情報保護學會論文誌
第13券 第1號, 2003. 2

암호프로토콜 논리성 자동 검증에 관한 연구*

권 태 경**, 양 숙 현***, 김 승 주****, 임 선 간****

An Experimental Study on the Semi-Automated Formal Verification of Cryptographic Protocols

Taekyoung Kwon**, Sookhyun Yang***, Seungjoo Kim****, Seongan Lim****

요 약

본 논문에서는 암호프로토콜의 논리성 검증을 위한 방법 중 하나인 SVO 로직을 바탕으로 한 자동 검증 방법과 실험에 대해서 다룬다. 먼저 기존 SVO 로직 자동 검증의 문제점을 도출한 후 자동 검증을 고려한 ASVO 로직을 설계하고 검증하였으며, Isabelle/Isar 시스템을 이용하여 구현하였다. 본 논문에서는 잘 알려진 NSSK 프로토콜 중심으로 추론 구성 사례를 소개하도록 한다. 결과적으로 이미 Denning-Sacco 공격에 취약한 것으로 알려진 NSSK 프로토콜의 문제점들을 ASVO 로직의 자동화 검증을 통해서 정확히 확인할 수 있었다. 그리고 최종적으로 ASVO 로직의 자동화 검증을 통해서 발견된 취약점들을 개선한 NSSK7 프로토콜을 설계하고 검증하였다.

ABSTRACT

This paper presents a semi-automated formal verification method based on the famous SVO logic, and discusses its experimental results. We discuss several problems on automating the SVO logic and design its derivative, ASVO logic for automation. Also the proposed method is implemented by the Isabelle/Isar system. As a result, we verified the well-known weakness of the NSSK protocol that is vulnerable to the Denning-Sacco attack, using our Isabelle/ASVO system. Finally, we refined the protocol by following the logical consequence of the ASVO verification.

Keyword : *인증 프로토콜, 프로토콜 분석, 인증 로직*

Ⅰ. 서 론

암호프로토콜(cryptographic protocol)이란 암호 기술을 이용하여 정해진 정보보호 목적을 달성하기 위한 규약을 일컫는다. 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 쉬운 작업이 아니며, 기존의 잘 알려진 소프트웨어 공학 도구들을 적용하는 경우에도 암호프로토콜에서 사용된 암호 기술의 특성과 특수한 요구사항으로 인하여 검증에는 어려움이 있다. 따라서 암호프로토콜의 체계적인 설계와 검증을 위한 형식방법(formal method)이 필요하다[6,7,14]. 형식방법의 연구는 주로 검증 단계의 형식 방법이 활발히 연구되었는데 그 이유는 명세나 설계

* 본 연구는 한국정보보호진흥원에서 지원하는 위탁과제로 수행하였습니다.(과제번호 2002-S-073)
** 세종대학교 컴퓨터공학부(tkwon@sejong.ac.kr)
*** 한국과학기술원 전자전산학과
**** 한국정보보호진흥원({sjkim, seongan}@kisa.or.kr)

단계에서 암호프로콜의 결점을 발견하기는 더욱 어렵기 때문이다[7]. 알려진 형식검증 방법에서 Syverson 등에 의해 제안된 SVO로직은 이미 구문 구조와 의미 구조의 안정성에 대한 증명이 이루어져서 건전한 로직으로 알려져 있지만, SVO로직을 통한 추론 경로는 다양해 질 수 있고 최적의 경로를 얻어내기 위해서는 적절한 공리 스키마 대입이 필요하다. 따라서, SVO로직을 이용한 수동적 논리성 검증은 검증자의 전문적 수준에 의해 그 효율성과 정확성이 크게 좌우된다.

본 논문에서는 기존의 형식 방법 중 하나인 SVO 로직에 대한 자동화 검증 방법을 제안한다. 먼저 II 장에서는 기존 연구에 대해서 살펴본 후, 본 연구의 방향을 기술한다. III장에서는 SVO 로직의 자동화의 문제점을 도출한후, 이에 대한 해결 방안으로 개선된 ASVO 로직과 자동화 방안을 제안한다. IV장에서는 구현된 Isabelle/ASVO 도구를 이용한 실험 내용을 기술한다. 그리고 V장에서 결론을 맺는다.

II. 암호프로토콜 자동 검증 기술

2.1 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 수학 모델을 통하여, 일련의 심볼로 표현된 프로토콜 메시지를 비트열로 해석하여 안전성을 확률적으로 분석하는 방법이다. 예를 들어 Bellare-Rogaway 모델, BCK 모델, BJM 모델, Shoup 모델 등이 이 범주에 속한다[2,17]. 하지만 이와 같은 방법은 자동화하기가 어렵다. 논리적 안전성 분석 방법은 형식 논리(formal logic) 또는 상태 탐색(state exploration) 방법 등의 잘 알려진 형식 방법을 이용하며, 프로토콜의 메시지를 추상적 심볼로 해석하여, 안전성을 논리적으로 분석하는 방법이다. 여기서 안전성을 논리적으로 분석한다는 것은, 암호프로토콜이 목적한 바를, 예를 들면 인증이나 키분배 등과 같이 안전성에 관련된 사항을, 논리 언어를 통하여 정형적으로 기술하고, 프로토콜이 주어진 가정하에서 이것을 논리적으로 정확하게 이루는지를 분석하는 것을 말한다.

암호프로토콜의 논리적 안전성 검증 방법은 공격 구성(attack construction) 방법과 추론 구성(inference construction) 방법으로 크게 나누어 볼 수 있다[11]. 우선 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 하는 방법을 일컫는다. 예를 들면, 범용 도구 CSP/FDR, Murϕ 등을 이용하는 방법과, Interrogator, NRL 프로토콜 분석기, Athena 등 전용 모델 검사기(model checker)를 구현하는 방법 등이 여기에 속한다[12,15,16,18]. 추론 구성 방법은 주로 확장 로직, 즉 일종의 modal 로직을 바탕으로 지식(knowledge) 또는 신뢰(belief) 분석을 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[1,6,10,19,22].

확률적 안전성 분석 방법을 통해서 얻을 수 있는 결과가 확률적으로 어떤 안전성 관련 성질을 만족한다는 증명이라면, 논리적 안전성 검증 방법을 통해서 얻을 수 있는 결과는 구체적인 공격 경로 혹은 논리적 추론 경로에 대한 구체적인 증명이다. 특히 확률적 안전성 분석 방법은 자동화가 불가능한 반면, 논리적 안전성 분석 방법은 자동화를 고려할 수 있다. 이와 같은 면에서 확률적 안전성 검증과 함께, 논리적 안전성에 대한 검증은 다른 한편의 유용한 방법이라고 할 수 있다.

2.2 암호프로토콜의 안전성 자동 검증 방법

논리적 안전성 분석의 자동화 방법은 논리적 안전성 검증 방법에 따라 두 가지로 나누어 볼 수 있다. 먼저 공격 구성 방법의 자동화는 모델 검사(model checking) 방법을 통하여 자동화하게 되는데, 이것은 시스템의 유한 모델을 구성하여, 해당 모델에서 원하는 특성이 성립하는지에 대하여 상태 탐색을 하는 것을 말한다. 하지만 이와 같은 방법은 쉽게 자동화해볼 수 있는 반면, 범용적 활용을 위해서는 시스템에 대한 전문적인 수정(modification)과 추론 기법의 병용을 요할 뿐만 아니라, 상태 폭증(state explosion) 발생으로 인하여 무한 루프에 빠지거나 오류를 발생시키기 쉽다는 단점이 있다.

한편 추론 구성 방법의 자동화는 자동 정리 증명(theorem proving) 방법을 통해서 자동화하게 되는데, 추론을 얻어내기 위한 각 단계를 자동화[1]하여 보다 쉽고 정확하게 추론을 얻어내도록 하는 것을 말한다.

이를 위해서는 Prolog와 같은 함수적 언어(functional language)로 구현하거나, 범용 정리 증명기를 이용하여 구현해야 한다. 이와 같이 반자동화를 통하여 구체적인 추론을 보다 쉽게 얻어낼 수 있지만, 총체적 자동화와는 상당한 거리가 있다.

2.3 연구의 방향

2.3.1 연구 목표

본 연구에서는 암호프로토콜의 대표적 유형에 해당하는, 인증 및 키분배 프로토콜의 논리적 추론을 위한 형식 논리의 자동화 도구를 개발하고자 한다. 이를 위하여 비록 자동 검증에 대해서는 취약하지만, 구문 구조와 의미 구조에 대해서는 안정적인 SVO 로직을 바탕으로 자동 검증 방법을 개발하도록 한다. 구현을 위해서는 자동 정리 증명기인 Isabelle/Isar 시스템을 사용하도록 한다[23].

먼저 SVO 로직 자동화의 문제점을 기술하고, SVO 로직이 만족하는 추론 기능을 바탕으로 자동 검증을 위한 ASVO(Automation-considered SVO) 로직[2]을 설계한다. ASVO 로직은 SVO 로직의 단순한 공리 스키마를 최대한 유지하도록 하며, 구문적 의미적 안정성을 유지하도록 한다. 또한 잘 알려진 인증 프로토콜인 NSSK(Needham-Schroeder Shared Key) 프로토콜을 통해서 ASVO 로직의 추론 구성 예를 소개한다.

2.3.2 관련 연구

직접 함수적 언어만을 사용하여 구현한 예로는, 1995년 Mathuria, Safavi-Naini, Nickolas 등에 의해서 제안된 GNY 로직 자동 검증 도구가 있다[13]. 이들은 주어진 프로토콜로부터 항상 유한한 개수의 논리식이 유도되도록 GNY 로직의 몇 가지 추론 규칙을 수정하였다. 하지만 결과적으로 Prolog로 구현된 소프트웨어의 정확성에 의존해야 하는 단점이 있다.

1996년 Brackin에 의해서 제안된 자동 검증 도구는 HOL 정리 증명기를 통해서 구현되었다[3,4]. 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. Brackin은 HOL을 통한 자동화를 위해서 GNY 로직을 수정한 BGNY 로직을 정의하였으며, 자동 검증 도구를 AAPA(Automatic Authentication Protocol Analyzer)라고 명명하였다[5]. 하지만 SVO 로직과 비교할 때 GNY 로직에 바탕을 둔 AAPA는, 로직 의미 구조의 안정성에 대해서 상대적으로 취약하다. Brackin은 AAPA를 통해서 Clark과 Jacob이 분류한 프로토콜 군의 일부에 대한 검증을 수행하였다[5,8].

2000년 Dekker에 의해서 제안된 자동 검증 도구인 C3PO는 Isabelle 정리 증명기를 통해서 구현되었다[9]. Isabelle/Pure를 통한 자동화를 위해서 SVO 로직을 수정한 SVD 로직을 제안하였으며, 이에 대한 자동화 도구를 C3PO라고 명명하였다. SVD 로직은 SVO 로직에 근간을 두고 있지만, 자동화를 위해서 단순한 SVO 로직의 공리들을 70여개에 이르는 규칙들의 집합으로 단편화하였다. 특히 해당 자동화 도구가 직접 다양한 프로토콜을 자동 분석하고 문제점을 발견한 사례에 대해서는 보고된 바 없다[21].

Ⅲ. 자동화를 위한 ASVO 로직

3.1 SVO 로직

3.1.1 SVO 로직 소개

1994년 Syverson과 van Oorschot가 제안한 SVO 로직은, modal 로직에 근간한 기존의 BAN 로직, GNY 로직, AT 로직, 그리고 vO 로직을 통합하여 보다 효과적이고 강력한 로직의 완성을 추구한 것이다[19~22]. 즉, 기존 신뢰 기반 로직(doxastic logic) 들의 장점만을 취합하였으며, 20개 정도의 간략한 공리 스키마로 재구성하였다. 또한 AT 로직을 따라서 의미 구조에 대한 안정성 증명도 수행하였다.신뢰기반으로 확장된 형식 논리서, SVO 로직은 다음과 같은 두 가지 추론 규칙을 갖는다.

MP) φ와 $\varphi \Rightarrow \psi$로부터 ψ를 추론한다.

Nec) $\vdash \varphi$ 로부터 $\vdash P$ believes φ를 추론한다.

---

1) 형식 논리에 대한 완전 자동화는 주어진 목표(goal)에 대한 추론을 자동화하는 것을 의미한다. 즉, 각 프로토콜은 여러 가지 목표를 가지며, 그 목표 또한 추론을 위해서 부목표(subgoal)로 나누어질뿐만 아니라, 결과적으로 이에 대한 사용자의 개입이 요구되므로, 오히려 사용자의 관점에서는 반자동화(semi-automation)라고 볼 수 있다.

2) SVO 로직의 자동화에 대한 선행 연구로는 SVD 로직을 구현한 C3PO가 유일하다[9]. 하지만 Isabelle/Pure를 사용해서 구현된 SVD 로직의 경우, SVO 로직을 단편화하였다는 면에서 본 연구와는 상당한 거리가 있다. 본 연구는 SVD 로직과 달리, 구문 구조를 복잡하게 단편화하지 않으며 SVO 로직의 구문적 의미적 구조를 바탕으로 ASVO 로직을 설계하였으며, Isabelle/Isar를 사용해서 구현하였다.

〔표 1〕 SVO 로직 수행 단계

단계	내용
0	프로토콜이 이루고자 하는 목표를 명세한다.
1	초기 상태에 대한 가정을 기술한다. 이 것을 initial state assumptions(ISA)라고 한다.
2	프로토콜을 기술한다. 이 것을 received message assumptions(RMA)라고 한다.
3	프로토콜의 메시지 이해에 대한 가정을 한다. 이 것을 comprehension assumptions(CA)라고 한다.
4	위의 단계에서 주어진 가정에 대한 해석에 대한 가정을 한다. 이 것을 interpretation assumptions(IA)라고 한다.
5	SVO 로직을 사용하여 프로토콜 주체들 간에 주어지는 신뢰를 도출한다.

여기서 MP는 predicate 로직에서 가장 일반적인 규칙 중 하나인, modus ponens에 해당하며, Nec은 generalization 규칙에 해당하는 necessitation을 의미한다. ⇒는 implication을 의미하며, φ와 ψ는 논리식, 즉 formula를, 그리고 $\vdash$는 theorem을 의미한다. SVO 로직은 수동적 추론에는 활용 가능하지만, 직접적인 자동화는 매우 어렵다. SVO 로직의 다양한 표기법과 자세한 사항에 대해서는 관련 논문[19~21]을 참고하기 바란다. 한편 SVO 로직을 이용한 수동적 추론, 즉 프로토콜 검증 단계를 살펴보면 [표 1]과 같다.

3.1.2 SVO 로직 자동화의 문제점

SVO 로직을 자동화하는데 있어서 문제가 되는 사항들이 어떤 것인지 살펴보도록 하자. [표 1]과 같은 SVO 로직의 검증 단계 중에서 특히 프로토콜에 대한 CA와 IA를 설정하는 과정은 검증자의 노력에 매우 의존적이다. 예를 들어서 NSSK 프로토콜 검증의 일부를 살펴보면 주체 A가 상대방 S로부터 받은 메시지 $\{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$에 대해서 다음과 같이 CA를 정의한다.

P15) A believes A received $\{N_A, B, *_1, *_2\}_{K_{AS}}$

또한 이에 대한 IA를 다음과 같이 정의한다.

P19) A believes A received $\{N_A, B, *_1, *_2\}_{K_{AS}}$

⇒ A believes A received

$$\left\{N_A, B, A \xleftrightarrow{K_{AB}} B, fresh(K_{AB}), *_2\right\}_{K_{AS}}$$

여기서 * 표시로 변환되는 부분은 해당 주체가 이해하지 못하는 값에 대한 표현이다. 하지만 우리는 추론에 대한 자동화를 고려할 때 SVO 로직의 몇 가지 문제점을 지적할 수 있다.

첫째, 프로토콜에 대한 CA를 규정하는 단계는 주체가 인식할 수 있는 메시지와 그렇지 못한 메시지를 수동적으로 구분해야한다는 면에서 자동화가 불가능하도록 만든다. 예를 들면, 위의 메시지를 P15의 메시지 $\{N_A, B, *_1, *_2\}_{K_{AS}}$로 변환하는 과정은, 프로토콜을 명확히 이해한 검증자에 의해서 직접 이루어져야 한다. 또한 프로토콜 증명 과정에서 이와 같은 메시지를 복원하는 작업 역시 수동적인 작업에 해당한다. 또한 이와 같은 작업을, 후향 증명(backward proof)으로는 이룰 수 없다.

둘째, 프로토콜에 대한 IA를 규정하는 단계는 BAN 로직이나 GNY 로직의 프로토콜 이상화(idealization) 단계에 해당하며, 역시 자동화가 불가능하도록 만든다. 예를 들면, P19의 메시지 $\left\{N_A, B, A \xleftrightarrow{K_{AB}} B, fresh(K_{AB}), *_2\right\}_{K_{AS}}$의 경우, K_{AB}에 대한 해석에 해당하는 $A \xleftrightarrow{K_{AB}} B$와 fresh($K_{AB}$)를 일방적으로 삽입하는 과정은, 역시 프로토콜을 명확히 이해한 검증자에 의해서 이루어져야한다. 또한 역시 이와 같은 작업을, 후향 증명으로는 이룰 수 없다.

셋째, 추론 도중에 강제적으로 해석되어야하는 모호한 부분들이 존재한다. 예를 들면, SVO 로직의 수신 공리(receiving axioms)에서와 같이 from Q라고 하는 상태 삽입은 역시 프로토콜을 명확히 이해한 검증자에 의해서 이루어져야한다.

넷째, formula나 함수의 메타 표기(meta notaton)에 대한 구체적인 표현이 필요하다. 예를 들면, F(X)와 같은 메타 함수에 대한 구체적인 정의와 이에 대한 확장이 필요하다.

다섯째, 추론 경로에 따라서 무한 순환을 발생시킬 수 있다. 따라서 무한 순환을 일으키지 않도록 주의가 필요하다.

마지막으로, Isabelle/Isar와 같이 HOL(Higher Order Logic)을 기반으로 논리를 구현하는 경우 유형 충돌을 명확히 피하도록 해야한다. 특히 term과 formula의 유형 구분으로 인하여, 연산자 정의에 어려움이 있다. 예를 들면, formula에 대한 연결(conjunction)과 term에 대한 쌍(pairing)을 구성하기 위한 명확한 연산 정의가 필요하다.

본 연구에서는 이와 같은 사항들을 고려하여, 비록 SVO 로직에 바탕을 두지만, 많은 부분을 개선한 ASVO 로직을 설계하였다.

3.2 ASVO 로직

3.2.1 ASVO 로직 구성

ASVO(Automation-considered SVO) 로직은 Isabelle/Isar로 구현된 Isabelle/ASVO 시스템을 통해서 암호 프로토콜의 인증 및 키 분배에 관한 반자동 검증을 지원하며, 또한 SVO 로직의 단순 명료성을 바탕으로 설계된 만큼 전문가에 의한 수동 증명도 가능하다. Isabelle/ASVO 시스템을 통한 증명 과정은 후향 증명으로 이루어진다[23]. ASVO 로직은 다음과 같이 명확한 가정으로부터 일정의 프로토콜 목표를 유도할 수 있는지, 논리적으로 검증하기 위한 도구이다.

$\vdash$ 가정1; 가정2; …, 가정 $n \Rightarrow$ 목표

ASVO 로직을 통하여 프로토콜을 검증하는 절차는 다음과 같다.

① 프로토콜을 기술한다.
② 프로토콜이 달성하고자 하는 목표를 기술한다.
③ 프로토콜의 초기 상태 가정(ISA: initial state assumption)을 기술한다. 이것은 프로토콜의 초기 설정과 관련된 주체의 신뢰를 명시한다.
④ 프로토콜의 수신 메시지 가정(RMA: received message assumption)을 기술한다. 이것은 프로토콜의 정확한 메시지 수신에 대한 주체의 신뢰를 명시한다.
⑤ 로직을 이용하여 프로토콜 참여 주체가 갖는 신뢰를 도출한다.

즉, ASVO 로직의 검증 절차는 [표 1]에서 살펴보았던 SVO 로직의 절차에 비해서 간소화되었음을 알 수 있다. 비록 검증 절차가 간소화되었지만, 역시 매우 숙련된 경우를 제외하고는 순수하게 수동으로만 검증할 경우 오류를 포함하기 쉽다. 따라서 Isabelle/ASVO 도구를 이용하여 반자동 검증을 할 경우 보다 명확한 검증 결과를 쉽게 산출할 수 있다. 한편 ASVO 로직의 공리 스키마는 [표 2]와 같이 구성된다. 기존의 SVO 로직보다는 구체화된 공리 스키마를 갖지만, 의미구조적으로 안정성을 유지한다.

본 논문에서는 ASVO 로직의 특징과, NSSK 프로토콜 검증을 위한 공리 스키마만을 부분적으로 소개하며, 구체적인 ASVO에 대한 자세한 내용은 별도의 논문에서 다루기로 한다.

〔표 2〕 ASVO 로직 공리 스키마

Believing 공리 스키마
Source Association 공리 스키마
Key Agreement 공리 스키마
Receiving 공리 스키마
Possession 공리 스키마
Key Ownership 공리 스키마
Saying 공리 스키마
Freshness 공리 스키마
Jurisdiction 공리 스키마
Nonce-Verification 공리 스키마
Symmetric Goodness 공리 스키마
Interpretation 공리 스키마
Recognition 공리 스키마

3.2.2 ASVO 로직의 특징

ASVO 로직은 SVO 로직의 자동화를 위하여 변형된 로직으로서 다음과 같은 특징을 갖는다.

1) SVO 로직의 단순한 구문 구조와 함께, 의미적 기본 구조를 최대한 유지하였다.
2) SVO 로직에서 CA와 IA 규정 단계를 제거하였으며, 그대신 이와 같은 부분을 논리적으로 해석할 수 있는 공리 스키마들을 정의하였다.
3) 각 formula의 연결과 term에 대한 쌍 구성을 위한 연산자를 정의하였다.
4) 메타 표기로 인한 모호성 제거를 위해서, 함수의 명확한 정의나 규칙을 포함하도록 하였다.
5) 무한 순환 발생을 피하기 위하여 predicate에 대한 구문 및 의미적인 선형 구조를 구성하였다.
6) Gödel의 두 번째 공리에 바탕을 두고, 후향 증명을 하도록 하였다. 따라서 P believes $\varphi \Rightarrow P$ believes ψ와 같은 추론 증명을 위해서 P believes ($\varphi \Rightarrow \psi$)에 대한 추론을 할 수 있다.

3.2.3 NSSK 프로토콜 검증을 위한 공리

ASVO 로직에서 NSSK 프로토콜의 자동 증명을 위해서 직접적으로 필요한 부분 공리만 [표 3]에 나열하였다. 이와 같은 표기법은 사용자가 쉽게 이해하고 알아볼 수 있도록 SVO 로직을 바탕으로 구성되었으며, Isabelle/ASVO 시스템에서는 Isabelle/Isar 언어로 변환되어야한다.

3.3 Isabelle/ASVO 시스템

3.3.1 Isabelle/ASVO 시스템 개요

정리 증명기인 Isabelle 시스템은 표준 ML 언어로 구현

〔표 3〕 ASVO 로직의 부분 공리 집합

A1) P believes $\varphi \wedge P$ believes $(\varphi \rightarrow \psi)$ $\Rightarrow P$ believes ψ
A2) P believes $\varphi \Rightarrow \varphi$
A5) P understands $X \wedge P$ received2 X from $Q \Rightarrow$ Q said $(X, P \overset{K}{\leftrightarrow} Q)$
A14) $(P$ has $X_1 \wedge \ldots \wedge P$ has $X_n) \Rightarrow$ $(P$ has F$(X_1, \ldots, X_n))$
A17) P said $(X_1, \ldots, X_n) \Rightarrow P$ said X_i
A17a1) P says $(X_1, \ldots, X_n) \Rightarrow$ P says $X_i \wedge P$ said $(X_1, \ldots, X_n)$
A18) fresh$(X_i) \Rightarrow$ fresh$(X_1, \ldots, X_n)$
A19) fresh$(X_1, \ldots, X_n) \Rightarrow$ fresh(F$(X_1, \ldots, X_n)$)
A20) $(P$ controls $\varphi \wedge P$ says $\varphi) \Rightarrow \varphi$
A21) (fresh$(X) \wedge P$ said $X) \Rightarrow P$ says X
A23) $(P \overset{K}{\leftrightarrow} Q \wedge P$ received $\{X\}_K) \Rightarrow$ P received2 X from Q
A24) R says $K \wedge R$ controls $P \overset{K}{\leftrightarrow} Q \Rightarrow$ R says $P \overset{K}{\leftrightarrow} Q$
A26) P has $X \Rightarrow P$ understands X
A27) P understands $X_i \Rightarrow$ P understands $(X_1, \ldots, X_n)$

〔표 4〕 ASVO 로직의 공개키 유형

	PKx (키의 소유자) (키의 종류)
$PK_{\phi}(P, K)$ 암호화를 위한 공개키	PKe principal principal
$PK_{\sigma}(P, K)$ 서명확인을 위한 공개키	PKd principal principal
$PK_{\delta}(P, K)$ 키 분배를 위한 공개키	PKs principal principal

되어, 연역 시스템을 구성하기 위한 일반적인 기반을 제공한다. 특히 Isabelle 시스템은 HOL내의 인터렉티브한 정리 증명에 중점을 두고 있다. 과거에는 일반 사용자도 ML 함수들, 예를 들어 goal 명령어, tactic, tactical 등을 직접 다루어야 했으나, 오늘날에는 Isar (Intellisable semi-automated reasoning)가 시스템에 포함되어 theory와 증명 단계에 맞추어진 해석 언어 환경을 제공한다[23].

본 논문에서는 ASVO 로직의 구현을 위해서 Isabelle/Isar 시스템을 사용하였다. 따라서 먼저 ASVO 로직을 Isar 언어로 번역하였으며, Isabelle 환경에서 구현 및 검증하였다. 구현된 ASVO 로직을 isatool을 이용하여 Isabelle 시스템에 build하였으며, 이와 같이 완성된 시스템을 Isabelle/ASVO라고 한다. 사용자는 Isabelle이 설치된 환경에서, 쉽게 ASVO 로직을 이용한 검증을 수행할 수 있다. Isabelle/ASVO 도구는 ASVO 로직을 이용한 후향 증명을 수행하게되며, 이와 같이 얻은 최종 결과를 통해서 ASVO 로직의 증명 과정을 산출할 수 있다. 특히 isatool document를 이용하여 생성된 theory들을 다양한 유형의 문서로 제작할 수 있다.

3.3.2 기본 데이터 유형

Isabelle/ASVO에서, ASVO로직은 유형 키(key,pk), 주체(principal) 등의 항(term)을 기본적인 구성 요소로 가지며, 이러한 항들로 이루어진 메시지(msg)와 논리식(formula)으로 로직을 표현한다. ASVO 로직의 기본적인 데이터 유형과 공리들은 모두 ASvo.thy 파일에 저장된다.

(1) 키 표현 방식

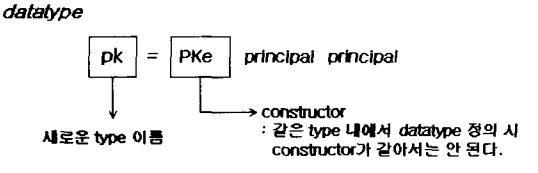
키는 대칭키 표현을 위한 key 유형과, 공개키 표현을 위한 pk 유형으로 나누어서 구현하였다.

1) key type

키는 Isabelle내에 이미 정의되어져 있는 유형인 'nat' (natural number)을 이용하여 표현한다. 키워드 types를 이용하여 key라는 새로운 유형을 선언하였는데, 이것은 단순히 ASvo.thy 파일 내에서 'nat' type을 key라는 이름으로 사용하여 theory의 가독성(readability)을 높이도록 한 것이다.

2) pk type

키의 소유자와 종류를 나타내는 PK(P, K) 구문은 Isabelle의 datatype을 이용하여 'pk'라는 새로운 유형으로 정의하여 표현한다. [표 4]를 참고하라.



(2) 주체

주체를 의미하는 유형 'principal'은 datatype을 이용하여 사용자 A와 B 그리고 키를 관리하는 등의 역할을 하는 Server를 표현하는 constructor들과 type 'pk'에서 키의 종류를 나타내기 위해서 추가된 constructor인 a, b로 이루어져 있다. 추가된 a와 b라는 constructor는

principal A, B와 다른 사용자를 뜻하는 것이 아니며 a와 A, b와 B 각각은 서로 동일한 사용자를 의미한다. 하지만 소문자 a, b는 Diffie-Hellman과 같은 키 교환 프로토콜의 Diffie-Hellman 공개키와, 합의 키를 표시할 때 사용되며, 대문자 A, B는 기타 프로토콜의 대칭키를 표시할 때 사용한다. 예를 들어, 주체에 기반한 키의 표기는 아래와 같이 한다.

· Key (Principal A) (Principal A)
: 주체 A : DH를 제외한 기타 프로토콜에서의 키

· Key (Principal a) (Principal a)
: 주체 A : DH 프로토콜에서의 A의 공개키

· Key (Principal A) (Principal B)
: A와 B 사이의 공유키

3.3.3 메시지와 논리식 유형

(1) 메시지 선언 및 정의

메시지(msg)는 키, 주체, 논스 등의 기본적인 항(term)들로 구성된다. 메시지 역시 Isabelle의 datatype을 이용하여 정의한다. 이렇게 정의되는 메시지의 종류는 Principal, Number, Nonce, Key, H(해쉬함수), Crypt, MPair(메시지 페어링), F0(Diffie-Hellman 변환 함수), F(일반함수) 등의 구성원을 통하여 결정된다.

(2) 논리식 선언 및 정의

논리식(formula)은 참 또는 거짓의 진리값을 갖는 구문을 의미하며, modal 연산자와 ASVO 로직의 각종 predicate으로 구성된다. 논리식 유형은 Isabelle의 datatype을 통해서 다음과 같이 구현되었다.

```
datatype
  formula =  Const          bool
           | Believes       msg       formula
           | Controls       msg       formula
           | Has            msg       msg
           | Received       msg       msg
           | Received2      msg       msg
           | Says           msg       msg
           | Said           msg       msg
           | Fresh          msg
           | From           msg       msg
           | Share          msg       msg       msg
           | PK             pk
           | Understands     msg       msg
           | vague          msg
```

[표 5]에서는 구체적인 논리식 구성 예를 보여준다.

[표 5] Isabelle/ASVO 의 논리식 구성 예제

	formula 유형
Const	Const T/F
Believes	Believes(Principal A)(Fresh(Nonce 1))
Controls	Controls(Principal Server) (Key(Principal A)(Principal B))
Has	Has(Principal A)(Nonce 1)
Received	Received(Principal A) (Key(Principal A)(Principal A))
Received2	Received2(Principal A) (Key(Principal A)(Principal A))
Says	Says(Principal A)(Nonce 1)
Said	Said(Principal A)(Nonce 1)
Fresh	Fresh(Nonce 1)
From	From(Principal A)(Nonce 1)
Share	Share(Principal A)(Principal B) (Key(Principal A) (Principal B))
PK	PK(PKe A A)
Understands	Understands(Principal A)(Nonce 1)
vague	vague(Nonce 1)

(3) 주요 변환

Isabelle/ASVO에서는 유형간 충돌을 방지하고, higer order 공리들의 사용을 위하여 다음과 같이 변환 함수들을 정의한다.

1) Trueprop

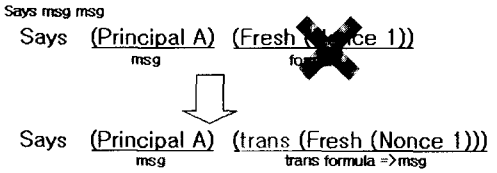
Trueprop은 입력 유형이 'formula'이고 출력으로서 Isabelle 시스템내에 이미 정의되어져 있는 유형인 'prop (proposition)'을 return하는 함수이다. 이 함수는 Isabelle/ASVO 내에 새롭게 정의된 'formula'가 T/F값을 갖는 하나의 명제로서 인식이 가능하도록 한다. 또한, Trueprop이라는 함수는 Isabelle에서 implicit coersion이 일어나므로 다른 일반적인 함수들과 다르게 직접적인 함수 호출 없이도 'formula'에서 'prop'으로 타입 변화가 자동으로 된다.

```
consts
  Trueprop :: " formula => prop"
              (input type)  (output type)
```

2) trans

함수 trans는 유형이 formula인 입력을 받아 그것을 message 유형으로 변환시켜주는 역할을 한다. 이 함

수를 통해서 ASvo 메시지(message)의 "성질 3) φ가 논리식이면, φ는 메시지이다"를 표현할 수 있다. 이러한 trans함수는 아래와 같이 사용한다.



즉, higher order 표현을 구현할 수 있도록 한다.

3.3.4 연산자와 공리 집합

(1) 연산자 선언과 정의

Isabelle/ASVO에서는 논리식 연산자로서 ¬(negation), ∧(and), |(or), →(implication)[3] 등이 새로운 유형인 formula에 적용될 수 있도록 새로 정의하였다.

```
consts
IMP :: "[ formula, formula ] => formula"  ("(_-->_)" 25)
AND :: "[ formula, formula ] => formula"  ("(_&_)" 35)
OR :: "[ formula, formula ] => formula"   ("(_|_)" 30)
NOT :: "formula => formula"               ("(~_)" 40)
```

연산자를 선언할 때에는 함수를 선언할 때와 마찬가지로 Isabelle의 consts를 사용하는데, syntax를 사용하여 선언할 수도 있다. 한편, 연산자에 대한 정의는 Isabelle 시스템의 defs, translations를 사용하거나, constdefs를 이용하여 선언과 정의를 동시에 할 수 있다. 따라서 다음과 같이 메시지 연결자를 정의하였다.

```
translations
"{|x, y, z|}" == "{|x, {|y, z|}|}"----------①
"{|x, y|}" == "MPair x y"    ----------②
```

위의 코드에서 ①은 {| |}의 연산자에 대한 right associativity를 정의하고 있으며, ②은 메시지 연결을 위한 MPair를 앞으로 {| |}으로 대체하여 사용하겠다는 것을 의미한다.

3) →와 ⇒의 차이점: →와 ⇒은 동일하게 논리적 implication을 뜻하지만, ⇒은 Isabelle에서 추론 규칙들(inference rules)을 표현하는 데 사용되며 theorem에서 ⇒의 왼쪽에 위치한 조건 부분과 ⇒의 오른쪽에 위치하는 결론 부분을 구분하는 역할을 한다. 반면, →은 higher-order logic에서 논리식들을 연결하는 역할을 한다.

(2) Isabelle/ASVO 공리 집합

Isabelle/ASVO의 공리 집합은 ASVO 로직의 공리 스키마를, Isabelle/ASVO의 기본 정의와 선언을 바탕으로 구현한 것이다. 따라서 구현에 의존적인 공리 인스턴스들이 발생할 수 있는데, 이와 같이 Isabelle/Isar문법에 이용하여 그대로 구현할 수 없었던 공리들에 한해서는 "Axi#"과 같은 형태로 구현 인스턴스로 명명하였다. 예를 들면, '{| X1,...,Xn |}'와 같은 메시지 구조에서 'Xi'를 선택하는 공리의 경우, Isabelle에서는 메시지를 이루는 항의 개수가 정확하게 정의되지 않는 '{| X1,...,Xn |}'과 같은 형태의 구현은 불가능하므로, 메시지의 항의 개수를 필요에 맞게 정하여 구현을 하였다. 하지만 이와 같은 구현 인스턴스는 필요 이상으로 발생하지는 않았으며, ASVO 로직의 모든 공리가 정확히 구현되었다.

Ⅳ. Isabelle/ASVO를 이용한 검증

4.1 NSSK 프로토콜 검증 준비

NSSK 프로토콜은 Needham과 Schroeder에 의해서 제안된 프로토콜로서, 30여년동안 암호 프로토콜 연구의 초석이 되어왔다. 그러한 가장 큰 이유는 최초의 본격적인 인증 프로토콜로 탄생하여, 약 10여년간 안전한 프로코콜로 여겨져 오며 Kerberos 등의 시스템에 응용되었지만, 그 후 BAN 로직의 등장과 더불어 프로토콜의 근본적인 오류가 발견되면서 이에 대한 많은 연구를 야기시켰기 때문이다. NSSK 프로토콜을 일반적인 프로토콜 표기법을 사용하여 나타내면 다음과 같다.

[프로토콜: NSSK]

1. $A \rightarrow S$: A, B, N_A
2. $S \rightarrow A$: $\{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$
3. $A \rightarrow B$: $\{K_{AB}, A\}_{K_{BS}}$
4. $B \rightarrow A$: $\{N_B\}_{K_{AB}}$
5. $A \rightarrow B$: $\{N_B - 1\}_{K_{AB}}$

즉, NSSK 프로토콜의 목표는 명확히 키분배와 인증에 관한 것이다.

4.1.1 프로토콜 목표

NSSK 프로토콜은 인증과 키 분배를 위해서 설계된 프로토콜로서, ASVO 로직을 통한 검증에서는 다

음과 같은 프로토콜 목표를 정의할 수 있다.

NSSK-G1: A believes $A \xleftrightarrow{K_{AB}} B$
NSSK-G2: A believes fresh(K_{AB})
NSSK-G3: B believes $A \xleftrightarrow{K_{AB}} B$
NSSK-G4: B believes fresh(K_{AB})
NSSK-G5: A believes B says $A \xleftrightarrow{K_{AB}} B$
NSSK-G6: B believes A says $A \xleftrightarrow{K_{AB}} B$

즉, 참여 주체 A와 B는 각각 세션키 K_{AB}의 유효성(공유/신규성)에 대해서 신뢰해야하며, 서로 상대방이 같은 키에 대해서 언급을 하는 사실을 신뢰해야 하는 것을 의미한다.

4.1.2 프로토콜 초기 가정

ASVO 로직을 통해서 검증하기 위해서는, 이미 언급한 바와 같이 프로토콜 가정인 ISA와 RMA를 정의해야한다. 이것은 각각 초기 설정에 대한 주체의 신뢰와, 프로토콜 메시지 수신에 대한 주체의 신뢰에 대해서 정의하는 것이다. 이 단계는 검증자가 프로토콜에 대한 간단한 이해를 하는 단계라고 할 수 있다. 먼저 각 주체 A와 B는 TTP에 해당하는 S와 각자 안전한 키를 공유하고 있음을 신뢰하여야 한다.

P1) A believes $A \xleftrightarrow{K_{AS}} S$
P2) B believes $B \xleftrightarrow{K_{BS}} S$

또한 S가 TTP로서 가질 수 있는 관할 권한에 대해서 신뢰하여야 한다. 즉, NSSK는 양자간의 키 분배를 관할하는 TTP를 가정한 프로토콜이다. 결과적으로 B가 갖는 신규성에 대한 가정은 생략하였다. 하지만 프로토콜의 초기 검증에서는 이와 같은 것들도 규정할 필요가 있다.

P3) A believes S controls $A \xleftrightarrow{K} B$
P4) B believes S controls $A \xleftrightarrow{K} B$
P5) A believes S controls fresh(K)

각 주체는 자신이 만든 논스 값에 대한 신규성 그리고 소유에 대한 신뢰를 하며, 상대방의 아이디를 식별할 수 있다. 이에 대한 가정 역시 실제 증명에 사용되는 것들만 요약하였다.

P6) A believes fresh(N_A)
P7) A believes A has N_A
P8) B believes fresh(N_B)
P9) B believes B has N_B
P94) B believes B understands A

하지만 이미 알려진 바와 같이 NSSK 프로토콜은 논리적인 취약점을 가지고 있는 프로토콜이며, ASVO 로직 분석을 통해서 이와 같은 취약점을 명확히 발견하고 수정할 수 있었다.

아래에 *표로 표시한 ISA들, 즉 P91, P92, P93은 NSSK 프로토콜이 앞에서 정의한 목표를 ASVO 로직을 통해서 이루기 위해서 추가적으로 필요했던[4] 가정들이며, 결과적으로 이와 같은 가정들은 각각 NSSK 프로토콜의 논리적 문제점을 제공하게 된다.

*P91) A believes A understands N_B
*P92) A believes fresh(N_B)
*P93) B believes fresh(K_{AB})

말하자면, P91에서는 주체 A가 상대방 B의 논스 값 N_B에 대해서 일방적으로 이해할 수 있어야한다는 것을 의미하며, P92와 P93에서는 주체 A와 B가 각각 상대방의 논스 값이나, TTP가 생성해준 세션 키에 대해서 신규성을 일방적으로 신뢰해야만 한다는 것을 의미한다. 하지만 이와 같은 초기 가정은, 프로토콜의 기본적인 명세에 대해서 위배되는 것이며 따라서 프로토콜을 이와 같은 가정이 필요하지 않도록 수정해야한다. 이것은 ASVO 로직을 통해서 수정된 NSSK7 프로토콜에서 찾아볼 수 있다.

한편, ISA를 기술한 후, 수신 메시지에 대한 주체의 신뢰 명세인 RMA를 다음과 같이 기술해야한다. 앞에서 설명한 바와 같이 ASVO 로직의 RMA 기술은 근본적으로 SVO 로직의 RMA 기술과 다르다. ASVO 로직의 RMA는 단지 각 주체가 프로토콜에 의해서 받기로 약속된 메시지 구조를 기술하고, 이와 같은 전체 메시지 구조에 대한 해당 주체의 신뢰를 나타내는 것이다. 따라서 그 단계가 매우 단순하다. 다만, P15와 같이 challenge-response에 대한 함수 표기

4) ASVO 로직 반자동 도구인 Isabelle/ASVO를 통해서 검증을 하면, P91, P92, P93과 같은 subgoal을 해결하기위한 요구가 발생한다. 따라서 위와 같은 추가적인 가정을 더 해야만 증명 과정을 통과할 수 있다. 한편 P92에 대해서는 다른 메시지 블록을 선택하여 대치할 수 있었다.

〔표 3〕 Isabelle/ASVO를 이용한 증명 사례 출력

```
proof (prove): step 3

goal(theorem (Goal), 3 subgoals):
[| ASvo.Trueprop (Believes (Principal B) (Received (Principal B) (Crypt (Key (Principal B) (Principal Server)) {|Key (Principal A) (Principal B), Principal A|})));
   ASvo.Trueprop (Believes (Principal B) (Share (Principal B) (Principal Server) (Key (Principal B) (Principal Server))));
   ASvo.Trueprop (Believes (Principal B) (Understands (Principal B) (Principal A)));
   ASvo.Trueprop (Believes (Principal B) (Controls (Principal Server) (Share (Principal A) (Principal B) (Key (Principal A) (Principal B))))) |]
==> ASvo.Trueprop (Believes (Principal B) (Share (Principal A) (Principal B) (Key (Principal A) (Principal B))))
 1. [| ASvo.Trueprop (Believes (Principal B) (Received (Principal B) (Crypt (Key (Principal B) (Principal Server)) {|Key (Principal A) (Principal B), Principal A|})));
       ASvo.Trueprop (Believes (Principal B) (Share (Principal B) (Principal Server) (Key (Principal B) (Principal Server))));
       ASvo.Trueprop (Believes (Principal B) (Understands (Principal B) (Principal A)));
       ASvo.Trueprop (Believes (Principal B) (Controls (Principal Server) (Share (Principal A) (Principal B) (Key (Principal A) (Principal B))))) |]
    ==> [redacted]
 2. [| ASvo.Trueprop (Believes (Principal B) (Received (Principal B) (Crypt (Key (Principal B) (Principal Server)) {|Key (Principal A) (Principal B), Principal A|})));
       ASvo.Trueprop (Believes (Principal B) (Share (Principal B) (Principal Server) (Key (Principal B) (Principal Server))));
       ASvo.Trueprop (Believes (Principal B) (Understands (Principal B) (Principal A)));
       ASvo.Trueprop (Believes (Principal B) (Controls (Principal Server) (Share (Principal A) (Principal B) (Key (Principal A) (Principal B))))) |]
    ==> ASvo.Trueprop (Said (Principal Server) {|Key (Principal A) (Principal B), Principal A|})
 3. [| ASvo.Trueprop (Believes (Principal B) (Received (Principal B) (Crypt (Key (Principal B) (Principal Server)) {|Key (Principal A) (Principal B), Principal A|})));
       ASvo.Trueprop (Believes (Principal B) (Share (Principal B) (Principal Server) (Key (Principal B) (Principal Server))));
       ASvo.Trueprop (Believes (Principal B) (Understands (Principal B) (Principal A)));
       ASvo.Trueprop (Believes (Principal B) (Controls (Principal Server) (Share (Principal A) (Principal B) (Key (Principal A) (Principal B))))) |]
    ==> ASvo.Trueprop (Believes (Principal B) (Controls (Principal Server) (Share (Principal A) (Principal B) (Key (Principal A) (Principal B)))))
```

는 RMA 단계에서 고려되어야 한다.

P11) S received(A, B, N_A)

P12) A believes A received $\{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$

P13) B believes B received $\{K_{AB}, A\}_{K_{BS}}$

P14) A believes A received $\{N_B\}_{K_{AB}}$

P15) B believes B received $\{F(N_B)\}_{K_{AB}}$

이와 같은 초기 명세를 마친 후, 각 목표에 대한 theorem을 기술한다. 그리고 해당 theorem에 대해서 ASVO 로직을 이용한 증명을 하게된다.

4.2 NSSK 프로토콜 검증

프로토콜의 분석 실행 과정은 명백하다. 먼저 프로토콜 목표에 대한 theorem을 정리하여야 한다. 그리고 이이서 각 theorem을 증명하기 위한 ASVO 로직의 공리 적용을 하여야한다. 이와 같은 과정은 반자동 검증 도구인 Isabelle/ASVO를 통해서 유도되는 subgoal들의 형태에 따라서 단계적으로 이룰 수 있다. 물론 최초의 가정은 RMA에서 비롯되며, 이와 관계가 있는 ISA들도 여기에 포함된다. 해당 목표에 대한 증명이 진행됨에 따라서, 선행 목표에서 비롯되어 증명된 theorem이나 lemma들 역시 후행 목표의 가정으로 포함될 수 있다. 다만 주의하여야 할 것은, 각 가정들의 modal 연산자의 주체가 해당 목표의 그것과 반드시 동일하여야 하다는 것이다. 이것은 ASVO 로직을 통한 자동 증명이 Gödel의 두 번째 공리에 바탕을 둔 증명이기 때문이다. 즉, ASVO 로직에서는 A2 공리와 Nec을 이용한 증명 과정을 허용한다

4.2.1 NSSK-G1에 대한 증명

NSSK-G1은 각각 주체 A가 키에 대해서 얻을 수 있는 두가지 성질, 즉 대칭키의 유효성과 신규성에 대한 증명이다. 따라서 두가지 theorem을 정의하고 이에 대해서 증명할 수 있다. 먼저 G1에 대한 theorem을 정의하고 이에 대한 증명 과정을 요약한다. 표기 ↬는(A1, A2, Nec, MP) 중에서 적절한 것을 사용하거나 그대로 다음 단계로 전이하는 것을 의미한다.

[Theorem NSSK-G1]

$\vdash$ P12 ; P1 ; P7 ; P6 ; P3 $\Rightarrow$

A believes $A \overset{K_{AB}}{\longleftrightarrow} B$

(증명)

P12↬P1↬A23↬P7↬A26↬A27↬A5↬A17↬P6↬A18↬*A21*↬A17a1↬P3↬A24↬A20(A1,A2,Nec,MP)

=> NSSK-G1: A believes $A \overset{K_{AB}}{\longleftrightarrow} B$

증명 과정을 요약하였지만, A21 공리 적용을 통해서

다음과 같은 중간 논리식을 얻을 수 있었다.

G1_D2a) A believes
S says($N_A, B, K_{AB}, \{K_{AB}, A\}_{K_{as}}$)

이것은 G2에 대한 증명에 활용된다.

[Theorem NSSK-G2] $\vdash$ P5 ; G1_D2a $\Rightarrow$
A believes fresh(K_{AB})

(증명)
G1_D2a↬A17a1↬P5↬A24a1↬A20 (A1,A2,Nec,MP)
=> NSSK-G2: A believes fresh(K_{AB})

이와 같이 NSSK 프로토콜의 목표 G1과 G2는 무난하게 증명되므로, 이에 대해서는 NSSK 프로토콜이 취약점을 갖지 않는다는 사실을 확인할 수 있다.

4.2.2 NSSK-G3에 대한 증명

NSSK-G3은 각각 주체 B가 키에 대해서 얻을 수 있는 두가지 성질, 즉 대칭키의 유효성과 신규성에 대한 증명이다. 따라서 역시 두가지 theorem에 대해서 증명해야한다. 다음과 같이 NSSK-G1과 유사한 유형의 theorem을 정의하고 증명을 한다. [표 3]의 출력결과는 다음의 theorem 증명 과정의 한 부분의 출력결과를 캡쳐한 것이다.

[Theorem NSSK-G3F]
$\vdash$ P13 ; P2 ; P94 ; P9 ; P8 ; P4 $\Rightarrow$
B believes $A \xleftrightarrow{K_{AB}} B$
*** 증명 불가

Isabelle/ASVO의 subgoal 출력인 ASvo.Trueprop(Fresh {|Key(Principal A)(Principal B), Principal A|})가 의미하는 바는, B가 메시지 $\{K_{AB}, A\}$에 대한 신규성을 확인할 수 있어야한다는 것이다. 따라서, 우리는 이와 같은 subgoal을 해결하기 위해서 다음과 같이 subgoal 자체를 가정으로서 준비해야하며, 결과적으로 새로운 theorem 정의와 증명이 필요하다.
P93) B believes fresh(K_{AB})

즉, 주체 B는 K_{AB}의 신규성을 항상 확인할 수 있어야한다는 조건이다. 하지만 이것은 증명을 완수하는 데는 필요하지만, 프로토콜에서 기본적으로 설정할 수 없는 가정이다. 결과적으로 이로 인하여 프로토콜이 안전하지 않게 된다.

[Theorem NSSK-G3]
$\vdash$ P13 ; P2 ; P94 ; *P93* ; P4 $\Rightarrow$
B believes $A \xleftrightarrow{K_{AB}} B$

(증명)
P13↬P2↬A23↬P94↬A27↬A5↬A17↬*P93*↬A18↬A21↬A17a1↬P4↬A24↬A20 (A1,A2,Nec,MP)
=> NSSK-G3: B believes $A \xleftrightarrow{K_{AB}} B$

이후 B에 관한 증명인, G4와 G6은 G3의 증명을 통해서만 증명 가능하다. 하지만 결과적으로 P93의 제거를 위해서 프로토콜의 수정이 필요하다. 본 논문에서는 이와 같은 관점에서 보완된 프로토콜인 NSSK7 프로토콜을 구성하였다.

4.2.3 NSSK-G5에 대한 증명

NSSK-G5는 주체 A가 상대방 B의 키 인증 사실을 확인하는데 대한 증명이다. 하지만 G3과 유사하게 해당 theorem을 증명할 수 없었으며, 결국 다음과 같이 P91을 포함한 theorem에 대해서 증명해야했다.
P91) A believes A understands N_B

이것은 증명 과정에서 사실상 A5 공리를 적용하기 위해서 요구되었다. 아래의 NSSK-G5 theorem은 P91을 포함하여 재증명된 결과이다. P91로 인하여 A5 공리가 적용됨을 확인할 수 있다.

[Theorem NSSK-G5]
$\vdash$ P14 ; G1; *P91* ; G2 $\Rightarrow$
A believes B says $A \xleftrightarrow{K_{AB}} B$

(증명)
P14↬G1↬A2↬A23↬*P91*↬A5↬G1↬G2↬A25↬A18↬A21↬A17a1 (A1,A2,Nec,MP)
=> NSSK-G5: A believes B says $A \xleftrightarrow{K_{AB}} B$

즉, NSSK-G3에서와 유사하게 P91로 인하여 안전성에 대한 문제점이 있음을 확인할 수 있다. NSSK7 프로토콜에서는 역시 이와 같은 문제점도 수정하였다.

4.3 NSSK 프로토콜의 문제점

결과적으로 NSSK 프로토콜은 목표 중에서 NSSK-G1과 NSSK-G2에 대해서는 기본적인 가정만으로 증명이 가능하지만, NSSK-G3과 NSSK-G5는 증명을 위해서 각각 P93과 P91이라는 비정상적인 가정을 필요로 하였다. 또한 NSSK-G4와 NSSKG6은 NSSK-G3의 증명 결과를 필요로 하므로, 역시 P93을 필요로 하였다. 따라서 NSSK 프로토콜의 문제점을 우리는 P91과 P93이라는 요구사항에서 발견할 수 있다.

4.3.1 NSSK 프로토콜에 대한 Denning-Sacco 공격

증명을 위해서 P93이 필요하였는데 이것은, B가 K_{AB}에 대한 신규성을 확인할 수 없었으며, 이와 같은 문제는 수정되어야한다는 것을 의미한다. 실제로 NSSK 프로토콜은 Dennig-Sacco 공격에 노출되는 프로토콜로서 그 취약성이 이미 밝혀진바 있다. 이것은 ASVO 로직의 분석 결과와 일치하는데, 즉 주체 B가 K_{AB}의 신규성을 확인할 수 없다는 사실에서 기인한다.

예를 들면 아래와 같이 이전 세션의 키 K_{AB}를 취득한 공격자 E는 A를 가장하여 B와 프로토콜을 수행할 수 있다.

$E(A) \rightarrow B$: $\{K_{AB}, A\}_{K_{BS}}$
$B \rightarrow E(A)$: $\{N_B\}_{K_{AB}}$
$E(A) \rightarrow B$: $\{N_B - 1\}_{K_{AB}}$

이때 키 K_{AB}의 신규성을 확인할 수 없는 주체 B는 A와의 정상적인 프로토콜 수행으로 오인하게 된다. 하지만 실제로 A는 본 프로토콜에 참여하지 않았다.

4.3.2 NSSK 프로토콜에 대한 Dumb Authentication 공격

또한 NSSK-G5의 증명을 위해서 P91이 필요하였다. 즉, A는 B로부터 수신한 메시지를 인식할 수 없었으며, 이와 같은 문제는 수정되어야한다는 것을 의미한다. 예를 들면, 다음과 같은 새로운 형태의 간단한 공격이 가능하다. 이것을 우리는 Dumb Authentication 공격이라고 부르기로 하자.

$A \rightarrow S$: A, B, N_A
$S \rightarrow A$: $\{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$
$A \rightarrow E(B)$: $\{K_{AB}, A\}_{K_{BS}}$
$E(B) \rightarrow A$: X
$A \rightarrow E(B)$: $\{\{X\}_{K_{AB}} - 1\}_{K_{AB}}$

즉, B를 가장한 공격자 E는 A와의 메시지 교환을 통해서, A가 마치 B와 프로토콜을 정상적으로 마친 것으로 오해하도록 할 수 있다. 하지만 실제로 B는 본 프로토콜에 전혀 참여하지 않았다.

4.4 NSSK7 프로토콜 설계

NSSK7 프로토콜은 ASVO 로직을 통해서 발견된 NSSK 프로토콜의 문제점을 보완 수정한 프로토콜이다. 즉, NSSK 프로토콜이 ASVO 로직을 이용한 증명을 위해서 NSSK-P91이나 NSSK-P93과 같은 비정상적인 가정을 필요로 했던 반면, NSSK7 프로토콜은 이와 같은 가정을 요구하지 않도록 보완한 것이다. 이것은 기존의 보완된 NSSK 프로토콜과 그 형태가 유사하지만, 명확히 ASVO 로직의 분석 결과를 따른다. ASVO 로직 분석을 바탕으로 수정한 NSSK7 프로토콜에 대해서 살펴본다.

4.4.1 NSSK7 프로토콜 도출

(1) NSSK-P91에 대한 보완

NSSK-P91의 문제점은 NSSK 프로토콜의 다음과 같은 단계에서 A가 B의 메시지를 인식할 수 있도록 해야한다는 것이다.

$B \rightarrow A$: $\{N_B\}_{K_{AB}}$

따라서, 메시지에 A가 기본적으로 식별할 수 있는 정보를 포함하도록 하며, 그 예로서 주체의 식별자를 포함하도록 한다. 따라서 다음과 같이 해당 메시지가 수정되며, 정상적인 가정을 추가하도록 한다.

$B \rightarrow A$: $\{A, B, N_{B_2}\}_{K_{AB}}$
P8) A believes A understands B(또는 A)

즉, A는 복호화된 메시지를 이해할 수 있으며, 이것은 A5와 같은 이해와 수신에 관련된 공리를 통해서 증명 가능하다는 것을 의미한다.

(2) NSSK-P93에 대한 보완

NSSK-P93의 문제점은 NSSK 프로토콜의 다음과 같은 단계에서 B가 세션키 K_{AB}의 신규성을 확인하는 방법이 필요하다는 것이다.

$A \rightarrow B$: $\{K_{AB}, A\}_{K_{BS}}$

따라서, 주체 B가 생성한 논스 값이 해당 메시지에 포함되도록 프로토콜을 수정하여야 한다. 따라서 새로운 논스 값을 S에게 전달하고 받도록 하기 위하여 다음과 같은 단계 추가가 필요하다.

$B \rightarrow A$: B, N_{B1}
$A \rightarrow S$: A, B, N_A, N_{B_1}
$A \rightarrow B$: $\{N_{B_1}, K_{AB}, A\}_{K_{BS}}$

결국 ASVO 로직을 통해서 NSSK 프로토콜은 7단계 프로토콜로 변환되며, 이것은 이미 언급한 바와 같이 기존의 보완 방법과 유사한 면이 있다. 예를 들면, GNY 로직을 통해서 보완된 프로토콜이나 Needham과 Schroeder가 보완한 것과 결과적으로 유사하다. 하지만, Needham과 Schroeder의 보완에서는 Dumb Authentication이 고려되지 않았으며, GNY 로직을 통한 보완에서는, 불필요한 암호화를 유지하였다[10]. NSSK7 프로토콜의 경우 ASVO 로직을 통해서 논리적으로 명확한 재구성을 하였으며, 보완 프로토콜 중에서 사실상 가장 효과적인 구성을 이루는 결과를 보여주었다.

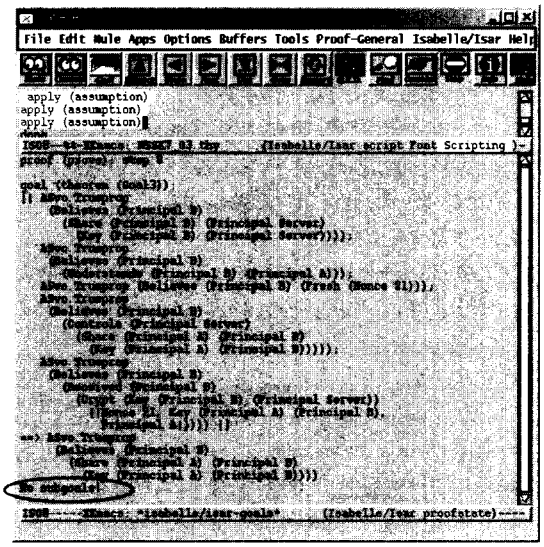
4.4.2 NSSK7 프로토콜

이와 같이 우리는 7단계로 이루어진 최적의 NSSK7 프로토콜을 얻을 수 있다.

$A \rightarrow B$: A
$B \rightarrow A$: B, N_{B1}
$A \rightarrow S$: A, B, N_A, N_{B_1}
$S \rightarrow A$: $\{N_A, B, K_{AB}, \{N_{B_1}, K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$
$A \rightarrow B$: $\{N_{B_1}, K_{AB}, A\}_{K_{BS}}$
$B \rightarrow A$: $\{A, B, N_{B_2}\}_{K_{AB}}$
$A \rightarrow B$: $\{N_{B_2} - 1\}_{K_{AB}}$

NSSK7 프로토콜은 ASVO 로직을 통하여 기존의 NSSK 프로토콜에서 발생하는 논리적인 문제점을 해결한 프로토콜이다. 위의 [그림 1]은 NSSK7-G3에 대해서, NSSK-P93과 같이 ISA로서는 불가능한 가정을 포함하지 않았음에도 무난히 증명되어[5], G3에 대한 추론 구성을 성공하는 Isabelle/ASVO의 화면을 보여주고 있다. NSSK7-G5 역시 NSSK-P91과 같은 가정 없이 증명되었다.

5) Isabelle/ASVO의 No subgoals! 메시지를 확인하라.



[그림 1] NSSK7 프로토콜의 G3 증명 예

이와 같이 구성된 NSSK7 프로토콜은 이미 언급한 바와 같이 GNY 로직을 통해서 재구성된 NSSK 프로토콜과 여러 가지로 가장 유사하다[10]. 그 이유는 BAN 로직이나 SVO 로직으로는 분석할 수 없었던 NSSK 프로토콜의 dumb authentication 취약성을 분석하는 면이 GNY의 recognizability 분석과 유사하기 때문이다.

V. 결론 및 향후 연구

본 논문에서는 암호프로토콜 논리성 자동 검증을 위한 방법으로서, SVO 로직의 자동화 방법을 제안하였다. 이를 위하여 먼저 SVO 로직 자동화의 문제점을 분석하였으며, 자동화를 고려한 ASVO 로직을 설계하였다. 또한 Isabelle 시스템상에서 자동 검증 도구인 Isabelle/ASVO를 구현하였다. 현재 본 논문에서 소개한 Isabelle/ASVO를 이용하여 NSSK와 NSSK7 프로토콜 이외에도 NSPK(Needham-Schroeder Public Key) 프로토콜에 대한 검증도 하였으며, 역시 발견된 취약성을 수정한 프로토콜 NSPK2와 NSPK3, 그리고 키 교환 프로토콜인 Diffie-Hellman 및 STS(Station-to-Station) 프로토콜에 대해서도 검증하였다. 한편 ASVO 로직에 대한 자세한 사항은 별도의 논문을 통해서 다루도록 한다.

참 고 문 헌

[1] 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.

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

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

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

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

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

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

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

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

[15] 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.

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

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

[18] 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.

[19] 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.

[20] P. Syverson and P. van Oorschot, "A unified cryptographic protocol logic," NRL Publication 5540-227, Naval Research Lab, 1996.

[21] P. Syverson and Iliano Cervesato, "The logic of authentication protocols," Lecture Notes in Computer Science, Vol. 2171, Springer-Verlag, 2002.

[22] 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.

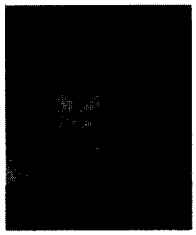
[23] T. Nipkow, L. Paulson, and M. Wenzel, "Isabelle/HOL", Lecture Notes in Computer Science, Vol. 2283, Springer-Verlag, 2002.

〈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 조교수, 정보보호학회 편집위원, TTA 암호분과 특별위원
<관심분야> 정보보호, 암호프로토콜 등

양 숙 현 (Sookhyun Yang)
2003년 : 연세대학교 컴퓨터과학과 졸업
2003년~현재 : 한국과학기술원 전자전산학과 석사과정
<관심분야> 정보보호, 암호이론 등

김 승 주 (Seungjoo Kim) 종신회원
1994년 : 성균관대학교 정보공학과 공학사
1996년 : 성균관대학교 대학원 정보공학과 공학석사(암호학 전공)
1999년 : 성균관대학교 대학원 정보공학과 공학박사(암호학 전공)
1999년~2002년 : 한국정보보호진흥원(KISA) 암호기술팀장
2003년~현재 : 한국정보보호진흥원(KISA) 기술표준팀장
2000년~현재 : 한국정보통신기술협회(TTA) 정보통신기술위원회 암호기술연구반 의장
<관심분야> 암호이론 등



임 선 간 (Seongan Lim) 종신회원
1985년 : 동국대학교 수학과 학사
1987년 : 서울대학교 수학과 석사
1995년 : Purdue 대학교 수학과 박사
1999년~현재 : 한국정보보호진흥원 암호기술팀 선임연구원
<관심분야> 암호프로토콜, 정보보호 등