Journal of Knowledge Information Technology and Systems(JKITS), Vol. 14, No. 6, pp. 711~718, December 2019
https://doi.org/10.34163/jkits.2019.14.6.014

**Journal of Knowledge Information Technology and Systems**

ISSN 1975-7700

http://www.kkits.or.kr

# Formal Specification of Cryptographic Security Protocols

**Gyesik Lee***

*Department of Computer Science and Engineering, Hankyong National University*

## ABSTRACT

We introduce a way of formal specification of cryptographic security protocols. We demonstrate how to use formal methods in verifying cryptographic security protocols. Each step of the verification is explained with a concrete example, the Needham-Schroeder public-key protocol. The style of describing the syntax is similar to that of the tool Scyther. It is simple and intuitive to use for the specification of security protocols. Our description corresponds also to the basic concepts of the tool ProVerif. Specifications of cryptographic security protocols are represented at the most abstract level: Messages are terms constructed from symbols and the attacker is modeled as a formal process. A formal language $L = (V, C, R, F)$ for public-key protocols consists of a set $V$ of variables for received messages, a set $C$ of local constant symbols, a set $R$ of role name symbols, and a set $F$ of function symbols. Then a protocol describes the behavior of each of the roles such as initiator, responder, key server, etc. In the specification, the behavior of each role is formalized as a transition system describing how to create messages, how to react to the received messages, and how to manipulate messages. Among many security properties, we illustrate how to formally handle secrecy and authentication.



**KEYWORDS:** Cryptographic security protocols, Formal specifications, Automatic verification, Formal methods, Scyther, ProVerif

**ARTICLE INFO:** Received 21 November 2019, Revised 2 December 2019, Accepted 7 December 2019.

*Corresponding author is with the Department of Computer Science and Engineering, Hankyong National University, 327 Jungang-ro Anseong-si Gyeonggi-do, 17579, KOREA.

*E-mail address*: gslee@hknu.ac.kr

## 1. 서 론

디지털 통신의 보안은 다양한 요소들에 의존하며 특히 암호화의 중요성이 크다. 디지털 통신의 암호화와 관련된 기술과 문제는 다양하다. 예를 들어, 암호화 알고리즘의 안전성, 알고리즘 구현의 신뢰성, 암호화 시스템의 정확성, 그리고 암호화 보안 프로토콜의 안전성 등이 주요 관련 요소들이다. 이 중에서 암호화 보안 프로토콜의 안전성을 제외한 다른 요소들의 명세에 대한 표준화는 널리 알려져 있다. 반면에 암호화 보안 프로토콜의 명세에 대해서는 일반적으로 합의된 표준화는 존재하지 않는다.

암호화 보안 프로토콜은 보안 연결을 사용하여 통신 당사자들의 사생활 보호와 데이터의 무결성을 보장하면서 서로 통신하는 데에 사용된다. 암호화 보안 프로토콜의 명세를 정형화(formalization)하면 이를 바탕으로 암호화 보안 프로토콜의 안전성을 검증할 수 있다. 예를 들어, 스카이터(Scyther)[1-3], 프로베리프(ProVerif)[4-6] 등 이미 잘 알려진 암호화 보안 프로토콜 자동 검증 도구들을 이용하여 프로토콜의 안전성을 검증하는 데에 활용될 수 있으며, 심지어 자동차 보안 시스템에서 사용되는 암호화 보안 프로토콜의 검증에 사용될 수 있다[7-9].

본 논문은 암호화 보안 프로토콜의 안전성 검증에 필수요소인 정형명세(formal specification)에 대한 한 가지 방법을 소개하며, 구체적인 논문 구성은 다음과 같다.

2절은 보안 프로토콜의 정형명세 언어와 암호화 프로토콜의 명세를 작성하는 방법을 소개한다. 3절에서는 공격자가 존재하는 네트워크에서 암호화 프로토콜 통신 참가자의 지식상태가 변화하는 과정인 상태전이를 정의하고 설명한다. 4장은 상태전이궤적의 성질을 이용하여 정형검증이 가능한 기밀성(secrecy)과 인증(authentication)을 정의하고 활용법을 소개한다. 끝으로 5장에서는 본 연구결과의 결론을 소개하며 마무리한다.

## 2. 보안 프로토콜 정형명세

먼저 암호화 보안 프로토콜 명세언어를 정의하고, 이후에 간단한 예제를 이용하여 암호화 보안 프로토콜을 검증하는 방법을 소개한다. 사용하는 예제는 니드햄-슈뢰더(Needham-Schroeder) 공개키 프로토콜이다. 아래에서 소개되는 명세언어는 암호화 보안 프로토콜 검증도구인 스카이터와 프로베리프의 작동원리를 바탕으로 하여 작성되었다.

### 2.1 정형명세 언어

니드햄-슈뢰더 공개키 프로토콜에 대한 언어

$$L = ( V, C, R, F )$$

은 다음 네 개의 집합으로 구성된다.

- $V$ : 수신된 메시지를 저장하는 변수들의 집합
- $C$ : 일회성 난수(nonce)를 가리키는 지역 상수 기호들의 집합
- $R$ : 역할 이름(role name) 기호들의 집합
- $F$ : 공개키 생성함수 $pk$, 개인키 생성 함수 $sk$, 순서쌍 생성 함수 $(\cdot,\cdot)$, 순서쌍을 분해하는 사영함수 $\pi_1$과 $\pi_2$, 암호화 함수 $enc$, 복호화 함수 $dec$ 등으로 구성된 집합

암호화 함수 $enc$와 복호화 함수 $dec$는 오류를 발생시키지 않으며, 일종의 블랙박스처럼 작동하고 몇 가지 관련 성질만 알려졌다고 가정한다.

$L$ 언어의 항(term)은 $dec(sk(r),enc(pk(r),m))$, $\pi_i(m_1,\ m_2)$ 등처럼 함수 기호와 변수, 상수 기호, 역할 이름 등의 조합으로 생성되며 특별한 제한 조건은 없다. 위에서 $r,m,m_1,m_2$ 등은 임의의 항을 가리킨다. 각 항마다 고유의 자료형이 존재하고 지정된 자료형의 인자들만 함수들의 인자로 사용될 수 있다. 또한 항들 사이의 대수적 관계가 다음과 같이 성립한다고 가정한다.

$$dec(sk(r),\ enc(pk(r),m)) = m$$
$$\pi_i(m_1,\ m_2) = m_i$$

함수들의 집합 $F$ 에 배타적 논리합(exclusive or) 연산자와 함께 아래 대수식을 추가하면 암호화 알고리즘이 약해질 수 있음이 잘 알려져 있다[10].

$$(x\oplus y)\oplus z = x\oplus(y\oplus z)$$
$$x\oplus y = y\oplus x$$
$$0\oplus x = x$$
$$x\oplus 0 = x$$
$$x\oplus x = 0$$
$$enc(k,x\oplus y) = enc(k,x)\oplus enc(k,y)$$

배타적 논리합을 스카이터, 프로베리프 등과 같은 자동 검증기에서 사용하는 것과 관련된 많은 이슈들이 존재한다[11].

## 2.2 프로토콜 명세

니드햄-슈뢰더 공개키 프로토콜 언어 $L$ = ( $V$, $C$, $R$, $F$ )를 이용하여 프로토콜을 작성할 수 있다. 프로토콜은 개시자(initiator), 반응자(responder), 키 서버(key server) 등 통신 당사자들의 역할을 명세한다. 즉, 메시지 생성, 수신된 메시지에 대한 대응, 메시지 조작 등의 연속적인 처리과정을 기술한다. 이 논문에서는 단순한 설명을 위해 통신이 하나의 공개채널을 통해서만 이루어진다고 가정한다.

**메시지 송신과 수신 이벤트** : Send와 Receive 함수를 이용한다.

- Send($i,r,m$) : $i$가 $r$에게 메시지 $m$ 보내기
- Receive($i,r,m$) : $r$이 $i$에게서 메시지 $m$ 받기

**역할 명세(role specification)** : 프로토콜은 통신 당사자들의 역할 명세로 이루어지며, 역할 명세는 유한 개의 수신 및 송신 이벤트의 연속으로 구성된다. 또한 각각의 역할은 자신의 개인키와 함께 다른 통신 참여자들의 역할 이름과 그들의 공개키에 대한 정보를 함께 갖는다고 가정한다.

니드햄-슈뢰더 공개키 프로토콜은 두 개의 역할 명세를 갖는다.

- 개시자(initiator):

| |
|---|
| Send( $I,\ R,\ \{N_i,I\}_{pk(R)}$ ); |
| Receive( $R,\ I,\ \{N_i,x\}_{pk(I)}$ ); |
| Send( $I,\ R,\ \{x\}_{pk(R)}$ ) |

그림 1. 니드햄-슈뢰더 공개키 개시자 프로토콜
Figure 1. Needham-Schroeder public key initiator protocol

- ✓ 역할 $I$가 자신이 생성한 메시지 $N_i$와 자신의 역할 이름 $I$를 $R$의 공개키를 이용하여 암호화한 후에 $R$에게 송신.
- ✓ 역할 $I$가 $R$로부터 $I$ 자신이 $R$에게 보낸 메시지 $N_i$와 함께 $R$이 임의로 생성한 메시지 $x$를 $I$ 자신의 공개키를 이용하여 암호화하여 보내진 메시지 수신.
- ✓ $R$에게서 수신하여 해독한 메시지 $x$를 $R$의 공개키를 이용하여 암호화한 후에 $R$에게 다시 송신.

- **반응자(responder):**

$\mathrm{Receive}(I,\ R,\ \{y, I\}_{pk(R)})$;
$\mathrm{Send}(R,\ I,\ \{y, N_r\}_{pk(I)})$;
$\mathrm{Receive}(I,\ R,\ \{N_r\}_{pk(R)})$

그림 2. 니드햄-슈뢰더 공개키 반응자 프로토콜
Figure 2. Needham-Schroeder public key responder protocol

- ✓ 역할 $I$가 생성한 임의의 메시지 $y$와 $I$ 자신의 역할을 $R$의 공개키를 이용하여 암호화하여 보내진 메시지를 $R$이 수신.
- ✓ $R$이 $I$로부터 수신한 메시지로부터 해독한 메시지 $y$와 함께 $R$ 자신이 임의로 생성한 메시지 $N_r$을 $I$의 공개키를 이용하여 암호화한 후 $I$에게 송신.
- ✓ $I$로부터 $R$ 자신이 보낸 메시지 $N_r$를 $R$ 자신의 공개키를 이용하여 암호화한 후에 보내진 메시지 수신.

**계산(computation)** : 각 역할을 담당하는 에이전트(agent)는 어떤 이벤트를 발생시키기 전에 항상 특정 계산을 먼저 실행한다. 반면에 이벤트가 발생한 후에는 각 에이전트의 지식 상태(knowledge state)가 변할 수 있으며 추가되는 지식은 특정 계산을 통해 얻어진다. 계산 내용은 사용하는 모델에 따라 다르며, 일반적으로 아래의 두 계산을 수행한다.

- 패턴 매칭: 모든 에이전트는 다루는 메시지의 패턴을 바로 알아챌 수 있다고 가정한다. 즉, 일화성 난수(nonce), 상대 에이전트 이름, 세션키, 순서쌍, 암호화 메시지 등등을 식별할 수 있다. 알려진 패턴에 어긋나는 메시지는 기본적으로 무시된다.
- 지식 추론: 공격자를 포함한 모든 에이전트는 패턴 매칭을 통해 알아낸 정보를 바탕으로 새 지식을 추론한다. 지식 추론은 에이전트의 기존 지식과 함께 합성, 분해, 암호화, 해독 등의 기능을 활용하여 이루어진다.

## 3. 지식상태 전이

지식상태 전이가 발생하는 과정을 묘사한다. 여기서는 각 역할 당사자들이 이벤트를 동시에, 그리고 무한 반복해서 발생시킬 수 있다고 가정한다.

### 3.1 공격자 모델

네트워크의 일부 또는 전체가 공격자에 의해 장악될 수 있다고 가정한다. 공격자의 지식상태에 따라 메시지를 낚아채거나 도청할 수 있으며, 가짜 메시지를 생성할 수도 있다. 또한 프로토콜 통신의 진행을 끊거나 교란할 수도 있다.

에이전트 $A$의 초기지식 $K^{0}{}_{A}$은 모든 통신 참가자들의 이름과 공개키, 그리고 자신의 개인키 등을 포함한다. 공격자 $E$의 초기지식 $K_E^0$는 신뢰할 수 없는 모든 에이전트들의 초기지식과 그들의 개인키를 포함한다. 공격자와 모든 에이전트들의 지식상태는 프로토콜 통신이 진행되는 과정에서 발생하는 메시지를 주고받는 이벤트를 통해 증가한다.

### 3.2 지역 지식상태

지역 지식상태(configuration state)는 프로토콜 통신과정 어느 순간에 관련된 모든 에이전트(agents)들의 지식상태 전체를 가리키며 다음과 같이 표기한다.

$$(K_E^i, \langle K_{A_n}^i \rangle_n)$$

여기서 $i$는 $i$번째 지역 지식상태를 의미한다. 공격자가 언제든지 새로운 이벤트 세션을 발생시켜 공격을 시도할 수 있기 때문에 에이전트의 수는 무한이라고 가정한다, 즉, $n = 0, 1, 2, 3, \ldots$ 이다.

프로토콜 $P$에 따라 통신이 진행될 때 지역 지식상태가 어떻게 변하는가를 상태전이(state transition) 규칙을 이용하여 명세할 수 있으며, 아래 세 규칙이 기본으로 사용된다.

- 새 이벤트 생성: 특정 에이전트가 새 이벤트를 시작한다. 해당 에이전트는 이벤트와 관련된 에이전트들에 대한 지식을 자신의 지식상태에 추가한다. 만약에 해당 에이전트가 공격자와 연합하는 경우 공격자와 모든 지식을 공유한다.
- 송신 이벤트: 에이전트가 메시지 $m$을 송신하면 모든 공격자에게 노출된다. 공격자는 자신의 컴퓨팅 능력을 활용하여 새 메시지로부터 새 정보를 획득한 후 다시 새 메시지 관련 이벤트를 발생시키거나 발생하기를 기다린다.
- 수신 이벤트: 새 메시지를 수신한 에이전트는 자신의 컴퓨팅 능력을 활용하여 새 정보를 획득한 후 다시 새 메시지 관련 이벤트를 발생시키거나 발생하기를 기다린다. 수신자의 컴퓨팅 능력에는 메시지의 가독성 여부 판단능력 등이 포함된다. 수신된 메시지 $m$의 가독성이 인정되면 수신 에이전트의 지식에 추가된다.

### 3.3 상태전이 궤적

초기 지역 지식상태에서 시작하여 앞서 설명한 상태전이 규칙들을 적용하여 지역 지식상태가 변하는 것을 상태전이(state transition)라 부르며. 이런 상태전이의 변화과정을 모아놓은 것이 상태전이 궤적(trace)이다. 프로토콜 $P$가 주어졌을 때 상태전이 궤적은 상태전이의 연속으로 구성되며, 상태전이는 아래 모양을 갖는다.

$$(K_E^i, \langle K_{A_n}^i \rangle_n) \rightarrow_{m_{i+1}} (K_E^{i+1}, \langle K_{A_n}^{i+1} \rangle_n)$$

위 상태전이는 $(i+1)$-번째 상태전이를 가리키며 사용된 기호들의 의미는 다음과 같다.

- $A_n$ ($n$=0, 1, 2,...): 이벤트가 무한정 발생함을 가정하기 때문에 무한이 많은 에이전트가 관여함을 의미함.
- $(K_E^i, \langle K_{A_n}^i \rangle_n)$: $i$번째 지역 지식상태
- $m_i$: $i$번째 이벤트에서 송신된 메시지

## 4. 상태전이 궤적 관련 성질

상태전이 궤적과 관련된 성질 두 개를 살펴본다.

### 4.1 기밀성

기밀성(secrecy)은 특정 정보가 프로토콜 통신에 관여한 에이전트 이외의 어느 누구에게도 노출되지 않는 성질을 의미하며, 이는 신뢰할 수 없는 네트워크 상에서 통신이 이루어지는 경우도 포함한다. 예를 들어, $A_{n_1}, \ldots, A_{n_p}$만 정보 $m$을 공유하고자 할 때 이들을 제외한 공격자나 다른 에이전트가 해당 정보를 어떤 식으로든 공유할 수 없어야 한다.

니드햄-슈뢰더 공개키 프로토콜의 경우 기밀성이 보장되지 못한다. Lowe가 1995년에 발견한 중간자 공격 (Man-in-the-middle attack)[12]에 의해 공격자가 다른 에이전트들 사이에 주고받는 정보를 획득할 수 있기 때문이다.

예를 들어, 아래 그림은 니드햄-슈뢰더 공개키 프로토콜의 통신과정을 보여주는 예제이다. 예제에

서 B 에이전트가 생성한 기밀 $N_b$가 공격자 E에게 노출될 수 있음을 알 수 있다.

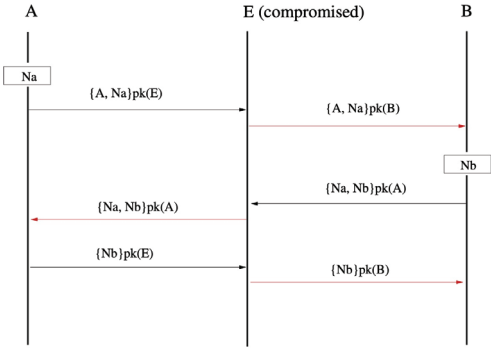


그림 3. 중간자 공격
Figure 3. Man-in-the-middle attack

### 4.2 인증(Authentication)

인증은 관련 성질에 따라 서로 다른 수준으로 구분한다. 예를 들어, Lowe[13,14]는 aliveness, recentness, (non-)injective agreement, (non-)injective synchronization 등으로 인증 관련 성질을 구분할 수 있다고 주장한다. 예를 들어, non-injective agreement는 다음과 같이 설명된다.

> 개시자 $i$와 반응자 $r$이 서로 동일한 프로토콜을 이용하여 통신하며, 사용되는 데이터 변수에 대해 양쪽 모두 동의한다.

injective agreement는 이에 더하여 개시자 $i$가 발생시키는 모든 이벤트에 반응자 $r$이 딱 한 번 반응함을 의미한다.

인증의 정형명세는 조금 더 복잡하며, 심지어 사용하는 명세언어에 따라 미세하게 달라진다. 여기서는 직관적인 이해를 돕기 위해 (non-)injective agreement를 상태전이 궤적을 이용하여 설명한다. 이를 위해, 아래 Claim 서술자(binary predicate) 를 사용한다.

- Claim($r$, NI-Agree) : non-injective agreement가 $r$ 역할 수행자에게 적용됨을 의미한다.

모든 상태전이 궤적에는 여러 개의 Claim 이벤트가 포함될 수 있다. $r$ 역할에 대해 non-injective agreement가 성립하려면 다음이 성립해야 한다.

> 상태전이 궤적의 $\ell$ 번째 단계에서 Claim 이벤트가 정직한 $A$ 에이전트에 의해 실행되고 해당 에이전트가 정직한 에이전트들과만 통신하였다면 그때까지 교신된 모든 메시지는 오직 정직한 에이전트들에 의해서만 생성되었다.

<그림 3>이 보여주듯이 니드햄-슈뢰더 공개키 프로토콜의 경우 non-injective agreement가 성립하지 않는다. 정직한 $B$ 에이전트는 정직한 $A$ 에이전트와 교신한다고 믿지만 실제로는 중간에 공격자 $E$ 가 $A$ 에이전트의 역할을 수행하는 것처럼 속이고 있기 때문이다.

## 5. 결 론

암호화 보안 프로토콜의 정형명세를 소개하였다. 또한 정형명세를 이용하여 보안과 관련된 성질들을 어떻게 다룰 수 있는지도 살펴보았다. 이를 위해 기밀성과 인증의 예를 사용하였으며, 사용되는 언어와 대상 성질에 따라 다양한 정의와 접근법이 있음을 니드햄-슈뢰더 공개키 프로토콜을 이용하여 설명하였다. 암호화 보안 프로토콜의 정형명세는 보안 프로토콜 자동검증을 위한 기초를 제공한다[15]. 따라서 이에 대한 연구가 진행되어 오고 있으며 산업체의 요구 또한 증가하고 있다. 관련 연구에 대한 보다 많은 관심이 중요하다.

## References


[1] C. Cremers, *The Scyther tool: verification, falsification, and analysis of security protocols*, Conference on Computer-Aided Verification (CAV), 2008.

[2] C. Cremers and S. Mauw, *Operational semantics and verification of security protocols*, Information Security and Cryptography, Springer, pp. 1-155 2012.

[3] B. T. Nguyen, C. Sprenger, and C. Cremers, *Abstraction for security protocol verification*, Journal of Computer Security, Vol. 26, No. 4, pp. 459-508, 2018.

[4] B. Blanchet, *Automatic verification of correspondences for security protocols*, Journal of Computer Security, Vol. 17, No. 4, pp. 363-434, 2009.

[5] B. Blanchet, *Automatic verification of security protocols in the symbolic model: the verifier ProVerif*, Foundations of Security Analysis and Design (FOSAD), pp. 54-87, 2013.

[6] N. Kobeissi, K. Bhargavan, and B. Blanchet, A*utomatic verification for secure messaging protocols and their implementations: a symbolic and computational approach*, EuroS&P 2017, pp. 435-450, 2017.

[7] H. Oguma, A. Yoshioka, M. Nishikawa, R. Shigetomi, A. Otsuka, and H. Imai, *New attestation based security architecture for in-vehicle communication*, GLOBECOM, pp. 1909-1914, 2008.

[8] G. Lee, H. Oguma, A. Yoshioka, R. Shigetomi, A. Otsuka, and H. Imai, *Formally verifiable features in embedded vehicular security systems*, IEEE Vehicular Networking Conference (VNC), pp. 1-7, 2009.

[9] T. Ohki, and A. Otsuka, *Theoretical vulnerabilities in map speaker adaptation*, ICASSP, pp. 2042-2046, 2017.

[10] S. Delaune, P. Lafourcade, D. Lugiez, and R. Treinen, *Symbolic protocol analysis for monoidal equational theories*, Information and Computation, Vol. 206, No. 2-4, pp. 312-351, 2008.

[11] R. Küsters, and T. Truderung, *Reducing protocol analysis with XOR to the XOR-free case in the horn theory based approach*, ACM Conference on Computer and Communications Security, pp. 129-138, 2008.

[12] G. Lowe, *An attack on the Needham-Schroeder public key authentication protocol*, Information Processing Letters, Vol. 56, No. 3, pp. 131-136, 1995.

[13] G. Lowe, *A hierarchy of authentication specifications*, IEEE Computer Security Foundations Workshop, pp. 31-44, 1997.

[14] P. J. Hopcroft, and G. Lowe, *Analysing a stream authentication protocol using model checking*, International Journal of Information Security, Vol. 3, No. 1, pp. 2-13, 2004.

[15] M. Avalle, A. Pironti, and R. Sisto, *Formal verification of security protocol implementations: a survey*, Formal Aspects of Computing, Vol. 26, No. 1, pp. 99-123, 2014.


## 암호화 보안 프로토콜의 정형명세


**이계식**

*한경대학교 컴퓨터공학과 부교수*



**요 약**

암호화 보안 프로토콜의 정형명세를 소개한다. 암호화 보안 프로토콜을 검증하기 위해 정형명세를 이

용하는 방법을 보여준다. 검증의 각 단계를 니드햄-슈뢰더(Needham-Schroeder) 공개키 프로토콜을 이용하여 구체적으로 설명한다. 사용되는 구문법(syntax)은 스카이터(Scyther) 툴의 구문법과 유사하며, 사용되는 명세는 간단하며 직관적이다. 우리가 사용하는 명세는 프로베리프(ProVerif)의 기본 개념과도 대응된다. 예를 들어, 메시지는 기호들의 나열로 표현되며 공격자는 정형화된 프로세서로 모델링된다. 니드햄-슈뢰더 공개키 프로토콜의 정형화된 언어 $L = (V, C, R, F)$ 로 구성되는데, $V$는 수신된 메시지를 위한 변수들의 집합, $C$는 일회성 난수(nonces)를 위한 지역 상수 기호 집합, $R$은 역할 이름 집합, $F$는 공개키/개인키 생성 함수들의 집합이다. 프로토콜은 개시자, 반응자, 서버 등의 각 역할이 수행해야 하는 행위를 지정한다. 이 프로토콜의 정형명세에서 각 역할의 행위는 상태전이 체계로 정형화되는데, 그 체계가 기술하는 내용은 메시지 생성방법, 수신된 메시지에 대한 반응, 메시지 조작방법 등이다. 많은 보안 관련 성질 중에서 여기서는 기밀성(secrecy)과 인증(authentication)을 정형적으로 다루는 방법에 대하여 기술한다.

## 감사의 글

본 논문은 2017학년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. NRF-2017R1D1A1B05031658).

**Gyesik Lee** received the bachelor's degree in the Department of Mathematics from Seoul National University in 1992. He received the M.S. degree and the Ph.D. degree in the Department of Mathematics and Computer Science from University of Muenster in 1996 and 2005, respectively. From 2005.12 to 2011.02, he had research positions at INRIA, AIST, and Seoul National University. He is an associate professor in the Department of Computer Science and Engineering at Hankyong National University since 2011. His current research interests include logic in computer science, data analysis, programming education through game programming. He is a member of the KKITS.
*E-mail address*: gslee@hknu.ac.kr