한국사물인터넷학회논문지 Vol. 1, No. 1, pp. 33-38, 2015 http://dx.doi.org/10.20465/KIOTS.2015.1.1.033

224비트 ECDSA 하드웨어 시간 시뮬레이션을 위한 테스트벡터 생성기

김태훈[1], 정석원[2]
[1]KEPCO KDN 보안운영팀, [2]목포대학교 정보보호학과

Test Vector Generator of timing simulation for 224-bit ECDSA hardware

Tae Hun Kim[1], Seok Won Jung[2]
[1]Security Development & Operation Team, KEPCO KDN Co. Ltd.
[2]Dept. of Information Security Engineering, Mokpo National University

요 약 하드웨어는 다양한 구조로 개발되고, 모듈들에 대한 시간 시뮬레이션을 할 때 각 클럭 사이클에 사용되는 변수들의 값을 확인할 필요가 있다. 본 논문은 224비트 ECDSA 하드웨어를 개발하면서 하드웨어 모듈의 시간 시뮬레이션을 위한 테스트 벡터를 제공하는 소프트웨어 생성기를 소개한다. 테스트 벡터는 GUI 형태와 텍스트 파일 형태로 제공된다.

주제어 : 전자서명, ECDSA, 테스트 벡터, 시간 시뮬레이션, 하드웨어 구조

Abstract Hardware are developed in various architecture. It is necessary to verifying value of variables in modules generated in each clock cycles for timing simulation. In this paper, a test vector generator in software type generates test vectors for timing simulation of 224-bit ECDSA hardware modules in developing stage. It provides test vectors with GUI format and text file format.

Key Words : Digital Signature, ECDSA, Test Vectors, Timing Simulation, Hardware Architecture

1. 서론

최근 PC 환경의 발전과 스마트폰의 상용화로 언제, 어디서나 유선과 무선으로 인터넷에 접속이 가능한 세상이 되었다. 또한 다양한 서비스를 위해서 지능형 디바이스들이 개발되고 사용자에게 유용한 정보를 실시간으로 제공하기 위해 디바이스들끼리 능동적으로 정보를 주고받는 사물인터넷 IoT(Internet of Things) 시대가 시작되었다. 많은 건물들이 에너지를 절약하기 위해서 센서를 사용하고 있으며, 홈 네트워크를 위해 다양한 스마트 디바이스들이 개발되었다. 또한 차량, 의료 서비스, 산업 공장의 지능화를 위해 스마트 다바이스가 사용되고 있는 실정이다[1].

사물인터넷은 다양한 통신환경과 여러 가지 스마트 디바이스를 사용한 융합 서비스가 창출되면서 보안 이슈가 생기게 되었다. 첫 째는 물리적인 보안으로 센서 인터페이스, 센서 데이터의 가로채기 등의 문제이다. 두 번째는 동작에 대한 보안으로 센서의 동작, 통신 시스템의 동작, 관리 시스템의 동작 등에서 발행하는 문제로 이는 일반적인 정보시스템의 보안과 비슷한 것이다. 세 번째는 데이터 보안으로 스마트 디바이스들이 처리하는 데이터, 통신 상에서 주고받는 데이터들이 도청되거나 변경되는

*교신저자 : 정석원(jsw@mokpo.ac.kr)
접수일 2015년 12월 9일

문제로 기존의 정보시스템보다 복잡한 형태로 나타난다[2].

스마트 디바이스는 서비스에 따라 다양한 제약적인 환경을 가지며 이러한 제약적인 환경에 적합한 정보보호 기술이 요구된다. 스마트 디바이스는 4-비트, 8-비트, 16-비트, 32-비트 기반의 프로세서에서 AES(Advanced Encryption Standard) 비밀 키 알고리즘, RSA 또는 ECDSA(Elliptic Curve Digital Signature Algorithm) 같은 공개키 알고리즘에 대한 구현이 연구되고 있다[3-7].

전자서명은 데이터의 무결성과 사용자 출처 인증을 동시에 제공하는 정보보호 기술이다. 대표적인 전자서명 기법으로 소인수분해의 어려움에 기반한 RSA 전자서명과 타원곡선 위의 이산대수 문제에 기반한 ECDSA가 있다. 112비트의 보안 강도를 가지기 위해서 RSA는 2048 비트 이상, ECDSA는 224비트 이상을 사용하는 것이 권고되고 있다[8].

공개키 알고리즘은 수학적 연산을 많이 사용하므로 범용 프로세서 위에 구현이 되면 처리 시간이 많이 소요된다. 이런 문제를 해결하기 위해 암호 프로세서를 개별적인 하드웨어로 구현하는 경우가 있다. 하드웨어로 공개키 알고리즘을 구현할 때, 하드웨어를 직렬, 병렬, 직렬-병렬 등 다양한 구조로 구성할 수 있으며, 각 구조에 따라 시간 시뮬레이션이 달라진다. 하드웨어 개발자는 하드웨어 구조에 따른 구현 결과가 제 기능을 하는지 확인해야 하며 이를 위해 테스트 벡터가 필요하다.

본 논문에서는 224비트 ECDSA를 FPGA 형태의 하드웨어로 구현할 때 ECDSA의 각 모듈의 시간 시뮬레이션에 필요한 테스트벡터를 생성하는 소프트웨어를 설명한다.

2. 목표 하드웨어 구조

2.1 ECDSA 전자서명

소수체 F_p 위에 정의된 타원곡선은

$$E: y^2 = x^3 + ax + b$$

의 형태를 갖는다. 여기에서 $a, b \in F_p$ 이다.

$E(F_p) = \{(s,t) \in F_p \times F_p \mid t^2 = s^3 + as + b\} \cup O$

는 덧셈 군이 된다. 여기에서 O는 무한원점으로 덧셈에 대한 항등원이다. 타원곡선 위의 점에 대한 덧셈은 다음과 같이 정의된다. 두 점 $P=(x_1,y_1)$ 과 $Q=(x_2,y_2)$에 대해 $P+Q=(x_3,y_3)$는

$$x_3 = (\frac{y_2-y_1}{x_2-x_1})^2 - x_1 - x_2,$$

$$y_3 = (\frac{y_2-y_1}{x_2-x_1})(x_1-x_3) - y_1$$

이고, $2P=(x_3,y_3)$은

$$x_3 = (\frac{3x_1^2+a}{2y_1})^2 - 2x_1,$$

$$y_3 = (\frac{3x_1^2+a}{2y_1})^2 (x_1-x_3) - y_1$$

이다[9].

목표 하드웨어는 SEC-2 표준인 P-224 타원곡선을 사용하고 있다. 이 곡선은

소수 $p_{224} = 2^{224} - 2^{96} + 1$ 이고

$a =$ FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFE FFFF FFFF FFFF FFFF FFFF FFFE

이고

$b =$ B405 0A85 0C04 B3AB F541 3256 5044 B0B7 D7BF D8BA 270B 3943 2355 FFB4

이다. 타원곡선 군의 생성원 $G(x,y)$는

$x =$ B70E 0CBD 6BB4 BF7F 3213 90B9 4A03 C1D3 56C2 1122 3432 80D6 115C 1D21

$y =$ BD37 6388 B5F7 23FB 4C22 DFE6 CD43 75A0 5A07 4764 44D5 8199 8500 7E34

이다. 여기에서 a, b, x, y의 값은 헥사 값 표현이다[10].

타원곡선 위의 전자서명 ECDSA는 위의 타원곡선 파라미터와 해시함수 H()에 대해 다음의 알고리즘으로 서명 값을 생성한다.

〈Table 1〉 ECDSA 알고리즘[9]

1. input private key d, message m.
2. Select $k \in_R [1, n-1]$
3. Compute $kP=(x_1,y_1)$ kP=() and convert x_1 to an integer t_1
4. Compute $r \equiv t_1 \pmod n$. If $r=0$ then goto step 2.
5. Compute $e=H(m)$
6. Compute $s=k^{-1}(e+dr)$. If $s=0$ then goto step 2..
7. output (r,s)

2.2 목표 하드웨어 구조

ECDSA를 <Table 1>의 알고리즘을 사용하여 구현하려면 step 2를 위한 난수 발생기, step 3을 위한 타원곡선 점 스칼라 곱셈기, step 4와 step 6을 위한 유한체 연산, step 5를 위한 해시함수가 필요하다.

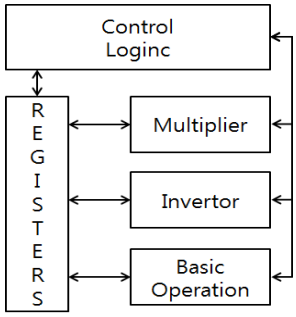
해시함수로 SHA-224를 사용하였고, 난수발생기는 SHA-224 해시함수를 사용한 구조를 택하였다. 2.1절의 두 점 덧셈과 한 점 두 배 연산을 이용하여 스칼라 k에 대한 kP를 구하는 여러 가지 방법이 제안되었다[11-13]. 이 중 목표 하드웨어는 다음의 스칼라 곱셈방법을 사용한다.

〈Table 2〉 스칼라 곱셈 알고리즘[13]

1. input P, $d=(d_{n-1}, d_{n-2}, \ldots, d_1, d_0)$
2. $Q[0] \leftarrow O$, $Q[1] \leftarrow O$, $R[0] \leftarrow O$, $R[1] \leftarrow P$
3. for i from 1 to n do
   - 3.1. $R[0] \leftarrow R[1]$
   - 3.2.. $R[1] \leftarrow 2R[1]$
   - 3.3. $Q[1] \leftarrow Q[0] + R[0]$
   - 3.4. $Q[0] \leftarrow Q[d_{i-1}]$
4. output $Q[0]$

스칼라 곱셈 알고리즘에서 step 3.2와 step 3.3은 연산이 독립적이므로 병렬로 연산하는 구조를 택하였고, 덧셈과 두 배 연산의 내부를 살펴보면 소수체 원소에 대한 덧셈, 뺄셈, 제곱, 곱셈, 역원 셈이 사용된다. 이 연산 중 역원 셈이 가장 많은 시간이 소요되며, $2P$의 y_3을 계산할 때 4번의 곱셈이 사용된다. 덧셈과 두 배 연산의 소요시간과 동작 구조를 같게 하기 위해서 역원 셈을 곱셈의 4배가 되도록 하였다. 그리고 역원 셈을 개선된 Mongomery 알고리즘을 사용하여 구현하였다[14]. 덧셈은 4비트 단위로 나누어 한 클럭 사이클에 계산이 되는 구조를 택하였다.

스칼라 곱셈의 구조는 [Fig. 1]과 같으며 면적의 효율성을 위해 곱셈기와 역원기를 한 개만 사용하는 구조를 사용하였다[15].

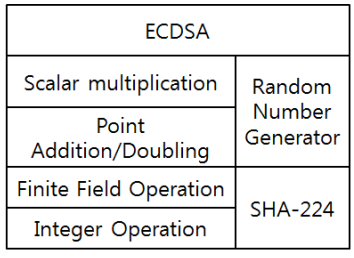


[Fig. 1] 스칼라 곱셈기 구조

3. 테스트 벡터 생성기

3.1 소프트웨어 구조

2장에서 설명하였듯이 목표 하드웨어는 병렬구조를 가지고, 하부 구조로 4비트 단위의 덧셈기를 사용한다. 하드웨어의 각 모듈에 대한 테스트 벡터를 만들기 위해서는 소프트웨어도 하드웨어와 같은 구조로 구현되어야 한다. 그러나 소프트웨어는 직렬구조로 작성되므로 하드웨어와 같게 만들 수는 없지만 하드웨어의 구조를 따라 거의 유사하게 구현할 수 있다. [Fig. 2]는 소프트웨어의 스택구조이다.[15]

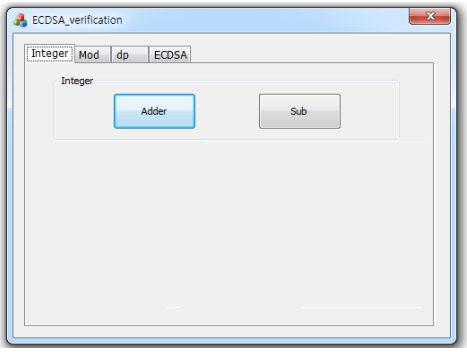


[Fig. 2] ECDSA 소프트웨어 스택

3.2 구현 결과

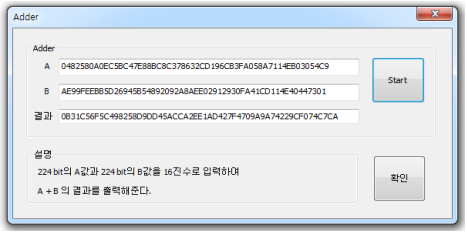
[Fig. 3]은 ECDSA 테스트 벡터 생성기의 화면으로 Integer 탭, Mod 탭, dp 탭, ECDSA 탭으로 구성된다[15].

Integer 탭은 224비트 정수 연산의 결과 값에 대한 테스트 벡터를 출력한다. 하드웨어가 4비트 단위로 계산하므로 4비트 단위의 값과 올림 수를 따로 출력할 수 있다. 테스트 벡터는 윈도우 화면에 제시하고 텍스트 파일에 저장도 한다.



[Fig. 3] ECDSA 테스트 벡터 생성기

[Fig. 4]는 정수 연산 중 덧셈에 대한 테스트 벡터의 GUI 화면 출력이고, <Table 3>은 테스트 벡터가 텍스트 파일에 저장된 형태이다.



[Fig. 4] 소수체 두 원소의 덧셈 GUI

〈Table 3〉 소수체 두 원소의 덧셈

a	0482580A0EC5BC47E88BC8C37863 2CD196CB3FA058A7114EB03054C9
b	AE99FEEBB5D26945B54892092A8A EE02912930FA41CD114E40447301
a+b	B31C56F5C498258D9DD45ACCA2EE 1AD427F4709A9A74229CF074C7CA

Mod 탭은 정수 연산을 한 후 소수 p로 나눈 나머지를 계산하여 결과를 윈도우와 텍스트 파일로 보여주는 것이다. dP 탭은 타원곡선 위의 점에 대한 스칼라 곱셈 결과를 창과 파일로 보여주는 것이다. ECDSA 탭은 타원곡선 전자서명의 결과를 창과 파일로 보여주는 탭이다.

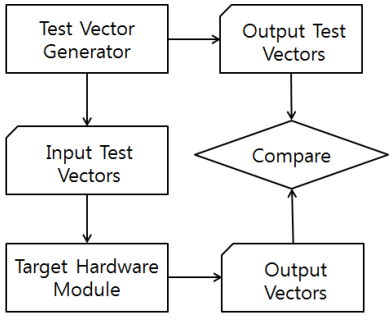
테스트벡터 생성기는 하드웨어 각 모듈에 대한 테스트 벡터를 만들어 줄 뿐 아니라 ECDSA 전자서명을 실행하면 하부 연산인 스칼라 곱셈, 모듈러 셈, 정수 셈의 결과가 파일로 저장된다. 따라서 하드웨어와 결과가 테스트벡터 생성기의 결과 값이 틀린 경우 하드웨어의 어느 모듈에서 결과가 잘못되었는지 확인을 할 수 있다.

3.2 하드웨어 테스트 방법

암호모듈에 대한 검증기법은 CMVP(Cryptographic Module Validation Program)에 의해 규정되어 있다[16]. ECDSA 전자서명에 대한 시험방법은 [17]에 제시되어 있다. 그런데 암호모듈 검증 시에는 완성된 제품에 대해 입출력을 시험하므로 이 시험에서 불합격을 하면 암호모듈의 어느 부분이 잘못 구현되었는지 알 수는 없다.

하드웨어는 HDL(Hardware Description Language) 언어로 코딩된 후 기능검증을 거치고 합성을 통한 결과물에 대해 시간 시뮬레이션을 시행한다. 시간 시뮬레이션이 성공하면 실제 FPGA 칩에 실행파일을 넣고 동작결과를 시험한다. 하드웨어 시간 시뮬레이션은 매 클럭 싸이클마다 외부 포트로 출력되는 값을 확인해야 하므로 합성 시간 및 시뮬레이션 시간이 많이 소요된다. 따라서 각 모듈에 대한 시간 시뮬레이션 시 하부 구조의 어느 부분이 잘못되었는지 확인을 할 수 있다면 개발과정을 단축할 수 있다.

제안하는 하드웨어 개발단계의 모듈에 대한 테스트 방법은 [Fig. 5]와 같다. 구현한 테스트 벡터 생성기는 임의로 테스트 벡터 쌍들을 만든다. 테스트 벡터 쌍 중 입력 벡터들을 목표 하드웨어 모듈로 제공하고 시간 시뮬레이션 테스트 벤치로 이를 사용하여 결과를 출력한다. 이 결과와 테스트 벡터 쌍 중 출력 벡터를 비교하여 목표 하드웨어 모듈이 정상 동작함을 확인한다.



[Fig. 5] 하드웨어 모듈 시험 제안 방법

4. 결론

사물인터넷 시대가 도래하면서 다양한 스마트 디바이

스가 만들어지고 있다. 그런데 스마트 디바이스의 시스템 환경이나 통신 환경이 취약하여 이에 대한 보안 기술이 필요하다. 전자서명 기술은 데이터 무결성과 출처인증을 동시에 제공하므로 스마트 디바이스 보안을 위해 널리 사용될 수 있다. 그런데 전자서명은 복잡한 수학연산을 통해 동작하므로 이를 하드웨어로 구현하였을 때 내부 모듈의 동작을 확인하기 위한 테스트 벡터를 만드는 것이 개발자에게는 어려운 과정 중 하나이다. 본 논문에서는 목표 ECDSA 전자서명 하드웨어의 구조에 따른 테스트 벡터 생성기를 구현하고 이를 통해 하드웨어 모듈을 검증하는 방법을 제시하였다. 테스트 벡터 생성기를 통해 하드웨어 개발의 속도를 증진시킬 수 있길 바라며 향후 여러 가지 암호 알고리즘에 대한 테스트 벡터 생성기를 개발할 예정이다.

ACKNOWLEDGMENTS

본 논문은 2013년도 산업통산자원부의 재원으로 한국에너지기술평가원(KETEP)의 지원을 받아 수행한 연구과제 입니다.(No. 2011T100100534)

REFERENCES

[1] John A. Stankovic, "Research Directions for the Internet of Things", Internet of Things Jour., IEEE, Vol.1, Issue 1, pp.3-9, 2014.

[2] C. Qiang, G. Quan, B. Yu and L. Yang, "Research on Security Issues of the Internet of Things", International Journal of Future Generation Communication and Networking, Vol.6, No.6, pp.1-10, 2013.

[3] J. Nisha, S. Saetang, C. Chen, S. Kutzner, S. Ling and A. Poschmann, "Feasibility and practicability of standardized cryptography on 4-bit micro controllers", In Selected Areas in Cryptography, pp.184-201, 2013.

[4] Z. Liu, H. Seo, J. Großschädl and H. Kim, "Reverse Product-Scanning Multiplication and Squaring on 8-bit AVR Processors", ICS, LNCS 8958, pp.158-175, 2015.

[5] D. Hong, J.-H. Lee, D.-C. Kim, D. Kwon, K. H. Ryu and D.-G. Lee, "LEA: A 128-bit block cipher for fast encryption on common processors", Information Security Applications, LNCS 8267, pp.3-27, 2014.

[6] Z. Liu, H. Seo, J. Großschädl and H. Kim, "Efficient Implementation of NIST-Compliant Elliptic Curve Cryptography for Sensor Nodes", ICICS2013, LNCS 8233, pp.302-317, 2013.

[7] S. Kumar, "Elliptic Curve Cryptography for Constrained Devices", A dotorial thesis. 2006.

[8] NIST, "Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths", NIST Special Publication 800-131A Rev.1, pp.6-7. 2015.

[9] D. Hankerson, A. Menezes and S. Vanstone, "Guide to Elliptic Curve Cryptography", Springer, 2004.

[10] Certicom Research, "SEC 2: Recommended Elliptic Curve Domain Parameters", SEC 2 (Draft) Ver. 2.0, 2010.

[11] J. S. Coron, "Resistance Against Differential Power Analysis for Elliptic Curve Cryptosystems", CHES'99, LNCS 1717, pp. 292-302, 1999.

[12] M. Joye, "Fast Point Multiplication on Elliptic Curves Without Precomputation", Arithmetic of Finite Fields (WAIFI 2008), LNCS 5130, pp.36-46, 2008.

[13] Y. J. Yoon, S. W. Jung, S. Lee, "Architecture for an Elliptic Curve Scalar Multiplication Resistant to Some Side-Channel Attacks", ICISC2003, LNCS 2971, pp.139-151, 2004.

[14] E. Saas and C. K. Koc, "The Montgomery Modular Inverse - Revisited", IEEE Trans. on Comp., Vol. 49, No. 7, pp.763-766, 2000.

[15] T. H. Kim and S. W. Jung, "Implementation of a Software Test Bench for Developing Stage of a 224-bit ECDSA Hardware Architecture", ICIoTC2015, pp.108-109, 2015.

[16] NIST, "Security requirments for cryptographic modules", FIPS PUB 140-2, 2002.

[17] NIST, "The FIPS 186-4 Elliptic Curve Digital Signature Algorithm Validation System(ECDSA2VS)", 2014.

김 태 훈(Tae Hun Kim) [정회원]

- 2010년 3월 ~ 2013년 8월: 목포대학교 공과대학 정보보호학과 (공학사)
- 2013년 9월 ~ 2015년 2월: 목포대학교 정보보호기술학협동과정 (공학석사)
- 2015년 6월 ~ 현재 : KDN 연구원

<관심분야>
암호 알고리즘 구현, 암호모듈 테스트, 보안관제

정 석 원(Seok Won Jung) [정회원]

- 1993년 2월 : 고려대학교 일반대학원 수학과(이학석사)
- 1997년 2월 : 고려대학교 일반대학원 수학과(이학박사)
- 1999년 2월 ~ 2001년 2월: (주) 텔리맨 책임연구원
- 2002년 3월 ~ 2004년 2월: 고려대학교 정보보호기술연구센터 조교수
- 2004년 3월 ~ 현재 : 목포대학교 정보보호학과 교수

<관심분야>

암호 알고리즘 설계, 부채널공격법, 스마트카드 보안