보안 기능을 위한 정형화 설계 방법 연구[1)]

유희준*○, 강은영*, 최진영*, 이성권**, 김우곤**
고려대학교 컴퓨터학과 정형기법연구실*
한국정보보호센터**

Formal Specifications for Security Functions

Hee-Jun Yoo*, Jin-Young Choi*, Sung-Kwon Lee**, Woo-Gon Kim**
Formal Methods Lab. Dept. CSE Korea University*
Korea Information Security Agency**

요 약

컴퓨터 기술의 발전으로 정보화 시대를 맞이한 현대에 있어서 "보안 기능의 정형화 설계 방법 연구"는 정보 보호와 완벽한 보안 때문에 매우 중요하다. 이러한 추세에서 전세계적으로 보안 시스템에 대한 등급을 나누고 있고, 국내에서도 한국정보보호센터에서 침입차단시스템에 대해 K1에서 K7까지의 등급을 매기고 있다. 하지만, 아직까지 국내에서 이 분야에 대한 연구가 거의 진행되고 있지 않아 K5이상의 고등급 제품의 개발 및 평가에 많은 어려움이 있는 실정이다. 본 논문은 실제 간단한 규모의 보안 시스템 설계에 직접 적용될 수 있을 수준의 정형기법을 제시하는 것을 목표로 한다. 여기서는 전자 서명에 관련된 인증 알고리즘인 MD4 알고리즘에 대해서 정형 명세 언어인 Z와 VDM을 적용해 본 결과를 기반으로 보안 시스템을 정형 명세한 경험을 기술한다.

1. 서 론

현대에 들어서면서 인터넷은 전자상거래(Electronic Commerce)라 새로운 문화를 만들었다. 이로 인해 사이버 기업(Cyber Company), 사이버 마켓(Cyber Market), 사이버 거래사회(Cyber Trading Community) 등과 같은 신종 기업문화가 탄생되고 있으며, 자신의 기업조직이나 국가의 범주를 넘어 전자적으로 비지니스를 실현할 수 있는 환경이 마련되고 있다.

이러한 전자상거래가 빠르게 실용화되기 위해서는 우선적으로 해결되어야 할 문제가 시스템의 보안에 관련된 문제이다. 현재 보안 문제를 해결하기 위해서 여러 시스템들이 개발되고 있으며, 이러한 보안 시스템에는 여러 보안 알고리즘이 적재되어 있다. 여기서 사용한 사용자 인증 알고리즘은 이러한 보안 알고리즘의 일종이다. 사용자 인증에 사용되는 해쉬 알고리즘에는 MD4[1], MD5[2], SHA[3]등이 있으며, 이러한 알고리즘에 대한 명확한 명세가 보안 시스템이 높은 보안 등급을 얻기 위해서 필수적으로 필요하게 되었다.

보안 관련 알고리즘 및 각종 프로토콜의 구현은 안전한 전자상거래, 개인의 사생활 보호, 국가적 차원의 안보와 직결되는 부분으로 일부 선진국에서는 오래전부터 이러한 정보보호시스템들에 대한 평가기준과 평가체계를 만들어 평가를 시행하고 있다. 최근우리나라에서도 이러한 제도의 필요성을 인식하고 한국정보보호센터를 통하여 관련 기준을 제정하고 평가를 시행하고 있다. 현재는 1998년 2월 23일 고시된 정보통신망 침입차단시스템 평가기준에 의하여 침입차단시스템(Firewall)에 대해 K1에서 K7까지의 등급을 부여하고 있다.[4] 이러한 평가제도는 앞으로 다른 종류의 정보보호시스템(침입탐지시스템, 스마트카드 등)에 대해서도 적용될 예정이다.

침입차단시스템 평가기준을 살펴보면 크게 보안기능요구사항과 보증요구사항으로 나누어진다. 보안기능요구사항에는 각 등급별로 요구되는 최소기능요구사항을 식별확인, 접근통제, 무결성, 비밀성, 감사기록 및 추적, 보안관리의 6가지로 분류하여 서술하고 있으며 보증요구사항에서는 침입차단시스템의 신뢰성을 확인하기 위하여 개발과정, 시험과정, 형상관리, 운영환경, 설명서, 취약성으로 분류하고 이와 관련된 문서들을 평가하게 된다. 보증요구사항의 평가는 시스템의 신뢰성을 보장하기 위한 중요한 과정으로 K5 이상의 고등급에서는 정형화된 명세를 요구하게 된다. 이와 같이 국내의 침입차단시스템 평가기준에서 높은 신뢰도를 보증하기 위하여 정형화를 요구하고 있지만 실제로 국내에서는 정형화 명세를 이용한 보안 알고리즘의 구현이나 제품의 개발은 거의 전무한 실정이다. 이러한 이유로 이에 관련한 연구가 절실하게 필요한 상태이다. 이러한 필요성에서 정형명세 언어인 Z[5]와 VDM[6]을 보안기능에 적용한 경험을 기술하고자 하는 것이다.

본 논문의 구성을 살펴보면, 2장에서는 보안 시스템을 안전하게 설계하기 위해 사용되는 정형명세에 대해서 설명하고, 3장에서는 Z와 VDM을 사용하여 MD4 알고리즘을 정형 명세하고, 4장에서 두 정형명세언어의 장단점을 기술한 후, 5장에서는 결론을 맺겠다.

2. 정형명세

하드웨어나 소프트웨어 시스템은 불가피하게 점점 더 규모, 기능면에서 커지고 복잡해진다. 이러한 복잡성의 증가 때문에 작은 에러가 존재할 가능성은 훨씬 커지고 있다. 이러한 복잡한 시스템에 대해서 과거와 같이 테스팅(testing)이나 시뮬레이션(simulation)만으로는 작은 에러를 찾아낼 수 가 없다.[7]

이에 대한 해결책으로 시스템 개발에 정형기법이 사용되고 있다. 정형기법은 수학과 논리학에 기반을 둔 방법으로 하드웨어나 소프트웨어 시스템을 명세하거나 검증하는 것이다. 수학적 기호를 사용하여 시스템의 명세를 작성하고 검증할 특성도 논리로 기술하여 시스템이 특성을 만족하는지를 수학적 성질을 이용하여 검증하므로 자연어가 내포하는 애매모호함이나 불확실성을 최소한 줄일 수 있다. 복잡한 시스템에 어떤 특성이 만족하는지를 검증하여 최소한으로 검증된 특성에 대해서는 믿을 수 있고 사용할 수 있다.

정형기법에는 정형 명세(formal specification)와 정형 검증(formal verification)으로 나누어진다.

정형 명세 언어는 자연어가 내포하고 있는 애매모호함이 없기 때문에 명세하고자 하는 시스템을 명확하게 기술할 수 있다. 각각의 정형 명세 언어마다 제공하는

1) 본 연구는 1999년 한국정보보호센터의 지원을 받은 것이다.

도구(Tool)를 이용하여 명세가 타입 오류(Type Error)없이 명확하게 명세되었는지를 검사하여 명세시 발생할 수 있는 오류를 제거해준다.

이러 정형 명세 언어에는 집합론(Set Theory), 논리(Logic)등에 기반을 하는 Z, VDM과 상태(State)를 기반으로 하는 Statechart[8], 그리고 마지막으로 CCS[9], CSP[10], ACSR과 같은 프로세스 알제브라(Process Algebra)를 기반으로 한 언어들이 있다.

3. 정형명세언어를 이용한 알고리즘 명세

이번 절에서는 Z와 VDM을 이용한 MD4 알고리즘의 정형 명세를 살펴보겠다.[11][12] 여기서는 명세의 일부만 언급한다.

3.1 데이터 타입, 불변자에 대한 명세

MD4 알고리즘에서 입력 데이터는 비트(bit)로 구성되어 있고, 비트의 순열(sequence)로 이루어진 워드(word)는 알고리즘에서 사용하는 해쉬 함수들의 처리 단위로 사용된다. Z에서는 데이터 타입을 대괄호로 표현하여 선언하면, 기본 데이터 타입과 같이 사용할 수 있다.

$$[\ Bit\ ]$$

$$Word : \text{seq } Bit$$
$$\#\ Word = 32$$

$$wordlength : \mathbb{N}$$
$$wordlength = 32$$

VDM에서는 등호를 이용하여 선언하고, 0, 1로 구성된 자연수라는 점을 명시해준다.

```
Bit=nat
inv Bit = Bit in set {0, 1};

Word = seq of Bit
inv Word = len Word = WordLength;

WordLength = 32;
```

그림 1 VDM의 타입과 불변자 정의

3.2 동작에 대한 명세

각 언어들이 명세에서 동작(Operation)을 어떻게 정의하는지를 살펴보겠다. 여기서는 패딩 비트를 더하는 메세지 패딩부분에 대해서만 언급한다. Z에서는 모든 동작이 스키마의 형태로 기술된다. 다음은 입력 메시지의 길이를 체크하고 패딩 비트를 더하는 동작에 대한 스키마들이다.

CheckLength

$$\Delta Convert$$
$$inputmessage? : \text{seq } Bit$$
$$inputmessage! : \text{seq } Bit$$
$$messagelength : \mathbb{N}$$

$$\#inputmessage? \geq 0$$
$$inputmessage! = inputmessage?$$
$$messagelength = \#inputmessage!$$

Padding1

$$\Delta Convert$$
$$inputmessage! : \text{seq } Bit$$
$$padding1 : \text{seq } Bit \rightarrow \text{seq } Bit$$

$$\forall inputmessage? : \text{seq } Bit \bullet$$
$$inputmessage! = inputmessage? \frown a$$

Padding2

$$\Delta Convert$$
$$ZeroBlock$$
$$inputmessage! : \text{seq } Bit$$
$$padding2 : \text{seq } Bit \rightarrow \text{seq } Bit$$

$$\forall inputmessage? : \text{seq } Bit \bullet$$
$$inputmessage!$$
$$= \textbf{if } num = 448$$
$$\textbf{then } inputmessage? \frown zeroblock1$$
$$\textbf{else if } num < 448$$
$$\textbf{then } inputmessage? \frown zeroblock2$$
$$\textbf{else } inputmessage? \frown zeroblock3$$

위와 같이 패딩된 메시지를 총 512비트로 만들기 위해서 다음과 같이 원시 메세지의 길이를 64비트로 만들어서 더해 준다.

MakeMessage

$$\Delta Convert$$
$$CheckLength$$
$$inputmessage! : \text{seq } Bit$$
$$inputmessage? : \text{seq } Bit$$
$$lengthtobit : \mathbb{N} \rightarrow \text{seq } Bit$$

$$\forall messagelength : \mathbb{N} \bullet$$
$$\#(\ lengthtobit\ messagelength\ ) = 64$$
$$inputmessage! = inputmessage? \frown$$
$$lengthtobit\ messagelength$$

이 부분의 VDM 명세는 그림 2와 같이 나타난다.

PadOutMessage와 MessagePadding이 패딩의 동작을 기술한 함수이고 메세지 패딩을 위한 최종 함수 Prelude가 정의된다. VDM의 함수 표현을 보면, 먼저 함수의 입력과 출력에 대한 데이터 타입을 기술한 후, 그 밑에 함수의 동작을 표현한다.

다음은 원하지 않는 입력이 들어온 경우에 대한 함수의 에러 처리에 관한 Z 명세이다.

ErrorBit

$$\Xi Convert$$
$$inputmessage? : \text{seq } Bit$$
$$rep! : Report$$

$$inputmessage? \notin \text{seq } Bit$$
$$rep! = is\_not\_bitmessage$$

```
PadOutMessage : MessageInBits → MessageInBits

PadOutMessage(M) ==

  if lenM mod 512 < 448

  then let NumPaddedBits = 448 - len M mod 512 in

       M∧MessagePadding(NumPaddedBits)

  else let NumPaddedBits = 960 - len M mod 512 in

       M∧MessagePadding(NumPaddedBits)


MessagePadding : nat → MessageInBits]

MessagePadding(NumPaddedBits) ==

   [1]∧ZeroPadding(NumPaddedBits - 1)


Prelude : MessageInBits → MessageInBits

Prelude(M) ==

 let PaddedMessage = PadOutMessage(M) in

 PaddedMessage∧ConvertNumberToDoubleWord(len M)
```

그림 2 VDM의 패딩 단계 명세

```
ErrorEmpty
ΞConvert
inputmessage? : seq Bit
rep! : Report
-----------------------
inputmessage? = 〈〉
rep! = empty_message
```

이러한 에러 조건은 이 절의 앞에서 기술한 정상적인 동작과 묶어서 다음과 같은 패딩 전체 동작 명세를 한다.

DoPadding ≙ CheckLength ∧ Padding1
∧ Padding2 ∧ MakeMessage ∧ ErrorBit
∧ ErrorEmpty

4. 정형명세언어의 장단점

이러한 정형명세언어를 사용함으로써, 우리는 설계하고자 하는 시스템의 명세가 에러 없이 명확하고 일관성 있게 기술되었는지를 검증할 수 있다. 여기서의 검증은 시스템의 올바른 동작에 대한 보장이 아닌 시스템 명세가 오류 없이 일관성(consistency)있게 기술되었음을 보장하는 것이다. 따라서, 이러한 정형명세언어를 사용함으로써 명세서부터 애매 모호함 없이 시스템을 설계할 수 있다.

여기서 사용한 Z와 VDM은 모두 수학적 기반을 가지고 있는 언어이기에 모두 매우 명확한 표현을 하고 있다. Z는 논리를 기반으로 칼큘러스적인 표현을 사용하여 여러 가지 특성을 표현하는 면에 있어서는 VDM보다 함축적인 표현을 한다. 하지만, 이러한 지식이 없는 사람이 명세를 이해하기에는 더 어렵다. VDM은 명세형식이 일반적인 함수형 언어들과 유사하며, pre, post, inv와 같은 표현으로 지금 줄이 어떠한 특성을 나타내는 가를 명세를 보는 사람들에게 알려주지만, Z는 모든 특성을 스키마안에 순서대로 기술한다.

5. 결 론

현재 급속히 발전하는 인터넷에서 사용자에 대한 정보 보호와 사용자 인증은 매우 중요하다. 비정상적인 사용자에 의해서 일어날 수 있는 전자 상거래 시스템의 피해는 현재도 일어나고 있으며, 앞으로는 더욱 큰 문제거리가 될 것이다. 이러한 흐름속에서 전세계의 각 나라들은 보안시스템에 대한 평가기준을 제정하고 평가하기 시작하였고, 여러 경우에 대한 비교가 이루어졌다. 그 결과 정형 명세로 구현된 알고리즘이 가장 정확한 명세가 이루어져 신뢰성이 가장 높다는 결론을 내렸다.

국내에서도 한국정보보호센터에서 보안 제품을 평가하면서 정형 명세가 이루어진 시스템에 대해 가장 높은 등급을 주는 것도 이러한 이유이다. 이로 인해, 보안 관련 알고리즘이나 시스템을 설계하는 설계자 입장에서는 정형 명세를 어떻게 이용해야 하는지가 중요한 문제로 대두되었다.

본 논문에서는 이러한 문제에 대한 방안으로 정형 명세 언어인 Z와 VDM을 이용하여 사용자 인증 알고리즘을 명세한 경험을 기술한 것이다. 본문에서 살펴보았듯이 정형 명세 언어를 이용한 명세는 알고리즘의 동작을 체계적이고, 명확하게 명세하여 애매모호함을 없애준다. 또한, 정형 명세 언어를 지원하는 도구를 이용하여 명세의 타입이 맞는지, 만족하고자 하는 증명이 옳게 동작하는 지를 보일 수 있다.

수행한 검사는 알고리즘의 정확성이 아닌 명세의 정확성에 관한 검사이다.

보안관련 시스템을 개발하는 개발자 입장에서도 자신이 개발하는 시스템이 정말로 자신의 생각과 같이 동작하는 지를 확인하기 위해서는 정형 명세가 반드시 필요하다. 그리고, 사업적인 측면에서도 개발한 시스템이 높은 보안 등급을 받을 경우, 고품질의 상품을 만들 수 있다. 신뢰도도 추가하기 때문에 많은 사용자가 믿고 사용함으로써 전자 상거래를 활성화 할 수 있다.

앞으로는 여러 정형 명세 언어를 이용하여 보안 기능 알고리즘을 명세하여, 어떤 정형 명세 언어가 보다 쉽고 명확하게 명세에 사용될 수 있는지에 관한 연구가 필요하다.

6. 참고문헌

[1] Ronald L. Rivest. The MD4 Message-Digest Algorithm, RFC 1320, MIT and RSA Data Security, 1992.

[2] Ronald L. Rivest. The MD5 Message-Digest Algorithm, RFC 1321, MIT and RSA Data Security, 1992.

[3] Secure Hash Standard. Federal Information Processing Standard Publication 180-1. 1995.

[4] 국내.외 정보 보호 시스템 평가 가이드, 한국 정보보호센터. 1998.11.

[5] Antoni Diller, Z An Introduction to Formal Methods. John Wiley & Sons. 1992.

[6] Andrew Harry. Formal Methods Fact File : VDM and Z. John Wiley & Sons. 1996.

[7] David L. Dill. John Rushby. Acceptance of Formal Methods : Lessons from Hardware Design, IEEE Computer. April 1996. Vol.29. No. 4. pp. 16-30

[8] David Harel. Statechart: A Visual Formalism For Complex Systems, Science of Computer Programming 8. 1987. 231-274.

[9] Robin Milner. Communication and Concurrency, Prentice Hall International(UK) Ltd.. 1989.

[10] C.A.R. Hoare, Communicating Sequential Processes. Pretice-Hall International(UK) Ltd.. 1985.

[11] 유희주. Z를 이용한 보안기능의 설계방법연구, 고려대학교 석사학위논문. 1999.7

[12] 김기수. 형식규격어를 이용한 보안표준의 기술 및 참조구현 코드 생성, 충남대학교 석사학위논문, 1996.2