암호 하드웨어 모듈의 신뢰성 검증 도구

경동욱[0] 김동규
부산대학교 컴퓨터공학과
dwkyoung@islab.ce.pusan.ac.kr[0], dkkim1@pusan.ac.kr

A Reliable Verification Tool for Testing Cryptographic Hardware Modules

Dong Wuk Kyoung[0], Dong Kyue Kim
School of Electrical and Computer Engineering, Pusan National University

요 약

암호 시스템들은 복잡한 연산과정을 수행하므로 소프트웨어적으로 구현할 경우 수행속도가 저하되는 단점이 있다. 이를 고속으로 수행하기 위하여 암호 시스템을 하드웨어적으로 구현하는 연구가 활발히 수행되고 있는 것이 현재의 추세이다. 암호 시스템을 하드웨어 모듈로 개발하는 과정 중에는 설계한 모듈이 올바르게 동작하는 지의 여부를 검증하는 과정이 필수적으로 포함된다. 기존의 방법은 검증하고자 하는 암호 알고리즘의 종류에 따라 검증도구를 다시 개발해야 하는 번거로움으로 시간과 비용의 낭비가 초래되었다. 본 논문에서는 기존의 검증 방법을 보완하는 방법으로 PC 기반의 소프트웨어 통제 하에서 어떤 종류의 암호 하드웨어 모듈에 대해서도 호환성을 갖춘 신뢰성 있는 검증 도구를 효과적으로 개발하였다.

1. 서론

컴퓨터 망 기술의 발달에 따라 정보공유라는 긍정적인 측면과 정보유출이라는 부정적인 측면이 함께 공존하게 되었다. 정보유출의 문제를 해결하기 위해 많은 암호 알고리즘이 개발되고 있는데 초기에는 소프트웨어적인 방법이 주류를 이루었으나 증가된 많은 정보를 빠르게 처리하기 위한 요구로 하드웨어 적인 방법으로 개발되고 있는 것이 최근 추세이다[1].

그러나 여기서 개발된 암호 하드웨어 모듈은 정확성을 검증하기 위해 많은 노력과 비용이 든다. 또한 호환성이 없는 하드웨어 모듈의 특성상 새로운 모듈이 개발될 때마다 일일이 새로운 검증 모듈을 개발하는데 이중의 낭비가 초래되는 실정이다.

본 논문에서 제시하는 검증도구는 어떤 암호 하드웨어 모듈에 대해서도 신뢰성을 보장할 수 있도록 검증할 수 있는 완벽한 호환성을 갖춘 범용의 검증도구를 설계하고 개발하여 기존의 호환성이 없었던 검증도구들이 갖는 문제점을 해결하였다.

2. 검증 도구의 구현

이 장에서는 검증도구의 구현 원리를 살펴보고 구체적인 구현방법을 설명하였다.

본 논문에서 제시하는 검증 도구의 전체적인 구성은 기능에 따라 [그림 1]와 같이 4가지 모듈로 나누어진다. 검증하려고 하는 Test모듈과 Test모듈에 필요한 입력과 결과 값을 저장하는 Register 모듈이 있고 또한 외부의 호스트와 통신을 담당하는 Interface 모듈이 있으며 마지막으로 PC기반에서 입력과 결과 그리고 다양한 검증 방법을 지원하는 소프트웨어인 Verify 모듈로 구성 되어있다.

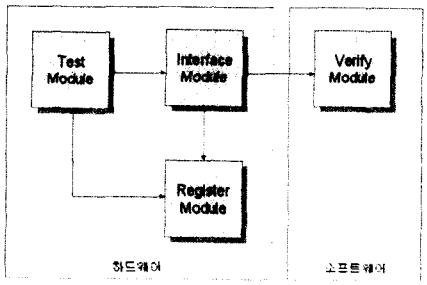

그림 1 검증도구의 4가지 모듈

2.1 Interface 모듈의 구조

FPGA 보드의 데이터 입출력을 담당하는 Interface 모듈은 검증도구에서 두 가지 역할을 한다. FPGA에서 프로그램과의 데이터 송수신 및 제어 기능을 한다. 이때 병렬통신으로 Test 모듈에 필요한 입력 값을 실시간으로 연산하여 결과를 전송하며 Test 모듈과 Register 모듈을 제어하는 역할을 동시에 담당한다.

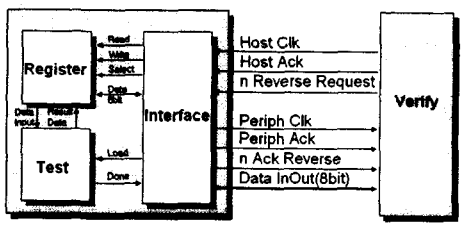

그림 2 검증도구의 전체 구성도

앞서 언급한 Interface 모듈의 전체 구성은 [그림 2]와 이때 암호화 및 복호화를 처리하기 위해 [그림 3]과 같이 병렬통신을 응용하여 개발하였다[2][3].

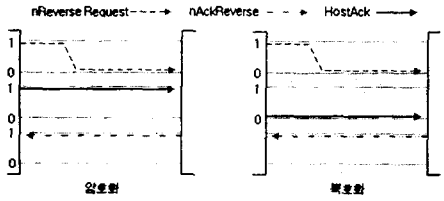

그림 3 암호화 및 복호화 과정의 신호

2.2 Register 모듈의 구조

앞서 살펴본 Interface 모듈은 Test 모듈에 필요한 입력을 병렬통신으로 8비트씩 송수신한다. 그런데 각 암호 알고리즘은 입력크기 및 Test 모듈의 입력 수가 알고리즘에 따라 다르다. 이에 따른 호환성을 실현하기 위해 유연한 저장공간의 구현이 필요하게 되었고 그 핵심이 바로 Register 모듈이다. [그림 4]는 AES처럼 Message와 Key를 입력으로 받는 일반 암호화 모듈을 예로 들어서 나타낸 것이다[4].

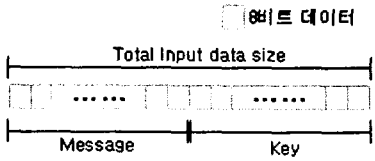

그림 4 Register 모듈의 저장 공간

Register 모듈의 진행과정을 [그림 5]와 같이 데이터 입력단계, Test 모듈로의 데이터 전송단계, 연산결과의 저장단계, 연산결과의 출력단계인 4단계로 구성된다.

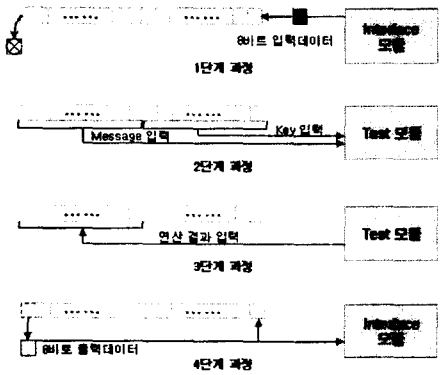

그림 5 Register 모듈의 진행 과정

1단계는 Test 모듈에 필요한 입력은 8비트 병렬통신을 담당하는 Interface 모듈로부터 전송받은 데이터를 전체 입력 크기에 맞게 반복해서 수행된다. 2단계는 입력받은 데이터를 각 암호 알고리즘에 따른 입력 개수와 크기에 맞게 데이터를 처리하여 Test 모듈로 전송한다. 3단계는 Test 모듈에서 처리된 연산결과를 수신하여 Register 모듈에 저장하며 이렇게 저장된 연산결과는 4단계에서 다시 Interface 모듈로 8비트씩 데이터를 전송하여 외부로 데이터를 출력한다.

2.3 Verify 모듈

Verify 모듈은 검증도구의 전체적인 기능을 총괄하여 제공하는 사용자 인터페이스 프로그램이다. 이 모듈의 기능 중 호환성 실현을 위한 두 가지 주목할 만한 기능인 입출력 인터페이스와 검증 알고리즘의 통합화가 있다. 또한 이 프로그램에서는 일반적인 4가지 검증방법을 제공한다. 먼저 입출력 인터페이스의 처리부터 구체적으로 살펴보겠다.

암호 알고리즘에 따라 입력 값이 바뀌면 변경된 값을 다시 FPGA 보드로 8비트씩 보내고 또한 연산결과를 8비트씩 받아서 보여주며 이 값과 소프트웨어를 이용한 결과 값이 서로 같은지 검사함으로써 어떠한 형태의 입력 값에 대해서도 검증을 할 수 있는 장점을 가진다.

이때 Test 모듈을 통해 출력된 하드웨어 연산결과의 정확성을 테스트 하기위해 소프트웨어적으로 데이터를 생성해서 비교하여 암호 하드웨어 모듈의 신뢰성 검증을 수행한다. 그런데 Test 모듈이 다른 암호 하드웨어 모듈로 바뀌면 Verify 모듈도 이에 맞는 비교 데이터를 생성하기위해 해당되는 알고리즘으로 수정해야하는 번거로움이 있다. 이러한 문제점을 해결하기위해서 수정해야 하는 번거로움을 동적링크라이브러리(Dynamic Link Library, 이하 DLL) 로 구현하였다. 이렇게 DLL을 이용하면 암호 하드웨어 검증 시 필요할 때마다 별도의 수정의 번거로움이 없이도 간단히 DLL만 로드 함으로써 암호 하드웨어 연산 결과와 실시간으로 비교가 가능하다. 이와 같이 DLL을 이용하면 보다 빠르고 효율적으로 검증을 수행할 수 있을 뿐만 아니라 호환성까지 보장한다.

또한 중요한 특징으로는 One Round, Known Answer Test, Random Base, Sequence Base 등과 같은 다양한 검증방법의 통합지원을 들 수 있다[5][6].

One Round 방법은 단순히 Test 모듈의 연산과정을 한 번의 수행으로 결과 값을 보여 주는 방법으로 DLL이 필요 없이 사용 할 수 있는 방법이다. Known Answer Test 방법(이하 KAT)은 암호화 모듈인 DES의 경우를 예로 들면 DES는 64비트의 Message와 Key를 사용한 암호 알고리즘이다. 이때 Key나 Message 둘 중하나를 16진수 00 00 00 00 00 00 00 00로 고정하고 나머지는 80 00 00 00 00 00 00 00에서 40 00 00 00 00 00 00 00으로 64비트의 값을 한 비트의 '1'을 쉬프트하여 그 결과를 검증하는 방법으로 이러한 검증은 NIST에서 제안하여 AES나 DES와 같은 암호화 모듈을 검증 시 사용하였다. 실제

완전한 검증을 위해서는 64비트를 검증하려면 2^{64} 만큼의 연산을 해야 하는데 이렇게 하면 시간이 많이 소모되므로 KAT를 사용하여 암호화 모듈의 정확성을 보장하도록 하는 방법을 제안하였다[5]. Random Base 방법은 일정한 규칙으로 검증할 때 그 규칙의 문제로 인하여 제대로 된 신뢰성을 보장 할 수 없을 때에 입력 값을 랜덤으로 결정하여 연산 결과 값을 검증하는 방법이다[6]. 그리고 마지막으로 Sequence Base 방법은 0에서부터 하나씩 증가된 값을 입력하여 그 결과를 비교해서 암호화 모듈의 신뢰성을 검증하는 방법이다. 이렇게 여러 가지 다양한 방법을 사용하여 연속적으로 Test 모듈의 신뢰성을 테스트하여 검증을 수행한다.

3. 결과

설계된 검증 도구는 하드웨어 부분과 소프트웨어 부분으로 나눌 수 있다. 우선 하드웨어 부분은 VHDL로 모델링 하였으며, Model Sim 5.2를 통해서 시뮬레이션을 담당하며 Synplify 7.0을 이용하여 VHDL로 모델링 한 회로를 컴파일 하였고 최종 Xilinx Foundation 3.1을 이용하여 합성하였다[6].

다음은 검증프로그램의 실행하는 모습을 [그림7]과 같이 나타내고 있다.

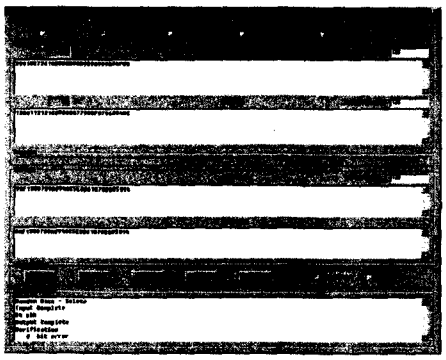

그림 7 검증 프로그램의 실행결과

다음은 여러 가지 다양한 암호 하드웨어 모듈을 Test한 결과를 정리 하였다.

표 1 검증도구의 테스트 결과 데이터 정리

Test 모듈	입력 크기	출력 크기	한 라운드 수행시간	면적
DES	128(bit)	64(bit)	1.54(second)	3,099(gate)
AES	256(bit)	128(bit)	2.48(second)	4,939(gate)
MD5	512(bit)	128(bit)	3.46(second)	8,565(gate)
SHA-1	512(bit)	160(bit)	3.57(second)	8,559(gate)
RSA	2048(bit)	1024(bit)	15.76(second)	29,538(gate)

Test 모듈로서 DES, AES, MD5, SHA-1, RSA 의 다양한 암호 하드웨어 모듈을 테스트 하였으며 [표 1]과 같은 결과를 얻었다[4][8][9]. [표 1]에서 입력크기는 Test 모듈에 필요한 입력 데이터 크기를 나타내며 출력 크기는 Test 모듈의 결과 데이터 크기를 나타내고 있다. 입력크기와 출력크기는의 영향으로 한 라운드 수행시간에서 RSA 모듈을 테스트 하였을때 가장 많은 시간이 소요된다. 그리고 RSA 모듈은 많은 입력비트로 인하여 Register 모듈의 저장 공간을 2048비트의 공간을 소요되므로 가장 많은 면적을 가진다.

4. 결론

최근의 많은 정보를 빠르게 처리하기 위해서 많은 암호화 알고리즘을 하드웨어로 구현하는 현 시점에서 검증은 중요한 위치에 있다. 하지만 이러한 검증을 하기위해서 Test 모듈에 따라 검증하기 위한 도구를 수정하고 만들기 위해서는 많은 시간과 자원이 낭비된다.

따라서 본 논문에서 제시하는 검증도구는 병렬통신을 이용하여 PC기반의 프로그램과 FPGA 보드사이에 데이터 통신을 가능하게 함으로써 일반적인 FPGA 보드를 Test 모듈에 최적으로 검증할 수 있으며 동적링크라이브러리를 이용하여 신뢰성과 더불어 호환성을 보장하는 암호 하드웨어 검증 도구를 개발하였다.

그러므로 어떤 암호 하드웨어 모듈에 대해서도 별도의 수정이나 변경이 없이도 적용 가능한 검증도구의 개발로 필요한 시간과 자원의 낭비를 줄였다.

5. 참고문헌

[1] 박창섭, "암호이론과 보안," 대영사, 1999.

[2] "Interfacing the Standard Parallel Port," http://www.beyondlogic.org.

[3] "Interfacing the Extended Capabilities Port," http://www.beyondlogic.org.

[4] "Announcing the ADVANCED ENCRYPTION STANDARD," Federal Information Processing Standards Publication 197, November 2001.

[5] "Description of Known Answer Tests and Monte Carlo Tests for Advanced Encryption Standard (AES) Candidate Algorithm Submissions," NIST, 1998.

[6] Janick Bergeron, "Writing Testbenches: Functional Verification of HDL Models Second Edition," Kluwer Academic Publishers, 2003

[7] Kevin Skahill, "VHDL for Programmable Logic," ADDISON WESLEY

[8] C. K. Koc, "High-Speed RSA Implementation," Technical Report TR 201, RSA Laboratories, November 1994.

[9] "SECURE HASH STANDARD," Federal Information Processing Standards Publication 180-2, August 2002.