情 報 保 護 學 會 誌
第 12 券 第 2 號, 2002. 4

암호프로토콜 논리성 검증도구 개발에 관한 연구

Towards Developing Formal Verification Tools for Cryptographic Protocols

권 태 경*, 김 승 주**, 송 보 연**

요 약

비형식적인 방법으로 부주의하게 설계 및 검증된 암호프로토콜은 시스템의 안전성에 대한 공격을 허용하는 중대한 결함이나 오류를 포함하기 쉽다. 이러한 암호프로토콜의 결함이나 오류를 모두 발견해내는 것은 결코 쉬운 작업이 아니며, 따라서 암호프로토콜의 체계적인 설계와 검증을 위한 정형화된 방법이 필요하다. 본 논문에서는 이 분야의 기술 동향에 대해서 살펴보고, 향후 연구에서 진행할 현실적인 개발 방법을 제안하도록 한다.

ABSTRACT

Incautiously designed and informally verified cryptographic protocols are error-prone and can allow an adversary to have the ideal starting point for various kinds of attacks. The flaws resulting from these protocols can be subtle and hard to find. Accordingly we need formal methods for systematic design and verification of cryptographic protocols. This paper surveys the state-of-the-art and proposes a practical developing method that will be implemented in the future study.

Ⅰ. 서 론

프로토콜 (protocol)이란 어떤 목적을 달성하기 위한 통신 주체간의 일련의 규약을 의미하며, 이와 같은 관점에서 암호프로토콜 (cryptographic protocol)이란 구체적으로 암호 기술을 이용하여 다양한 정보보호 목적을 달성하기 위한 규약을 일컫는다. 암호프로토콜은 그 종류가 다양하며 분산시스템의 정보보호 서비스를 위해서 반드시 필요하다.

비형식적인 방법 (informal method)으로 부주의하게 설계 및 검증된 암호프로토콜은 시스템의 안전성에 대한 공격을 허용하는 중대한 결함이나 오류를 포함하기 쉽다. 이러한 암호프로토콜의 결함이나 오류를 모두 발견해내는 것은 결코 쉬운 작업이 아니며, 따라서 암호프로토콜의 체계적인 설계와 검증을 위한 정형화된 방법이 필요하다[12,13,26,36]. 기존의 잘 알려진 소프트웨어 공학 도구들을 적용할 수 있지만, 암호프로토콜에서 사용된 암호 기술의 특성과 특수한 요구사항으로 인하여 검증에는 어려움이 있다. 따라서 국외에서는 이미 암호프로토콜의 체계적인 설계와 검증을 위한 다양한 형식 방법 (formal method)의 연구가 이루어져왔다. 특히, 형식 명세 (formal specification), 형식 제작 (formal construction), 형식 검증 (formal verification) 등 세 가지 단계에서 체계적으로 연구되고 있다. 암호프로토콜의 결함이나 오류를 명세나 설계 단계에서 발견하기는 더욱 어려우며, 따라서 검증 단계에서의 형식 방법이 가장 활발히 연구되었다[13]. 이것을 논리성 검증도구 (formal verification tool)라고 부른다.

일반적 형식 방법에 대한 기반이 약한 국내에서는 미국 및 유럽과 같은 선진국가들과의 기술격차가 매

* 세종대학교 소프트웨어 공학과 (tkwon@sejong.ac.kr)
** 한국정보보호진흥원 ({sjkim, bysong})kisa.or.kr입니다.
본 연구는 한국정보보호진흥원에서 지원하는 위탁과제로 수행하였습니다. (과제번호 2001-S-092)

우 큰 편이며, 특히 암호프로토콜에 대한 형식 방법 연구에 관해서는 더욱 취약하다. 본 논문에서는 이 분야의 기술 동향에 대해서 살펴보고, 현실적인 개발 방법을 제안하도록 한다. 먼저 II장에서는 암호프로토콜을 위한 형식 방법의 종류와 연구 필요성에 대해서 살펴본다. III장에서는 Meadows의 분류에 따른 주요 방법들을 살펴본다. IV장에서는 현재 진행중인 개발 방법을 소개하고, V장에서 결론을 맺는다.

II. 암호프로토콜을 위한 형식 방법 연구

2.1 암호프로토콜의 안전성 검증의 필요성

암호프로토콜의 특성상 부주의한 설계로 인하여 의도하지 않은 오류가 포함되기 쉽다. 이와 같은 오류는 대부분 프로토콜의 원래 목적에 대한 치명적인 결함을 제공하게 된다[13,21,30,34]. 하지만 기존의 범용 소프트웨어 공학 도구만을 이용하여 프로토콜의 안전성에 대한 형식적인 검증을 하기는 매우 어렵다. 또한 다양한 형태의 메시지 블록을 갖는 암호프로토콜에 대해서 확률적인 검증만으로 안전성을 논하는데는 한계가 있다. 따라서 국외에서는 암호프로토콜의 안전성 검증을 위한 논리성 도구 개발이 중요한 연구 분야로 인식되고 있다.

2.2 암호프로토콜의 안전성 검증 방법

암호프로토콜의 안전성 검증을 위한 방법은 확률적 안전성 분석과 논리적 안전성 분석으로 크게 나누어 볼 수 있다. 먼저 확률적 안전성 분석 방법은 랜덤 오라클 모델과 같은 수학적 모델 (확률 기반)에 근거하며, 프로토콜의 메시지를 비트 스트링으로 해석하여 확률적 보안 성질을 분석하는 방법이다. 예를 들어 Bellare-Rogaway 모델, BCK 모델, BJM 모델, Shoup 모델 등이 이 범주에 속한다[3,4,6,35]. 논리적 안전성 분석 방법은 형식 논리, 상태 탐색 (state exploration) 등을 이용하며, 프로토콜의 메시지를 추상적 심볼로 해석하여, 논리적 보안 성질을 분석하는 방법이다. 즉, 논리적 안전성이란 프로토콜의 논리적 정확성 (correctness)에서 비롯된다고 할 수 있다. 이것은 구체적으로 비밀성 (secrecy), 인증 (authentication) 등의 안전성 관련 성질을 일컫는다.

암호프로토콜의 논리적 안전성 검증 방법은 공격 구성 (attack construction) 방법과 추론 구성 (inference construction) 방법으로 크게 나누어 볼 수 있다[21]. 공격 구성 방법은 프로토콜을 상태 기계 (state machine)들의 상호 작용으로 간주하고 모든 가능한 상태들에 대한 탐색을 통해서 불안전한 상태에 도달하는 구체적인 공격 경로를 찾도록 한다. 예를 들면, 범용 도구 CSP/FDR (Communicating Sequential Process / Failures Divergence Refinement), Murϕ 등을 이용하는 방법과, Interrogator, NRL 프로토콜 분석기, Athena 등 전용 모델 검사기 (model checker)를 구현하는 방법, 대수적 성질에 근거한 Spi Calculus, 스트랜드 공간 모델 (strand space model) 등이 여기에 속한다[1,18,19,22,29,33,36,40,41].

추론 구성 방법은 Hoare 로직에서 비롯된 형식 논리 (modal logic), 즉 지식 (knowledge)과 신뢰 (belief) 분석을 위한 형식 논리를 통해서, 논리적 안전성에 관련된 목적을 정확하게 이루는지 구체적인 추론을 하도록 한다. 최초의 전용 논리 도구인 BAN 로직을 필두로 GNY 로직, AT 로직, vO 로직, SvO 로직 등의 신뢰 기반 로직과, CKT5, KPL 등의 지식 기반 로직이 잘 알려져 있다[2,5,12,20,37,38]. 이와 같은 추론 구성 방법은 HOL (Higher Order Logic), Isabelle, Convice와 같은 범용 정리 증명기 (theorem prover)를 통해서 자동화할 수 있다[8,15,23].

2.3 본 연구의 필요성

암호프로토콜의 안전성 검증 방법 중 확률적 안전성 분석 방법을 통해서 얻을 수 있는 결과가 확률적으로 어떤 안전성 관련 성질만을 만족한다는 증명이라면, 논리적 안전성 검증 방법을 통해서 얻을 수 있는 결과는 구체적인 공격 경로 혹은 프로토콜의 논리적 정확성에 대한 구체적인 증명이다. 이와 같은 면에서 논리적 안전성에 대한 검증은 유용한 방법이라고 할 수 있다. 하지만 대부분의 공격 구성 방법은 상태 폭증 (state explosion) 발생으로 인하여 무한 루프에 빠지거나 오류를 발생시키기 쉬우며, 대부분의 추론 구성 방법은 논리적 안전성을 비밀성 (secrecy) 보다는 인증 (authentication) 및 키 확증 (key confirmation) 등의 성질에 두고 있다. 또한 프로토콜의 손쉬운 명세를 통한 효율적인 자동 검증, 프로토콜의 명세 및 설계 단계에서 안전성 부여, 그리고 검증 결과와 실제 안전성과의 안정적 의미 연결 등, 아직 연구되어야할 사항들이

많다. 이와 같은 문제를 해결하고 국제적인 연구 수준에 도달하기 위해서는, 여러 방법에 대한 신중한 검토와 연구를 위한 많은 시간을 필요로 한다. 따라서 본 연구에서는 다음과 같은 요구 사항을 만족할 수 있는 효과적인 개발 방안을 제시하는 것을 목적으로 한다.

1) 높은 자동화를 통해서 사용자가 중간에 개입하는 회수를 최소화 해야한다.
2) 간단한 시스템 구성을 통해서 사용자가 쉽게 이해하고 사용할 수 있도록 해야한다.
3) 이미 잘 알려진 논리성 검증 기술에 기반하여 그 기술적 근거를 명확히 해야한다.
4) 분석 결과가 의미하는 바가 명확해야한다.
5) 논리적 안전성과 실제 안전성 문제와의 관계가 명확해야한다.
6) 자동화된 검증은 그 결과가 명확하고 항상 종료할 수 있어야 한다.
7) 짧은 기간 동안 개발하고 쉽게 갱신할 수 있도록 개발되어야 한다.

즉, 수동적인 검증보다는 자동 검증이 요구되며, 무엇보다도 사용자의 개입이 최소화하는 기술이 필요하다. 특히 이미 잘 알려진 기술들을 바탕으로 짧은 시일에 구현할 수 있고 쉽게 갱신할 수 있는 방법이 바람직하겠다. 이와 같은 요구 사항을 만족할 수 있는 자동화 기술의 개발이 필요하다.

III. Meadows의 분류에 따른 비교 분석

3.1 Meadows의 유형 분류

C. Meadows가 분류한 네 가지 유형을 정리하면 다음과 같다[24,26,36]. 유형 1은 다른 유형들과 달리 이미 존재하는 기존의 범용 명세언어를 활용하는 반면, 나머지 유형들은 암호프로토콜 분석을 위해서 필요한 방법을 만들도록 한다. 유형 4의 대수적 방법은 유형 2와 유형 3에서 간혹 공유된다. 각 유형별 연구 사례는 [표 1]에 기술되어있다.

1) 유형 1: 범용 명세언어 및 검증도구 응용
2) 유형 2: 전문가시스템 (expert system) 개발
3) 유형 3: 지식 (knowledge)과 신뢰 (belief) 분석을 위한 형식 논리 개발
4) 유형 4: 대수적 term-rewriting 성질에 근간한 형식 모델 개발

본 장에서는 Meadows의 유형 분류를 바탕으로 여러 가지 방법들을 살펴본다.

3.2 범용 명세언어 및 검증도구 응용 방법

이 유형은 소프트웨어공학의 범용적 측면에서 암호프로토콜의 논리적 정확성을 검증하지만, 정확성과 안전성의 의미 연결이 어려우며 따라서 암호프로토콜 분석에 적합하지 않다.

3.2.1 유한 상태 기계 (FSM: finite state machine)

Sidhu와 Varadharajan은 각각 암호프로토콜을 유한 상태 기계만을 사용하여 명세 및 검증하는 방법을 제안하였다[40]. 이 방법에서는 프로토콜의 각 주체를 상태 다이어그램으로 기술하는데, 먼저 초기 상태를 정의한 후 메시지의 전송/수신에 따라서 상태를 전이시키게 된다. 즉, 이것을 다음과 같은 개념을 사용해서 방향그래프로 표현한다.

P^{-n} : P transmits message n

P^{+n} : P receives message n

이와 같은 방법은 프로토콜의 다양한 실행모델을 설정하여 분석할 수 있으며, 도달 분석 (reachability analysis) 기술을 사용하여 최종적으로 수락 (accept) 상태로 전이되는지에 따라서 프로토콜의 정확성 여부를 판단하게 된다. 명세 그대로 프로토콜 공학적 측면에서 프로토콜의 정확성을 판단한다는 특징이 있다[49]. 하지만 이와 같은 방법은 능동적 공격자에 대한 안전성을 판단하기 어렵다는 문제점이 있다. 그 이유는 능동적 공격자를 모델링할 경우 상태 폭증 문제를 발생시키기 때문이다.

3.2.2 Ina Jo

Kemmerer는 소프트웨어 개발과 정확성 검증을 위한 범용 도구로 사용되었던 형식명세언어 Ina Jo를 이용하여 암호프로토콜을 검증하는 방법을 제안하였다[22]. 이 방법 역시 상태 기계로 시스템을 표현하여, 반드시 정의된 상태 전이에 따라서만 각 상태의 상태변수가 갖는 값이 변화하도록 한다. 먼저 암호프로토콜을 Ina Jo로 명세하고 모든 상태에서 만족되어야할 요구사항을 명세한다. 즉, Ina Jo의 상수, 변수, 변환, 공리, 규범들을 각각 기술한다.

〔표 1〕 암호프로토콜 논리성 검증 방법의 분류

유형	형식검증 도구	연구자(발표년도)
1	1. 상태기계와 같은 범용 기술(description) 기법 응용	D. Sidhu et. al. (1986) V. Varadharajan(1989)
	2. Ina Jo 응용	R. Kemmerer et. al.(1989/1994)
	3. LOTOS 응용	V. Varadharajan(1990)
	4. Petri Net 응용	B. Nieh et. al.(1992)
	5. CSP/FDR 응용	A. Roscoe(1995)
	6. CSP/FDR 응용	G. Lowe(1996)
	7. HOL 응용	E. Snekkenes(1995)
	8. HOL 응용 - Convince	R. Lichota et. al.(1996)
	9. HOL 응용 - Isabelle	G. Bella et. al.(1997)
	10. Murφ(phi) 응용	J. Mitchell et. al. (1997)
2	1. Interrogator	J. Millen et. al.(1987)
	2. NRL 프로토콜 분석기	C. Meadows(1991/1995)
	3. 규칙 기반 시스템	D. Longley(1992)
3	1. Rangan 로직	P. Rangan(1988)
	2. Moser 로직	L. Moser(1989)
	3. BAN 로직	M. Burrows et. al.(1990)
	4. GNY 로직	L. Gong et. al.(1990)
	5. CKT5 로직	P. Bie(1990)
	6. KPL 로직	P. Syverson(1990)
	7. AT 로직	M. Abadi et. al.(1991)
	8. GS 로직	K. Gaarder et. al.(1991)
	9. CSNP 로직	E. Campbell et. al.(1992)
	10. MB 로직	W. Mao et. al.(1993)
	11. vO 로직	P. van Oorschot(1993)
	12. SvO 로직	P. Syverson et. al.(1994)
	13. BGNY 로직	S. Brackin (1996)
	14. WK 로직	G. Wedel et. al (1996)
	15. SVD 로직	A. Dekker (2000)
4	1. Dolev-Yao 모델	D. Dolev et. al.(1983)
	2. Merrit Narrowing 모델	M. Merrit(1983)
	3. Toussaint 모델	M-J. Toussant(1992)
	4. Spi calculus 모델	M. Abadi et. al.(1998)
	5. 스트랜드 공간 모델	F. Fabrega et. al. (1996)

그리고 Ina Jo 정리를 구성하여 규범을 만족하는지 증명한다. 하지만 Ina Jo 증명만으로 안전성을 보장할 수 없으며, 규범 명세를 위해서 결국 설계자가 모든 가능한 공격방법을 미리 알아야한다는 조건이 따른다.

3.2.3 LOTOS

Varadharajan은 LOTOS (Language of Temporal Ordering Specification)를 사용하여 인증프로토콜을 분석하도록 제안하였다[41]. OSI 관련 시스템을 위해서 개발된 LOTOS는 시스템에서 발생하는 사건의 순서가 명세되어 있는 프로세스들의 집합으로 시스템을 모델링한다. 하지만 결국 LOTOS는 암호프로토콜 검증을 위해서 적합하지 않다는 결론을 내렸다[41].

3.2.4 Petri Net

Nieh와 Tavares는 Petri Net을 사용하는 방법을 제안하였으며, 암호프로토콜 모델링을 위해서 Coloured Petri Net을 사용하였다[13,36]. 하지만 이 방법 역시 전사 테스트를 해야하므로 능동적인 공격자를 포함할 경우 역시 상태 폭증 문제가 발생한다.

3.2.5 CSP/FDR

Roscoe는 CSP-FDR을 사용하는 방법을 제안하였다[33]. 프로토콜에 참여하는 모든 에이전트, 즉, 모든 통신 주체와 공격자를 CSP로 모델링하며, 이

에 대한 자동 검증을 위해서 FDR을 사용한다. FDR에는 CSP로 모델링된 프로토콜의 명세와 구현이 주어지며, FDR은 이에 대한 일치 여부를 검사한다. 하지만 이 방법은 사용자가 정의한 범위 안에서만 검증이 가능하며, 범위 밖의 문제점에 대해서는 다룰 수 없다. 즉, 범위 안에서 공격 방법이 발견되지 않더라도, 범위 밖에서 발견되지 않는다고는 할 수 없다.

3.2.6 Murφ (phi)

Mitchell 등은 범용 상태 계수기 (state enumeration tool)로 알려진 Murφ를 사용해서 암호프로토콜의 안전성을 검증하는 방법을 제안하였다[29]. 이것은 CSP를 통한 모델 검사와 유사한 방법이지만, CSP를 이용하는 방법보다 더욱 구체적이며 효율적인 것으로 알려졌다. 프로토콜과 프로토콜의 요구되는 성질을 Murφ 언어로 모델링한 후 모든 도달 가능한 상태가 이와 같은 명세를 만족하는지 자동적으로 검증한다.

3.3 전문가시스템 개발

이 유형은, 암호프로토콜의 문제점을 발견할 경우 곧바로 이 문제점을 통한 공격시나리오를 만들 수 있다는 장점이 있다. 하지만 결과를 통해서 암호프로토콜의 안전성을 보장하기 어려우며 주어진 공격 방법에 대한 안전성을 검증할 수는 있지만 알려지지 않은 공격방법을 찾아내기 어렵다. 또한 전사탐색을 하므로 비효율적인 경우가 많다. 이러한 전반적인 단점을 NRL 프로토콜 분석기는 유형 4의 대수 시스템 모델과 정리 증명 (theorem proving)으로 많이 극복하였다.

3.3.1 Interrogator

인증 프로토콜 분석을 위한 최초의 전문가 시스템으로서 Prolog 언어로 구현되었다. 이 시스템은 프로토콜 명세와 데이터 항목을 입력받아 메시지 히스토리를 출력하며, 프로토콜을 각 주체 당 통신 프로세스들의 집합으로 간주한다. 결국 모델 검사를 위한 상태 기계 접근 방법이며, 따라서 프로토콜 참여자는 통신 기능을 갖는 상태 기계로 해석되고 이들 간의 메시지는 공격자에 의해서 갈취된다고 가정한다[28].

안전성은 메시지 전송에 의한 상태 전이를 통해서 분석되며, 메시지 히스토리를 통해서 공격자가 어떤 경로로 정보를 얻게 되는지 알 수 있다. 즉, 이미 공격자가 비밀 정보를 알았다고 가정한 불안전한 상태를 최종 상태로 정의하고, Interrogator는 이 상태에 도달하는 모든 경로를 탐색한다. 결국 매우 많은 경로들이 분석을 통해서 생성된다. 만약 이런 경로 중에서 단 하나라도 시작 상태에서 출발한 경로라면, 구체적인 안전 취약성이 발견되었다고 할 수 있다.

하지만 이와 같은 최종 상태를 정의하기 어려우며, 상태 폭증을 해결할 수 있는 방법이 없다.

3.3.2 NRL 프로토콜 분석기

NRL 프로토콜 분석기는 Interrogator의 단점을 극복하도록 새롭게 설계된 전문가 시스템으로서 역시 Prolog 언어로 구현되었다[24,26]. NRL 프로토콜 분석기는 모델 검사를 수행한다는 면에서 Interrogator와 유사한 형태를 갖지만, 구체적인 방법에는 큰 차이가 있다. 특히 이것은 유형 4의 기반인 Dolev-Yao 모델에 근간하므로 유형 4로 분류되기도 한다. 하지만 Dolev-Yao 모델과의 근본적인 차이점은 Dolev-Yao 모델이 단순히 공격자의 목적을 비밀값을 발견하는 것으로 한정하는 반면, NRL 프로토콜 분석기의 기본 모델은 모든 참여자의 지역 상태 변수를 포함하여 공격자의 목적을 더욱 다양하게 설정하도록 한 것이다.

NRL 프로토콜 분석기의 가장 큰 특징은 모델 검사와 정리 증명을 병행하며, 이를 위해서 유형 4의 성질을 공유하는 것이다. 즉 암호프로토콜을 구성하는 메시지 원소들이 대수적 시스템을 통해서 약분 규칙 (reduction rule)을 따르도록 한다. 결국 프로토콜 메시지를 습득한 공격자는 term-rewriting 시스템에서 이와 같은 약분 규칙을 따라서 목적을 달성하려는 것으로 볼 수 있다. 또 한 가지의 특징은, 탐색 공간과 프로토콜 라운드의 무한함을 허용하는 것이다. 예를 들면, 실제로 안전한 암호키의 개수는 유한하지만 전사 탐색이 불가능하므로 NRL 모델에서는 이것을 무한하다고 가정한다. 마지막으로 NRL 프로토콜 분석기는 최초 상태에서 불안전한 최종 상태에 도달하는 경로를 자동적으로 검색하는 것뿐만 아니라, 유형 4에서 주로 고려되는 방법과 마찬가지로 불안전한 상태에서 후위 탐색을 할 경우 무한 루프에 도달함을 정리로써 증명하여, 자동 검색을 위한 공간을 축소하도록 한다.

NRL 프로토콜 분석기는 최근 IKE 프로토콜 및

SET 프로토콜과 같이 다중 구성을 갖는 복잡한 프로토콜의 분석을 위해서 프로토콜 명세 언어, 정리 증명기, 질의 구조 등 부분적으로 수정되었다[27]. 특히 명세 언어인 CAPSL (Common Authentication Protocol Specification Language)을 사용하기 위한 번역기도 개발되었다[11].

3.4. 지식과 신뢰 분석을 위한 형식 논리 개발

이 유형은 지식과 신뢰, 혹은 둘 중 한 가지의 분석을 위해서 개발된 형식 논리를 이용하여 암호프로토콜의 요구사항을 모델링하고 분석하는 방법이다. 지식에 근간을 둔 논리를 epistemic 논리라고 하며, 신뢰에 근간을 둔 논리를 doxastic 논리이라고도 한다[36].

기존의 일반적인 논리적 분석방법은 복잡했던 반면, 1989년 처음으로 암호프로토콜의 분석을 위해서 개발된 doxastic 논리, 즉 BAN 로직은 이것을 매우 단순화시켰다[12]. 이 유형의 방법들은 암호프로토콜을 일종의 분산 알고리즘으로 해석하며, 프로토콜 명세언어와 추론 규칙을 통하여 프로토콜이 주어진 요구사항을 만족하는지 검사한다.

3.4.1 BAN 로직

1989년 Burrow, Abadi 그리고 Needham이 제안한 BAN 로직은 암호프로토콜 분석을 위한 최초의 논리 도구로 알려져 있다[12]. BAN 로직을 통해서 암호프로토콜을 분석하는 방법 중 가장 첫 단계는 먼저 프로토콜을 BAN 로직 항 (term)으로 재구성하는 프로토콜 이상화 (protocol idealization) 과정을 수행하는 것이다. 이것은 비형식적인 방법으로 유도된다. 그리고 역시 비형식적인 방법으로 프로토콜 초기 상태에 대한 가정 (assumption)을 유도하여 BAN 로직 항으로 기술한다. 이어서 프로토콜의 각 구문 (statement)에 논리식 (logical formula)을 첨부하여 각 구문이 수행된 후의 상태에 대한 정의를 한다. 마지막으로 추론 규칙에 따라서 각 프로토콜 참여자의 신뢰 상태에 대한 추론 분석을 한다. 하지만 이 방법의 가장 큰 문제점은 프로토콜 모델을 위한 많은 가정을 필요로 한다는 것이다. 특히 이와 같은 가정을 비형식적인 방법으로 다루게 된다. 예를 들면 프로토콜 참여자는 모두 믿을 수 있어야 하며 또한 자신의 비밀값을 절대로 유출하지 않는다는 가정, 적용된 암호 시스템은 절대적으로 안전하다는 가정, 메시지의 redundancy로 인하여 복호화할 경우 항상 키에 대한 확인을 할 수 있다는 가정, 프로토콜 참여자는 자신이 생성한 메시지를 항상 확인할 수 있다는 가정 등을 필요로 한다. 더구나 BAN 로직은 메시지에 대한 목격 (seeing)과 이해 (understanding)를 명확히 구별하지 않는 등 로직의 의미론 (semantics)과 관련해서 매우 취약하다[2,30]. 또한 신뢰에 바탕을 둔 로직으로서 프로토콜 참여자의 지식에 대한 형식적인 모델링은 전혀 시도하지 않았다. 다만 묵시적으로 프로토콜 이상화 과정에서만 비형식적인 방법으로 다루어진다. BAN 로직은 논리성 검증 방법의 큰 가능성과 함께 많은 문제점을 보여주어, 이를 개선한 많은 논리성 검증 방법들이 등장하는데 기여하였다[7,30]. 이후 BAN 로직의 영향을 받아서 제안된 로직들은 BAN 로직의 기본적인 프로토콜 분석 절차를 따른다.

3.4.2 GNY 로직

1990년 Gong, Needham, 그리고 Yahalom에 의해서 제안된 GNY 로직은 BAN 로직의 몇 가지 중요한 단점들을 극복하기 위한 목적으로 제안되었다[20]. 즉, 프로토콜 참여자를 전적으로 신용하지 않으며, 메시지의 redundancy를 무조건적으로 가정하지 않는다. 또한 프로토콜 참여자의 신뢰로부터 소유를 분리하여 별도 구별했다는 특징이 있다. 하지만 이와 같은 목적을 위해서 기존의 BAN 로직의 배에 가까운 추론 규칙이 추가되어 분석을 어렵게 만들며, 의미론에 대해서는 여전히 취약하다는 문제점이 있다.

3.4.3 AT 로직

1991년 Abadi와 Tuttle에 의해서 제안된 AT 로직은 BAN 로직의 의미론 취약점을 개선하여 논리를 재구성하였다는데 가장 큰 의미가 있다[2]. 추론 규칙을 재구성하여 비록 개수는 더 많지만 그 구조를 더욱 단순화했으며, BAN 로직에서의 프로토콜 참여자에 대한 묵시적 신용 가정을 완전히 제거하였다. 그리고 신뢰에 대한 재정의를 통해서 신뢰가 자원에 의존적인 지식에 근거하도록 모델링하였다. GNY 로직과 마찬가지로 프로토콜 참여자의 소유와 신뢰를 구분하였다.

3.4.4 vO 로직

1993년 van Oorschot에 의해서 제안된 vO로

직은 Diffie-Hellman과 같은 키 일치 프로토콜 (key agreement protocol)을 위한 로직을 BAN 로직에 추가하였다는데 가장 큰 의미가 있다[16,39].

3.4.5 SvO 로직

1994년 Syverson과 van Oorschot가 제안한 SvO 로직은, 기존의 BAN 로직, GNY 로직, AT 로직, 그리고 vO 로직을 통합하여 보다 효과적이고 강력한 로직의 완성을 추구한 것이다[38]. 즉, 기존 로직들의 장점만을 취합하였으며 20개만의 추론 규칙으로 재구성하였다. 또한 AT 로직의 개념을 따라서 로직의 의미론도 형식적으로 재구성하였다.

3.4.6 CKT5

1990년 Bieber에 의해서 제안된 CKT5 로직은 신뢰 기반의 BAN 로직과 달리 지식 기반의 epistemic 로직으로 잘 알려져 있다[5]. 따라서 주로 암호프로토콜에서 발생하는 지식의 진화에 대한 검증을 수행할 수 있으며 특히 어떤 시간에 누가 무엇을 알고 있는지 검증하게 된다. 메시지의 목격과 메시지의 중요한 의미 또는 성질을 이해하는 것을 명확히 구분했으며, 지식에 기반한 행동을 기술하는데 유리하지만, 실제로 인증 프로토콜의 안전성을 분석하는데 부정확하고 불충분한 것으로 알려져 있다[34].

3.4.7 KPL

1990년 Syverson에 의해서 제안된 KPL 로직은 CKT5와 유사한 epistemic 로직이지만, 유형 4와 같이 대수 시스템 성질을 공유한다[37]. KPL 로직의 가장 큰 장점은 공격자가 프로토콜 참여자의 비밀키를 알고 있다는 사실을 쉽게 표현할 수 있는 것이지만, 이것이 프로토콜 분석에 주는 이점의 정도가 얼마나 되는지 알 수 있는 구체적인 분석 사례가 없다. 한편 이 방법을 통한 분석은 검증 자체가 매우 복잡해진다는 사실이 잘 알려져 있다[34].

3.5 형식 논리의 자동화

형식 논리의 자동화를 위해서는 Prolog와 같은 함수적 언어 (functional language)로 구현하거나, 자동화된 범용 정리 증명기를 이용하여 구현된다. 주로 이용되는 정리 증명기로는 HOL, Isabelle, Convince 등이 있으며, 이를 위한 언어로는 ML (Meta Language)이 사용된다.

3.5.1 Mathuria 등의 GNY 자동화

1995년 Mathuria, Safavi-Naini, Nickolas 등에 의해서 제안된 GNY 로직 자동 검증 도구는 Prolog 언어로 구현되었다[23]. 논리 도구의 자동화를 위해서 중요하게 다루어져야 할 부분은 추론 규칙의 구조이다. 따라서 Mathuria 등은 주어진 프로토콜로부터 항상 유한한 개수의 논리식이 유도되도록 GNY 로직의 몇 가지 추론 규칙을 수정하였다. 하지만 이와 같은 방법은 완전한 자동화를 이루기 어려우며, Prolog로 구현된 소프트웨어의 정확성에 의존해야 하는 단점이 있다.

3.5.2 HOL을 이용한 GNY 자동화

1996년 Brackin에 의해서 제안된 GNY 로직의 자동 검증 도구는 HOL 정리 증명기를 통해서 구현되었다[8]. HOL에서는 정리를 Standard ML 컴파일러의 유형 검사 (type checking)를 통해서 확인하며, 정리의 정확성에 대한 증명을 표준 HOL 정리 증명 도구를 이용한 소프트웨어로 수행할 수 있다. 따라서, 예를 들면 Prolog 구현과 같이 별도로 구현된 소프트웨어의 정확성에 의존해야하는 부담이 없다. Brackin은 HOL을 통한 자동화를 위해서 GNY 로직을 수정한 BGNY 로직을 정의하였으며, 프로토콜과 프로토콜이 가져야할 성질에 대한 명세를 쉽게 할 수 있도록 ISL (interface specification language)이라는 프로토콜 명세 언어를 정의하였다[9]. 이 도구를 AAPA (Automatic Authentication Protocol Analyzer)라고 명명한다[10]. BGNY 로직은 연결 (conjunction)과 쌍(pairing)을 위한 명확한 연산, 프로토콜의 중간 단계에서 프로토콜의 요구사항 명세 기능, 다중 키 암호 및 다중 해쉬 함수를 사용 기능, MAC 사용 기능, 임의의 계산 값을 키로 사용할 수 있는 기능, 키 교환 알고리즘 사용 기능 등 대폭적인 확장을 하였다. 하지만 여전히 의미론에 대한 취약성을 갖는다. Brackin은 AAPA를 통해서 Clark과 Jacob이 분류한 프로토콜 군에 대한 검증을 수행하였다[10,14].

3.5.3 Isabelle을 이용한 SvO 자동화

2000년 Dekker에 의해서 제안된 SvO 로직의 자동 검증 도구는 Isabelle 정리 증명기를 통해서 구현되었다[15]. Isabelle을 통한 자동화를 쉽게 구현하기 위해서 SvO 로직을 수정한 SVD 로직을 제

안하였으며, 자동화 도구에 대한 GUI 환경을 구현하였다. 이것을 C3PO라고 명명한다. SVD 로직은 SvO 로직을 기반으로 자동화뿐만 아니라 Kripke 의미론을 바탕으로 로직의 의미론에 대해서도 충분히 다루었다.

3.6 대수적 성질에 근간한 형식 모델 개발

이 유형에서는 암호프로토콜을 대수 시스템으로 모델링한다. 따라서 term-rewriting 성질에 근간한 약분 규칙을 정하고 이 규칙에 의해서 암호프로토콜을 구성하는 메시지 원소들을 정리해나갈 수 있다. 이 유형에서 가정하는 것은 암호프로토콜을 구성하는 메시지 원소들이 대수적 시스템을 통해서 약분 규칙을 따른다는 사실이다. 결국 프로토콜 메시지를 습득한 공격자는 term-rewriting 시스템에서 이와 같은 약분 규칙을 따라서 목적을 달성하려는 것으로 볼 수 있다. 하지만 이 유형이 갖는 가장 큰 문제점은 암호프로토콜 분석이 복잡해진다는데 있다.

3.6.1 Dolev-Yao 모델

Dolev-Yao 모델은 유형 4의 근간이 된 최초의 모델이다[17]. Dolev-Yao 모델이 보여준 가장 유익한 방법은 먼저 프로토콜을 수행하는 네트워크가 이미 공격자의 제어 아래에 있다고 가정하는 것이다. 이와 같은 가정은 이미 많은 모델들, 즉 NRL 프로토콜 분석기, Athena, Bellare-Rogaway 모델, BCK 모델, BJM 모델, 그리고 Shoup 모델 등 많은 분석 모델에서 활용되었다[3,4,6,24,26,35,36]. 즉, 이와 같이 가정된 공격자는 프로토콜 메시지에 대한 수정이나 파괴, 모든 통신 트래픽에 대한 도청뿐만 아니라 시스템의 합법적인 사용자에게 가능한 모든 연산들 즉, 암호화나 복호화 같은 연산들까지도 할 수 있다. 단지 공격자가 모르는 것은 프로토콜 참여자가 공유하고 있는 비밀값이며 결국 이 모델에서는 단순히 공격자의 목적을 비밀값을 발견하는 것으로 한정하게 된다. 이와 같은 비밀값의 발견은 시스템 자체를 질의 기계로 여기고 term-rewriting 규칙에 따라서 비밀값을 발견하도록 시도한다. 결국 공격자가 term-rewriting 시스템을 다루는 것이라고 할 수 있다. 하지만 이와 같은 분석 방법은 결국 비밀성 문제점을 발견하는 것에만 국한되며 또한 이전 상태의 정보를 각 참여자가 기억할 수 있는 방법이 없다는 단점이 있다.

이것을 해결하기 위한 연구도 이루어졌다.

먼저 비밀성 이외의 문제점 발견을 위해서 1983년 Merrit가 공격자의 목적을 일반화시켰으며 1992년 Toussaint는 가가 참여자가 이전 상태의 정보를 기억할 수 있도록 모델링하기 위해서 Merrit 시스템에 근간하여 전반적인 지식을 유도할 수 있도록 시도하였다[13,36].

3.6.2 스트랜드 공간 모델

Fabrega, Herzog, 그리고 Guttman에 의해서 연구된 스트랜드 공간 모델은 프로토콜의 정확성에 대한 단순한 모델과 간편한 증명 생성을 위해서 제안된 방법이며, 프로토콜과 프로토콜의 증명에 대해서 그래프를 통한 기술이 가능하도록 하였다[18,19].

여기서 스트랜드란 프로토콜 참여자에게 발생하는 연속된 사건을 의미하며, 스트랜드 공간이란 다양한 프로토콜 참여자와 공격자의 스트랜드 집합을 의미한다. 즉, 스트랜드 공간은 참여자의 스트랜드들과 공격자의 스트랜드들, 그리고 각 스트랜드들의 사상(mapping)인 trace 등으로 구성된다. 또한 스트랜드 공간은 서로 연관된 스트랜드로 이루어진 번들(bundle)을 포함하는데, 이것은 프로토콜의 모든 메시지 교환을 표현할 수 있는 정도의 크기를 가지며 우선 순위 (precedence) 관계를 갖는다. 즉, 번들은 어떤 설정 아래에서의 프로토콜의 실행을 표현한다.

이와 같은 방법을 통한 프로토콜 분석을 위해서는 각 참여자의 스트랜드들과 공격자의 스트랜드들을 만든 후, 스트랜드 공간을 구성한다. 예를 들면, 프로토콜 개시자의 스트랜드들, 응답자의 스트랜드들, 그리고 공격자의 스트랜드들로 구성된다. 그 후 번들과 우선 순위에 대한 lemma들을 이용해서, 공격자의 한계점에 대해서 증명하도록 한다. 프로토콜이 정확성을 가질 경우는 전형적으로 각 번들 마다 합법적인 참여자의 스트랜드 하나씩만으로 구성된다. 이 때 각 스트랜드는 프로토콜 참여자의 신분, 논스, 세션 키 등에 합의해야 한다.

Song, Berezin, Perrig 등은 1999년에 이와 같은 스트랜드 공간 모델을 이용한 자동 검증 도구인 Athena를 제안하였다[36]. 자동 검증을 위해서 스트랜드 공간 모델을 위한 형식 논리를 설계하였으며, 이것으로 표현된 논리식을 검증하는 자동 절차를 개발하였다.

3.6.3 Spi calculus

Abadi에 의해서 연구된 Spi calculus는 기존의 Pi calculus를 사용하여 암호프로토콜을 추상적 단계에서 기술하도록 하는 것이다[1]. Pi calculus가 암호 기능에 대한 표현 방법을 갖지 않았던 관계로 이것을 Spi calculus로 확장한 것이다. 즉, Spi calculus는 Pi calculus 보다는 덜 추상적인 단계에서 프로토콜을 분석하게 되지만, 역시 복잡한 분석을 필요로 한다.

IV. 논리성 검증 도구 개발 방안

4.1 자동 검증 기술

본 논문의 2.3절에서 제시한 요구사항을 만족하는 논리성 검증 도구의 개발을 위해서는 수동적인 검증보다는 자동 검증이 요구되며, 무엇보다도 사용자의 개입이 최소화하는 기술이 필요하다. 특히 이미 잘 알려진 기술들을 바탕으로 짧은 시일에 구현할 수 있고 쉽게 갱신할 수 있는 방법이 바람직하다.

이미 앞에서 설명한 바와 같이, 논리성 검증 도구의 자동화 방법은 모델 검사 기법을 통한 공격 구성 방법과, 정리 증명 기법을 통한 추론 구성 방법으로 나누어 볼 수 있다. 위의 요구 사항에 대체적으로 부합하는 자동화 검증 방법은 다음과 같다.

1) 정리 증명기를 통한 형식 논리의 자동화 방법
2) 모델 검사와 정리 증명을 병합한 방법

여기서 물론 정리 증명을 이용하는 방법은 형식 논리의 자동화 방법으로서 추론 구성에 해당하며, 모델 검사와 정리 증명을 병용하는 방법은 여전히 공격 구성에 해당한다.

모델 검사 방법의 가장 큰 문제점은 방대한 상태 탐색에서 비롯되는 상태 폭증과 무한 루프로 인한 탐색 실패 등이다. 따라서 NRL 프로토콜 분석기나 Athena와 같이 정리 증명을 통해서 자동 탐색할 상태를 크게 줄이는 방법이 효과적이다[25,36]. 하지만 이와 같은 방법은 자동화를 높이더라도, 사용자가 질의와 처리를 위해서 자주 개입해야 하며, 방대한 상태 폭증을 해결하기 위한 방법 마련이 간단하지 않다. 또한 개발과 검증에 실제로 상당한 시간이 소요된다.

반면 일반적으로 잘 알려지고 이미 검증된 정리 증명기와 또한 많은 연구가 진행되어온 암호프로토콜을 위한 형식 논리를 이용하여 자동화 검증 도구를 개발할 경우, 사용자는 프로토콜의 초기 명세 후 적은 개입과 명확한 결과로 인하여 편리하게 검증을 할 수 있다. 또한 비교적 짧은 기간 동안에 개발할 수 있을 뿐만 아니라 쉽게 갱신할 수 있다는 장점이 있다. 이와 같은 추론 구성 방법은 공격 구성 방법이 프로토콜의 정확성에 대해서 구체적인 공격 방법을 결과로 출력하는 것과 비교할 때, 특정 목적을 논리적으로 정확하게 이루는지 검사하는 것이다. 즉, 결과가 의미하는 논리적 정확성과 안전성과의 관계를 표현하는 의미론이 정의되어야 한다. 따라서 형식 논리의 선택과 개선, 그리고 자동화에 대한 구현 등 많은 부분이 신중하게 이루어져야 한다.

4.2 형식 논리 자동화의 타당성

이미 3.5절에서 설명한 바와 같이 기존의 잘 알려진 형식 논리 기술을 기반으로 자동화된 검증 도구를 개발한 좋은 사례로는 Acra Systems의 Brackin이 개발한 AAPA와 Australian National University의 Dekker가 개발한 C3PO 등이 있으며, 개발을 위한 구체적인 참고 모델이 될 수 있다[8,15]. 하지만 이와 같은 방법은추론 구성으로 분류되는 전적인 정리 증명 방법으로서, 공격 구성으로 분류되는 모델 검사 방법과 비교할 때 안전성 분석에 대한 결과가 상대적으로 취약할 수 있다. 하지만, 정확성과 안전성에 대한 의미론 정의 및, 상태 정보 (state information) 및 정보 흐름 (information flow)을 위한 술부 (predicate)들을 추가함으로써 안전성 분석 기준을 정의할 수 있다[31].

여러 가지 공격 구성 방법들은 구체적인 공격 방법을 찾아내거나, 이에 대한 안전성을 확인할 수 있는 반면, 상태 폭증 문제, 전체적인 안전성 보장 결여, 사용 불편함 등의 단점을 갖는다. 따라서 이와 비교할 때 추론 구성 방법들은 안전성에 대한 부분을 보완하여 자동화를 이룬다면 보다 적합한 방법이 될 수 있을 것이다.

요약하면 기존의 잘 알려진 형식 논리 기술을 기반으로 자동화된 검증 도구를 개발하는 방안은 다음과 같은 장점을 갖는다.

1) 기술적으로 완전 자동화가 가능
2) 이미 검증되고 잘 알려진 기술들 활용

3) 검증자가 쉽게 이해하고 사용할 수 있음
4) 비교적 짧은 기간에 개발할 수 있음

따라서 본 연구에서는 정보 흐름을 위한 술부를 추가하고, 안전성과 관련된 의미론에 대한 정의를 갖는 형식 논리를 자동화하는 것이 가장 바람직하다고 본다.

4.3 형식 논리 기반 도구가 검증할 수 있는 항목

주로 모델 검사기를 이용하는 공격 구성 방법과 주로 정리 증명기를 이용하는 추론 구성 방법이 검증할 수 있는 항목들은 상이할 수 있다. 여기서는 추론 구성 방법을 통해서 검증할 수 있는 항목을 나열한다.

추론 구성 방법은 논리적 증명을 위해서는 목표(goal)들과 부목표 (subgoal)들을 정의하고 추론 규칙을 적용하여 프로토콜의 수행이 그와 같은 목표를 만족하는지 확인하도록 한다. 추론 규칙에 대해서는 구문론뿐만 아니라 명확한 의미론를 정의하여, 어떠한 결과가 만족할 경우 그 결과에 대한 조건이 항상 만족함을 보장할 수 있어야 한다. 추론 구성 방법은 구체적인 공격 방법이나 공격 방법이 없다는 사실을 검증 결과로 출력하지는 않지만, 이와 같은 의미론 정의를 통해서 목표나 부목표에 대한 언역이 불가능할 경우 유사한 사실을 확인할 수 있다.

4.3.1 사용자 인증

1) 상대방 인증

가) 메시지 수정 공격 가능 여부 논리적으로 확인

추론 규칙 중에서 메시지에 대한 Seeing 규칙, Believing 규칙, 그리고 Understanding 규칙과 Jurisdiction 규칙 등을 통해서, 프로토콜 참여자가 최초에 가지고 있던 신뢰 가정이 메시지 수신으로 인하여 어떠한 결과를 만들어 가는지 확인할 수 있다. 예를 들어 man-in-the-middle 공격에 의하여 수정된 메시지가 유효한 결과를 산출하는 경우라면, 수정되지 않은 정상적인 메시지 수신 후에는 목표 혹은 부목표를 이루기 위한 신뢰와 지식을 발전시킬 수 없는 경우이다. 즉, 최초에 가정된 신뢰와 지식에서 크게 벗어날 수 없는 경우일 것이다. 하지만, 메시지 수신을 통해서 명확한 신뢰와 지식의 발전을 얻을 수 있고, 목표 혹은 부목표를 이루는 결과를 얻는다면, 메시지 수정에 대해서도 확인할 수 있다는 사실을 논리적으로 확인할 수 있다.

나) Impersonation 공격 가능 여부 논리적으로 확인

프로토콜의 논리적 검증을 시작하는 가정 중에서 가장 중요한 것이 프로토콜 참여자 명시와 이들의 신뢰 및 지식에 대한 초기화이다. 프로토콜 검증 단계에서 논리적으로 어떤 비밀값에 근거하여 상대방의 신원에 대한 확인 (인증) 결과를 얻을 수 있다면, 공격자가 어떤 비밀값 없이 참여자를 가장(impersonation)할 수 없다는 사실을 논리적으로 도출할 수 있다.

2) 상호 인증

상호 인증의 논리적 결과는 앞에서 언급한 상대방 인증에 대한 중복 확인을 의미한다. 즉, 상대방을 인증할 뿐만 아니라, 상대방이 자신에 대한 인증을 했다는 사실을 논리적으로 확인한다면, 상호 인증을 이룬다고 할 수 있다.

4.3.2 키 분배

1) 세션키 확인

프로토콜 참여자 간에 공유하는 어떤 비밀값에 대한 신뢰와 지식은 초기 가정에서 정의되어야 한다. 하지만 프로토콜의 수행을 통해서 얻을 수 있는 세션키에 대한 신뢰와 지식은 구체적인 검증을 통해서 논리적으로 확인할 수 있어야 한다. 이와 같은 확인 여부는 프로토콜 검증의 목적에 포함되며, 정상적인 참여자가 의도한 동일 세션키를 모든 참여자가 얻게 되는지에 대한 논리적 확인을 통해서 키 설정에 대한 공격 가능 여부도 확인할 수 있다.

2) 상호 세션키 확인

상호 세션키 확인의 논리적 결과는 앞에서 언급한 세션키 확인에 대한 중복 확인을 의미한다. 즉, 자신이 상대방과 동일한 세션키를 얻었다는 사실을 확인할 뿐만 아니라, 상대방이 자신과 동일한 세션키를 얻은 것을 확인한 사실을 논리적으로 확인한다면, 상호 세션키 확인을 이룬다고 할 수 있다.

4.3.3 각 단계에서 노출될 수 있는 정보 확인

메시지의 정보 흐름을 프로토콜 검증 단계에서 제어하면, 구체적으로 정상적인 초기 가정 없이 얻을

〔표 2〕 논리 도구들의 규칙 구성

논리	BAN 로직	GNY 로직	BGNY 로직	SvO 로직	SVD 로직
규칙	Belief Saying Message-Meaning Nonce-Verification Seeing Freshness Jurisdiction Symmetry	Being-Told Possession Freshness Recognizability Message Interpret. Jurisdiction	Null-Object Self-Belief Possession Reception Belief Acquisition Believes Conveyed Believes Fresh Believes Recognize Believes Shared Sec.	Believing Source Association Key Agreement Receiving Seeing Comprehending Saying Jurisdiction Freshness Nonce-Verification Symmetry Having	Believing Encryption-Related Strict Subterm Loose Subterm Recognize Subterm Seeing Understanding Said Jurisdiction Freshness

수 있는 정보에 대해서 논리적으로 확인할 수 있다. 이것은 프로토콜의 각 단계에서 노출될 수 있는 정보를 논리적으로 확인할 수 있다는 것을 의미한다.

4.3.4 각 단계에서 잘못 이해될 수 있는 정보 확인

메시지를 추상화하므로 각 메시지의 구조를 논리적으로 표현하고 검증할 수 있다. Understanding 규칙과 Seeing 규칙을 이용하여 메시지의 명확성에 대해서 논리적인 확인을 할 수 있다.

4.3.5 각 단계에서 메시지의 Freshness 확인

Freshness 규칙을 통해서 현재 세션에서 만들어진 메시지인지 여부를 논리적으로 확인할 수 있다. 현재성에 대한 확인은 Recognizing, Jurisdiction, Believing 등의 주요 규칙을 적용하기 위한 필수적인 사항이다.

4.4 논리성 검증 도구 개발 모델

본 논문에서 제안하는 형식 논리의 자동화를 통한 논리성 검증 도구 개발 방법은 형식 논리 설계 모델 및 자동화 도구 개발 모델로 나누어 고려할 수 있다.

4.4.1 형식 논리 설계

형식 논리 설계는 기존의 암호프로토콜 형식 논리를 바탕으로 구문론과 의미론을 정의해야한다.

1) 구문론 정의

먼저 구문론을 위해서는 프로토콜의 항에 해당하는 심벌 및 메타심벌, 기본 연산자, 그리고 기본 구문 구성에 대한 정의를 한다. 여기에는 BAN 로직의 심벌 체계와, SVD 로직의 연산자 구성, 그리고 SvO 로직의 구문 구성을 활용한다. 구문 구성을 위해서는 술부에 대한 정의가 필요한데, 정보 흐름 술부를 이 단계에서 고려한다[31].

2) 추론 규칙 정의

이어서 암호 연산 규칙과 추론 규칙을 정의한다. 여기에는 SvO 로직의 암호 연산 규칙과 GNY 로직 및 SvO 로직의 추론 규칙을 기반으로 규칙을 구성한다. 여기에는 주체의 신뢰 확인을 위한 Believing 규칙, 키 분배 등 암호 연산 확인을 위한 Cryptography 규칙, 현재성 확인을 위한 Freshness 규칙, 권한 확인 및 신뢰 도출을 위한 Jurisdiction 규칙, 최초 소유와 메시지 송수신에 의한 소유 확인을 위한 Possession 규칙, 메시지 전송과 수신 확인을 위한 Saying 규칙, 메시지 목격 확인을 위한 Seeing 규칙, 메시지 해석을 위한 Term Relation 규칙, 메시지 이해 여부 확인을 위한 Understanding 규칙 등이 정의되어야 한다. 이와 같은 규칙 정의를 위해서는 〔표 2〕에 나열되어있는 기존 로직들의 규칙에 기반하도록 한다. 특히, 자동화를 위해서 BGNY 로직과 SVD 로직의 구성이 참조되어야 한다.

3) 의미론 정의

의미론 정의를 위해서는 각 구문 및 규칙의 상관관계를 비형식적으로 기술한 후, 이에 대한 의미론 모델을 확립해야한다. 예를 들면, SVD 로직과 같이 Kripke 의미론 모델 등을 통해서 정의할 수 있다[15]. 의미론을 통해서 논리의 완전성 (completeness)와 안정성 (soundness)를 보여줄 수 있으며, 이것

은 추론 분석이 출력하는 논리적 안전성 결과에 실제 안전성과 관련된 의미를 부여하는 단계이므로 매우 중요하다. 구문론과 의미론 정의는, 설계한 형식 논리를 이용한 수동 분석 과정을 거치며 개선되어야 한다.

4.4.2 자동화 도구 개발

먼저 형식 논리를 설계할 때 자동화를 위해서, HOL이나 Isabelle과 같은 범용 정리 증명기 구성을 고려해야 한다. 하지만 구체적인 자동화는 형식 논리를 설계한 후 범용 정리 증명기를 통해서 이루어진다. 자동화 도구 개발을 위해서는 설계한 형식 논리를 범용 정리 증명기에서 제공하는 메타 논리(meta logic)를 통해서 정형화한다. 즉, 고차 논리인 메타 논리를 확장하여 설계한 형식 논리를 정의한다. 자동화 도구 개발에는 ML 언어를 이용하게 된다.

4.4.3 인터페이스 개발 및 확장

자동화 도구를 사용하기 위해서는 프로토콜과 프로토콜의 목표에 대한 형식적인 명세를 하고, 검증 결과를 얻기 위한 인터페이스가 필요하다. 별도의 인터페이스가 없을 시에는 ML 언어를 통해서 입력과 출력을 얻게 된다. 예를 들면, AAPA에서는 ISL과 같은 별도의 인터페이스 언어가 제공되며, C3PO에서는 GUI를 통해서 단계별로 검증하도록 한다[8,9,15]. 이와 같은 인터페이스는 사용자를 위한 명세 언어 개발과, 이에 대한 해석기 개발, 그리고 논리 자동화 도구를 통해서 얻은 결과를 Athena나 NRL 프로토콜 분석기와 같은 모델 검사 자동화 도구와 연결할 수 있는 번역기의 개발이 필요하다.

4.5 향후 연구

향후 연구에서는 4.4절의 모델, 즉 추론 구성 방법의 논리성 검증 도구를 개발한다. 먼저 형식 논리 개발을 위해서는 논리적 정확성과 논리적 안전성의 의미론적 연결, 논리적 안전성의 의미론을 통한 구체적인 안전성 분석 기능 정의 (4.3절 참조), 정보 흐름 술어를 통한 상태 정보 확인 등을 고려한 형식 논리의 설계를 한다. 이것은 GNY 로직, BGNY 로직, SvO 로직, SVD 로직의 추론 규칙과, SvO 로직 및 SVD 로직의 의미론에 바탕을 둔다. 범용 정리 증명기 Isabelle을 통한 자동화 도구를 개발한 후, 잘 알려진 명세 언어를 수용한 번역기와 인터페이스, 그리고 모델 검증 도구와의 연결을 위한 번역기를 개발한다.

V. 결 론

본 연구에서는 암호프로토콜을 위한 형식 방법의 연구동향을 분석하였으며, 논리성 검증 도구의 요구사항 및 이에 대한 개발 방안을 제시하였다. 먼저 암호프로토콜을 위한 형식 방법 연구의 동향을 살펴보면, 프로토콜의 확률적 안전성과 논리적 안전성에 대해서 별도로 연구가 진행되어 왔으며 기본적인 모델에 있어서 부분적으로 관련이 있지만, 구체적인 안전성 분석에 있어서 서로의 연관성은 매우 적다는 사실을 알 수 있다. 형식 방법의 적용 단계는 명세, 제작, 검증의 세 단계로 나눌 수 있는데, 암호프로토콜에 대한 검증 단계가 가장 활발히 연구되어 왔지만, 최근에는 명세 및 제작 단계에 대한 형식 방법 적용 연구가 구체적으로 이루어지고 있다.

암호프로토콜의 논리성 검증 도구는 크게 공격 구성 방법과 추론 구성 방법으로 분류할 수 있으며, Meadows의 분류에 의하면, 유형 1의 범용 도구 사용 방법, 유형 2의 전문가 시스템 개발 방법, 유형 3의 형식 논리 개발 방법, 유형 4의 대수적 형식 모델 개발 방법 등으로 보다 상세히 나눌 수 있다. 본 연구에서는 이와 같은 분류에 따라서 각 종 방법을 분석하였다. 특히 형식 논리의 자동화를 위한 방법으로, 모델 검사를 이용한 방법과 정리 증명을 이용한 방법으로 분류하여 논리성 검증 도구의 개발 방안을 제시하였다. 구체적인 논리성 검증 도구의 개발 방안은 잘 알려진 형식 로직 중에서 SVO 로직과 GNY 로직을 바탕으로, 정보 흐름 술부 등을 수용한 형식 로직을 설계하며, 이에 대한 자동화를 위해서 Isabelle과 같은 범용 정리 증명기를 활용하는 것이다. 이와 같은 방법은 기술적으로 완전 자동화가 가능하며, 이미 검증되고 잘 알려진 기술들 활용하고, 검증자가 쉽게 이해하고 사용할 수 있을 뿐만 아니라, 비교적 짧은 기간에 개발하고 확장할 수 있다는 장점을 갖는다.

참 고 문 헌

[1] M. Abadi and A. Gordon, "A calculus for cryptographic protocols: The Spi calculus," Technical Report SRC RR

149, Digital Equipment Corporation, Systems Research Center, January 1998.

[2] M Abadi and M. Tuttle, "A semantics for a logic of authentication," In Proc. of the Tenth Annual ACM Symposium on Principles of Distributed Computing, pp. 201-216, 1991.

[3] M. Bellare and P. Rogaway, "Entity authentication and key distribution," In Advances in Cryptology - CRYPTO 93, pp. 232-249, 1994.

[4] M. Bellare, R. Canetti, and H. Krawczyk, "A modular approach to the design and analysis of authentication and key exchange protocols," In ACM Symposium on Theory of Computing, 1998.

[5] P. Bieber, "A logic of communication in a hostile environment," In Proc. of the IEEE Computer Security Foundation Workshop, pp. 14-22, June 1990.

[6] S. Blake-Wilson, D. Johnson, and A. Menesez, "Key agreement protocols and their security analysis," In Sixth IMA International Conference on Cryptography and Coding, 1997.

[7] C. Boyd and W. Mao, "On a limitation of BAN logic," In Advances in Cryptology - Eurocrypt 93, pp. 240-247, 1993.

[8] S. Brackin, "A HOL extension of GNY for automatically analyzing cryptographic protocols," In Proc. of the IEEE Computer Security Foundation Workshop, June 1996.

[9] S. Brackin, "An interface specification language for automatically analyzing cryptographic protocols," In Proc. of the ISOC Network and Distributed System Security, February 1997.

[10] S. Brackin, "Evaluating and improving protocol analysis by automatic proof," In Proc. of the IEEE Computer Security Foundation Workshop, June 1998.

[11] S. Brackin, C. Meadows, and J. Millen, "A CAPSL interface for the NRL Protocol Analyzer, " In Proc. of IEEE ASSET 99, March 1999.

[12] M. Burrows, M. Abadi, and R. Needham, "A logic of authentication," Technical Report SRC RR 39, Digital Equipment Corporation, Systems Research Center, February 1990.

[13] L. Buttyan, "Formal methods in the design of cryptographic protocols," EPFL SSC Technical Report No. SSC/1999/038, November 1999.

[14] J. Clark and J. Jacob, "A survey of authentication protocol literature: version 1.0," http://www.cs.york.ac.uk/~jac/, November 1997.

[15] A. Dekker, "C3PO: a tool for automatic sound cryptographic protocol analysis," In Proc. of the IEEE Computer Security Foundation Workshop, June 2000.

[16] W. Diffie and M. Hellman, "New directions in cryptography," IEEE Transactions on Information Theory, vol. 22, pp. 644-654, 1975.

[17] D. Dolev and A. Yao, "On the security of public key protocols," IEEE Transactions on Information Theory, vol. 29, pp. 198-208, 1983.

[18] F. Fabrega, J. Herzog, and J. Guttman, "Honest ideals on strand spaces," In Proc. of the IEEE Computer Security Foundation Workshop, June 1998.

[19] F. Fabrega, J. Herzog, and J. Guttman, "Strand spaces: Why is a security protocol correct?," In Proc. of the IEEE Symposium on Research in Security and Privacy, 1998.

[20] L. Gong, R. Needham, and R. Yahalom, "Reasoning about belief in cryptographic protocols," In Proc. of the IEEE Symposium on Research in Security and Privacy, pp. 234-248, 1990.

[21] S. Gritzalis, D. Spinellis, and P.

Georgiadis, "Security protocols over open networks and distributed systems: formal methods for their analysis, design, and verification," Computer Communications, vol. 22, no. 8, pp. 695-707, May 1999.

[22] R. Kemmerer, "Analyzing encryption protocols using formal verification techniques," IEEE Journal on Selected Areas in Communications, vol. 7, no. 4, pp. 448-457, October 1989.

[23] A. Mathuria, R. Safavi-Naini, and P. Nickolas, "On the automation of GNY logic," Australian Computer Science Communications, vol. 17, no. 1, pp. 370-379, 1995.

[24] C. Meadows, "A system for the specification and analysis of key management protocols," In Proc. of the IEEE Symposium on Research in Security and Privacy, pp. 182-195, 1991.

[25] C. Meadows, "Applying formal methods to the analysis of a key management protocol," Journal of Computer Security, vol. 1, no. 1, pp. 5-35, 1992.

[26] C. Meadows, "Formal verification of cryptographic protocols: A survey," In Advances in Cryptography - Asiacrypt 94, pp. 135-150, 1995.

[27] C. Meadows, "Analysis of the Internet Key Exchange protocol using NRL Protocol Analyzer," In Proc. of the IEEE Symposium on Security and Privacy, May 1999.

[28] J. Millen, S. Clark, and S. Freedman, "The Interrogator: Protocol security analysis," IEEE Transactions on Software Engineering, vol. 13, no. 2, pp. 274-288, February 1987.

[29] J. Mitchell, M. Mitchell, and U. Stern, "Automated analysis of cryptographic protocols using Murφ," In Proc. of the IEEE Symposium on Security and Privacy, May 1997.

[30] D. Nessett, "A critique of the Burrows, Abadi and Needham logic," Operating Systems Review, vol. 24, no. 2, pp. 35-38, April 1990.

[31] R. Peri, W. Wulf, and D. Kienzle, "A logic of composition for information flow predicates," In Proc. of the IEEE Computer Security Foundation Workshop, June 1996.

[32] C. V. Ramamoorthy, Personal Communications, 2001

[33] A. Roscoe, "Modelling and verifying key-exchange protocols using CSP & FDR," In Proc. of the IEEE Computer Security Foundation Workshop, June 1995.

[34] A. Rubin and P. Honeyman, "Formal methods for the analysis of authentication protocols," Technical Report CITI TR 93-7, October 1993

[35] V. Shoup, "On Formal Models for Secure Key Exchange," IBM Zurich Research Lab., 1999.

[36] D. Song, S. Berezin, and A. Perrig, "Athena: a novel approach to efficient automatic security protocol analysis," In Journal of Computer Security, vol. 9, no. 1, pp. 47--74, 2001.

[37] P. Syverson, "Formal semantics for logics of cryptographic protocols," In Proc. of the IEEE Symposium on Research in Security and Privacy, pp. 32-41, 1990.

[38] P. Syverson and P. van Oorschot, "On unifying some cryptographic protocol logics," In Proc. of the IEEE Symposium on Research in Security and Privacy, pp. 14-28, 1994.

[39] P. van Oorschot, "Extending cryptographic logics of belief to key agreement protocols," In Proc. of the ACM Conference on Computer Communications Security, pp.232-243, 1993.

[40] V. Varadharajan, "Verification of network security protocols," Computers and Security, vol. 8, no. 8, pp. 693-708, 1989.
[41] V. Varadharajan, "Use of a formal description technique in the specification of authentication protocols," Computer Standards and Interfaces, vol. 9, pp. 203-215, 1990.

〈著 者 紹 介〉

권 태 경(Taekyoung Kwon)
종신회원
1992년 : 연세대학교 컴퓨터과학과 졸업
1995년 : 연세대학교 컴퓨터과학과 석사
1999년 : 연세대학교 컴퓨터과학과 박사
1999년~2000년 : U.C. Berkeley Post-Doc.
2001년~현재 : 세종대학교 컴퓨터공학부 소프트웨어공학과 전임강사, 정보보호학회 편집위원, TTA 암호분과 특별위원
관심분야 : 정보보호, 암호 프로토콜, 인증 및 키 관리

김 승 주 (Seungjoo Kim)
종신회원
1994년 2월 : 성균관대학교 정보공학과 공학사
1996년 2월 : 성균관대학교 대학원 정보공학과 공학석사 (암호학 전공)
1999년 2월 : 성균관대학교 대학원 정보공학과 공학박사 (암호학 전공)
1998년 12월~현재 : 한국정보보호진흥원(KISA) 암호기술팀장
2000년 6월~현재 : 한국정보통신기술협회(TTA) 정보통신기술위원회 암호기술연구반 의장
관심분야 : 암호이론

송 보 연 (Boyeon Song)
1994년 3월~1998년 2월 : 한국항공대학교 통신정보공학과 졸업
1998년 3월~2001년 2월 : 한국정보통신대학원 정보공학과 석사
2001년 2월~현재 : 한국정보보호진흥원 암호기술팀 연구원
관심분야 : 암호프로토콜, 정보보호