Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.29, NO.4, Aug. 2019 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2019.29.4.697

8비트 마이크로컨트롤러 유닛 상에서 Rainbow 최적화 구현 및 분석*

홍 은 기,† 조 성 민, 김 애 영, 서 승 현‡
한양대학교

Compact Implementation and Analysis of Rainbow on 8bits-Microcontroller Uunit*

Eungi Hong,† Seong-Min Cho, Aeyoung Kim, Seung-Hyun Seo‡
Hanyang University

요 약

본 논문에 이차식 기반 서명인 Rainbow를 8비트 MCU(Microcontroller Unit)에 적용하기 위해 최적화 하는 방안을 검토하고 구현한다. 양자 컴퓨터가 개발되면서 기존의 암호 알고리즘 특히, 서명 기법의 보안성을 위협함에 따라 IoT 기기에도 양자내성을 갖춘 서명 기법을 적용해야할 필요성이 있다. 현재 제안된 양자내성암호는 격자 기반, 해쉬 기반, 코드 기반 그리고 다변수 이차식 기반 암호 알고리즘 및 서명 기법들이 있는데, 특히 다변수 이차식 서명 기법은 기존의 서명 기법과 비교해 속도가 빨라 IoT 기기에 적합하다. 그러나 키의 길이가 크고 연산이 많아 IoT 기기 중 메모리와 성능에 큰 제약이 있는 8비트 MCU에는 기존의 구조 그대로 구현하기 어려워 이에 적합한 최적화가 필요하다. 따라서 본 논문에서는 다변수 이차식 서명 기법인 Rainbow를 8비트 MCU에 키를 나누어 저장하는 방안과 연산방식을 최적화하여 메모리 소모가 적고 연산 속도가 빠르게 알고리즘을 개선하고, 구현해본 후 각 최적화 방식에 따른 성능을 비교한다.

ABSTRACT

In this paper, we propose and implement a method to optimize Rainbow for 8 bit MCU(Microcontroller Unit). As quantum computers have been developed, the security of existing cryptography, especially the signature algorithms, has been threatened, so it is necessary to apply a signature scheme with a quantum-resistance to IoT devices. Currently, the proposed PQC(Post Quantum Cryptography) are lattice-based, hash-based, code-based, and MQ(Multivariate Quadratic)-based cryptographic algorithms and signature schemes. In particular, MQ-based signature schemes are faster than conventional signature schemes and are suitable for IoT devices Do. However, it is difficult to apply 8-bit MCU, which has a large key length and large number of computations, to the memory and performance of IoT devices. In this paper, we propose a method of storing Rainbow, which is a MQ-based signing scheme, in 8-bit MCU by dividing the key and optimizing the computation method. By reducing the memory consumption and improving the algorithm speedily, Compare performance.

Keywords: NIST Post Quantum Cryptography Competition, Multivariate Quadratic-based Signaiture Algorithm, Rainbow, 8bits Microcontroller Unit

Received(01. 14. 2019), Modified(06. 10. 2019), Accepted(07. 10. 2019)

* 본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 대학ICT연구센터지원사업의 연구결과로 수행되었음(IITP-2019-2018-0-01417).

* 본 연구는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No.2018R1A2B6006903).

† 주저자, heg0403@hanyang.ac.kr

‡ 교신저자, seosh77@hanyang.ac.kr(Corresponding author)

I. 서 론

현재 양자컴퓨터 개발은 슈퍼컴퓨터의 성능에 준하는 성능 기준으로 제시된 50큐비트(Qubit)를 넘어가는 단계까지 진행되었다. Intel은 2018년 1월에 48큐비트 양자 칩을 개발했다고 발표했으며, Google은 같은 해 3월에 72큐비트 양자 칩 브리슬콘(Bristlecone)을 선보이면서 양자컴퓨터가 상용화되는 것이 가시화되었다[1]. 양자컴퓨터 상용화가 실현되면 연산 복잡도가 높아 처리 할 수 없었던 양자 알고리즘들을 구현할 수 있게 된다. 이 알고리즘들 중 쇼어의 양자 소인수분해 알고리즘(Shor's algorithm)을 이용하면 기존의 RSA와 ECDSA의 안전성을 보장하던 인수분해 문제와 이산로그 문제(DLP; Discrete Logarithm Problem)가 다항시간(polynomial time) 내에 풀리게 되어 더 이상 안전하지 않게 된다[2]. 그래서 미 국립표준기술연구소(NIST)에서는 이에 대응하기 위해 양자내성암호(PQC; Post Quantum Cryptography) 공모전을 진행하여 기존의 PKI 기반 암호 알고리즘 및 서명 기법들을 대체할 수 있는 양자내성암호 알고리즘 표준화 작업을 진행 중이다.

양자내성암호는 격자 기반(lattice-based) 암호, 해쉬 기반 암호(hash-based PQC), 코드이론 기반 암호(code-based PQC)와 다변수 이차식 기반 암호(MQ; Multivariate Quadratic-based PQC)로 분류된다. 이 중 다변수 이차식 기반 서명 기법은 기존의 RSA-3072, ECDSA-256과 같은 수준의 보안수준으로 각각 비교해 서명은 약 136배와 3배, 검증은 약 2배와 7배로 매우 빨라 여러 가지 환경의 하드웨어에 적용되어 효과적으로 사용할 수 있으며, 특히 연산능력에 제약이 있는 IoT에 적용되기에 적절하다[3]. 현재 다변수 이차식 기반 서명 기법은 Intel x64환경(MAGMA, Sage, AVX-2)에서 최적화되어 있으며, 8비트 AVR 프로세서 환경 등에서는 최적화 구현이 되어 있지 않다[4].

대표적인 다변수 이차식 기반 암호인 Rainbow는 RSA-3072, ECDSA-256과 각각 비교해 공개키가 각각 362배, 2176배 크고, 개인키가 34배, 1093배 크다[3]. 이 때문에 상대적으로 고성능의 스마트폰이나 드론, 라즈베리파이에는 그대로 구현할 수 있으나, 8비트 AVR 프로세서 환경의 MCU 및 센서 등 낮은 메모리와 성능을 갖는 기기에는 그대로 구현하기 어렵다. 따라서 본 논문에서는 낮은 메모리와 성능을 갖고 있는 기기인 8비트 MCU에 Rainbow의 키 길이를 줄이지 않고 연산을 고속화하여 구현할 수 있도록 프로그램 메모리에 공개키나 개인키를 상수로 분할하여 저장 및 운용하는 방법과 서명 생성 및 검증에 사용되는 연산 고속화 방법을 제시하고 이를 구현 및 분석한 결과를 보인다.

본 논문의 구성으로 2장에서는 제안 기법에 필요한 다변수 이차식 기반 서명과 이에 대한 최적화 방안을 서술하며, 3장 및 4장에서는 메모리와 연산 성능의 제약이 아주 큰 8비트 MCU 상에서 Rainbow를 적용 및 고속화 하는 방안을 서술하고, 이를 구현 후 분석한 결과를 보인다.

II. 관련연구

본 장에서는 다변수 이차식 기반 서명 기법에 대한 개요와 이 기법에 대한 최적화 방안을 서술한다.

2.1 다변수 이차식 기반 서명 기법

다변수 이차식 기반 서명의 비밀키는 각 변수들의 이차식의 계수들로 이루어진 중앙 맵 F와 역행렬이 가능한 아핀행렬 S와 T로 이루어지며 공개키로는 비밀키들이 합성된 F'가 된다. 이 다변수 이차식 기반 서명 기법의 안전성은 다변수 이차식 문제(problem MQ)와 다항식의 확장 동형성 문제(problem EIP;Extended Isomorphism of Polynomial)로 보장된다[5].

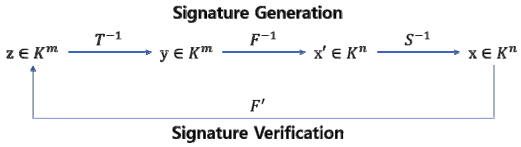


Fig. 1. Signature generation/verification in MQ Signature

- **다변수 이차식 문제**

$$y^k = f^k(x_1,\ldots,x_n) = \sum_{i,j=1}^{n} \alpha_{ij}^k x_i x_j + \sum_{i=1}^{n} \beta_i^k x_i + \gamma \quad (k = 1, 2, \ldots, m)$$

다변수 이차식 문제는 위와 같이 n개의 변수를

가진 이차 다항식 $m(m<n)$개가 주어졌을 때 $Y=(y_1,y_2,...,y_m)$들에 대한 답인 $X=(x_1,x_2,...x_n)$를 찾는 문제다. 이 문제는 다항시간 내에 풀지 못하는 문제(NP-hard problem)로 증명되어 있다.

● **다항식의 확장 동형성 문제**

EIP 문제는 공개키 $F'=S\circ F\circ T$가 주어 졌을 때 아핀행렬 S와 T를 찾는 문제이다 이는 다항시간 내에 풀지 못하는 것으로 증명되어 있지는 않지만 그만큼 어려운 문제로 알려져있다.

UOV(Unbalanced Oil and Vinegar) 서명은 다변수 이차식 기반 서명기법의 대표적인 초기 모델이다. UOV는 오일(oil)과 비네가(vinegar) 즉 기름과 식초가 층을 구분하여 담겨 있는 것에서 착안하여 n개의 변수를 섞이지 않는 두 종류의 변수(비네가 변수와 오일 변수)로 구분하여 이차식으로 구성해 키로 사용한다. 비밀키로 중앙행렬F와 아핀행렬S를 사용하고, 공개키로는 F와 S가 합성된 F'을 사용한다. UOV의 보안 수준은 오일과 비네가 변수의 개수에 의해 영향을 받는데 이때, 비네가 변수의 개수가 오일 변수의 개수보다 많아야 하며, 이 비네가 변수 개수와 오일 변수 개수의 차가 너무 크지도 작지도 않아야 한다[6][7]. 이 특징들을 충족시키게 되면, 서명 할 메시지의 길이보다 서명의 크기가 두 배 이상이 되어 비효율적이다. 이를 개선한 것이 Rainbow 서명이다. Rainbow 서명은 계층을 여러 개로 늘려 UOV와 같은 보안 정도에서 키의 길이와 서명 생성 속도를 개선하였으며, 비밀키로 중앙행렬 F와 아핀행렬S, T를 사용하고 공개키로는 이 F와 S, T를 합성한 F'를 사용하여 EIP문제를 강화하였다[6][8]. Rainbow는 NIST의 양자 내성 암호 공모전에 제출되었고, 제출된 다른 다변수 이차식 기반 서명 기법에는 LUOV와 HiMQ 등 7개가 있다.

● **Rainbow 서명**

K: GF(갈로아 필드)
o_1, o_2, v_1: 정수
$n=o_1+o_2+v_1$, $m=v_1+o_1+o_2$

위와 같은 파라미터를 가진 Rainbow는 서명을 위한 비밀키로 S, F, T를 사용한다.

$F=(F_1,\ F_2)$
$F_1=(f^{1_1},\ \ldots,\ f^{v_1+o_1})$, $F_2=(f^{1_2},\ \ldots,\ f^{v_2+o_2})$,
아핀행렬 $S: K^n\Rightarrow K^n$
아핀행렬 $T: K^m\Rightarrow K^m$

또한 서명을 검증하기 위한 공개키는 비밀키들이 합성된 $F'=S\circ F\circ T$를 사용한다.

위는 2개의 계층 F_1과 F_2를 가진 Rainbow이다[8][9]. 서명 생성을 위해서는 첫 번째 비네가에 속한 변수 $(x_1,\ \ldots,\ x_{v_1})$는 무작위로 선택된 값 v_1개를 변수로 채운 뒤 첫 번째 오일에 속한 변수$(x_{v_1+1},\ \ldots,\ x_{v_1+o_1})$로 구성된 일차식을 생성하고, 이 일차식의 해를 구해 앞의 비네가 변수와 합한$(x_1,\ \ldots,\ x_{v_1+o_1})$을 두 번째 비네가 변수라 정의하고, 이 비네가를 이용해 $(x_{v_1+o_1+1},\ \ldots,\ x_n)$로 구성된 일차식의 해를 찾아 모든 변수를 모아 놓으면 서명 $X=(x_1,\ \ldots,\ x_n)$가 된다. 이 과정은 계층이 늘어나도 동일하게 반복하면 된다. 다음으로 서명 검증은 이렇게 생성된 서명 X를 공개키 F'에 대입하여 확인한다.

2.2 다변수 이차식 기반 서명 기법 최적화 방안

GF(2^m)상의 다변수 이차식 기반 서명 기법이 고속으로 실행 가능 하도록 최적화하는 주요 방안은 GF(2^m)상의 사칙 연산 최적화, 컴파일러 내장 명령어인 SIMD(Single Instruction Multiple Data) 명령어 기반 병렬 연산 그리고 다항식의 계수에 대한 재배치를 통한 최적화가 있다.

● GF(2^m)상의 사칙 연산 최적화

GF(2^m)상의 덧셈은 비트 연산 XOR로 수행되며, 한 번의 XOR연산 명령어는 m 게이트만을 소비한다. GF(2^m)상의 뺄셈은 덧셈과 동일하다.

곱셈과 나눗셈은 일반적으로 한 연산 명령어당 O(m^2) 의 복잡도를 갖고 있어 덧셈에 비해 매우 비싼 연산이다. 다변수 이차식 기반 서명 기법에서는 지정된 메시지 길이 또는 서명 길이만큼의 변수들에 대한 이차식 및 일차식 구성으로 곱셈의 사용이 주를 이룬다. 이에 곱셈에 대한 고속화가 해당 서명 기법의 고속화로 직결된다. a/b와 같은 나눗셈은 $a\times b^{-1}$

과 같이 곱셈에 대한 역수를 이용하여 계산한다.

주요 곱셈 방법은 쉬프트 연산 및 덧셈 연산의 비트 시리얼을 이용한 곱셈, 다양한 비트 연산 조합의 비트 병렬을 이용한 곱셈, 하위 필드를 이용한 곱셈, 로그 및 안티로그 테이블을 이용한 곱셈, 라지 룩업 테이블을 이용한 곱셈 등이다.

쉬프트 연산 및 던셈연산의 비트 시리얼을 이용한 곱셈은 피연산자를 비트시리얼로 변환 후 비트 연산자 중 이동 비트 수 당 $\times 2$의 결과를 갖는 왼쪽 쉬프트 연산과 GF(2) 상의 덧셈인 XOR 연산을 이용해 곱셈을 하는 방식이다. 쉬프트 연산을 이용한 곱셈 예는 다음과 같다.

$$\begin{aligned} 5\times 5 &= 101_2 \times 101_2 \\ &= 101_2 \times 100_2 + 101_2 \times 001_2 \\ &= 101_2 \ll 2 + 101 \ll 0 \end{aligned}$$

유한체 상에서 곱셈의 최종 결과는 이 쉬프트 연산 결과를 기약다항식으로 나눈 나머지로 (쉬프트 곱셈 연산의 값) mod (기약다항식)를 계산한 결과이다. 예를 들면, 기약다항식 $x^8+x^4+x^3+x+1$을 갖는 GF(256) 상에서 38×19의 곱셈 결과는 두 수의 쉬프트 연산결과인 1011010010_2에 대해 기약다항식으로 나눈 나머지 값으로 $(1011010010)_2 \bmod (100011011)_2$ $= 10011100_2 = 156$이 된다.

비트 연산 조합의 비트 병렬을 이용한 곱셈은 FFT (Fast Fourier Transform) 기반 비트 연산을 여러 개의 CPU 또는 계산 전용칩에 적용한 것으로 하드웨어 기반 곱셈 병렬화 방식이다. $a=(a_{L-1},\ldots,a_0)$, $b=(b_{L-1},\ldots,b_0)$에 대한 곱셈은 다음과 같은 과정을 통하여 $O(L\log L)$ 시간에 곱셈을 한다.

$$w = e^{(2\pi i/L)}$$

$\tilde{a_i} = \sum^{L-1_0} w^{is} a_i$ 일 때, $\tilde{c_i} \equiv \tilde{a_i}\tilde{b_i}$라 하면,

$$\begin{aligned} c_r &= \frac{1}{L}\sum^{L-1_{i=0}} \tilde{c_r} w^{-ri} = \frac{1}{L}\sum^{L-1_{i=0}} \tilde{a_r}\tilde{b_r} w^{-ri} \\ &= \frac{1}{L}\sum_{i,j,k} w^{-ri} a_j w^{rj} b_k w^{rk} = \sum_{j,k} \delta_{r,j+k} a_j b_k \\ &= \sum_{j+k\equiv r \bmod L} a_j b_k = \sum_{j=0}^{r-1} a_j b_{r-j} \end{aligned}$$

가 됨을 이용하여 $O(L\log L)$ 시간에 곱셈을 한다.

하위 필드를 이용한 곱셈은 주어진 필드보다 더 큰 하위 필드로 재표현이 가능한 GF의 속성을 이용한 곱셈이다. 즉, GF(2^8)상의 2차 prime polynomial로 나머지 연산 mod가 적용된 GF(2^4)의 값들은 GF(2^8)로 재표현이 가능하다. 그러면, GF(2^8) 상의 곱셈은 하위 필드 GF(2^4) 상의 연산 사용이 가능하다. 예를 들면, GF(2^8) 상의 α에 대하여 프라임 이차식 $x^2+\alpha x+1$일 때, $a=a_0+a_1x$와 $b=b_0+b_1x$의 곱셈 $a\times b$는 다음과 같이 정리된다.

$$\begin{aligned} a\times b &= (a_0+a_1x)(b_0+b_1x) \\ &= (a_0b_0+a_1b_1)+(a_0b_1+a_1b_0+\alpha a_1b_1)x \end{aligned}$$

여기에서 $(a_0,a_1)\times(b_0,b_1)$에 대해 새로 구성된 계수 결합 (y_0,y_1)은 다음과 같다.

$$\begin{aligned} y_0 &= a_0b_0+a_1b_1 \\ y_1 &= a_0b_1+a_1b_0+\alpha a_1b_1 \end{aligned}$$

이를 회로로 살펴보면, Fig. 2와 같이 4개의 곱셈기와 1개의 스칼라, 3개의 덧셈기로 구성되어, 적용되는 게이트 수는 증가했으나 연산 수행 시간은 좀 더 줄인다. 이렇게 GF(2^{2m})상의 곱셈은 GF(2^m)상의 연산을 사용 가능하므로 GF(2^8) 곱셈은 GF(2^4) 연산으로, 다시 GF(2^4) 곱셈은 GF(2^2) 연산을 사용하여 좀 더 빠르게 GF(2^8) 상의 곱셈을 수행한다[10].

로그 및 안티로그 테이블을 이용한 곱셈은 로그 연산 및 역수 속성을 이용한 곱셈이다. 즉, $2^j=k$에

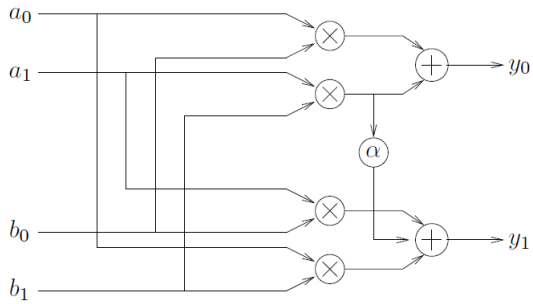


Fig. 2. The multiplier circuit of the sub field re-representation method of GF(2^{2m})

Table 1. Log and anti-log table for multiplication of GF(2^8)

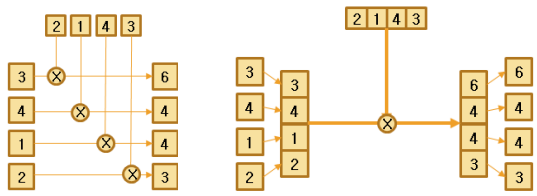
i	0	1	2	3	4	5	6	7
$\log_2(i)$	-	0	1	3	2	6	4	5
$\log_2^{-1}(i)$	1	2	4	3	6	7	5	-

대해서 $\log_2(k)=j$로 표현 할 수 있고, 이를 다시 역으로 나타내면 $\log_2^{-1}(j)=k$로 표현이 가능하다. 이러한 관계를 이용하여 모든 발생 가능한 로그 및 로그 역수 값을 미리 계산해놓은 테이블을 구성하고, 이 표를 이용하여 좀 더 효율적인 곱셈을 수행 할 수 있다. 예를 들어, GF(2^8)에 대한 로그 테이블은 [표 1]와 같으며, 이 표를 이용한 곱셈$a\times b$는 $ab=\log_2^{-1}(\log_2(a)+\log_2(b))$ 로 계산한다[11].

라지 룩업 테이블을 이용한 곱셈은 GF(2^8)상의 값의 범위인 0~255 값에 대해 모든 곱셈 경우를 계산해 표로 저장해놓고 곱셈 시 별도의 계산 없이 표에 접근하는 것만으로 곱셈 결과를 획득한다. 다만 그만큼 메모리를 많이 소비하게 된다. GF(2^8)의 경우, 프로그램 실행 시 메모리를 256×256바이트만큼 차지하지만, 곱셈을 위한 별도의 시간은 필요하지 않고 단순히 해당 곱셈 결과의 위치를 찾아가는 메모리 접근 시간만 소비한다. 이에, 연산 속도보다 메모리 접근 속도가 현저히 느린 경우에는 전체 서명 기법의 수행이 오히려 라지 룩업 테이블 사용의 경우보다 어느 정도의 연산을 수반하는 로그/안티로그 테이블 기반 수행이 더 빠를 수 있다.

● SIMD 명령어 기반 병렬 연산

SIMD 기반의 연산은 Fig. 3과 같이 하나의 변수에 여러 개의 데이터를 배정하고, 한 번의 연산 명령으로 해당 데이터들의 연산이 모두 수행이 되는 방식이다. 즉, 이 방식의 기존 방식과의 차이는 한 번의 CPU 클럭 동작으로 하나의 결과와 여러 개의 결과가 나오는 차이를 의미한다. SIMD 기반에서 사용되는 변수는 레지스터를 사용하며, 이 레지스터는 CPU의 발전에 따라 그 크기가 64 bits -> 128 bits -> 256 bits -> 512 bits로 확대되었고, 이와 관련한 벡터 확장 명령 집합을 MMX, SSE, AVX 2, AVX3(512) 등으로 제조사에서 제공하고 있다. 현재 주로 사용 가능한 명령 집합은 SSE와 AVX2 이다. 256 bits 크기의 레지스터 YMM의 개수는 32비트 시스템에서는 총 16개로 YMM0~YMM15을 사용할 수 있으며, 64비트 시스템에서는 YMM16~YMM31로 32개가 증가되며, 한번의 명령으로 처리될 수 있는 데이터의 양이 많아져 프로그램 구현 시 최대한 사용할수록 효율이 높아진다[12].



(a) SISD Op. (b) SIMD Op.

Fig. 3. SISD operation & SIMD operation

다변수 이차 다항식을 프로그래밍 상 일반적인 SISD(Single Instruction Single Data) 곱셈연산과 SIMD 기반 병렬 연산을 비교해보면, 변수가 n개인 이차 다항식 m개에 대한 곱셈연산을 위해서는 $n(n+1)/2\times m$개의 연산을 해야 하지만 SIMD 연산을 진행하면 $n(n+1)/2\times m/32$의 연산만으로 완료할 수 있다.

● 다항식의 계수에 대한 재배치를 통한 최적화

다항식 연산 운용 알고리즘에 따라 다항식의 계수 배치는 재정렬 되어야 한다. 특히, SIMD 병렬연산을 적용했을 때 1차원 배열을 사용함에 있어 배열 원소의 구성 및 저장 순서는 특정 원소가 있는 메모리 위치를 찾아가는 상대주소 연산과 해당 배열을 이용하는 전체 연산 횟수를 감소하여 결과적으로 프로그램의 실행시간을 줄여준다.

Fig. 4는 변수가 8개인 다항식 4개로 이루어진 키 배치 별 구조의 형태를 나타낸 것이다. 이를 변수 n개인 다항식 m개로 일반화하고 8바이트 병렬연산을 진행하게 된다고 가정하고 이 문제를 풀게 되면 사용 되는 연산은 덧셈과 곱셈을 동시에 연산(곱셈*덧셈 연산)하는 것과 덧셈 연산 두 가지를 이용한다.

이 경우 Fig. 4-(a)는 2차 다항식 당 곱셈*덧셈 연산 n회, 덧셈연산 $n-1$회로 전체 다항식을 풀기 위해선 $((n)+(n-1))\times m$회의 연산을 해야 한다. Fig. 4-(b)는 곱셈*덧셈연산 $n/2+1$회 덧셈연산 $n-1$회로 전체 $((n/2+1)+(n-1))\times m$회의 연산을 해야 한다. 마지막으로 Fig. 4-(c)는 각각의 다항식 별로 같은 변수들이 갖는 계수들을 배치한 것이

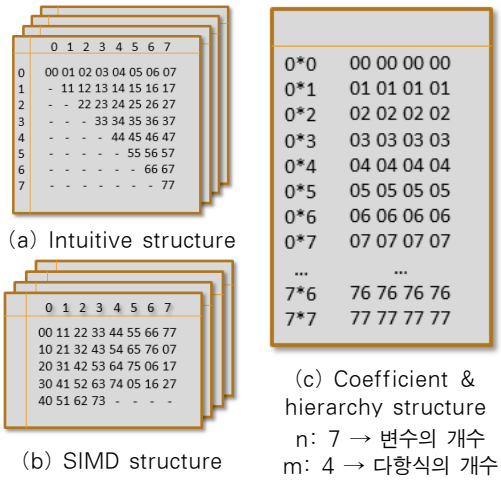


Fig. 4. Coefficient of MQ(Multivariate Quadra-tic) polynomial(key) arrangement reordering method

다. 이 구조는 덧셈 연산이 필요하지 않으며 곱셈*덧셈 연산으로 $n(n+1)/2$회의 연산을 하게 되면 전체 문제가 풀리게 된다. 이 때문에 연산의 횟수가 가장 적은 Fig. 4-(c)가 가장 효율적이다[13].

III. 8비트 마이크로컨트롤러 유닛(MCU)의 Rainbow 구현을 위한 최적화 방안

2012년에 Peter Czypek 등은 UOV, Rainbow, enTTS와 같은 다변수 이차식 기반 서명을 8비트 MCU(MicroController Unit)에 최적화 구현하였다[14]. Peter Czypek는 Rainbow에 특수한 형태의 아핀 맵 S를 구성하는 등가 키(Equivalent key) 방식[15]을 적용해 공개키의 크기를 12 KB ~ 136 KB로 줄여 ATxmega128a1에 구현했다. 그러나 해당 연구에서 사용한 파라미터들은 NIST에서 요구한 보안 수준을 충족시키지 못하여 NIST 공모에 제출된 Rainbow[16]에는 포함되지 않았다.

Table 2. 8-bit microcontroller resource

8bit MCU	Program Memory (Bytes)	Data Memory (Bytes)	EEPROM (Bytes)
ATmega328p	32,768	2,048	1,024
ATmega128	131,072	4,096	4,096
ATxmega128a1	139,264	8,192	2,048
ATxmega256a3	270,336	16,384	4,096

현재 NIST Rainbow에는 키 길이를 줄이는 기술이 적용되지 않았으며, 공개키의 길이가 148 KB ~ 1,321 KB이다. 이 크기의 키는 Table 2에 보이는 것과 같이, [14]에서 사용한 ATxmega128a1에는 구현할 수 없다. 따라서 본 장에서는 NIST의 양자내성암호 공모전에 제출된 Rainbow 중 파라미터(GF(256), 40, 24, 24) 버전을 좀 더 큰 메모리를 갖고 있는 ATxmega256a3bu에 구현하기 위한 알고리즘 개선에 대해서 다룬다.

3.1 8비트 MCU 고려사항

Rainbow 서명은 키 길이의 크기가 크고 연산의 양이 많기 때문에 초당 연산 횟수가 낮은 프로세서를 가진 기기에서는 키를 저장할 수 없거나 서명 생성이나 검증의 속도가 상대적으로 느리다. 이 때문에 서명 및 검증 알고리즘과 키의 구성 그대로 8비트 마이크로컨트롤러를 이용하는 MCU에 적용할 수 없다. 그래서 본 최적화 구현에서는 서명 생성 및 검증 알고리즘의 메모리 사용 효율성 및 수행 속도를 향상시킬 방안 모색에 초점을 맞추었다. 따라서 키를 줄이는 기법이 아닌 나누어서 저장하는 기법을 구현하고 하위필드 연산을 이용한 곱셈 연산과 곱셈 테이블을 이용한 곱셈 연산을 사용하였다.

Rainbow 파라미터(GF(256), 40, 24, 24)를 기준으로 공개키가 약 187.7 KB, 비밀키가 약 140 KB로, 8비트 마이크로컨트롤러의 데이터 메모리(SRAM)에 입력 혹은 저장이 불가능하기 때문에 프로그램 메모리(flash memory)에 저장해야 한다. 그렇게 되면 대표적으로 아두이노(arduino)에 쓰이는 ATmega328p나 ATmega128에는 키 생성 과정에

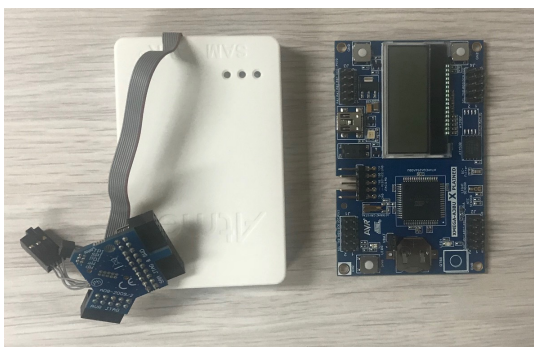


Fig. 5. In-circuit debugger; Atmel Ice & XMEGA-A3BU Xplained

서 길이를 줄이지 않는 한 구현이 불가능하고, 상대적으로 강한 연산 능력을 가진 XMEGA시리즈 중에서도 ATxmega256 이상이 되어야 비밀키나 공개키 저장이 가능한 관계로 Table 2에서 확인할 수 있는 ATxmega256a3bu 마이크로컨트롤러를 사용한 XMEGA-A3BU XPLD를 사용하였다. ATxmega256A3은 무인기의 원격 전송 장치로 이용되는 등 IoT에 활발히 적용되는 8비트 마이크로컨트롤러 중 하나이다.

3.2 키 배치 최적화

Rainbow는 파라미터(GF(256), 40, 24, 24)를 기준으로 공개키가 약 187.7KB이며, 개인키는 약 140KB이다. 하지만 선택한 8비트 MCU의 데이터 메모리는 16KB이기 때문에 통상적으로 공개키와 개인키가 저장할 수 없다. 또한 8비트 MCU의 허용 배열 최대 크기가 32KB이기 때문에 하나의 변수에 키를 모두 포함시키기 어렵다. 따라서 다른 종류의 내장 메모리인 프로그램 메모리에 데이터에 대한 강제/명시적 접근과 키 분할 및 인덱싱을 이용하여 8비트 MCU에서 다변수 이차식 기반 서명을 고속 연산하기 위한 최적화를 진행하였다.

- **프로그램 메모리에 데이터에 대한 강제/명시적 접근**

8비트 MCU의 일반적인 세그먼트(segment)를 무시하고 재배치한다. 기본적으로 변수들이 할당 되는 데이터 메모리가 아닌 프로그램 메모리에 상수 주소 지정 방식을 이용하여 강제 할당하였다. 프로그램 메모리의 상수에 대한 특정 접근 명령은 8비트로 동작하고, 접근 가능한 영역이 한정돼 있으며, 각각의

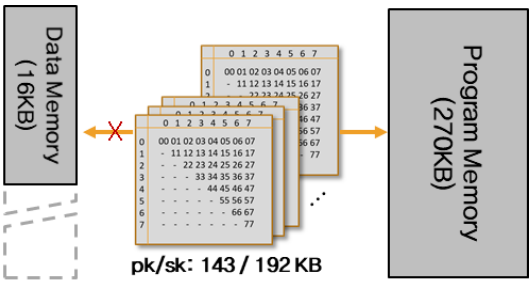


Fig. 6. Since the SRAM area, which is the data memory, is small, the key must be stored in the program memory.

주소(index)는 16비트로 구성되어 있다.

- **키 분할 및 인덱싱**

Fig. 7에서 표현한 것과 같이 공개키 혹은 개인키 전체를 배열 최대 크기보다 작은 여러 개의 배열 조각으로 분할해서 저장해야 하며, 서명 생성이나 검증과정에서 키들을 쉽게 사용할 수 있도록 분할된 배열 조각에 대한 주소를 할당하는 링커 구성 배열을 생성한다.

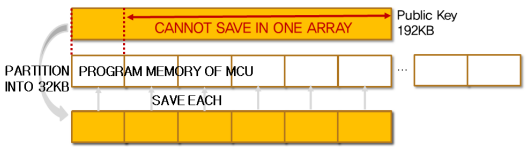


Fig. 7. Key partition and Indexing

배열을 나눌 때 중요한 점은 너무 작게 나눌 경우 주소를 저장하는 배열이 커져 데이터 메모리 소모가 커지고, 코드의 길이 또한 길어짐에 따라 프로그램 메모리 또한 초과될 수 있다. 따라서 성능 최대화를 크기와 개수를 검토해야 한다. 본 최적화를 진행할 때 우리는 공개키를 88×48의 선형함수 행렬 1개와 44×48의 이차함수 행렬 89개, 1×48의 상수 행렬 1개로 분할했고, 개인키는 88×88 그리고 48×48의 아핀 행렬(affine matrix) 각각 1개와 각각 88개와 48개의 벡터 1개씩 저장하는 것은 기존의 Rainbow와 같이 유지하고, 1계층(layer)의 중앙 행렬(central matrix)은 24×24의 오일(oil) 선형 행렬 1개, 40×24의 비네가(vinegar)-오일 이차항 행렬 24개, 40×24의 비네가 선형 행렬 1개 그리고 $410\times1\times24$의 비네가-비네가 이차항 행렬 2개와 1×24의 상수 행렬로 구성되었으며, 2계층은 24×24의 오일 선형 행렬 1개, 64×24의 비네가-오일 이차항 행렬 24개, 64×24의 비네가 선형 행렬 1개 그리고 $1040\times1\times24$의 비네가-비네가 이차항 행렬 2개와 1×24의 상수 행렬 1개로 이루어져 있다. 기술한 바와 같이 분할한 이유는 메모리 소모를 최대한 작게 하고, 서명 생성과 검증과정의 소스코드 구성의 편의성 때문이고 Rainbow의 파라미터와 기기의 용도에 따라 더 크게 혹은 작게 구성해도 무관하며, Fig. 8과 같이 프로그래밍 한다.

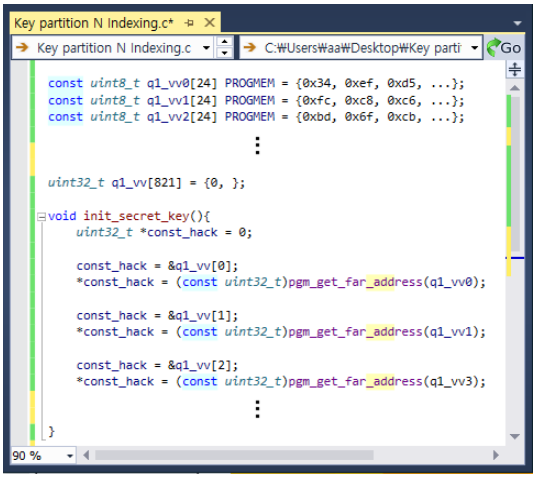


Fig. 8. Key partition & indexing C code

3.3 연산 모듈 최적화

8비트 MCU가 IoT로 사용될 때 낮은 연산과 빈번한 연산 요청으로 반복적인 서명 생성이나 검증을 해야 한다. 따라서 연산 고속화 방안이 필요한데 이는 기본 자료형을 활용한 병렬처리와 프로그램 구조 경량화, 저비용 연산 구성을 이용하여 해결했다.

● **기본 자료형을 활용한 병렬처리**

8비트 MCU는 SIMD가 지원되지 않기 때문에 병렬연산을 수행하기 위해서는 기본 자료형 중 가장 크기가 큰 자료형 long을 이용하여 Fig. 9와 같이 구현한다. 기본 자료형을 활용한 병렬 연산은 1 Byte의 unsigned char의 포인터로 4 Bytes의 unsigned long에 접근하여 한번의 4개의 데이터를 쓰고 읽어서 비트 연산(Ex. 각 바이트별로 Fig. 2의 $GF(2^8)$ 곱셈연산)을 수행한다. 본 연구에서는 이러한 방식을 이용하여 AVX2(32 바이트 크기의 자료형)기반 SIMD를 대체하는 또 다른 형태의 SIMD를 구현해 병렬연산을 수행한다. 또한 다변수 이차식 구조에 따른 동일 연산에 대한 반복 수행을 고려하여 메모리에 한번 불러들인 데이터를 최대한 사용할 수 있도록 데이터 로딩의 순서를 전략적으로 구성하였다. 메모리에 적재되는 데이터의 순서는 연산 한번에 사용되는 데이터의 수 및 메모리의 크기(1바이트, 4바이트, 32바이트 크기의 자료형 등)에 따라 다양하게 구성될 수 있다.

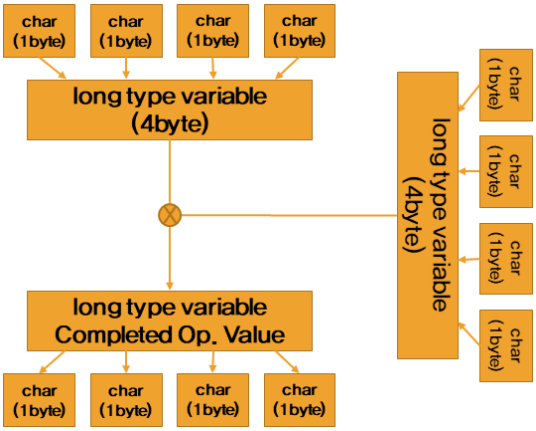


Fig. 9. Parallel operation using unsigned long type variable

● **프로그램 구조 경량화**

이차식 변수들을 서명으로 사용하는 다변수 이차식 기반 서명 특성 상 프로그래밍에 중첩된 반복문(루프)이 많이 사용된다. 이러한 루프는 스택 내에 세그먼트 구성에 소요되는 명령어의 사용과 시간이 늘어나 서명 생성과 검증에 많은 연산을 하는 다변수 이차식 기반 서명 특성 상 연산 속도에 큰 영향을 미친다. 그래서 반복문의 중첩 깊이가 깊지 않도록 배열로 저장되어 있는 키나 서명 값들의 탐색방식 및 순서를 재구성하고 재구성된 탐색순서에 적합한 반복문을 적용하였다. 수정 적용된 반복문은 3 또는 4중첩의 깊이를 갖는 루프에서 2 또는 3중첩의 루프를 갖는 구조로 경량화 되었다. 하지만 이러한 루프의 해제를 통한 루프 중첩의 완화는 모든 반복문 부분에서 그 효과를 보이지 않고 루프 블록의 크기가 커져 캐시(instruction cache)에 적합하지 않은 경우도 발생했다. 이러한 경우에는 캐시를 사용하지 못해 성능이 떨어지므로, 해당 부분의 루프 해제에 대한 결정은 실제로 프로그램을 실행하여 측정된 수행시간을 비교 대조해 루프 해제에 의한 연산 비용이 줄어드는 경우에만 이뤄진다[13].

● **저비용 연산 구성**

기존의 Rainbow의 곱셈 연산은 2.2절에서 설명한 GF상의 사칙연산 최적화 방식 중 하위 필드 연산을 이용해 GF(256) 상의 곱셈을 GF(16)으로 GF(16) 상의 연산을 GF(4)으로 재표현하여 연산을 최적화하는 방식을 이용했다. 따라서 이 방식을 ATxmega256a3bu 보드에 맞게 재구성 했으며, 또한 2.2절에서 언급한 또다른 최적화 방식인 라지 룩업

테이블과 로그 및 안티로그 테이블을 이용한 연산으로 재구성하여 3가지 버전으로 구축한다.

하위필드를 이용한 곱셈 연산 최적화와 로그 및 안티로그 테이블을 이용한 곱셈 연산 최적화는 통상적으로 적용이 가능하나, 라지 룩업 테이블을 이용한 곱셈 연산 최적화를 적용한 경우에는 3.1장에서 기술한 바와 같이 배열의 최대크기 문제로 256×256의 테이블은 하나의 배열 변수 선언으로 이용할 수 없다. 그래서 키 분할 및 인덱싱을 이용하여 동일한 분할 반식을 이용하여 저장해야 하며, 본 최적화를 적용할 때 우리는 100단위로 잘라서 0~99 와 100~199, 200~255와 곱하는 테이블 각각 3개로 총 9개의 테이블을 분할했다.

IV. 최적화 후 동작 확인 및 비교

본 장에서는 서명 생성과 검증을 각각 프로그래밍을 해 MCU에 이식한 결과를 보인다. 서명 생성의 경우 Fig. 10-(a)와 같이 왼쪽 LED가 점등하며 서명 생성이 완료되면 Fig. 10-(b)와 같이 양쪽의 LED가 모두 점등한다. 검증은 Fig. 10-(c)와 같이 오른쪽 LED가 점등하며 검증이 완료되면 Fig. 10-(d)와 같이 양쪽의 LED가 점등을 한다.

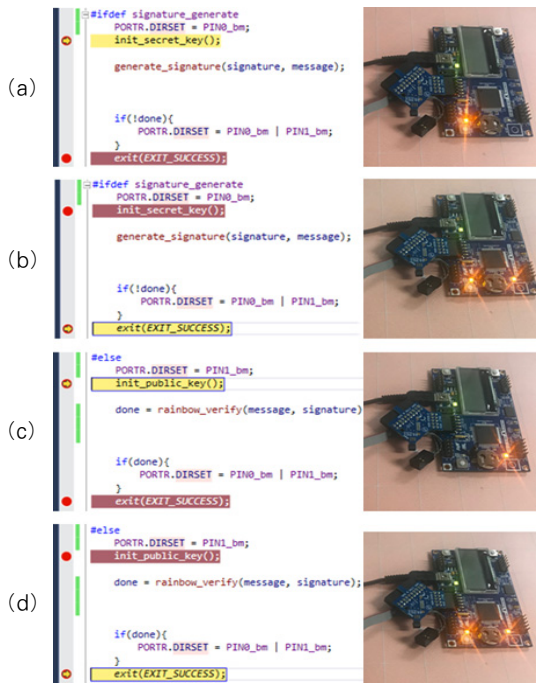


Fig. 10. Application Example

Table 3. Comparison of instruction cycle & resource consumption of algorithms with multiplication using log and anti-log table and large lookup table

Mult Method	Program memory Usage(Bytes)		Instruction cycle (Cycle ×1000)	
	Sign	Verify	Sign	Verify
Subfield	161,664 (59.8%)	198,704 (73.5%)	38,724	71,138
Large Lookup Table	221,172 (81.8%)	263,452 (97.5%)	14,395	11,678
Log & Antilog Table	148,962 (55.1%)	196,942 (72.9%)	9,897	5,533

다음의 Table 3을 보면, 로그 및 안티로그 테이블 기반 곱셈을 사용한 알고리즘이 메모리 소모율과 서명 생성과 검증의 명령 주기(instruction cycle)가 가장 적다. 이에 따라 메모리 소모 및 곱셈 속도를 고려하여, ATxmega256a3bu 보드에 구현하기에 적합한 곱셈 최적화 방식은 로그 및 안티로그 테이블을 이용한 방식이다. Table 3에 의하면 서명 생성과 검증의 속도가 큰 차이를 보이는데, 이는 Fig. 1에서 보인 다변수 이차식 기반 서명의 생성과 검증과정이 다른 구조로 이루어지기 때문이다. 서명 생성 과정에서는 다항식에 대한 변수의 대입을 위한 곱셈 외에 가우시안 소거법이나, 아핀행렬 연산 등 상대적으로 적은 곱셈이 필요한 연산을 더 많이 사용하는 반면 서명 검증 과정은 다항식에 대한 변수의 대입을 위해 단순한 곱셈연산을 많이 진행하는 구조로 이루어져 있다. 따라서 하위필드를 이용한 연산을 적용하면 서명생성 과정에서, 테이블을 이용한 연산을 적용하면 서명검증 과정에서 적은 명령 주기를 갖는다.

V. 결 론

본 논문은 IoT의 확대 및 양자 컴퓨터의 개발로 IoT 보안 솔루션에 적용된 공개키 암호를 대체할 수 있는 양자 내성 암호 구현이 필요하여 NIST 양자 내성 암호 공모전에 제출된 다변수 이차식 기반 서명 기법 중 Rainbow를 8비트 MCU 상에 최적화 할 수 있는 방안에 대해 다루고 구현하였다.

최적화 대상은 8비트 MCU인 XMEGA-A3BU XPLD이며, 메모리 명시적 할당 및 전체 연산비용 감량을 이용해 최적화 하였다. 중첩, 반복 등의 완화하거나 해제하는 기본적인 임베디드 프로그래밍 최적화 기법을 적용하였으며, 프로세서 구조별 메모리 운용방식, 메모리 접근 속도 등의 상이함을 고려해 저비용의 곱셈방식으로 구현해 각각 비교해보았으며, SIMD 미지원 기기에서의 병렬화는 기본 자료형 중 가장 작은 자료형의 포인터와 가장 큰 자료형을 활용해 병렬화 하였다.

또한, 곱셈 최적화 구현을 통한 실험결과는 라지 룩업 테이블을 이용한 것이 로그 및 안티로그 테이블을 이용한 것보다 메모리 소모는 크나 연산 속도는 빠를 것이라 예상했지만, 특정 프로세서에서 로그 및 안티로그 테이블을 사용한 연산이 더 빠른 것으로 확인하였다. 이러한 실험결과는 프로세서의 종류에 따라 로그 및 안티로그 테이블을 사용하여 연산을 적용하는 것이 성능과 메모리 소모 측면에서 효율적이다.

따라서 서명 생성과 검증에 사용되는 연산을 최적화하는 기법은 현재까지 상용화된 IoT 기기 등에 응용하여 더 높은 편리성, 효율성, 보안성을 제공할 것으로 기대되며, 향후 더 다양한 기기에 구현해 고찰해 볼 필요성이 있다. 또한, 구현 암호 알고리즘 개선, 알고리즘 내에서 다른 방식의 최적화를 고안 및 적용 그리고 엣지컴퓨팅 등의 기법을 이용한 최적화 방안 또한 강구하여 최종적으로 IoT 환경에 적합한 양자내성 암호 알고리즘 기반 보안/인증 프로토콜을 설계해야 할 필요성이 있다.

References

[1] Google AI Blog, "A Preview of Bristlecone," https://ai.googleblog.com/2018/03/a-preview-of-bristlecone-googles-new.html Mar. 2018

[2] Sandeep Kumar Rao, Dindayal mahto, Dilip Kumar Tadav, and Danish Ali Khan "The AES-256 Cryptosystem Resists Quantum Attacks," proc. of Internatinal Journal of Advanced Research in Computer Science, vol. 8, no. 3, pp. 404-408 March.-April 2017

[3] Kyung-Ah Sim, Cheol-Min Park, and Namhun Koo, "An Existential Unforgeable Signature Scheme Based on Multivariate Quadratic Equations," proc. of Advances in Cryptology - ASIACRYPT 2017. ASIACRYPT 2017, LNCS, vol. 10624, pp. 37-64, Nov. 2017

[4] Tae-hwan Park, Hwa-jeong Seo, Ga-ram Lee, and Ho-won Kim, "Post Quantum Cryptography Latest Software Implementation Trends," Journal of Korean Institute of Information Security and Cryptology, 27(6), pp. 21-28, Dec. 2017

[5] ECRYPT "Multivariatee Quadratic Public-Key Cryptography Part 2: Big Field Schemes," https://2017.pqcrypto.org Jun. 2017

[6] Peter Czypek, "Implementing Multivariate Quadratic Public Key Signature Schemes on Embedded Devices," Diss. Ph. D. thesis, Diploma Thesis, Chair for Embedded Security, RUB, 2012

[7] Aviad Kipnis, Jacques Patarin and Louis Goubin, "Unbalanced Oil and Vinegar schemes," Advanced in Cryptology - EUROCRYPT'99, LNCS, vol. 1592, pp. 206-222, 1999

[8] Jintai Ding and Dieter Schmidt, "Rainbow, a New Multivariate polynomial Signature scheme," Applied Cryptography and Network Security, LNCS, vol. 3531, pp. 164-175, 2005

[9] Zhiniang Peng and Shaohua Tang, "Circulant Rainbow: A New Rainbow Variant with Shorter Private Key and Faster Signature Generation," IEEE Access vol. 5, pp. 11877-11886, Jun. 2017

[10] James S. Plank, Kevin M. Greenan, and Ethan L. Miller. "Screaming fast Galois field arithmetic using intel SIMD instructions." Conference: FAST: USENIX Conference on File and Storage Technologies, vol. 11, pp. 299-306 2013.

[11] James Westall, and James Martin, "A

n Introduction to Galois Fields and Reed-Solomon Coding," ISchool of Computing Clemson University Clemson, SC, 2010, 29634-1906.

〔12〕 David A. Patterson, and John L. Patterson, "Computer Organization and Design: the Hardware/Software Interface,", 2nd Ed. Morgan Kaufmann, pp. 751, 1998

〔13〕 Youjin Kim, "Optimization of Embedded Programming C code," Hanbit media, pp. 161~174, 215~225, 2008

〔14〕 Peter Czypek, Stefan Heyse, Enrico Thomae, "Efficient Implementations of MQPKS on Constrained Devices," in In International Workshop on Cryptographic Hardware and Embedded Systems, LNCS, vol. 7428, pp. 374-389, 2012

〔15〕 Albrecht Petzoldt, Enrico Thomae, et al, "Small Public Keys and Fast Verification for Multivariate Quadratic Public Key Systems," in In International Workshop on Cryptographic Hardware and Embedded Systems, LNCS, vol. 7428, pp. 475-490, 2011

〔16〕 Jintai Ding, Ming-Shing Chen, et al, "Rainbow. NIST submission," 2017

〈저자소개〉

홍 은 기 (Eungi Hong) 학생회원
2018년 2월: 한양대학교 ERICA 전자시스템공학과 공학사
2018년 3월: 한양대학교 전자공학과 석사과정
〈관심분야〉 IoT 보안, 임베디드 시스템 보안, 양자암호

조 성 민 (Seong-Min Cho) 학생회원
2019년 2월: 한양대학교 ERICA 전자공학부 공학사
2019년 3월: 한양대학교 전자공학과 석사과정
〈관심분야〉 IoT 보안, 임베디드 시스템 보안, 양자암호

김 애 영 (Aeyoung Kim) 정회원
2000년: 한신대학교 정보처리학과 이학사
2003년: 이화여자대학교 컴퓨터학과 공학석사
2012년: 이화여자대학교 컴퓨터공학과 공학박사
2012년~2015년: 이화여자대학교 컴퓨터공학과 연구교수
2016년~2018년: 국가수리과학연구소 암호기술연구팀 박사후연구원
2018년 5월~현재: 한양대학교 공학기술연구소 연구교수
〈관심분야〉 IoT 보안, 암호알고리즘 최적화 구현, 블록체인 보안, 인증, 생체정보 보안

서 승 현 (Seung-Hyun Seo) 정회원
2000년: 이화여자대학교 수학과 이학사
2002년: 이화여자대학교 컴퓨터학과 공학석사
2006년: 이화여자대학교 컴퓨터학과 공학박사
2006년 12월~2010년 1월: 금융보안 연구원 주임연구원
2010년 2월~2012년 2월: 한국인터넷 진흥원 선임연구원
2012년 2월~2014년 5월: 미국 퍼듀대학교 컴퓨터학과 박사후연구원
2014년 6월~2015년 2월: 고려대학교 정보보호대학원BK21+ 사업단 연구교수
2015년 3월~2017년 2월: 고려대학교 세종캠퍼스 수학과조교수
2017년 3월~현재: 한양대학교 ERICA 캠퍼스 전자공학과부교수
〈관심분야〉 IoT 보안, 블록체인 보안, 암호프로토콜 설계 및응용