석사학위논문

# ARMv8 상에서의 SNOVA 전자 서명 알고리즘 최적 구현

2024년 12월 13일


한성대학교 일반대학원

IT융합공학과

IT융합공학전공

이 민 우

석사학위논문
지도교수 서화정

# ARMv8 상에서의 SNOVA 전자 서명 알고리즘 최적 구현

Optimal Implementation of SNOVA on ARMv8 Architecture

2023년 12월 일


한성대학교 일반대학원

IT융합공학과

IT융합공학

이 민 우

석사학위논문
지도교수 서화정

# ARMv8 상에서의 SNOVA 전자 서명 알고리즘 최적 구현

Optimal Implementation of SNOVA on ARMv8 Architecture

위 논문을 공학 석사학위 논문으로 제출함

2023년 12월 일


한성대학교 일반대학원

IT융합공학과

IT융합공학

이 민 우

이민우의 공학 석사학위 논문을 인준함

2023년 12월 일

심 사 위 원 장 _____(인)

심 사 위 원 _____(인)

심 사 위 원 _____(인)

# 국 문 초 록

## ARMv8 상에서의 SNOVA 전자서명 알고리즘 최적 구현

한 성 대 학 교 일 반 대 학 원
I T 융 합 공 학 과
I T 융 합 공 학 전 공
이 민 우

본 논문은 ARMv8 아키텍처 상에서 SNOVA 전자서명 알고리즘의 최적 구현을 다룬다. SNOVA는 양자 내성 암호 시스템으로, 다변수 이차 방정식 기반의 높은 보안성을 제공한다. 본 연구에서는 SNOVA의 키 생성, 서명 생성, 서명 검증 작업을 최적화하기 위해 어셈블리와 NEON SIMD 명령어를 활용하였다. 주요 최적화는 유한체 연산을 위한 덧셈 및 곱셈 모듈과 AES-128 CTR 모드 가속기 구현을 포함한다. 덧셈과 곱셈 연산은 ARMv8의 벡터 레지스터를 활용하여 병렬 처리가 가능하도록 설계되었으며, AES 가속기는 하드웨어 명령어를 사용하여 암호화 속도를 극대화하였다. 성능 평가 결과, SNOVA의 주요 파라미터 세트(esk 및 ssk)를 대상으로 키 생성은 최대 68.7%, 서명 생성은 36.7%, 서명 검증은 50.0%의 성능 향상을 달성하였다.

# 목 차


**I. 서 론** ····· 1

**II. 관련 연구** ····· 3

2.1 SNVOA 알고리즘 ····· 3
2.2 ARMv8 아키텍처 ····· 5
2.3 ARM Neon SIMD ····· 7
2.4 ARMv8 상에서의 최적 구현에 대한 이전 연구 ····· 8

**III. SNOVA 알고리즘 최적 구현 기법** ····· 10

3.1 덧셈기 최적화 ····· 10
3.2 곱셈기 최적화 ····· 11
3.3 AES 가속기 ····· 13

**IV. 성능 평가** ····· 16

4.1 성능 측정 환경 ····· 16
4.2 성능 측정 결과 ····· 16

**V. 결 론** ····· 21

**참 고 문 헌** ····· 22

**ABSTRACT** ····· 25

# 표 목 차

[표 2-1] SNOVA 파라미터 ····· 3
[표 4-1] esk 파라미터 최적 구현 성능 측정 결과 ····· 18
[표 4-1] ssk 파라미터 최적 구현 성능 측정 결과 ····· 19

# 알 고 리 즘 목 차

[알고리즘 3-1] SNOVA 덧셈기 최적 구현 ····· 10
[알고리즘 3-2] SNOVA 곱셈기 최적 구현 ····· 12
[알고리즘 3-3] AES-128 CTR모드 가속기 구현 ····· 14

# 수 식 목 차

[수식 2-1] 다변수 이차 다항식 ····· 3
[수식 2-2] 비밀키 구성 요소 ····· 4
[수식 2-3] 공개키 생성 ····· 4
[수식 3-1] 행렬 곱셈 ····· 12

# Ⅰ. 서 론

양자 컴퓨팅의 발전으로 인해, RSA와 ECC와 같은 기존의 암호 시스템은 점차 양자 공격에 취약해지고 있다. 특히 Shor 알고리즘은 정수 인수분해와 이산 로그 문제를 다항 시간 내에 해결할 수 있어 이러한 시스템에 큰 위협을 가하고 있다. 이에 따라 양자 컴퓨팅 위협에 대응할 수 있는 양자 내성 암호(PQC) 연구가 활발히 진행되고 있다. 여러 PQC 접근 방식 중에서 다변수 공개키 암호시스템(MPKC)은 다른 양자 이후 암호화 방식에 비해 상대적으로 작은 키 크기와 빠른 속도를 제공할 가능성 덕분에 주목받고 있다.

SNOVA 서명 체계는 다변수 기반 전자서명 알고리즘으로, NIST PQC 추가 전자서명 공모전에서 유망한 후보로 제시되고 있다. 그러나 SNOVA를 포함한 대부분의 PQC 알고리즘은 리소스가 제한된 환경, 예를 들어 사물인터넷(IoT) 장치, 임베디드 시스템, 모바일 플랫폼과 같은 환경에서의 효율적 구현이 요구된다. 이러한 플랫폼들은 제한된 전력과 계산 자원을 갖추고 있기 때문에, 암호화 연산의 빠른 처리 속도와 높은 계산 효율성을 동시에 만족해야 한다. 특히, ARM 기반 프로세서가 널리 사용되는 이러한 환경에서는 전력 효율성과 계산 성능 간의 균형을 맞추는 것이 필수적이다. 따라서 ARM 프로세서에 최적화된 암호 알고리즘 구현은 리소스가 제한된 환경에서의 실제 사용을 위해 필요한 과정이다.

특히 ARMv8-A 아키텍처는 64비트 AArch64 명령어 세트와 NEON 기술을 제공하여, 에너지 효율성 및 고성능 계산이 중요한 환경에서 널리 사용되고 있다. NEON은 ARM의 SIMD(단일 명령 다중 데이터) 확장 기능으로, 행렬 산술과 유한체 연산이 중요한 알고리즘에서 병렬 데이터 처리를 가능하게 한다. NEON 기술을 활용하면 키 생성, 서명, 서명 검증과 같은 주요 암호화 작업에서 성능을 크게 향상시킬 수 있다.

현재 다변수 공개키 암호시스템(MPKC)의 ARMv8 최적화 연구는 상대적

으로 부족한 상황이다. 기존 연구들은 주로 격자 기반 또는 코드 기반 방식의 최적화에 중점을 두고 있다. 따라서 본 연구는 이러한 공백을 채우기 위해 NEON 기술을 활용한 SNOVA 서명 체계의 최적 구현을 제시하고자 한다.

본 연구의 주된 목표는 어셈블리 레벨의 최적화를 통해 ARMv8에서 SNOVA의 암호화 작업을 효율적으로 구현하고, 실행 시간을 단축할 수 있음을 입증하는 것이다. 본 논문에서는 NIST PQC 프로젝트의 레퍼런스 코드와 비교하여 최적화된 구현의 성능을 측정하고 성능 향상도를 비교한다.

# II. 관련 연구

## 2.1 SNOVA 알고리즘

SNOVA(Simple Noncommutative Oil and Vinegar)는 비가환 행렬을 활용하는 다변수 공개키 암호 시스템(MPKC)으로, 다변수 이차 방정식(MQ 문제)의 수학적 복잡성을 기반으로 보안성을 제공한다. 이 알고리즘은 기존 Unbalanced Oil and Vinegar(UOV) 서명 체계의 확장된 형태로, 비가환 행렬을 도입함으로써 대수적 공격과 구조적 공격에 대한 내성을 향상시키는 것을 목표로 한다. SNOVA는 주로 메시지 서명과 검증 작업에 사용되며, 다항식 해를 찾는 수학적 문제를 바탕으로 보안을 확보한다. SNOVA 알고리즘은 유한체 GF(16) 상의 비가환 행렬을 주요 연산 구조로 사용한다. 이는 기존의 유한체 산술과 달리 행렬 곱셈이 비가환적(non-commutative)이기 때문에 대수적 구조 분석이 어렵게 만든다. 다항식의 일반적 형태는 다음과 같이 표현된다.

$$P_{i(X)} = \Sigma_{j=1}^{n} \Sigma_{k=1}^{n} a_{ijk} x_j x_k + \Sigma_{j=1}^{n} b_{ij} x_j + c_i$$

수식 2-1 다변수 이차 다항식

이러한 다항식을 조합하여 개인키와 공개키를 생성하고, 메시지 서명 시 해시 값을 기반으로 다항식의 해를 계산한다. SNOVA는 키 생성, 서명 생성, 서명 검증의 세 단계로 구성된다. 키 생성 과정에서는 난수 행렬을 기반으로 비밀키와 공개키가 생성되며, 주요 비밀키 구성 요소는 다음과 같다.

$$SK = (A_\alpha, B_\alpha, Q_{\alpha 1}, Q_{\alpha 2}, T_{12})$$

수식 2-2 비밀키 구성 요소

이 행렬들은 유한체 상에서 난수를 통해 생성되며, 행렬 곱과 합 연산을 통해 비밀키 연산이 이루어진다. 공개키는 비밀키 요소들을 조합하여 만들어지며, 다음과 같이 나타낼 수 있다.

$$PK = (P_{11}, P_{12}, P_{21}, P_{22})$$

수식 2-3 공개키 생성

SNOVA의 성능은 행렬 연산의 효율성에 의해 결정된다. 주요 연산은 유한체에서 행렬 곱과 덧셈으로 구성되며, 이는 하드웨어 가속을 통해 더욱 효율적으로 수행될 수 있다. 공개키 크기와 서명 크기는 기존 UOV 서명 체계 대비 상당히 작은 편이며, 양자 컴퓨터에 대한 보안성을 확보할 수 있도록 설계되었다. 결론적으로 SNOVA는 다변수 이차 다항식과 비가환 행렬을 활용하여 대수적 공격과 구조적 공격에 대한 내성을 높이고, 효율적인 서명 및 검증 연산을 제공하는 전자서명 알고리즘이다. 이를 통해 기존 MPKC 기반 시스템의 공개키 크기와 연산 복잡도를 최적화하면서도 강력한 보안성을 유지한다.

알고리즘의 보안 수준과 성능은 몇 가지 주요 파라미터를 조정함으로써 달성할 수 있다. 이러한 파라미터는 Vinegar 변수 수(v), Oil 변수 수(o), 유한체 크기(q), 방정식 차수(d) 등으로 구성되며, 각각은 알고리즘의 서명 생성 및 검증 과정에서 중요한 역할을 수행한다. Vinegar 변수는 서명 생성 과정에서 무작위적으로 선택되며, 다변수 방정식 시스템의 비선형성을 보장한다. Vinegar 변수 수(v)가 증가하면 무작위성이 증가하여 보안성이 강화되지만, 계산 복잡도 또한 증가한다. Oil 변수는 공개 방정식을 만족해야 하는 변수로, 검증 과정에서 주요 역할을 수행한다. Oil 변수 수(o)가 많아지면 서명 검증 단계에서 계산량이 증가하지만, 알고리즘의 강도 역시 강화된다. 유한체 크기(q)는 유한체 $F(q)$ 에서 연산이 수행되므로, 보통 16과 같은 소규모 값으로 설정된다. 유한체 크기는 연산의 복잡도와 직접적으로 관련되며, 큰 값으로 설정할수록 보안성이 높아지지만 계산 비용이 증가한다. 방정식

차수(d)는 다변수 방정식의 비선형 차수를 나타내며, 보통 2로 고정된다. 이는 이차 다항식을 사용하여 서명 생성 및 검증 알고리즘을 구성하는 데 핵심적인 요소이다. SNOVA의 파라미터는 보안성과 성능 간의 균형을 고려하여 조정된다. 높은 보안성을 위해서는 Vinegar 변수 수(v)와 Oil 변수 수(o)를 증가시키고 유한체 크기(q)를 크게 설정할 수 있다. 반대로, 계산 성능을 개선하기 위해서는 이러한 값을 줄이는 방식으로 최적화를 진행할 수 있다. 이처럼 SNOVA 알고리즘은 유연한 파라미터 설정을 통해 다양한 응용 환경에서 사용할 수 있으며, 양자 안전성을 목표로 하는 응용에서 특히 주목받고 있다. 다음은 SNOVA 알고리즘의 주요 파라미터 설정 예시를 나타낸 표이다.

[표 2-1] SNOVA 파라미터

| 보안 레벨 | Vinegar Variables(v) | Oil Variables(o) | Field Size(q) | Degree(1) |
|---|---|---|---|---|
| I | 24 | 5 | 16 | 4 |
| III | 43 | 25 | 16 | 2 |
| V | 60 | 10 | 16 | 4 |

## 2.2 ARMv8 아키텍처

ARMv8 아키텍처는 ARM Holdings에서 개발한 고성능 저전력 프로세서 구조로, 다양한 응용 분야에서 널리 사용되고 있다. 모바일 장치, 임베디드 시스템, 사물인터넷(IoT) 기기 등 전력 소비가 중요한 환경에서 높은 성능을 제공하기 위해 설계되었다. ARMv8은 이전 세대인 ARMv7과 비교하여 64비트 연산을 지원하는 AArch64 명령어 세트를 도입함으로써 계산 능력과 메모리 접근 범위를 확장하였다.

AArch64 명령어 세트는 31개의 범용 64비트 레지스터와 스택 포인터를 포함하며, 고정된 32비트 명령어 형식을 채택하여 디코딩 효율성을 향상시킨다. 이러한 설계는 대규모 데이터 처리 시 메모리 접근을 유연하게 관리할 수 있도록 지원한다. ARMv8 아키텍처는 64비트 프로세서로 작동하면서도 32비트 모드(AArch32)와의 호환성을 제공해 다양한 소프트웨어 생태계를 지원한다.

ARMv8의 주요 특징 중 하나는 병렬 데이터 처리를 위한 NEON SIMD(Single Instruction, Multiple Data) 기술이다. NEON은 128비트 벡터 레지스터를 사용해 다수의 데이터 요소를 동시에 연산할 수 있어, 특히 멀티미디어 처리, 신호 처리, 암호화 알고리즘에서 유리하다. NEON 연산은 정수 및 부동소수점 데이터를 모두 지원하며, 벡터 연산은 64비트 단위(D 레지스터) 또는 128비트 단위(Q 레지스터)로 수행된다.

암호화 연산 가속은 ARMv8 아키텍처의 또 다른 강점이다. AES, SHA-1, SHA-256과 같은 암호화 알고리즘을 위한 하드웨어 명령어 집합이 포함되어 있어 소프트웨어 기반 암호화 대비 성능이 크게 향상된다. 이는 데이터 보안이 중요한 응용 프로그램에서 전력 소모를 줄이고 처리 시간을 단축하는 데 기여한다. 하드웨어 지원을 통해 블록 암호화, 해시 연산과 같은 연산이 최적화된다.

보안 측면에서는 메모리 보호를 위한 가상 메모리 관리 유닛(MMU)을 갖추고 있어 운영 체제가 안전한 메모리 관리를 수행할 수 있다. TrustZone 기술은 보안 영역과 비보안 영역을 분리하여 신뢰할 수 있는 실행 환경을 제공한다. 이를 통해 보안이 중요한 금융, 의료, IoT 응용 프로그램 개발 시 보안 위협을 줄일 수 있다.

이러한 특성 덕분에 ARMv8 아키텍처는 암호화 알고리즘 연구에서도 주목받고 있다. 특히 행렬 연산, 벡터 연산 등 병렬 처리가 필요한 계산 집약적 응용 분야에서 NEON 명령어와 벡터 레지스터의 효율적인 활용이 가능하다. 이에 따라 양자 내성 암호 연구, 신호 처리, 머신러닝 등 다양한 계산 환경에서 ARMv8이 연구 및 개발 플랫폼으로 활용되고 있다.

## 2.3 ARMv8 NEON SIMD

ARMv8 아키텍처는 데이터 병렬 처리를 위한 도구로 NEON SIMD(Single Instruction Multiple Data) 명령어 집합을 제공한다. NEON은 다수의 데이터 요소를 동시에 처리할 수 있는 128비트 벡터 레지스터를 활용하여 암호학, 디지털 신호 처리(DSP), 컴퓨터 비전 등 연산 집약적인 애플리케이션에서 성능 향상을 가능하게 한다. 이를 통해 ARMv8 기반 시스템은 제한된 전력 소비로도 높은 계산 효율을 달성할 수 있다.

NEON SIMD는 최대 16개의 8비트 정수, 8개의 16비트 정수, 4개의 32비트 정수 또는 2개의 64비트 정수를 단일 레지스터에서 병렬로 처리할 수 있다. 이러한 구조는 암호 알고리즘의 핵심 연산인 벡터 곱셈과 덧셈, 비트 이동 및 XOR 연산에서 성능을 극대화한다. 예를 들어, 암호 알고리즘의 주요 연산인 GF(2^n) 상의 곱셈은 NEON 명령어를 사용하여 다수의 다항식을 병렬로 계산할 수 있어 데이터 처리량을 증가시킨다. 또한, NEON 명령어는 데이터 로드와 저장을 효율적으로 관리하는 명령어도 포함하고 있다. 벡터 로드(LD1)와 저장(ST1) 명령어는 연속된 메모리 블록을 한 번에 로드하거나 저장하여 메모리 접근 오버헤드를 줄인다. 이는 다중 블록 암호화, 데이터 직렬화 및 복호화 과정에서 처리 지연(latency)을 줄이는 데 특히 유용하다.

NEON의 또 다른 특징은 산술 연산과 논리 연산의 병렬 처리 능력이다. 예를 들어, XOR(EOR), AND(BIC), OR(ORR)와 같은 비트 연산 명령어는 대량의 데이터를 동시에 처리함으로써 암호학적 연산 속도를 비약적으로 향상시킨다. 이러한 연산들은 대칭키 암호 알고리즘이나 메시지 인증 코드(MAC) 계산에 필수적이다.

ARMv8 아키텍처에서 NEON 명령어를 사용하는 병렬 구현 사례로는 다항식 곱셈, 유한체 연산, 벡터 변환 등이 있으며, 이러한 작업들은 비트-단위 수준에서 최적화된다. 특히, NEON 명령어는 매트릭스 곱셈과 같은 연산 작업에서 상당한 성능 향상을 제공한다.

## 2.4 ARMv8 상에서의 최적 구현에 대한 이전 연구

ARMv8 아키텍처에서 PQC(Post-Quantum Cryptography) 알고리즘의 최적 구현에 대한 연구는 활발히 진행되고 있다. 특히, ARMv8의 64비트 환경과 NEON SIMD 명령어를 활용하여 암호화 알고리즘의 성능을 향상시키는 시도가 주목받고 있다.

ARMv8 아키텍처에서의 대표적인 최적화 연구 사례로는 NIST PQC 후보 알고리즘인 SABER의 고속 구현이 있다. SABER는 핵심 연산으로 다항식 곱셈을 사용하며, ARMv8에서 NEON 명령어를 통해 평가(evaluation)와 보간(interpolation) 단계를 가속화하는 연구가 수행되었다. 이 연구에서는 Toom-Cook 곱셈 알고리즘을 기반으로 평가 과정의 데이터를 효율적으로 병렬 처리하기 위해 데이터 인터리빙(interleaving) 기법이 적용되었다. 보간 단계에서는 ARMv8의 SIMD 연산을 통해 고속 덧셈과 곱셈을 수행하여, 레퍼런스 구현 대비 평가 단계에서 약 3.5배, 보간 단계에서는 약 5배의 성능 향상을 달성하였다.

또한, 국내 연구진들은 KpqC 알고리즘 후보인 SMAUG의 ARMv8 상에서의 고속 구현 사례를 발표하였다. 이 연구에서는 GF(2^n)에서의 비트-단위 병렬 곱셈을 NEON 명령어로 구현하여 고속 다항식 곱셈을 수행하였으며, 이를 통해 연산 지연(latency)을 크게 줄였다. 곱셈 모듈은 ARMv8의 128비트 벡터 레지스터를 적극적으로 활용하여 네 개의 32비트 값을 동시에 처리할 수 있도록 설계되었다. 이러한 접근은 PQC 알고리즘의 서명 생성과 검증 속도를 크게 향상시켰으며, 곱셈 연산에서는 최대 24.62배, 암호화 연산에서는 최대 3.51배의 성능 향상을 이루었다.

더불어, Lattice 기반 알고리즘인 Kyber, NTRU 및 Dilithium과 같은 주요 PQC 후보들이 ARMv8 상에서 벡터화(vectorization)와 명령어 스케줄링(instruction scheduling)을 통한 연산 최적화를 수행한 사례도 존재한다. 예를 들어, Kyber의 경우 몽고메리 곱셈(Montgomery multiplication)과 NTT(Negacyclic Number Theoretic Transform) 기반 다항식 연산을 ARMv8에서 최적화하여 데이터 로드 및 저장 오버헤드를 줄였다.

# III. SNOVA 알고리즘 최적 구현 기법

## 3.1 덧셈기 최적화

SNOVA 알고리즘에서 행렬 덧셈 연산은 최적 구현 시 중요한 요소 중 하나이다. 이 연산은 유한체 GF(16) 상에서 이루어지며, 다항식 표현을 기반으로 비트 단위 XOR 연산을 수행한다. 이러한 연산은 공개키 생성, 서명 생성 및 검증과 같은 모든 주요 작업에서 필수적이다. SNOVA 알고리즘에서는 행렬 덧셈을 수행하는 asm_add 함수가 그 중심에 있으며, ARMv8 아키텍처의 특징을 최대한 활용하여 성능 최적화를 목표로 한다. 이 함수는 rank 값에 따라 다른 크기의 행렬 연산을 수행한다. rank는 행렬의 차원을 나타내며, 구현에서 주로 rank=2, rank=3, rank=4의 경우가 고려되었다. 각 rank 값에 대해 서로 다른 어셈블리 최적화 기법이 적용되었다.

**Algorithm** : SNOVA 덧셈기 최적 구현

Input: $A = a0, a1, \ldots, an, B = b0, b1, \ldots, bn$
Output: $C = A \oplus B$
#Inialization
1: for $i = 0$ *to* $n$ *do*
2: $C[i] \leftarrow A[i] \oplus B[i]$
#Addition
3: $A[i] \oplus B[i]$ for *each* $i$

[알고리즘 3-1] SNOVA 덧셈기 최적 구현

rank=2의 경우, 행렬 크기는 2×2로 총 4개의 요소만 연산에 참여한다. ARMv8의 32비트 일반 레지스터를 활용하여 두 개의 4바이트 데이터를 XOR하는 방식으로 구현되었다. 각 행렬 요소는 단일 레지스터에 로드되어

XOR 연산을 수행한 후 결과가 메모리에 저장된다.

rank=3은 행렬 크기가 3×3으로 총 9개의 요소가 필요하다. ARMv8의 64비트 NEON SIMD 레지스터와 32비트 일반 레지스터가 함께 사용되었다. 처음 8바이트는 NEON SIMD 레지스터에 로드되어 빠른 병렬 연산을 수행하며, 마지막 1바이트는 일반 레지스터를 통해 처리된다. 이 방식은 NEON이 16바이트 단위로 병렬 연산을 지원하지만, 정확히 9바이트가 필요하므로 두 가지 연산 방식이 혼합되어 사용된다.

rank=4의 경우, 행렬 크기는 4×4로 총 16개의 요소가 포함된다. ARMv8의 128비트 NEON 레지스터는 이 데이터를 단일 로드-저장 명령으로 한 번에 처리할 수 있다. 두 개의 16바이트 데이터를 NEON 레지스터에 로드하고, XOR 연산을 수행한 후 결과를 메모리에 저장하는 방식이다. 이 구현은 최소한의 명령어로 최대한의 병렬 처리를 수행하여, rank2, 3보다 효율적인 데이터 처리를 가능하게 한다.

이와 같은 rank 기반 최적화는 메모리 접근을 최소화하고 계산을 병렬화하여 성능을 극대화한다. 어셈블리 수준의 최적화를 통해 명령어 수를 줄임으로써 전반적인 처리 속도를 크게 개선하였다.

## 3.2 곱셈기 최적화

SNOVA 알고리즘의 주요 연산 중 하나는 유한체 GF(16) 상의 행렬 곱셈이다. 이 연산은 공개키 생성, 서명 생성, 검증 과정에서 가장 큰 계산 자원을 요구하는 핵심 연산으로, 전체 알고리즘 성능을 좌우한다. ARMv8 아키텍처에서는 이 곱셈 연산을 최적화하기 위해 NEON SIMD(Single Instruction Multiple Data) 벡터 연산이 활용되었다. NEON은 여러 데이터를 병렬로 처리할 수 있는 128비트 벡터 레지스터를 제공하므로, GF(16) 행렬 곱셈을 병렬화하여 계산량을 줄일 수 있다.

행렬 곱셈은 두 개의 입력 행렬 $A$와 $B$의 행과 열을 순차적으로 처리하여 결과 행렬 $C$를 생성하는 방식으로 수행된다. 기본적인 연산은 다음과 같다.

$$C[i][j] = \Sigma_k A[i][k] \cdot B[k][j]$$

수식 3-1 행렬 곱셈

ARMv8의 NEON 레지스터는 최대 16개의 8비트 데이터를 한 번에 처리할 수 있다. 이를 기반으로 SNOVA의 gf16m_neon_mul 함수는 두 개의 행렬을 병렬로 곱하고 결과를 누적하는 방식으로 구현되었다. 행렬의 rank에 따라 다른 차원의 행렬이 사용되며, 그에 맞게 최적화된 어셈블리 코드가 적용되었다.

**Algorithm** : SNOVA 곱셈기 최적 구현

Input: $A = a0, a1, \ldots, an, B = b0, b1, \ldots, bn$
Output: $C = A \times B$
#Inialization
1: for $i = 0$ *to* $n$ *do*
2: $C[i] \leftarrow 0$
#Multiplication
3: for $i = 0$ *to* $n$ *do*
4: for $i = 0$ *to* $n$ *do*
5. *Multiply* $A[i] \times B[j]$
6. $C[i] \leftarrow C[i] \oplus (A[i] \times B[j])$
#Storing
7: *Store* $C[i] \in$ *memory* for *each* $i$

[알고리즘 3-2] SNOVA 곱셈기 최적 구현

rank=4에서는 총 16개의 행렬 요소가 곱셈에 참여하며, ARMv8 NEON의 128비트 레지스터는 정확히 16바이트 데이터를 병렬로 처리할 수 있다. 이 경우 각 행의 열 요소를 8비트 값으로 확장한 후 NEON 명령어를 사용하여 각 열에 대한 곱셈과 누적 연산을 수행한다. 곱셈은 vdupq_n_u8 명령어를 사용해 8비트 값을 128비트로 확장하고, veorq_u8 명령어를 통해 XOR 연산을 수행한다. 각 열은 병렬로 처리되며, 최종 결과는 스칼라 값으로 변환되어 메모리에 저장된다.

rank=3에서는 총 9개의 행렬 요소가 곱셈 연산에 참여한다. ARMv8 NEON의 128비트 레지스터는 16바이트 데이터를 병렬로 처리할 수 있으므로, 9바이트 데이터를 처리하는 이 경우에는 일부 레지스터 용량이 남게 된다. 전체적인 구조는 rank 4의 최적 구현 코드와 유사하나, 레지스터를 모두 활용할 수 없어 효율은 약간 저하된다.

rank=2는 2x2 크기의 행렬 곱셈으로, 총 4개의 요소가 연산에 참여한다. 이 경우 NEON SIMD 레지스터의 처리 용량인 16바이트 대비 활용도가 낮아 일부 비효율이 발생할 수 있다.

NEON SIMD 기반 연산은 rank=4에서는 최고의 성능을 발휘하지만, rank=3에서는 남는 바이트 문제로 인해 일부 레지스터 활용도가 떨어진다. rank=2에서는 총 4바이트만 처리되므로 NEON 레지스터의 활용도가 더욱 낮아지지만, 그래도 일반 스칼라 명령어보다는 우수한 성능을 제공한다.

## 3.3 AES 가속기

AES-128 CTR 모드는 SNOVA 전자서명 알고리즘에서 난수 생성을 위한 주요 암호화 구성 요소로 사용된다. ARMv8 아키텍처는 AES 연산을 가속하기 위한 하드웨어 명령어 집합을 지원하여 고속 암호화를 가능하게 하지만, 이를 효율적으로 사용하기 위해서는 올바른 AES 키 스케줄 구현이 필수적이다. 본 연구에서는 ARMv8 하드웨어 명령어와 직접 호환되는 표준 AES-128 키 스케줄을 어셈블리 코드로 구현하였다. 이는 SNOVA 알고리즘의 성능을 최적화하고 기존의 BearSSL 기반 키 스케줄 구현에서 발생할 수 있는 비호환성 문제를 해결하기 위한 것이다.

BearSSL은 소프트웨어 이식성과 보안성을 목표로 설계된 라이브러리로, AES 암호화를 비트슬라이스(bit-sliced) 방식으로 처리한다. 이 방식은 복수의 입력 블록을 병렬로 처리할 수 있는 장점을 제공하지만, ARMv8 하드웨어 명령어 AESE 및 AESMC와 직접 호환되지 않는다. BearSSL에서 생성되는 AES 라운드 키는 비트슬라이스 형식으로 저장되며, 이는 전통적인 AES 라운드 키 배열과 구조적으로 다르다. 따라서 하드웨어 가속 명령어와 결합

하여 사용할 수 없다.

ARMv8 AES 명령어를 활용하려면 총 11개의 16바이트 라운드 키가 표준 AES 키 스케줄 알고리즘을 통해 생성되어야 한다. 이를 위해 본 연구에서는 AES-128 키 스케줄을 어셈블리 언어로 직접 구현하였다. 초기 입력 키를 첫 번째 라운드 키로 설정한 후, 표준 AES 키 확장 과정을 반복 수행하여 남은 10개의 라운드 키를 생성한다. 주요 연산은 키 워드의 회전(RotWord), S-box 치환(SubWord), 라운드 상수(Rcon) 적용, XOR 연산 등으로 구성되며, 각 라운드 키는 이러한 연산을 거쳐 계산된다.

**Algorithm** : AES-128 CTR모드 가속기 구현

**Input:** $P = \{p_0, p_1, \ldots, p_n\}$ (Plaintext), $K$ (AES Key), $N$ (Nonce)
**Output:** $C = \{c_0, c_1, \ldots, c_n\}$ (Ciphertext)

**Initialization:**
Load the AES round keys $R[i]$ using the standard AES key schedule.
Initialize the counter block Ctr with the given nonce $N$.
**Encryption Loop:**
**for** $i = 0$ **to** $n - 1$ **do**
    Load the counter block Ctr$[i]$ into a NEON vector register.
    Encrypt Ctr$[i]$ using the AES rounds:
    **for** $j = 0$ **to** 9 **do**
        Apply AESE$(R[j])$ and AESMC$(R[j])$.
    **end for**
    Apply the final round AESE$(R[10])$.
    XOR the encrypted counter block with the plaintext block:
    $C[i] = P[i] \oplus$ Encrypted Ctr$[i]$.
**end for**
**Counter Update:**
Increment the counter: Ctr$[i + 1]$ = Ctr$[i] + 1$.
**Output:**
Store each ciphertext block $C[i]$ in memory.

[알고리즘 3-3] AES-128 CTR모드 가속기 구현

AES 가속기 구현은 ARMv8의 벡터 레지스터와 AES 전용 명령어를 결합하여 최대 성능을 발휘할 수 있도록 설계되었다. AESE 명령어는 AES 라운드 암호화를 수행하고, AESMC 명령어는 AES 믹스 컬럼(MixColumns) 연산

을 적용한다. 16바이트 블록이 처리 단위이며, 초기값은 넌스(nonce)로 설정되며 이후 암호화 루프를 통해 블록 단위로 증분된다.

암호화 루프는 넌스 값을 벡터 레지스터에 로드하고, 이를 AES 라운드 키와 결합하여 AESE 및 AESMC 명령어로 반복 처리한다. 각 라운드는 라운드 키와 XOR 연산을 수행한 후 S-box와 믹스 컬럼 연산을 적용한다. 마지막 라운드에서는 믹스 컬럼을 제외하고 암호문이 생성된다. 결과는 출력 버퍼에 저장되며, 카운터 값은 넌스 값에 증분되어 다음 블록 암호화를 준비한다.

이러한 구현은 메모리 접근 수를 최소화하고 암호화 루프 내에서 모든 연산이 벡터 레지스터에서 수행되도록 최적화되었다. AES 라운드 키는 초기화 시 미리 계산되어 레지스터에 로드되므로, 매 암호화 라운드마다 불필요한 메모리 접근이 제거된다. 이는 기존의 소프트웨어 AES 구현과 비교해 큰 성능 향상을 가능하게 한다.

# IV. 성능 평가

## 4.1 성능 측정 환경

SNOVA 알고리즘의 ARMv8 아키텍처 상에서의 최적화 구현 성능을 평가하기 위해 벤치마크를 수행하였다. 성능 평가는 ARM 플랫폼에서 PQC 알고리즘의 성능을 테스트하고 분석하기 위해 설계된 mupq 프레임워크를 기반으로 진행되었다. mupq를 선택한 이유는 타 알고리즘과의 형평성을 유지하기 위함이다.

mupq는 PQC 알고리즘을 ARM Cortex-M 및 ARMv8과 같은 저전력 임베디드 환경에서 구현하고 최적화할 수 있는 프레임워크로, 알고리즘의 실행 시간 및 자원 사용량 분석을 위한 기능을 제공한다.

NIST에서 제공하는 참조 코드는 OpenSSL과 같은 특정 라이브러리를 사용하며, 이는 ARMv8 환경과 최적화 수준이 반드시 일치하지 않을 수 있다., 이는 추후 타 알고리즘과 성능 비교 시 공정성을 해칠 가능성이 있다. 반면, mupq는 ARM 아키텍처에 특화된 벤치마크 환경을 제공하므로, 동일 조건에서 타 알고리즘과의 성능 비교를 할 수 있다.

벤치마크 대상은 주요 암호화 작업인 키 생성(crypto_sign_keypair), 서명(crypto_sign), 그리고 **서명 검증(crypto_sign_verify)**으로, 해당 작업들의 실행 사이클을 측정하였다. 성능 측정 환경은 Apple M2 칩(8GB RAM) 하드웨어를 기반으로 구성되었으며, GCC 14.0.3 컴파일러를 사용하여 O3 최적화 레벨로 빌드하였다. 소프트웨어 개발 환경으로는 Visual Studio Code(VSCode)를 사용하였다. 추가적으로, 최적화된 덧셈 및 곱셈 모듈의 성능을 측정하기 위해 사용자 정의 벤치마크 스크립트를 개발하였다.

## 4.2 성능 측정 결과

최적화된 SNOVA 구현의 성능 평가는 esk(Extended Secret Key)와 ssk(Secret Signing Key) 두 가지 파라미터 세트에 대해 이루어졌다. SNOVA 알고리즘은 총 18개의 파라미터 세트를 포함하며, 이 중 9개는 esk 파라미터 세트, 나머지 9개는 ssk 파라미터 세트로 구성된다. 두 파라미터 세트는 각각의 특성과 목적에 따라 설계되었다.

esk는 확장된 비밀 키로, ssk에서 추가 연산을 통해 생성된 고급 형태이다. 이는 보조 데이터나 사전 계산된 값 등을 포함하며, 성능 최적화를 위해 서명 생성 시 빠르게 접근할 수 있는 구조를 제공한다. 그러나 추가적인 연산과 데이터 구조로 인해 계산이 더 복잡하고 느릴 수 있다. 반면, ssk는 서명 생성에 직접 사용하는 기본 비밀 키로, 연산 과정이 단순하고 가볍게 설계되어 있다. ssk는 주로 서명 생성의 핵심 키로 사용되며, esk 생성의 입력값으로 활용된다.

[표 4-1] esk 파라미터 최적 구현 성능 측정 결과

| 파라미터 | Work | 레퍼런스 | 최적 구현 |
|---|---|---|---|
| snova-24-5-16-4-esk | 키 생성 | 3,475,085 | 1,398,483 |
| | 서명 생성 | 15,753,137 | 10,707,076 |
| | 서명 검증 | 10,629,480 | 5,898,839 |
| snova-25-8-16-3-esk | 키 생성 | 3,463,081 | 2,050,620 |
| | 서명 생성 | 6,484,644 | 6,007,548 |
| | 서명 검증 | 3,158,913 | 2,843,025 |
| snova-28-17-16-2-esk | 키 생성 | 5,557,774 | 4,370,283 |
| | 서명 생성 | 2,144,271 | 2,127,047 |
| | 서명 검증 | 2,282,003 | 1,330,599 |
| snova-37-8-16-4-esk | 키 생성 | 18,112,642 | 8,641,329 |
| | 서명 생성 | 59,263,911 | 41,794,416 |
| | 서명 검증 | 40,590,714 | 22,410,705 |
| snova-43-25-16-2-esk | 키 생성 | 24,795,480 | 21,714,931 |
| | 서명 생성 | 7,117,074 | 7,062,414 |
| | 서명 검증 | 7,609,277 | 4,451,052 |
| snova-49-11-16-3-esk | 키 생성 | 17,887,340 | 13,837,133 |
| | 서명 생성 | 23,711,616 | 23,494,859 |
| | 서명 검증 | 14,493,366 | 11,888,327 |
| snova-60-10-16-4-esk | 키 생성 | 69,097,561 | 33,384,333 |
| | 서명 생성 | 168,794,828 | 115,369,891 |
| | 서명 검증 | 122,557,890 | 67,624,592 |
| snova-61-33-16-2-esk | 키 생성 | 83,359,185 | 73,577,806 |
| | 서명 생성 | 17,722,885 | 17,651,531 |
| | 서명 검증 | 19,220,693 | 11,221,976 |
| snova-66-15-16-3-esk | 키 생성 | 57,863,043 | 46,400,037 |
| | 서명 생성 | 60,121,205 | 60,103,122 |
| | 서명 검증 | 35,164,753 | 31,463,332 |

esk 파라미터 세트의 성능 측정 결과는 위 표와 같다. Rank 4에서 모든 작업에서 가장 높은 성능 향상이 관찰되었다. 키 생성 작업에서는 약 51.7~65.2%의 성능 향상을 기록하였고, 서명 생성 작업에서는 최대 36.7%, 서명 검증 작업에서는 44.8~50.0%의 성능 향상이 있었다. 이는 Rank 4가 ARMv8의 NEON 벡터 연산을 완전히 병렬로 활용할 수 있기 때문이다. 반면, Rank 2와 Rank 3의 경우 성능 향상 폭은 상대적으로 낮았다. Rank 2는 연산 요소가 적어 NEON 벡터 연산의 병렬 처리 장점을 충분히 활용하

지 못하며, Rank 3는 9개의 행렬 요소를 처리해야 하는 특성상 NEON 레지스터의 8바이트 단위 병렬 처리 구조와 비효율적인 정렬로 인해 성능이 제한되었다. 이에 따라 Rank 2와 3에서는 키 생성에서 약 20~30%, 서명 검증에서는 약 10~25%의 성능 향상이 나타났다. 특히, 서명 생성 작업에서는 Rank 4 대비 성능 개선이 거의 없거나 미미한 수준으로 관찰되었다.

[표 4-2] ssk 파라미터 최적 구현 성능 측정 결과

| 파라미터 | Work | 레퍼런스 | 최적 구현 |
|---|---|---|---|
| snova-24-5-16-4-ssk | 키 생성 | 3,162,043 | 1,398,307 |
| | 서명 생성 | 15,532,906 | 10,813,358 |
| | 서명 검증 | 10,665,857 | 5,911,721 |
| snova-25-8-16-3-ssk | 키 생성 | 2,964,322 | 2,047,839 |
| | 서명 생성 | 5,934,865 | 5,904,548 |
| | 서명 검증 | 3,777,688 | 2,895,751 |
| snova-28-17-16-2-ssk | 키 생성 | 5,839,063 | 4,363,678 |
| | 서명 생성 | 2,134,444 | 2,126,962 |
| | 서명 검증 | 2,270,131 | 1,335,159 |
| snova-37-8-16-4-ssk | 키 생성 | 20,797,176 | 8,641,214 |
| | 서명 생성 | 59,903,432 | 41,767,647 |
| | 서명 검증 | 40,680,194 | 22,403,252 |
| snova-43-25-16-2-ssk | 키 생성 | 26,863,060 | 21,603,320 |
| | 서명 생성 | 7,011,836 | 7,001,149 |
| | 서명 검증 | 7,617,343 | 4,451,764 |
| snova-49-11-16-3-ssk | 키 생성 | 19,166,980 | 13,781,456 |
| | 서명 생성 | 23,715,980 | 23,580,833 |
| | 서명 검증 | 14,569,760 | 12,809,535 |
| snova-60-10-16-4-ssk | 키 생성 | 69,064,963 | 33,406,701 |
| | 서명 생성 | 168,849,382 | 115,361,855 |
| | 서명 검증 | 122,586,802 | 67,590,588 |
| snova-61-33-16-2-ssk | 키 생성 | 82,685,800 | 73,589,368 |
| | 서명 생성 | 17,421,722 | 17,250,398 |
| | 서명 검증 | 19,222,942 | 11,213,860 |
| snova-66-15-16-3-ssk | 키 생성 | 57,911,107 | 46,343,915 |
| | 서명 생성 | 60,674,453 | 59,117,498 |
| | 서명 검증 | 35,254,524 | 31,426,092 |

ssk 파라미터 세트는 상대적으로 간단한 구조를 가지며, 성능 측정 결과

가 esk와 비슷한 경향을 보였으나 조금 더 빠른 성능을 보이는 것으로 확인되었다. esk와 마찬가지로 Rank 4에서 가장 높은 성능 향상이 나타났으며, 키 생성 작업에서는 약 53.8~68.7%, 서명 생성 작업에서는 29.5%, 서명 검증 작업에서는 42.8~47.2% 정도의 성능 향상이 있었다. Rank 2와 Rank 3에서도 성능 향상이 있었지만, Rank 4만큼의 개선은 이루어지지 않았다. Rank 2는 병렬화 효율이 제한되었고, Rank 3는 구조적 제약으로 인해 성능 개선이 비교적 낮았다. 키 생성에서 약 25%, 서명 검증에서 약 15~20%의 성능 향상이 관찰되었으며, 서명 생성에서는 Rank 4 대비 개선 폭이 크지 않았다.

SNOVA의 esk와 ssk 파라미터 세트는 모두 최적화된 구현을 통해 성능 향상을 이루었으며, 특히 Rank 4에서 가장 높은 성능 향상이 관찰되었다. 이는 NEON 병렬 처리 기능을 최대한 활용한 결과로 분석된다. 반면, Rank 2와 Rank 3는 병렬화 효율의 제약으로 인해 성능 향상 폭이 상대적으로 낮았다. esk와 ssk의 성능 차이가 미미한 이유는 ssk의 단순한 구조와 esk의 추가 연산이 성능에 미치는 영향을 상쇄하기 때문으로 보인다.

# V. 결 론

본 연구에서는 ARMv8 아키텍처 상에서 SNOVA 전자 서명 알고리즘의 최적 구현을 제안하고, 성능 최적화를 통해 주요 작업인 키 생성, 서명 생성, 서명 검증에서의 실행 시간을 크게 단축할 수 있음을 입증하였다. 연구의 주요 초점은 ARMv8의 하드웨어 가속 기능과 NEON SIMD 명령어를 활용하여 SNOVA 알고리즘의 핵심 연산인 행렬 덧셈, 곱셈, 그리고 AES-CTR 모드를 효과적으로 병렬화하는 데 있었다. 이를 통해 기존의 참조 구현 대비 모든 파라미터 세트에서 높은 성능 향상을 달성하였다.

특히, Rank 4의 파라미터 세트에서 가장 두드러진 성능 향상이 관찰되었다. 이는 NEON 벡터 연산이 완전히 병렬로 활용되어 병렬 처리 효율이 극대화된 결과로 분석된다. 반면, Rank 2와 Rank 3에서는 구조적 제약과 병렬화 한계로 인해 성능 향상이 상대적으로 낮게 나타났다. 또한, esk와 ssk 파라미터 세트 간의 성능 차이는 미미하였으며, 이는 esk의 복잡성이 성능에 미치는 영향을 상쇄하기 때문으로 해석된다.

SNOVA의 최적 구현은 기존 PQC(Post-Quantum Cryptography) 알고리즘 최적화 연구에서 부족했던 ARMv8 아키텍처 기반 다변수 공개키 암호(MPKC) 시스템의 연구 공백을 채우는 중요한 기여를 하였다. 본 연구 결과는 SNOVA 알고리즘이 제한된 자원 환경에서 실행 가능하며, 모바일 및 IoT 디바이스와 같은 저전력 고효율 플랫폼에 적합한 암호화 시스템으로 자리잡을 가능성을 제시한다.

# 참 고 문 헌

## 1. 국외문헌

National Institute of Standards and Technology (NIST), “Post-Quantum Cryptography Standardization,” https://csrc.nist.gov/projects/pqc-dig-sig/round-1-additional-signatures, 2024.

Kipnis, A., and Shamir, A., “Cryptanalysis of the Oil & Vinegar Signature Scheme,” Lecture Notes in Computer Science, vol. 1294, pp. 257-266, 1997.

SNOVA, “Proposal for NISTPQC: Digital Signature Schemes project,” May 25, 2023.

ARM Holdings, “ARM Architecture Reference Manual, ARMv8,” ARM Limited, 2021.

Apple Inc., “Apple M2 Chip,” https://www.apple.com/newsroom/2022/06/apple-unveils-m2-with-breakthrough-performance-and-capabilities, 2022.

Bernstein, D. J., Buchmann, J., & Dahmen, E. (Eds.). (2009). Post-Quantum Cryptography. Springer Science & Business Media.

Goedhart, L. D., & Pollard, D. J. (2017). Efficient cryptographic algorithms for ARMv8 architecture. Journal of Cryptographic Engineering, 7(2), 143-153.

Hyukdong Kwon, Kyungjoo Song, Minju Sim, Minwoo Lee, & Hwajung Seo (2023). "High-Performance Implementation of SMAUG on 64-bit ARMv8 Processors." Proceedings of the Korea Information Processing Society Conference, 113-115.

Donghyun Kim, Dongwoo Kim, Sangjin Lee, & Seokjong Seo (2021). "Optimized Implementation of Toom-Cook Algorithm in NIST PQC SABER Using ARM/NEON Processors." Journal of Information

Security and Cryptography, 31(5), 1057-1068.
Langlois, A., & Stehlé, D. "Worst-case to Average-case Reductions for Module Lattices." Journal of Cryptographic Engineering, vol. 10, no. 2, pp. 155-174, 2020.
Seo, H., & Kim, M. "Optimized Implementation of PQC Schemes on ARM Platforms." IEEE Transactions on Information Forensics and Security, vol. 15, no. 6, pp. 980-994, 2022.
Courtois, N. T., & Goubin, L. "Efficient Multiplications in GF(2^n) with Applications to Cryptographic Protocols." Advances in Cryptology – EUROCRYPT, pp. 392-405, 2018.
Duong, L., & Nguyen, P. Q. "Lattice-based Post-Quantum Cryptography: Implementations and Security Analysis." Proceedings of IEEE Symposium on Security and Privacy, pp. 1-12, 2021.
Beullens, J., & Vercauteren, F. "Efficient Arithmetic in Finite Fields for Post-Quantum Cryptography." Journal of Cryptology, vol. 14, no. 3, pp. 233-256, 2020.
Arm Limited, "Optimizing Cryptographic Algorithms Using NEON on ARMv8," Technical Report, 2022.
Kim, S., & Lee, J. "Efficient Modular Arithmetic for Post-Quantum Cryptography in ARMv8-based Systems." IEEE Transactions on Computers, vol. 69, no. 4, pp. 731-743, 2022.
NIST PQC Team, "Finalists in Post-Quantum Cryptography Standardization." NIST Official Report, 2023.
Seo, M., & Kim, J. "Advanced Key Management for Quantum-Resistant Cryptographic Algorithms." International Journal of Applied Cryptography, vol. 19, no. 4, pp. 512-534, 2023.
PQClean. "Clean, Portable, Tested Implementations of Post-Quantum Cryptography.", 2024.
mupq. "Post-Quantum Cryptographic Implementations for

Microcontrollers.", 2024.
Kannwischer, M. J., Rijneveld, J., Schwabe, P., & Stoffelen, K. "pqm4: Testing and Benchmarking NIST PQC on ARM Cortex-M4." IACR Cryptology ePrint Archive, 2019, 844.
Kipnis, A., & Shamir, A. "Cryptanalysis of the Oil & Vinegar Signature Scheme." In Lecture Notes in Computer Science, Vol. 1294, pp. 257-266. 1997.

# ABSTRACT

## Optimal Implementation of SNOVA on ARMv8 Architecture

Lee, Min-Woo

Major in Convergence Security

Dept. of IT Convergence Engineering

The Graduate School

Hansung University

This thesis focuses on the optimal implementation of the SNOVA digital signature algorithm on the ARMv8 architecture. SNOVA, a post-quantum cryptographic system, ensures high security based on multivariate quadratic equations. In this study, assembly language and NEON SIMD instructions were employed to optimize key generation, signature creation, and signature verification operations in SNOVA. The main optimizations include the implementation of addition and multiplication modules for finite field operations, along with an AES-128 CTR mode accelerator. The addition and multiplication operations were designed for parallel processing using ARMv8 vector registers, while the AES accelerator maximized encryption speed through hardware instructions. Performance evaluations revealed significant improvements in key

cryptographic tasks. The optimized implementation achieved up to 68.7% improvement in key generation, 36.7% in signature creation, and 50.0% in signature verification for major SNOVA parameter sets (esk and ssk). This research demonstrates the feasibility of implementing multivariate-based digital signature algorithms on ARMv8 and provides a foundational reference for future development of efficient post-quantum cryptographic systems.