2048-비트 RSA 공개키 암호 프로세서

조욱래* · 신경욱*

*금오공과대학교

2048-bit RSA Public-key Crypto-processor

Wook-Lae Cho* · Kyung-Wook Shin*

*Kumoh National Institute of Technology

E-mail : jodnrfo2@kumoh.ac.kr

요 약

2048-bit의 키 길이를 지원하는 공개키 암호 프로세서 RSA-2048을 설계하였다. RSA 암호 연산에 사용되는 핵심 기능블록인 모듈러 곱셈기는 Word-based Montgomery Multiplication 알고리듬으로 설계하였으며, 모듈러 지수 승은 L-R binary exponentiation 알고리듬으로 설계하였다. 2048-bit의 큰 정수를 저장하기 위한 레지스터를 메모리로 대체하고, 곱셈기에 필요한 최소 레지스터만 사용하여 전체 하드웨어 자원을 최소화 하였다. Verilog HDL로 설계된 RSA-2048 프로세서를 RTL-시뮬레이션을 통해 기능을 검증하였다. 작은 소형 디바이스들 간에 인증 및 키 관리가 중요해짐에 따라 설계된 RSA-2048 암호 프로세서를 하드웨어 자원, 메모리가 제한된 응용 분야에 활용 할 수 있다.

키워드

RSA, 공개키 암호, 몽고메리 모듈러 곱셈 알고리듬, 모듈러 곱셈기

Ⅰ. 서 론

IoT는 각종 산업 분야에 적용되어 많은 부가가치를 창출하고 있으며, 홈/가전, 의료, 교통 등 각 분야에서 본격적으로 시장이 활성화 되고 있다. IoT 기술의 활성화 및 신규 서비스 창출을 위해 정보보안은 반드시 제공해야 하는 필수적 요소이며, 정보보호가 담보되지 않은 환경에서 치명적인 보안 위협들이 발생할 것으로 예상된다[1].

정보보안을 위하여 암호기술들이 다양하게 사용되고 있으며, 대칭키 암호, 해시, 공개키 암호 등을 이용한 무결성, 기밀성, 사용자 및 기기간의 인증, 키분배 기술이 IoT 보안의 핵심요소이다. 공개키 암호와 대칭키 암호는 정보보안에서 상호 보완적인 기능을 수행하며, 송/수신자 간의 인증, 키 분배, 무결성 검증 등을 수행하는 공개키 암호 시스템의 중요성이 날로 강조되고 있다.

대표적으로 사용되는 공개키 암호인 RSA는 두 개의 큰 정수를 곱한 수의 인수분해가 어려운 점에 안정성을 둔다. RSA 공개키 암호는 현재 에너지/건설/환경/교통/헬스케어/스마트 홈 등 분야별로 다양하게 활용되고 있으며, 한국인터넷진흥원(KISA)에서는 2011년부터 2,048 비트 이상의 키 길이를 지원하는 RSA 암호를 권장하고 있다.

본 논문에서는 2,048 비트의 키 길이를 지원하는 RSA 공개키 암호 프로세서를 설계하고, FPGA 구현을 통해 하드웨어 동작을 검증하였다. 2장에서는 RSA-2048 암호 프로세서 구조를 설명하고, 3장은 기능검증 및 FPGA 구현을 설명하며, 4장에서 결론으로 마무리 짓는다.

Ⅱ. 몽고메리 모듈러 곱셈 알고리듬

큰 정수 A와 B에 관한 몽고메리 모듈러 곱셈 알고리듬은 식 (1)과 같이 표현된다. A와 B는 $A,B < N$이고, $R=2^k$과 N은 $\gcd(R,N)=1$을 만족하며, N의 범위는 $2^{k-1} < N < 2^k$이다.

$$Z = A \cdot B \cdot R^{-1} \bmod N \qquad (1)$$

그림 1은 워드 기반(word-based) 몽고메리 모듈러 곱셈 알고리듬의 슈도코드를 나타낸다[2]. 각각의 k 비트 정수(A, B, N)에 관해 w 비트의 워드 s 개로 분할하여 연산을 진행한다. 두 개의 반복 루프로 구성되며, 외부 반복루프에서는 승수를 워드 단위로 스캔하여 내부 반복루프에 사용

$Input:$ $A = \{a_{s-1}, \cdots, a_1, a_0\}_{2^w}$
$B = \{b_{s-1}, \cdots, b_1, b_0\}_{2^w}$
$N = \{n_{s-1}, \cdots, n_1, n_0\}_{2^w}$
$R = 2^k$
$W = 2^w$
$n' = -n_0^{-1} \bmod W$
$Output:$ $Z = \{z_{s-1}, \cdots, z_1, z_0\}_{2^w}$
$= ABR^{-1} \bmod N$

1: $Z = 0;\ u = 0;\ S = 0;$
2: ***for*** $i = 0$ ***to*** $s - 1$ ***do***
3: $S = z_0 + a_i b_0;$
4: $C_a = S/W;\ z_0 = S \bmod W;$
5: $x_i = z_0 n' \bmod W;$
6: $S = z_0 + x_i n_0;$
7: $C_b = S/W;\ z_0 = S \bmod W;$
8: ***for*** $j = 1$ ***to*** $s - 1$ ***do***
9: $S = z_j + a_i b_j + C_a;$
10: $C_a = S/W;\ z_0 = S \bmod W;$
11: $S = z_j + x_i n_j + C_b;$
12: $C_b = S/W;\ z_0 = S \bmod W;$
13: ***end for***
14: $S = C_a + C_b + u;$
15: $u = S/W;\ z_0 = S \bmod W;$
16: ***end for***
17: ***if*** $Z > N$ ***then*** $Z = Z - N;$

Fig. 1 Word-based Montgomery multiplication algorithm

될 부분곱을 생성한다. 내부 반복루프에서는 매 반복연산마다 부분곱의 가산과 리덕션이 수행된다. 식 (1)과 그림 1에서 볼 수 있듯이, 몽고메리 모듈러 곱셈 알고리듬은 곱셈 결과에 R^{-1}이 포함되므로, 매 곱셈 연산마다 R을 곱하여 R^{-1}을 제거해야한다. 그러나 그림 2와 같이 전처리 과정인 매핑과 후처리 과정인 역매핑을 적용하면, 몽고메리 곱셈마다 R을 곱할 필요가 없어지기 때문에 몽고메리 곱셈을 보다 효율적으로 구현할 수 있다.

III. RSA-2048 암호 프로세서 구조

그림 3은 2,048 비트의 키 길이를 지원하는 RSA-2048 프로세서의 전체 구조이며, WMM (Word-based Montgomery Multiplier) 블록, 제어 블록, 2,048 비트 메모리 6개로 구성된다. 메모리 KeyM은 2,048 비트의 키(공개키 또는 개인키) 값을 저장하며, 메모리 Msg는 입력되는 평문/암호문 값을 저장한다. WMM 블록은 몽고메리 모듈러 곱셈을 32 비트 단위로 연산하며, 그 중간 결과는 메모리 MMz에 저장된다. 메모리 RSAt는 RSA 암호화/복호화의 중간 결과를 저장하고, 메

$MM(Montgomery\ Multiplication)$
Mapping: $MM(A, R^2) = A \times R^2 \times R^{-1} = AR \bmod N$
Mapping: $MM(B, R^2) = B \times R^2 \times R^{-1} = BR \bmod N$
Multiplication: $MM(AR, BR) = AR \times BR \times R^{-1} = ABR \bmod N$
re-mapping: $MM(ABR, 1) = ABR \times 1 \times R^{-1} = AB \bmod N$

Fig. 2 Mapping and re-mapping process for Montgomery modular multiplication

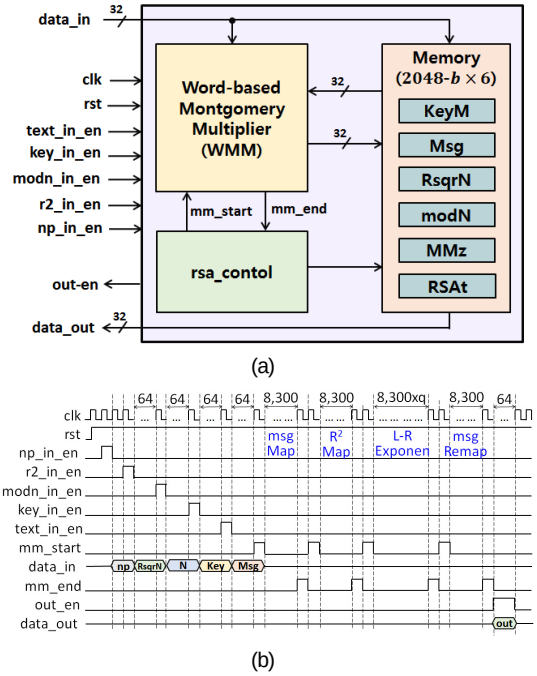

Fig. 3 (a) Overall architecture and (b) timing diagram of the RSA-2048 processor

모리 RsqrN은 매핑인자 $R^2 \bmod N$을 저장하며, 메모리 modN은 모듈러 상수 N을 저장한다.

RSA-2048의 동작 타이밍 도는 그림 4-(b)와 같다. 매핑인자 $R^2 \bmod N$, 모듈러 상수 N, 공개키/개인키, 그리고 평문/암호문이 각각 32 비트 단위로 64 클록 주기에 걸쳐 순차적으로 입력되어 해당 메모리에 저장된다. 데이터 입력이 완료 된 후, mm_start 신호에 의해 전처리 매핑, 모듈러 멱승 연산, 역매핑이 순차적으로 진행되어 RSA 암호 또는 복호 연산이 수행된다. 연산이 완료되면, 암호/복호문이 32비트 단위로 64 클록 주기 동안 out_en 신호와 함께 출력된다.

IV. 기능검증

RSA-2048 프로세서를 Verilog HDL로 설계하여 RTL-simulation을 통해 기능검증을 수행하였다. 그림 4는 RSA-2048 프로세서의 기능검증 결과의 일부를 보인다. 데이터를 모두 입력 받으면, Msg_mapping을 수행한다. r2_mapping 연산 후에 Key_scan을 통해 유효한 키 길이를 확인한 후 L-R 이진 멱승 연산을 시작하며, 연산이 완료되면 역매핑 후에 out_en 신호와 함께 결과를 32 비트씩 64 클록 주기에 걸쳐 출력한다.

RSA-2048 프로세서는 공개키 $e(10001_{16})$를 사용한 암호화에 185,724 클록 사이클이 소요되고, 2,048 비트의 개인키 d를 사용한 복호화에 25,561,076 클록 사이클이 소요된다. 설계된 RSA-2048 프로세서를 0.18㎛ CMOS 표준 셀 라

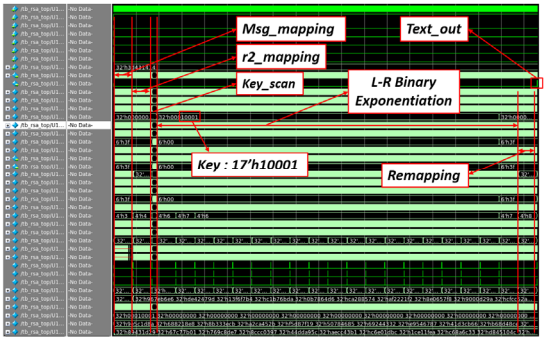

Fig. 4 RTL simulation result of RSA-2048 processor

이브러리로 합성한 결과, 100 MHz의 동작 주파수에서 12,540 GE와 12k bit(2,048 비트의 메모리 블록 6개)의 메모리로 구현되었다. 최대 동작주파수 165 MHz에서 암호화 연산에 1.12 ms 가 소요되며, 복호화 연산에 157.91 ms가 소요된다.

V. 결 론

2,048 비트의 키를 지원하는 RSA 공개키 암호 프로세서를 설계하고, RTL-simulation을 통해 기능검증을 수행하였다. RSA 연산에 핵심이 되는 모듈러 곱셈기를 워드 기반 몽고메리 모듈러 곱셈(WMM) 알고리듬을 이용하여 설계하였다. 곱셈기에 필요한 최소 레지스터만 사용하였으며, 큰 정수를 저장하기 위해 메모리를 사용함으로써 전체적인 하드웨어 자원을 최소화하였다. RSA-2048 프로세서를 100 MHz의 동작주파수에서 암호프로세서는 12,540 GE, 12 킬로비트의 메모리로 구현되었다. 최대 165 MHz로 동작 가능하며, RSA 암호화와 복호화 연산에 각각 1.12 ms와 154.91 ms가 소요된다. 본 논문에서 설계된 2,048 비트 RSA 암호 프로세서는 제한된 하드웨어 자원을 갖는 디바이스 간의 인증 또는 키 관리를 위한 공개키 기반 보안 하드웨어 구현에 활용이 가능하다.

ACKNOWLEDGMENTS

· This work was supported by Korea Institute for Advancement of Technology (KIAT) grant funded by the Korean government (Ministry of Trade, Industry & Energy, HRD Program for Software-SoC convergence) (No. N0001883). The authors also are thankful to IDEC for EDA software support.

참고문헌

[1] IoT Security Alliance, IoT Common Security Principle v1.0 [Internet]. Available: http://www.kisa.or.kr/public/laws/laws3_View.jsp?mode=view&p_No=259&b_No=259&d_No=67&ST=T&SV=.

[2] A. Miyamoto, N. Homma, T. Aoki and A. Satoh, "Systematic Design of RSA Processors based on High-radix Montgomery Multipliers," IEEE Transactions on Very Large Scale Integration(VLSI) Systems, vol. 19, no. 7, pp. 1136-1146, July. 2011.

[3] J. Shao, L. Wu and X. Zhang, "Design and implementation of RSA for dual interface bank IC card," *2013 IEEE 10th International Conference on ASIC (ASICON)*, pp 1-4, 2013.

[4] M. S. Kim, Y. S. Kim and H. S. Cho, "Design of Cryptographic Hardware Architecture for Mobile Computing," *Journal of Information Proceeding Systems*, vol. 5, no. 4, pp. 187-196, Dec. 2009.

[5] X. Zheng, Z. Liu and B. Peng, "Design and Implementation of Ultra low power RSA processor," *Wireless Communications, Networking and Mobile Computing WiCom'08 4th International Conference on*, pp. 1-5, 2008.