C언어로 개발된 검증필 암호모듈을 자바환경에서 활용하기 위한 방안

최현덕* · 이재훈* · 이옥연†

*국민대학교 (금융정보보안 대학원)

Application of C Language Based Cryptographic Module with KCMVP in Java Environment

Hyunduk Choi* · Jaehoon Lee* · Okyeon Yi†

*Kookmin University (Department of Finantial Information Security, Graduate School)

E-mail : {jakehdc, guderian88, oyyi}@kookmin.ac.kr

요 약

최근 들어 각종 보안 사고가 발생하면서 이전과는 다르게 보안에 대한 인식이 매우 중요해 졌다. 국내에는 국가차원에서 암호모듈의 안전성을 검증하는 제도인 KCMVP를 시행하고 있으며, 다양한 언어로 개발된 암호모듈들이 KCMVP 검증을 받고 있다. 암호모듈 검증시 해당 모듈뿐만 아니라 동작하는 환경까지 평가하기 때문에 다른 환경에서의 동작은 제도적으로 허용하지 않는다. 하지만 동일한 환경에서는 C언어로 개발된 검증필 암호모듈을 자바로 동작시키는 것이 가능하며, 본 논문은 이러한 암호모듈의 활용 방안을 제안한다. 이를 통해 암호모듈 개발에 들어가는 막대한 시간과 비용을 절약할 수 있으며, 이론적으로는 자바로 개발된 다른 검증필 암호모듈보다도 더 나은 성능을 기대할 수 있을 것이다.

ABSTRACT

Due to recent arise of cybercrime, importance of cyber security is highlighted more than ever. Korean government has been running Korea Cryptographic Module Validation Program, namely KCMVP, to validate security level of cryptographic modules for public organizations: indeed, many are achieving the validation. According to the program, operating environments for any specific cryptographic module are fixed. In other words, running validated module in other software environment is strictly prohibited. However, this paper asserts that it is possible for a C language based module to operate in Java based environment as long as the module is running on a validated environment. We expect this paper to help saving great amount of money and time for developing another cryptographic modules for the same operating environment. Also, this method will provide an idea for developing faster modules.

키워드

KCMVP, 암호모듈, 검증제도, JNI

Ⅰ. 서 론

썬 마이크로시스템즈사가 1995년 자바를 선보인 이래, 자바는 플랫폼에 독립적이라는 강점을 앞세워 지속적인 인기를 누리고 있다. 자바는 범용성을 극대화 시킨 프로그래밍 언어로서 지금도 C언어와 함께 가장 많이 사용되는 언어 중 하나이다. 최근에는 스마트폰뿐만 아니라, 각종 스마트 디바이스에서까지 자바의 활용 가치는 매우 높은데, 가장 큰 이유는 자바가 하나의 소스코드로 거의 모든 플랫폼에서 동작할 수 있기 때문이다.

하지만 자바가 널리 보급되고 확산됨에 따라,

보안이라는 이슈가 필연적으로 동반된다. 정보시스템을 운영하고 있는 기업 및 기관 중에는 해킹과 같은 정보화의 부작용으로 인해 막대한 손실을 보는 경우가 발생하는데, 이를 방지하기 위한 기술 역시 수요가 급증함에 따라 활발히 연구되고 있다. 북미를 비롯한 세계 각국은 국가기관에서 다루는 중요자료의 위조, 변조, 훼손, 유출 등을 막기 위해 안전성이 검증된 암호모듈의 사용을 의무화하고 있으며, 국가차원에서 암호모듈 검증제도를 운영하고 있다[1].

국내 암호모듈 검증제도인 KCMVP는 암호모듈의 안전성을 보장하기 위해서 암호모듈 자체의 안전성뿐만 아니라, 해당 모듈이 실제로 동작하는 환경에서의 안전성까지도 평가한다. 따라서 KCMVP를 목적으로 개발되는 암호모듈은, 개발 단계에서부터 운용환경을 결정하게 된다. 소프트웨어 암호모듈의 경우, 모듈이 지정된 운영체제에서 동작할 때의 안전성을 평가하기 때문에, 명확한 타겟 운영체제가 존재한다. 이렇게 운영체제 및 운용환경의 보안까지도 평가하기 때문에 범용성을 가지는 암호모듈을 검증 받기는 불가능에 가까운 것이 현실이다.

제도적 관점에서 볼 때, 검증필 암호모듈이라도 지정된 운용환경 이외의 다른 환경에서 동작시킬 경우, 검증의 효력을 잃게 된다. 따라서 검증필 암호모듈의 부분적인 수정이 필요하거나 운용환경이 추가되는 경우, 재검증을 시행하지만, 역시 상당한 시간과 비용이 요구된다. 이는 다양한 플랫폼에서의 범용성을 추구하는 자바에게는 매우 치명적이다. 자바로 개발된 암호모듈은 여러 환경에서 동작할 수는 있지만, 검증 받지 않은 환경에서의 동작은 제도적으로 안전성을 보장 받지 못한다. 자바의 특성상 C언어보다는 약간의 성능 저하가 발생할 수밖에 없기 때문에, 범용성을 잃은 자바 암호모듈은 시장에서 경쟁력을 갖기 어려울 수 있다.

본 논문은 JNI 기술의 활용을 통해 C언어로 개발된 암호모듈의 형상을 온전히 유지하면서 자바로 동작시킬 수 있는 방안을 제안한다. 이는 다만 자바에서 C언어가 동작하는 것일 뿐 운용환경이 바뀌거나 암호모듈의 소스코드를 수정하는 것이 아니므로 KCMVP 검증필을 유지할 수 있다. 이러한 암호모듈은 C언어와 자바에서 모두 운용 가능하기 때문에 별도로 암호모듈 개발에 들어가는 시간과 비용을 절감할 수 있다.

II. 관련연구

2.1 KCMVP (Korea Cryptographic Module Validation Program)

CMVP는 1995년 미국 NIST(National Institute of Standards and Technology)와 캐나다 주정부의 CSE(Communications Security Establishment)가 공동으로 개발한 암호모듈 검증제도로서, 표준에 따라 구현된 암호 모듈 및 알고리즘의 보안적합성을 평가하는 제도이다[1].

국내에서 시행되는 암호모듈 검증제도인 KCMVP는 국가사이버안전센터 (National Cyber Security Center)에서 총괄하며, 전자정부법 시행령 제69조와 “암호모듈 시험 및 검증지침”에 의거하여, 국가기관 및 공공기관에 도입되는 암호모듈의 안전성과 구현 적합성을 검증하는 제도를 통칭한다. 이 제도는 비밀로 분류되지 않은 중요 정보의 보호를 위한 암호모듈을 검증하는 목적으로 시행된다[2].

이미 수많은 암호 알고리즘과 운용모드가 세계 여러 학자들의 이론과 연구를 통해 그 안전성을 인정받았음에도 불구하고 추가적으로 KCMVP를 운영해야하는 이유는, 암호모듈의 구현과정 또는 운용환경에서 새로운 취약점이 발생할 수 있기 때문이다.

표준 문서에 따라 알고리즘을 구현하더라도, 개발자마다 다른 구현 방식을 사용하며, 구현된 여러 알고리즘들이 하나의 모듈로서 동작할 때, 데이터를 처리하는 과정에서 새로운 취약점이 발생할 수도 있다. 또한 암호모듈이 안전하게 구현되었더라도 해당 모듈이 동작하는 환경에 따라 다른 취약점이 발생할 수 있는데, KCMVP는 암호모듈의 구현 적합성뿐만 아니라 운용환경의 보안성까지 평가해서 암호모듈의 최소한의 안전성을 보장하고 있다.

2.2 C언어와 자바의 특성

C언어와 자바는 그 탄생부터 목적이 다르기 때문에 단순히 어느 언어가 더 좋다고 말 할 수는 없다. 개발자들이 상황에 따라 더 유리하거나 편리한 언어를 사용해서 개발하면 된다.

C언어의 경우, 기계어로 컴파일 되기 때문에 다른 프로그래밍 언어들 중에서도 우수한 성능이 강점으로 꼽힌다. 하지만 환경에 따라 다양한 컴파일러가 존재하고, 제공되는 라이브러리가 조금씩 다르기 때문에, 컴파일러에 따라 원본 소스코드의 수정이 필요한 경우가 발생할 수 있다.

반면, 자바는 다른 프로그래밍 언어들과는 다르게 컴파일된 코드가 플랫폼에 독립적으로 동작한다. 자바 컴파일러는 자바 언어로 작성된 프로그램을 바이트코드라는 특수한 바이너리 형태로 변환시키고, 이 바이트코드를 실행하기 위해서는 JVM(Java Virtual Machine)이 필요하다. 이것이 바로 자바의 가장 큰 강점인데, 자바는 이 가상 머신만 설치되어 있으면 원본 소스코드의 수정 없이 아무 플랫폼에서나 동작할 수 있다. 따라서 자바로 개발된 프로그램은 CPU나 운영체제의 종류에 관계없이 JVM을 설치할 수 있는 시스템에서는 어디서나 실행할 수 있으며, 이 점이 다양한 플랫폼이 존재하는 IT 시장의 특성과 맞아떨어져 현재까지도 지속적인 인기를 유지하

고 있다.

자바의 가장 큰 경쟁력은 바로 범용성이라 할 수 있는데, 이를 위해 만들어진 JVM이 동시에 자바의 가장 큰 약점이 되기도 한다. C언어의 경우, 기계어로 직접 컴파일 되서 실행되지만, 자바는 class파일로 컴파일 된 뒤 JVM을 거쳐서 실행되는데, 인터프리터 방식의 자바 특성상 C언어보다 느리다는 단점이 존재한다. 또한 운영체제나 운용환경의 도움을 받아 동작하는 C언어와는 다르게, 자바는 JVM이라는 별도의 운용환경을 구성하여 동작하므로 메모리 관리와 같은 운용적 측면에서 부족한 부분이 존재한다. 또한 JVM은 이미 운용환경임에도 불구하고, 윈도우나 리눅스 수준의 안전성을 제공하지 못하기 때문에, 또 다시 운영체제에게 의존적일 수밖에 없다. 즉, 범용성을 위해 속도를 포기한 자바를 다시 보안적 관점에서 바라보면, 결국 안전성을 위해 그 범용성마저 포기할 수밖에 없게 된다.

Ⅲ. JNI

3.1 JNI 활용 목적

앞선 논의를 종합해 본다면, JNI 기술의 활용 목적은 명확하다. KCMVP는 보안이슈가 동반됨에 따라 더 이상 선택이 아닌 필수가 되었고, 이는 암호모듈 자체의 안전성뿐만 아니라 운용환경이 제공하는 보안의 안전성까지도 검증하게 된다. 하지만 JVM에 의존적으로 동작하는 자바의 경우, JVM만으로는 안전성을 보장받기 어렵기 때문에 JVM이 실행되는 운용환경까지 검증의 대상이 되는데, 이렇게 검증 받은 자바 암호모듈은 범용성을 잃게 된다. 물론 상황에 따라 범용성을 추구하지 않는 자바용 암호모듈을 개발할 수도 있겠다. 하지만 암호모듈의 성능이 중요한 경우, C언어로 개발해서 JNI 기술을 활용하는 것이, 성능뿐만 아니라 두 가지 언어로 동작할 수 있다는 경쟁력을 얻을 수 있을 것이다.

3.2 JNI 활용 방안

JNI 기술의 활용 방안을 설명하기에 앞서, 본 논문의 목적은 KCMVP를 위한 JNI 기술의 활용 제안이기 때문에 상세한 설명은 생략한다. 여기서 사용된 암호모듈은 우분투 환경에서 C언어로 개발되었으며, 검증필을 받은 뒤, 다시 우분투 환경에서 JNI 기술을 적용 시켰다. 아래는 해당 환경에 대한 정보를 표로 기술하였다.

표 1. 환경정보

환경	버전
운영체제	ubuntu 12.04 (64bit)
컴파일러	gcc 4.6.3
자바(JDK)	java 1.8.0_11

JNI 기술을 활용하기 위해선, 우선 C언어로 검증 받은 암호모듈의 모든 API 목록이 필요하다. 한 개의 API라도 더해지거나 빠질 경우, 암호모듈의 형상이 변하므로 검증의 효력을 잃게 된다. 그 다음, 자바파일(.java)을 만들어서 네이티브 API들을 불러오도록 한다. javah 명령어를 사용하면 자동으로 헤더파일을 만들 수 있다. 이 헤더파일에 선언된 API들로 다시 C파일(.c)을 만들어 준 뒤, 링크만 걸어주면 자바에서 해당 API들이 호출되는 것을 볼 수 있다.

3.3 검증필 보존의 근거

그렇다면 C언어로 개발된 암호모듈이 자바로 동작하는데도 불구하고 왜 검증의 효력이 유지되는지를 논의하고자 한다.

JNI 기술은 단순히, C언어로 개발된 암호모듈이 자바에서 동작하도록 연결시키는 역할을 할 뿐이다. 암호모듈은 C언어로 개발이 완성된 이후에 어떠한 수정도 없다. 모든 API를 추가나 제거 없이 그대로 사용했기 때문에, 암호모듈의 유한상태모델이 온전히 유지되며 운용환경의 변경도 없기 때문에 실제로 암호모듈은 검증 받은 상태를 그대로 유지하고 있다. 또한 암호모듈이 자바로 동작은 하지만 실제로는 JVM에서 동작하는 것이 아니라, C언어로 동작하게 된다. 사실상 C언어로 개발된 암호모듈의 형상을 온전히 유지하면서, 자바라는 케이스를 씌운 것뿐이다. KCMVP 정책에 따라 암호모듈의 변경 여부를 판단할 수 있는 가장 간단하고 확실한 방법이 MAC(Message Authentication Code)값과 해쉬값을 확인하는 것인데, JNI 기술은 이 모든 조건을 충족시킨다.

Ⅳ. 결 론

지금까지 자바와 C언어의 특성 및 장·단점을 살펴보고, KCMVP 관점에서 JNI 기술을 활용과 암호모듈의 안전성 보장에 관한 논의를 했다. 현재 자바와 C언어는 가장 대중적으로 사용되는 언어이며, 서로 다른 장점이 있기 때문에 어느 언어로 암호모듈이 개발되어도 그 활용가치는 높을 것이다. 특히 자바만을 다루는 기업이라면, 오히려 C언어로 암호모듈을 개발하는 것이 더 많은 시간과 비용을 투자해야 할 것이다. 하지만 이미 C언어로 검증 받은 암호모듈이 존재하거나 혹은 자바의 가장 큰 장점인 범용성을 어쩔 수

없이 포기해야하는 상황이라면, 본 저자는 JNI 기술이 시간과 비용을 절감할 뿐만 아니라 암호모듈의 성능마저 기대할 수 있는 방안이라 확신한다. 이 방안을 통해 더욱 활용가치가 높은 암호모듈이 개발되고 사용된다면, 보안과 서비스 측면에서 한 단계 도약하는 계기가 될 수 있을 것이다.

"본 연구는 미래창조과학부 및 정보통신기술연구진흥센터의 정보통신·방송 연구개발사업의 일호나으로 수행하였음. [10039140 , 스마트 디바이스용 칩(ARM7/9/11, UICC 등)에 최적화된 암호(ARIA, SEED, KCDSA 등)의 국가 인증 모듈 및 배포 체계 개발]"

참고 문헌

[1] IT보안인증사무국, 국내·외 상용 암호모듈 검증정책, 정보과학회지 제25권 제5호, 2007.5.

[2] 국가사이버안전센터, 암호모듈검증 - 개요 및 체계, http://service1.nis.go.kr/