되며, 만일 해당 속성을 만족시키지 않을 경우 반례를 보여주어, 어떤 공격 시나리오가 가능한지 분석하도록 도와준다. 마지막으로, 발견된 공격 시나리오를 바탕으로 보안 프로토콜의 취약점을 수정하면 된다.

2.2 EKE 프로토콜

이 장에서는 Casper를 이용한 보안 프로토콜 검증 방법의 이해를 돕기 위해, Casper 사용 매뉴얼[14]에 소개된 내용을 바탕으로 EKE 프로토콜의 안전성 분석 예제를 소개하도록 하겠다.

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. 다음은 EKE 프로토콜의 메시지 순서도를 보여주고 있다.

Message 1. A -> B : {A, PK(A)Passwd(A,B)
Message 2. B -> A : {{PK(A)}Kab}Passwd(A,B)
Message 3. A -> B : {Na}Kab
Message 4. B -> A : {Na, Nb}Kab
Message 5. A -> B : {Nb}Kab

EKE 프로토콜 메시지 순서도에서, A와 B는 각각 호스트를 나타내며, PK(A)와 PK(B)는 각 호스트의 공개키를 의미하고, Na와 Nb는 각 호스트의 임의 난수를 가리킨다. 그리고 Kab와 Passwd(A,B)는 각각 A 와 B의 세션키와 패스워드를 나타낸다.

2.3 EKE 프로토콜 Casper 명세

· 본 장에서는 아래에 명세된 EKE 프로토콜 예제를 통해, Casper를 통해 명세된 각 섹션 헤더의 사용법 및 의미 설명을 통해, 보안 프로토콜 표현 방법을 구체적으로 보여주고자 한다.

```
#Free variables
a, b : Agent
na, nb : Nonce
passwd, kab : SessionKey
PK : Agent -> PublicKey
SK : Agent -> SecretKey
InverseKeys = (passwd, passwd), (kab, kab), (PK, SK)

#Processes
INITIATOR(a,na,passwd,kab) knows PK, SK(a)
RESPONDER(b,nb,passwd,kab) knows PK

#Protocol description
0.    -> a : b
1. a -> b : {a, PK(a)}{passwd}
2. b -> a : {{kab}{PK(a)}}{passwd}
3. a -> b : {na}{kab}
4. b -> a : {na,nb}{kab}
5. a -> b : {nb}{kab}

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

#Actual variables
Anne, Bob, Mallory : Agent
Passwd, Kab : SessionKey
Na, Nb, Nm : Nonce
InverseKeys = (Passwd, Passwd), (Kab, Kab)

#Functions
symbolic PK, SK

#System
INITIATOR(Anne, Na, Passwd, Kab)
RESPONDER(Bob, Nb, Passwd, Kab)

#Intruder Information
Intruder = Mallory
IntruderKnowledge = {Anne, Bob, Mallory, SK(Mallory), PK, Nm}
```

Casper를 이용한 보안 프로토콜 명세는 일반적으로 8개의 섹션 헤더로 구성된다 : #Free varia-

bles, #Processes, #Protocol description, #Specification, #Actual variables, #Functions, #System, #Intruder Knowledge.

2.3.1 #Free variables

보안 프로토콜에서 사용되는 변수 및 함수 타입을 정의한다.

a와 b는 Agent 타입으로 호스트를 지칭하며, na와 nb는 Nonce 타입으로 난수를 나타낸다. passwd와 kab는 각각 사용자의 패스워드와 세션키를 의미하고 있다. PK와 SK는 각각 공개키와 개인키를 반환하는 함수를 표현한다. 그리고 Inverse-Keys 표현을 통해, 암호키와 복호키의 키쌍을 나타낸다.

2.3.2 #Processes

CSP 프로세스로 표현하게 될 각 호스트들의 역할, 변수 및 함수들을 정의한다.

a는 통신을 시작하는 송신자 역할을 담당하며, a와 na, passwd, kab 매개변수를 갖고 있으며, 다른 호스트들의 공개키들과 자신만의 개인키를 알고 있다고 가정하고 있다. 이와 마찬가지로 b는 메시지를 수신하는 수신자 역할을 담당하며, b와 nb, passwd, kab 매개변수를 갖고 있으며, 다른 로스트들의 공개키들과 자신만의 개인키를 알고 있다고 가정하고 있다.

2.3.3 #Protocol description

보안 프로토콜상에서 상호 전달되는 메시지들을 정의한다.

정수 숫자 0, 1, 2, 3 들은 전달되는 메시지의 순서를 표현하게 된다. 0번 메시지는 통신을 시작하는 A 호스트가 누구와 통신을 해야 하는지 명시적으로 알려주기 위해 사용되어 진다. 그리고 {m}{k}로 표현되는 기호는 k 키로 암호화된 메시지 m을 나타내게 된다.

2.3.4 #Specification

보안 프로토콜에서 검증하고자 하는 보안 속성을 정의한다. 보안 속성이란 크게 비밀성(confidentiality)과 인증(authentication) 속성으로 나눌 수 있다. 각 속성에 대한 의미는 다음과 같다.

비밀성이란 신뢰하는 호스트들이 공유하고 있는 비밀 정보 m은 그 밖의 다른 호스트들에 노출되어서는 안된다는 의미이다. 그리고 인증이란 만일 A호스트가 B호스트에 인증 받는다고 하면, B는 A하고의 통신을 완료하였을 경우, A는 그 이전단계에서 B하고 통신을 수행하고 있어야 된다는 의미이다. 또한 인증이란 송신자에 의한 수신자의 인증과 수신자에 의한 송신자의 인증 두 가지 측면을 생각할 수 있다.

Secret 기호는 비밀성을 나타내고, Agreement는 인증 속성을 표현하기 위해 사용된다. 좀더 상세히 말해서, 'Secret(a, na, {b})'는 "a는 중요 정보 na를 b하고만 공유하고 있다고 믿는다"는 의미로 해석되며, 'Agreement(a,b,{na,nb})'는 "a는 인증정보 na와 nb를 이용해서 b에 인증을 받는다"는 의미로 풀이된다.

2.3.5 #Actual variables

실제 통신 호스트들의 변수이름과 타입을 정의한다.

#Free variables 섹션 헤더가 통신 호스트들의 추상적 표현이라면, #Actual variables는 통신에 참여하는 실제 호스트들을 나타내게 된다.

Anne 와 Bob은 각각 a와 b의 실제 호스트 이름을 의미하고, Mallory는 공격자 호스트의 이름을 가리킨다. Na와 Nb는 각각 Anne 와 Bob 호스트가 생성하는 임의 난수이며, Nm은 공격자 Mallory가 생성하는 임의난수가 된다.

2.3.6 #Functions

#Free variables에서 기술된 함수를 정의한다.

PK는 공개키를 생성하는 함수 이고, SK는 개인키를 생성하는 함수이다.

2.3.7 #System

#Processes에서 기술한 추상화 모델에 매핑되는 실제 호스트들의 역할과 정보를 정의한다.

'INITIATOR(Anne, Na, Passwd, Kab)'는 Alice 호스트는 송신자로서 Anne, Na, Passwd와 K류 정보를 알고 있다는 의미이며, 'RESPONDER(Bob, Nb, Passwd, Kab)'는 Bob은 수신자로서 Bob, Nb, Passwd와 Nb 정보를 알고 있다는 의미로 해석된다.

2.3.8 #Intruder Information

공격자 호스트의 이름과 사전정보지식을 정의한다.

모델체킹을 기반으로한 보안 프로토콜 검증결과는 공격자 모델의 구성정도에 따라 그 결과가 달라질 수

있다. 즉, 공격자가 어떤 사전지식 과 공격 능력을 갖고 있느냐에 따라 보안 취약점이 달라지기 때문이다. CSP기반 공격자 모델은 일반적으로 다음과 같은 공격행위를 갖고 있다고 가정한다.

첫째, 상호 호스트들간에 전달되는 메시지를 가로챌 수 있다.

둘째, 정상적인 수신자에게 전달되는 메시지를 가로 막을 수 있다.

셋째, 정상적인 송신가인 것처럼 위장하여 메시지를 보낼 수 있다.

앞의 EKE 프로토콜 예제에서, 공격자의 이름은 Mallory 이며, 공격자는 각 호스트들의 이름인 Anne, Bob, Mallory, 자신이 생성하는 임의 난수 Nm, 모든 호스트들의 공개키 PK와 자신만의 개인키 SK(Mallory)를 알고 있다고 가정하고 있다.

2.4 EKE 프로토콜 분석 결과

Casper의 컴파일 기능을 이용해서, 앞에서 설명한 Casper 소스코드를 자동으로 CSP 코드 생성하게 된다. 생성된 CSP 코드를 FDR 도구에 입력하게 되면 그림 2와 같은 화면이 나타나게 된다.

2.3.4장에서 언급한 보안 속성을 검증하면, 다음과 같은 반례들을 발견하게 된다.

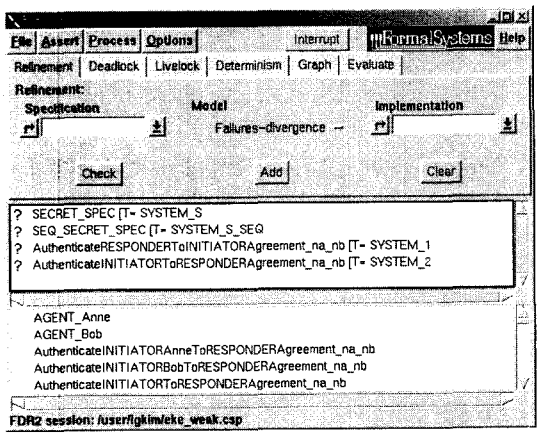


그림 2. FDR 검증도구

- Agreement(b, a, [na, nb])에 대한 반례를 보여주는 CSP 이벤트

```
env.Anne.(Env0,Anne,〈〉)
send.Anne.Anne.(Msg1,Encrypt.(Passwd,
〈Anne,PK__.Anne〉),〈〉)
receive.Anne.Bob.(Msg1,Encrypt.(Passwd,
〈Anne,PK__.Anne〉),〈〉)
send.Bob.Anne.(Msg2,Encrypt.(Passwd,
〈Encrypt.(PK__.Anne,〈Kab〉)〉),〈〉)
receive.Anne.Anne.(Msg2,Encrypt.(Passw
d,〈Encrypt.(PK__.Anne,〈Kab〉)〉),〈〉)
send.Anne.Anne.(Msg3,Encrypt.(Kab,〈Na
〉),〈〉)
receive.Anne.Bob.(Msg3,Encrypt.(Kab,〈Na
〉),〈〉)
signal.Running1.RESPONDER_role.Bob.
Anne.Na.Nb
receive.Anne.Anne.(Msg4,Encrypt.(Kab,
〈Na,Nb〉),〈〉)
signal.Commit1.INITIATOR_role.Anne.
Anne.Na.Nb
```

- Agreement(a, b, [na, nb])에 대한 반례를 보여주는 CSP 이벤트

```
env.Anne.(Env0,Mallory,〈〉)
send.Anne.Mallory.(Msg1,Encrypt.(Passwd,
〈Anne,PK__.Anne〉),〈〉)
receive.Anne.Bob.(Msg1,Encrypt.(Passwd,
〈Anne,PK__.Anne〉),〈〉)
send.Bob.Anne.(Msg2,Encrypt.(Passwd,
〈Encrypt.(PK__.Anne,〈Kab〉)〉),〈〉)
receive.Mallory.Anne.(Msg2,Encrypt.(Pas
swd,〈Encrypt.(PK__.Anne,〈Kab〉)〉),〈〉)
send.Anne.Mallory.(Msg3,Encrypt.(Kab,
〈Na〉),〈〉)
receive.Anne.Bob.(Msg3,Encrypt.(Kab,〈Na
〉),〈〉)
send.Bob.Anne.(Msg4,Encrypt.(Kab,〈Na,
Nb〉),〈〉)
receive.Mallory.Anne.(Msg4,Encrypt.(Kab,
〈Na,Nb〉),〈〉)
signal.Running2.INITIATOR_role.Anne.
Mallory.Na.Nb
signal.Commit2.RESPONDER_role.Bob.
Anne.Na.Nb
```

위의 CSP 이벤트들은 CSP 전문가가 아니면 분

석하기 어렵다. 하지만, Casper의 interpret 명령어 기능을 이용하게 되면 인증 속성을 위배하는 다음과 같은 공격시나리오를 발견할 수 있다.

0. A -> M(B) : {A, PK(A)}Passwd
1. M(A) -> B : {A, PK(A)}Passwd
2. B -> M(A) : {{Kab,}{PK(A)}}Passwd
3. M(B) -> A : {{Kab,}{PK(A)}}Passwd
4. A -> M(B) : {Na}Kab
5. M(A) -> B : {Na}Kab
6. A -> M(B) : {Na, Nb}Kab
7. M(B) -> A : {Na, Nb}Kab
8. A -> M(B) : {Nb}Kab
9. M(A) -> B : {Nb}Kab

'M(B)' 기호는 정상적인 호스트 B에게 전달되는 메시지를 가로채거나 혹은 정상적인 호스트인 B인 것처럼 위장하는 공격자 Mallory를 나타낸다. 위의 공격시나리오를 통해, 중요 정보 {Nb}Kab가 공격자 Mallory에게 노출되는 것을 알 수 있고, 그 결과 EKE 프로토콜에서는 인증 속성을 만족하지 않는다는 사실을 확인할 수 있다.

EKE 프로토콜 예제에서, 공격자는 정상적인 호스트들 중간에 위치해서 메시지를 가로채고 정상적인 호스트인 것처럼 위장할 수 있기 때문에, 결국, 인증에 필요한 {Nb}Kab 정보를 가로채게 되기 때문에 인증속성 'Agreement(b,a,{na,nb})'와 'Agreement (a,b,{na,nb})'를 위반하게 되는 것이다.

인증속성을 위반하는 보안 취약점을 막기 위해서는 공격자의 man-in-the-middle 공격을 통한 메시지 재전송 공격을 차단하여야 한다. 그러기 위해서는 EKE 프로토콜을 다음과 같이 수정할 수 있다.

Message 1. A -> B : {A, PK(A)Passwd (A,B)
Message 2. B -> A : {{PK(A)}Kab}Passwd(A,B)
Message 3. A -> B : {Na}Kab
Message 4. B -> A : {B, Na, Nb}Kab
Message 5. A -> B : {Nb}Kab

안전한 EKE 프로토콜은 초기버전에 비해서 단지, 두 번째 메시지에 B의 식별자를 추가하였을 뿐이지만, 공격자의 메시지 재공격을 막아 주는 효과를 발휘하게 된다.

3. BCY 프로토콜 명세 및 분석

3.1 BCY 프로토콜

BCY 프로토콜은 공개키 방식과 대칭키 방식을 혼합한 보안 프로토콜로서, Diffie-Hellman 및 MSR+DH 키 교환 프로토콜을 사용하고 있다[15]. Tom Coffey는 GNY를 이용하여 BCY 프로토콜의 안전성 검증하였다[16]. 하지만, 이 논문에서는 서비스 제공자와 사용자에게 공개키를 분배하는 제 3의 인증기관 S를 명세하지 않고, BCY 프로토콜의 안전성을 검증하였다. Belief Logic을 기반으로한 GNY는 보안 프로토콜의 비밀성이나 인증속성을 증명하기 보다는 상호 호스트의 믿음관계의 신뢰성을 증명하든데 그 목적이 있다. 따라서, Tom Coffey의 논문은 BCY 프로토콜에서 V와 U 호스트가 세션키 정보 SK1과 SK2를 서로 신뢰하지 못한다는 사실을 증명하였다. 또한, GNY를 이용한 증명방법은 자동화된 증명도구를 지원하지 못한다는 단점을 갖고 있다.

- BCY 프로토콜 동작 절차

Message 1. V -> U : {V, Kvd+, Kvm +}Ks-
Message 2. U -> V : {Ru}{Kvm+, {{U, u+}Ks-}Ru
Message 3 : V -> U : {dataV}SK1
Message 4 : V -> U : {dataU}SK2

앞에서 언급한 BCY 프로토콜에서 V는 서비스 제공자 이며, U는 무선 이동통신 사용자를 의미하게 된다. 그리고 메시지 순서상에는 명시하지 않았지만, S는 인증서를 방행하는 제3의 인증기관을 나타낸다. 즉, Ks- 는 인증기관의 개인키를 의미하게 된다. 메시지에 표기된 보다 상세한 기호는 표 1을 참조하기 바란다.

서비스 제공자 V가 Msg1을 사용자 U에게 전송한 후에, 사용자는 KK2 = {Kvd+}Ku-를 생성하고, 세션키 SK2 = {Ru}KK2를 계산하게 된다. 이와 마찬가지로, 서비스 제공자가 Msg2를 수신한 후, KK1 = {Ku+}Kvd- 를 생성하고, 세션키 SK1 = {Ru}{KK1}을 계산하게 된다. SK1과 SK2는 Diffie-Hellman 키 교환 방식을 나타내기 때문에 KK1=KK2, SK1=SK2의 관계가 성립한다.

표 2. BCY 프로토콜 기호 및 의미

기호	의 미
U	사용자의 식별자
V	서비스 제공자의 식별자
S	인증기관의 식별자
Rx	X 호스트에서 생성한 임의 난수
Kx+	X 호스트의 공개키
Kx-	X 호스트의 개인키
KK	키 암호화 키(KK1 = KK2)
SK	세션키(SK1 = SK2)

3.2 BCY 프로토콜 Casper 명세

제 3.1 장에 기술한 BCY 프로토콜을 토대로, Casper를 이용하여 BCY 프로토콜의 행위 및 보안 요구사항을 명세하였다. 본 논문에서는 추가적으로 인증기관 S의 행위도 Casper 모델에 추가하였다. Casper 모델은 기본적으로 8개의 헤더로 나누어지지만, 중요한 #Protocol description, #Specification, #Intruder Knowledge 및 #Equivalences 섹션 명세 부분에 대해서만 기술하도록 하겠다. 그림 1은 BCY 프로토콜에 대한 Casper 명세를 보여주고 있다. 그리고 본 논문에서는 암호 알고리즘은 안전하기 때문에 공격자가 암호키를 알지 못한 상태에서 암호문을 통해 암호키와 평문을 알아내는 것은 불가능하다고 가정하고 있다.

```
#Protocol description
0.    -> v : u
1a. s -> v : {v, pkvd, pkvm}{SSK(s)} % digV
1b. s -> u : {u, pku}{SSK(s)} % digU
2. v -> u : digV % {v, pkvd, pkvm}{SSK(s)}
3. u -> v : {ru}{pkvm}, {digU % {u, pku}
            {SSK(s)}}{ru}
4. v -> u : {{{dataV}{ru}}{pku}}{skvd}
5. u -> v : {{{dataU}{ru}}{pkvd}}{sku}

#Specification
Secret(v, ru, [u])
Secret(u, ru, [v])
Agreement(v, u, [ru, pku, skvd])
Agreement(u, v, [ru, pkvd, sku])

#Intruder Information
Intruder = Mallory
IntruderKnowledge = {Vendor, User, Sam,
Mallory, Nm, PKvd, PKvm, PKu, PKm,
SKm, SPK(Sam), Rm}

#Equivalences
forall nu, pkvd, pku, skvd, sku, ru.
{{{nu}{ru}}{pkvd}}{sku} = {{{nu}{ru}}{pku}}
{skvd}

forall nv, pkvd, pku, skvd, sku, ru.
{{{nv}{ru}}{pku}}{skvd} = {{{nv}{ru}}{pkvd}}
{sku}
```

#Protocol description 섹션 헤더는 보안 프로토콜상의 메시지 전송을 표현하기 위해 사용된다. s는 인증기관, v는 서비스 제공자 이고 u는 이동 단말기 사용자를 의미한다. 0번 메시지에서는 명시적으로 사용자 u가 서비스 제공자 v와 통신을 해야 한다는 사실을 알려주고 있다. pkvd와 pkvm은 v의 두 공개키를 나타내며, pku는 u의 공개키를 의미한다. SSK(s)는 s 인증기관의 개인키를 나타내게 되며, {v, pkvd, pkvm}{SSK(s)} % digV 는 인증기관 s의 개인키로 서명된 인증서를 의미하게 된다. 이와 마찬가지로, {u, pku}{SSK(s)} % digU도 인증기관 s의 개인키로 서명된 인증서를 표현하고 있다.

#Specification 섹션 헤더는 검증하고자 하는 보안속성을 표현하는데 사용된다. Secret 기호는 비밀성을 나타내며, Agreement 기호는 인증 속성을 표현하는 부분이다. 예를 들어, Secret(v, ru, [u])는 "서비스 제공자 v는 임의 난수 ru를 사용자 u 하고만 공유하고 있다고 믿는다"는 의미로 해석되며, Agreement(v, u, [ru, pku, skvd])는 "서비스 제공자 v는 사용자 u에게 ru, pku, skvd 정보를 이용하여 인증을 받는다"는 의미로 해석된다.

#Information knowledge는 공격자의 사전지식을 표현하기 위해 사용된다. 공격자의 사전지식을 어떻게 구성하느냐에 따라, 보안 취약점 탐지 유무가 결정된다. 본 논문에서 BCY 프로토콜에대한 공격자 호스트의 이름은 Mallory 이며, 그는 모든 호스트의 공개키, 자신의 개인키 그리고 Ru와 동일한 기능을 하는 자신의 임의난수 세션키 Rm을 알고 있다고 가정하고 있다.

#Equivalences는 메시지 표현상의 상호 교환 법칙을 적용하기 위해 사용된다. 예를 들어, 메시지 {{m}k1}k2 와 {{m}k2}k1는 같은 의미로 해석되기 때문에 수학적으로 다음과 같이 표현할 수 있다.

$$\forall k1, k2, m \cdot \{\{m\}k1\}k2 = \{\{m\}k2\}k1$$

위의 표현식은 #Equivalence 섹션 헤더 부분에 다음과 같이 명세하게 된다.

forall k1, k2, m. {{m{k1}}{k2}={{m}{k2}}{k1}

그림 1의 #Equivalence 부분에 기술된 내용의 Diffie-Hellman 키 교환 방식을 표현하고 있다. 따라서, SK1=SK2이기 때문에 {{ru}{pkvd}}{sku} ={{ru}{pku}}{skvd} 로 나타낼 수 있다.

결과적으로, #Protocol description 헤더에 기술된 4, 5번 메시지는 {{{nu}{ru}}{pkvd}}{sku} ={{{nu}{ru}}{pku}}{skvd}와 {{{nv}{ru}}{pku}} {skvd}={{{nv}{ru}}{pkvd}}{sku}로 해석될 수 있다. 그림 1의 명세상에, 만일 #Equivalence 섹션에 Diffie-Hellman 키 교환 방식에 대한 내용을 기술하였을 경우, 어떠한 공격 취약점도 찾아내지 못하게 된다.

3.3 검증 결과

FDR 모델체커 도구를 이용해서 BCY 프로토콜이 위에서 언급한 비밀성 및 인증 속성을 만족하고 있는지 확인해 보았다. 그 결과 비밀성(Secret(v, ru, [u]), Secret(u, ru, [v])) 및 인증 속성(Agreement(u, v, [ru, pkvd, sku]))을 위반하는 보안 취약점을 발견 할 수 있었다.

- 인증 속성 보안 취약점

1a. S -> I(V) : {V, PKvd, PKvm}{SSK(S)}
1b. S -> I(U) : {U, PKu}{SSK(S)}
0. -> V : User
1a.I(S)-> V : {V, PKvd, PKvm}{SSK(S)}
2. V -> I(U) : {V, PKvd, PKvm}{SSK(S)}
3. I(U) -> V : {Rm}{PKvd}, {{U, PKu} {SSK (S)}}{Rm}
4. V -> I(U) : {{{Nv}{Rm}}{PKm}}{SKu}

위의 보안 취약점에서 I(V) 기호는Vendor로 위장하거나 Vendor가 수신하는 메시지를 가로채는 공격자를 의미하게 된다. 결국, BCY 프로토콜의 보안 취약점은 공격자가 V의 공개키 PKvd를 가로채서 다시 이용하는 재사용 공격(replay attack)이 가능하다는 것을 확인할 수 있었다. 이 공격을 방어하기 위해서는 메시지에 타임 스탬프(timestamp)를 표기하거나 혹은 인증서버에서 인증서 만료일을 알려주는 메시지를 보내주어 특정 시간이 지난 후에, V의 공개키를 재사용하는 취약점을 막아주어야만 한다.

3.4 수정된 BCY 프로토콜

BCY 프로토콜의 취약점을 개선하기 위해, 두 가지사항을 고려하였다. 첫째, 인증서버 S가 V와 U에게 분배하는 인증서의 신뢰성을 높여, 재사용 공격을 막도록 타임 스탬프를 추가하였다. 둘째, V와 U가 각각 Ru와 Rv 정보를 이용하여 세션키(SK)를 생성하도록 수정하였다. 4번 메시에서, (+)는 XOR 연산을 이용한 Vernam 암호화 알고리즘을 나타내고 있다.

Message 1. S -> V : {V,Kvd+,Kvm+, TSv}Ks-
Message 2. S -> U : {U,Ku+,TSu}Ks-
Message 3. V -> U : {V,Kvd+,Kvm+,TSv}Ks-, Rv
[U computes : Ru]
Message 4. U -> V : {Ru (+) Rv}Kvm+, {U, Ku+}Ks-
[U and V compute : SK = (Ru (+) Rv)]
Message 5. V -> U : {dataV}SK
Message 6. V -> U : {dataU}SK

수정된 BCY 프로토콜을 Casper/FDR 도구를 이용하여 검증한 결과 세션키 SK={Ru (+) Rv} 정보를 공격자가 가로챌 수 없고, V와 U는 세션키를 이용하여 상호 호스트를 신뢰하고 인증할 수 있다는 사실을 검증할 수 있었다.

Ⅲ. 결 론

보안 프로토콜이 사용되는 통신 환경에 따라, 암호화 알고리즘 및 키 교환 방식이 달라 질 수 있다.

그러므로, 안전한 보안 프로토콜을 설계하는 것은 더욱더 복잡하고 어려워지고 있다.

본 논문에서는 설계단계에서 보안 프로토콜의 안전성을 검증하는 방법 및 도구를 분류 하였다. 그리고 모델체킹 방법 중 널리 사용되고 있는 Casper/CSP 및 FDR 도구를 이용하여 EKE 프로토콜 예제를 분석 하는 과정을 상세히 보여줌으로써, 보안 프로토콜 검증 및 안전한 보안 프로토콜 설계에 대한 이해를 높이고자 하였다. 마지막으로, 무선 네트워크 환경에서 사용되는 BCY 프로토콜의 안전성을 분석하고, 보안 취약성을 수정한 새로운 BCY 프로토콜을 제안하였다.

참 고 문 헌

[1] 박영희, 정병천, 이윤호, 김희열, 이재원, 윤현수, "Diffie-Hellman 키 교환을 이용한 확장성을 가진 계층적 그룹키 설정 프로토콜," 정보보호학회논문지, 13(5), pp. 3-15, 2003.

[2] 권정옥, 황정연, 김현정, 이동훈, 임종인, "일방향 함수와 XOR을 이용한 효율적인 그룹키 관리 프로토콜 : ELKH," 정보보호학회논문지, 12(6), pp. 93-112, 2002.

[3] G. Lowe, "Breaking and Fixing the Needham-Schroeder Public-Key Protocol," TACAS 96, pp. 147-166, 1996.

[4] P. E. Varner, "Formal Methods as an Environmental Catalyst for Emergent Security in System Design and Construction," December 12, 2002.

[5] M. Zalewski, "Remote vulnerability in SSH daemon crc32 compensation attack detector," Available from : http://razor.bindview.com/publish/advisories/adv_ssh1crc.html, February 2001.

[6] Formal Systems(Europe) Ltd. Failure Divergence Refinement-FDR2 User Manual, Aug. 1999.

[7] K. L. McMillian, "Symbolic Model Checking," PhD thesis, Carneigie Mellon University, May 1992.

[8] G. J. Holzman, "The Model Checker SPIN," *IEEE Trans. on Software Engineering*, vol. 23, Issue 5, pp. 279-295, May 1997.

[9] M. Burrows, M. Abadi and R. Needham. "A Logic of Authentication," In Proceeding of the Royal Society, Series A, 426, 1871, pp. 233-271, December 1989

[10] L. Gong, R. Needham and R. Yahalom, "Reasoning about Belief in Cryptographic Protocols," Proceedings 1990, IEEE Symposium on Research in Security and Privacy, pp. 234-248, 1990.

[11] 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.

[12] C.A.R. Hoare, Communicating Sequential Processes, Prentice-Hall, 1985.

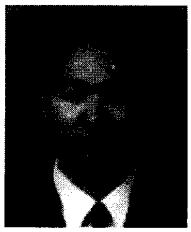
[13] G. Lowe, "Casper: "A compiler for the analysis of security protocols," 10th IEEE Computer Security Foundations Workshop, 1997.

[14] Gavin Lowe, "Casper : A Compiler for the Anaysis of Security Protocols," User Manual and Tutorial, Version 1.3, 1999.

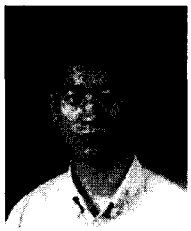
[15] M. J. Beller, L.-F. Chang and Y. Yacobi, "Privacy and authentication on a portable communications system," Proceedings of the International Computer Symposium, vol.1, pp. 821-829, 1994.

[16] T. Coffey and R. Dojen, "Analysis of a mobile communication security protocol," Proceeding of the 1st international symposium on Information and communication technologies, pp. 322-328, 2003.

〈著者紹介〉

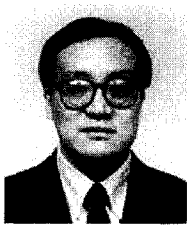


김 일 곤 (Il-Gon Kim)
1993년~2000년 : 경기대학교 영어영문학과 졸업
2000년~2002년 : 고려대학교 컴퓨터학과 석사 졸업
2002년~현재 : 고려대학교 컴퓨터학과 박사 과정
〈관심 분야〉 정형기법, 소프트웨어 공학, 보안 프로토콜, 보안 운영체제



전 철 욱 (Hyun-Seok Kim)
1996년~2003년 : 명지대 정보통신학과 졸업
2003년~2005년 : 고려대학교 컴퓨터학과 석사 졸업
〈관심 분야〉 정형기법, 보안 프로토콜, 인공지능

김 현 석 (Chul-Wuk Jeon)
1996년~2000년 : 육군사관학교 경제·경영학과 졸업
2004년~현재 : 고려대학교 컴퓨터학과 석사 과정
〈관심 분야〉 정형기법, 인터넷 네트워크 프로토콜, 무선 네트워크 보안



최 진 영 (Jin-Young Choi)
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
〈관심 분야〉 컴퓨터이론, 정형기법(정형 명세, Formal verification), 실시간 시스템, 분산 프로그래밍 언어, 소프트웨어 공학

강 인 혜 (In-Hye Kang)
1987년 : 서울대학교 컴퓨터공학과(공학사)
1989년 : 서울대학교 컴퓨터공학과(공학석사)
1997년 : University of Pennsilvania(공학박사)
1998년~2000년 : 숭실대학교 컴퓨터학부 객원교수
2000년~2002년 : (주)시큐아이닷컴 부장
2002년~현재 : 서울시립대학교 기계정보공학과 조교수
〈관심분야〉 소프트웨어공학, 컴퓨터 보안, 정형기법