GF(p) 224-비트 ECC와 2048-비트 RSA를 지원하는 공개키 암호 프로세서

성병윤 · 신경욱

금오공과대학교

A Public-Key Cryptography Processor Supporting GF(p) 224-bit ECC and 2048-bit RSA

Byung-Yoon Sung · Kyung-Wook Shin

Kumoh National Institute of Technology

E-mail: sungby0809@kumoh.ac.kr

요 약

GF(p)상 타원곡선 암호(ECC)와 RSA를 단일 하드웨어로 통합하여 구현한 공개키 암호 프로세서를 설계하였다. 설계된 EC-RSA 공개키 암호 프로세서는 NIST 표준에 정의된 소수체 상의 224-비트 타원곡선 P-224와 2048-비트 키 길이의 RSA를 지원한다. ECC와 RSA가 갖는 연산의 공통점을 기반으로 워드기반 몽고메리 곱셈기와 메모리 블록을 효율적으로 결합하여 최적화된 데이터 패스 구조를 적용하였다. EC-RSA 공개키 암호 프로세서는 Modelsim을 이용한 기능검증을 통하여 정상동작을 확인하였으며, 0.18㎛ CMOS 셀 라이브러리로 합성한 결과 11,779 GEs와 14-Kbit RAM의 경량 하드웨어로 구현되었다. EC-RSA 공개키 암호 프로세서는 최대 동작주파수 133 MHz이며, ECC 연산에는 867,746 클록주기가 소요되며, RSA 복호화 연산에는 26,149,013 클록주기가 소요된다.

키워드

Elliptic curve cryptography, RSA, Public-key cryptosystem, Montgomery multiplier

I. 서 론

네트워크를 통하여 정보를 송·수신하는 IoT (Internet of Things)가 발달함에 따라 IoT에 적합한 보안이 요구되고 있다. 이에 따라 제한적인 하드웨어 자원을 갖는 IoT 디바이스의 스펙을 고려하여 수많은 경량 암호 기술들이 등장하고 있다. 그러나 IoT 네트워크 환경을 위한 경량 통신 프로토콜은 아직 완벽하게 표준화되지 않았다. 따라서 IoT 디바이스는 외부와의 네트워크 연결을 위한 범용 암호 기술을 지원할 수 있어야 하며, 그 중 공개키 암호 알고리듬이 필요한 기술 중 하나이다. 공개키 암호 알고리듬은 RSA[1]와 타원곡선 암호[2] (elliptic curve cryptography)가 대표적으로 사용되고 있다. RSA는 최근까지 대표적인 공개키 암호로 여러 분야에 사용되었다. RSA는 네트워크 및 기술의 발달로 점점 더 높은 강도의 보안성이 요구됨에 따라 키 길이의 증가로 인하여 연산 소요시간이 증가하였다. 최근 RSA를 대체하는 공개키 암호의 사용이 확대되고 있다. RSA를 대체하는 공개키 알고리듬으로 ECC의 사용이 확대되고 있으며, ECC는 RSA보다 짧은 키 길이를 사용하여도 동일한 보안 강도를 갖는다. 따라서 IoT 디바이스는 RSA와 ECC를 모두 지원할 수 있는 범용성을 가져야 한다[3].

본 논문에서는 RSA와 ECC를 동시에 지원하는 공개키 암호 프로세서를 IoT 디바이스에 적합한 경량 하드웨어로 설계하였다. 본 논문의 EC-RSA 프로세서는 워드기반 몽고메리 곱셈기를 기반으로 설계 되었다[4]. 워드기반 몽고메리 곱셈기는 32-비트의 배수가 되는 데이터의 곱셈연산이 가능하다. 서로 다른 길이의 곱셈 연산이 요구되는 RSA와 ECC 하드웨어 구조에 워드기반 몽고메리 곱셈기를 공유하여 자원 소모를 줄였다. II장에서는 EC-RSA 프로세서의 하드웨어 구성에 대해 간략하게 설명한다. 시뮬레이션 결과를 III장에서 기술하고, IV장에서 결론을 맺는다.

II. EC-RSA 공개키 암호 프로세서 설계

EC-RSA 공개키 암호 프로세서는 키 길이 2048-

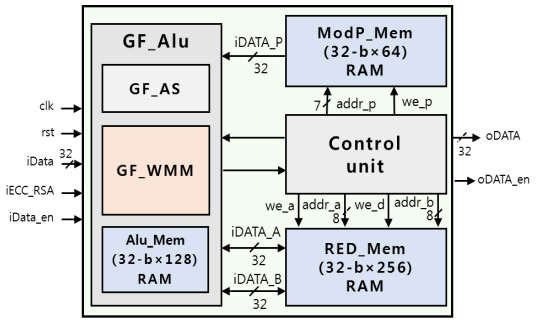


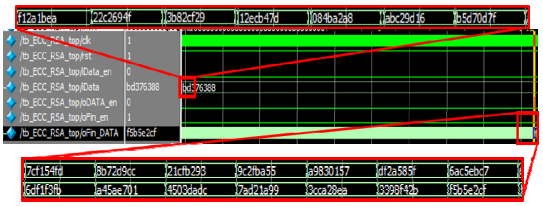
Fig. 1. Architecture of EC-RSA processor,

비트의 RSA와 소수체 상의 P-224 타원곡선 암호를 지원한다. 그림 1은 EC-RSA 공개키 암호 프로세서의 내부 구조이며, 두 개의 RAM 블록(ModP_Mem, RED_Mem), GF_Alu 블록 그리고 제어블록으로 구성된다. 두 개의 RAM 중 ModP_Mem은 모듈러 연산을 위한 모듈러 값을 저장하며, RED_Mem은 정수 k, 타원곡선 생성점의 좌표값과 RSA와 ECC 연산에 필요한 초기값과 암·복호 연산의 중간 결과값이 저장된다. GF_Alu 블록에서는 RSA와 ECC의 암호·복호를 위한 산술연산이 수행되며, GF_AS, GF_WMM 그리고 Alu_Mem으로 구성된다. GF_AS에서는 32-비트 단위의 모듈로 가산·감산 연산이 수행되며, GF_WMM에서는 32-비트 단위의 모듈로 곱셈연산이 수행된다. Alu_Mem에는 모듈로 연산의 중간 결과값이 저장된다.

EC-RSA 프로세서의 연산 순서는 데이터 입력, 매핑, RSA 및 ECC를 위한 산술연산, 리매핑 그리고 데이터 출력 순이다. 매핑 과정은 ECC와 RSA 연산에 사용되는 파라미터를 워드기반 몽고메리 곱셈기에 적합한 형태로 변환하는 과정이며, 리매핑은 변환된 데이터를 일반 데이터 형태로 역변환하는 과정이다. ECC 모드의 연산은 스칼라 곱셈과 좌표계 변환으로 구성된다. 스칼라 곱셈은 부채널 공격에 강한 몽고메리 레더(Montgomery ladder) 알고리듬이 사용되었으며, 점 연산은 자코비안(Jacobian's) 좌표계로 구현되었다. 자코비안 좌표계는 $(x,\ y,\ z)$로 표현되는 3차원 좌표계이므로 모든 스칼라 곱셈이 완료된 후에는 2차원 아핀(affine) 좌표계로 변환되어야 한다. 좌표계 변환은 스칼라 곱셈이 완료된 점 $(x,\ y,\ z)$를 점 $(x/z^2,\ y/z^3)$으로 연산해주는 과정이다. 좌표계 변환을 위해서 역원 연산이 한번 수행되며, 역원 연산은 곱셈기만으로 연산 가능한 페르마의 소정리(Fermat's little theorem)를 적용하여 구현하였다. RSA 모드의 모듈러 멱승연산에는 left-to-right 이진 알고리듬을 적용하였다.

III. 시뮬레이션 기능검증

EC-RSA 공개키 암호 프로세서는 Modelsim을

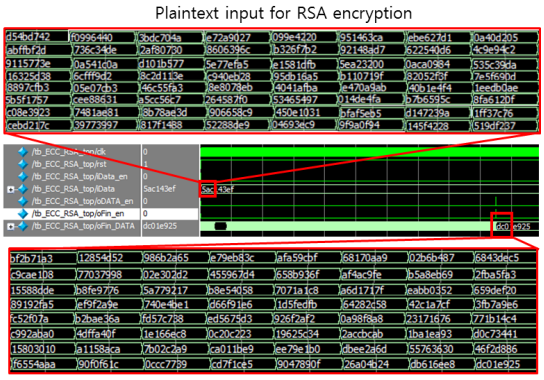


(a)

정수 k 0xf12a1bea22c2694f3b82cf2912ecb47d084ba2a8abc29d16b5d70d7f
좌표 x 0x7cf154fd8b72d9cc21cfb2939c2fba55a9830157df2a585f6ac5ebc7
좌표 y 0x6df1f3fba45ae7014503dadc7ad21a993cca28ea3398f42bf5b5e2cf

(b)

Fig. 2. Functional simulation results (a) ECC mode (b) software result.



(a)

----------Plaintext----------
0xd54bd742f09964403bdc704ae72a9027099e4220951463caebe627d10a40d205
0xabff bf2d736c34de2af807308606396cb326f7b292148ad7622540d64c9e94c2
0x9115773e0a541c0ad101b5775e77efa5e1581df b5ea232000aca0984535c39da
0x16325d386cf f f9d28c2d113ec940eb2895db16a5b110719f82052f3f7e5f690d
0x8897cf b305e07cb346c55f a38e8078eb4041af bae470a9ab40b1e4f41eedb0ae
0x5b5f1757cee88631a5cc56c7264587f053465497014de4f ab7b6595c8f a6120f
0xc08e39237481ae818b78ae3d906658c9450e1031bf af5eb5d147239a1f f37c76
0xcebd217c39773997817f148852288de904693ec99f9a0f94145f4228519df237
----------Cypertext----------
0xbf2b71a312854d52986b2a65e79eb83caf a59cbf68170aa902b6b4876843dec5
0xc9cae1087703799802e302d2455967d4658b936f af4ac9f eb5a8eb692f ba5f a3
0x15588ddeb8f e97765a779217b8e540587071a1c8a6d1717f eabb0352659def 20
0x89192fa5ef 9f2a9e740e4be1d66f91e61d5f edf b64282c5842c1a7cf3fb7a9e6
0xfc52f07ab2bae36af d57c738ed5675d3926f2af 20a98f8a8231716767 71b14c4
0xc992aba04df f a40f1e166ec80c20c22319625c342accbcab1ba1ea93d0c73441
0x15803010a1158aca7b02c2a9ca011be9ee79e1b0dbee2a6d5576363046f 2d886
0xf6554aaa90f0f61c0ccc7739cd7f1ce59047890f26a04b24db616ee8dc01e925

(b)

Fig. 3. Functional simulation results (a) RSA mode (b) software result.

이용한 RTL 시뮬레이션 결과를 Python으로 구현한 소프트웨어 결과와 비교하여 기능검증을 하였다. 그림 2는 ECC 모드 동작에 대한 RTL 의 시뮬레이션 결과값과 Python을 이용한 ECC 연산의 결과값을 보여준다. NIST FIPS 186-2에 정의되어 있는 P-224 타원곡선 파라미터를 사용하였으며, 224-비트 정수 "F12A1BEA 22C2694F 3B82CF29 12ECB47D 084BA2A8 ABC29D16 B5D70D7F"를

이용하여 ECC 연산을 수행하였다. ECC 연산에는 총 867,746 클록 주기가 소요되며, 모든 연산이 완료되면 oDATA_en 신호와 함께 x, y 좌표가 출력된다. 출력된 x, y 좌표는 그림 2-(b)의 Python 소프트웨어 결과와 정확하게 일치함을 확인할 수 있다.

그림 3은 EC-RSA 프로세서의 RSA 모드 동작에 대한 RTL 기능검증 결과와 Python을 이용한 소프트웨어 구현 결과의 비교를 보이고 있다. RSA 암호를 위한 공개키 e는 일반적인 값 10001_{16}가 사용되었으며, 평문은 Python 소프트웨어의 랜덤함수를 이용하여 생성하였다. RSA 암호화와 복호화 연산에 각각 189,828과 26,149,013 클록 주기가 소요된다. RSA 암호연산이 완료되면 oDATA_en 신호와 함께 암호문이 출력되며, 그림 3-(b)의 Python 소프트웨어 결과와 정확하게 일치함을 확인할 수 있다.

IV. 결 론

키길이 2048-비트의 RSA와 소수체 상 P-224의 타원곡선 암호를 단일 하드웨어로 통합하여 구현한 EC-RSA 공개키 암호 프로세서를 설계하였다. EC-RSA 공개키 암호 프로세서는 0.18㎛ CMOS 셀 라이브러리로 합성한 결과 11,779 GEs와 14-Kbit RAM의 경량 하드웨어로 구현되었다. 최대 동작주파수 133 MHz이며, ECC의 스칼라 곱셈 연산에는 867,746 clock cycles이 소요되며, RSA 복호화 연산에는 26,149,013 clock cycles이 소요된다. 본 논문의 EC-RSA 공개키 암호 프로세서는 경량화 구현을 특징으로 하므로, 하드웨어 자원이 제한된 IoT와 무선센서 네트워크 (WSN) 등의 보안에 활용될 수 있다.

ACKNOWLEDGMENTS

- This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education (No. 2017R1D1A3B03031677)
- Authors are thankful to IDEC for EDA software support.

참고문헌

[1] R. Rivest, A. Shamir, and L. Adleman, “A method for obtaining Digital Signatures and Public-Key Crypto-systems,” *Communications of the ACM*, vol. 21, no. 2, pp. 120-126, Feb. 1978.

[2] NIST Std. FIPS PUB 186-2, *Digital Signature Standard (DSS)*, National Institute of Standard and Technology (NIST), Jan. 2000.

[3] Seoul national university of science and technology, “Study on application of public key algorithm and hash function in IoT envuronment,” *KISA*, Oct. 2016.

[4] P. L. Montgomery, “Modular Multiplication Without Trial Division,”*Mathematics of Computation,* vol. 44, no. 170, pp. 519-521, Apr. 1985.

[5] W. L. Cho, K. W. Shin, “2,048 bits RSA public-key cryptography processor based on 32-bit Montgomery modular multiplier,” *Journal of the Korea Institute of Information and Communication Engineering*, vol. 21, no. 8, pp. 1471-1479, Aug. 2017.