엑셀 환경에서의 SHA-3 알고리즘 구현

안상현, 원희정, 강주성, 염용진*
국민대학교*
{angry5424, hiijng1220, jskang, *salt}@kookmin.ac.kr

Implementation of SHA-3 Algorithm in Excel Environment

SangHyun Ahn, Heejeung Won, Ju-Sung Kang, Yongjin Yeom*
Kookmin University

요 약

해시함수는 가변 길이의 입력에 대해 고정된 길이의 해시값으로 출력하는 함수로, 주로 데이터의 무결성 검증 및 인증을 위한 암호 시스템의 핵심 요소이다. 본 논문에서는 NIST의 표준 해시함수인 SHA3-224를 엑셀 환경에서 구현하여 내부 연산 과정을 가시화하였으며, 이를 통해 SHA-3 알고리즘에 대한 이해를 돕고 구현의 정확성을 검증하기 위한 분석 도구를 제공하고자 한다.

Ⅰ. 서론

암호학적 해시함수는 데이터의 무결성 검증, 전자서명, 인증 등 현대 암호 시스템의 운용에 있어 필수적인 요소이다. Merkle Damgård 구조 기반의 해시함수인 MD5, SHA-1에서 충돌 저항성에 대한 취약점이 발견됨에 따라, 미국 국립표준기술연구소(National Institute of Standards and Technology, 이하 NIST)는 SHA-1 사용에 제한을 두고 보안 강도를 개선한 SHA-2를 표준으로 사용할 것을 권고하고 있다[1]. 그러나 SHA-2 또한 동일한 Merkle Damgård 구조를 기반으로 하고 있어, 잠재적인 구조적 취약점에 대한 우려가 있다. 이에 따라 NIST는 기존 설계 방식과 상이한 Sponge 구조 기반의 SHA-3 표준을 선정하여 구조적 다양성을 확보하고자 하였다[2]. 이러한 SHA-3 및 이의 XOF(Extendable-Output Function)인 SHAKE는 최근 NIST에서 발표한 양자내성암호(Post-Quantum Cryptography) 표준 규격에 필수적으로 포함되면서 향후 암호 체계의 핵심적인 기술로 그 역할이 확대되고 있다.

한편, 암호 알고리즘의 교육 및 가시화를 목적으로 DES, AES, SHA-2 등을 엑셀과 같은 스프레드시트 환경에서 구현한 선행 연구가 수행된 바 있다[3, 4]. 따라서 본 논문에서는 SHA3-224 알고리즘을 엑셀 환경에서 구현함으로써, 암호화 과정의 내부 상태 변화를 가시화하고 구현 정확성을 검증할 수 있는 도구를 제공한다.

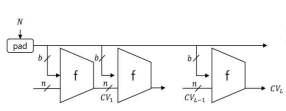
Ⅱ. SHA-3

2.1 해시함수

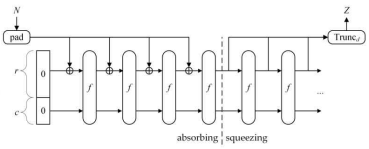
해시함수란 가변 길이의 입력 데이터를 고정 길이의 해시값으로 출력하는 일방향 함수이다. 암호학적 해시함수는 역상 저항성, 제2 역상 저항성, 충돌 저항성을 보장함으로써, 해시값으로부터 원본 데이터를 복원하거나 동일한 해시값을 갖는 입력 쌍을 찾는 것을 계산적으로 불가능하게 한다. 이를 바탕으로 암호학적 해시함수는 데이터의 무결성 검증 및 위·변조 탐지 등의 핵심 기술로 사용되며, 현대 암호 프로토콜의 필수적인 기반 기술로 활용된다[1].

2.2 Sponge 구조

SHA-1, SHA-2 알고리즘은 Merkle Damgård 구조를 기반으로 압축 함수를 반복 적용하여 해시값을 생성하는 반면, SHA-3는 Sponge 구조를 기반으로 내부 상태 변수 state에 순열 함수를 반복 적용하여 해시값을 생성하도록 설계되었다. Sponge 구조는 입력 메시지 블록을 state와 XOR 한 후 순열 함수를 거쳐 state를 갱신하는 과정을 모든 메시지 블록에 대해 반복하여 메시지 정보를 흡수하는 Absorbing phase와 최종적으로 갱신된 state로부터 해시값을 추출하는 Squeezing phase로 구성된다.



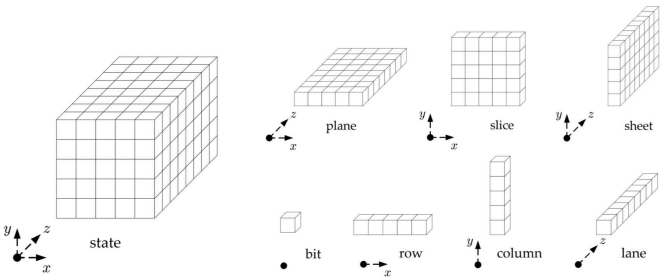
[그림 1] Merkle Damgård 구조



[그림 2] Sponge 구조[2]

2.3 SHA-3 state 표현과 Step mapping

SHA-3는 $5 \times 5 \times 64$ 크기의 3차원 state 단위로 연산을 수행하며, state 구조를 도식화하면 [그림 3]과 같다. 특히 state에서 z축 방향으로 배열된 64비트 열을 Lane으로 정의하고, 각 Lane은 $[x, y]$ 좌표로 나타낸다.



[그림 3] SHA-3의 state 표현[2]

SHA-3 알고리즘의 순열 함수는 총 24라운드로 구성된다. 순열 함수의 한 라운드를 이루는 다섯 가지의 step mapping 함수를 각각 $\theta, \rho, \pi, \chi, \iota$라 하며, $\theta \rightarrow \rho \rightarrow \pi \rightarrow \chi \rightarrow \iota$ 순서로 각 함수를 거쳐 state를 갱신하며, 이 과정에서 state 내부 비트의 확산이 이루어진다[2].

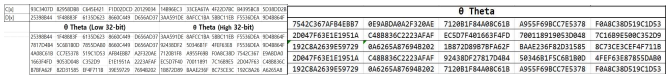
Ⅲ. 엑셀에서의 SHA3-224 구현

기존의 SHA-3 구현은 주로 메시지에 대한 해시값은 확인할 수 있지만, Step Mapping 과정에서의 state 변화를 파악하기 어렵다. 본 논문에서는 state를 시각화하기에 적합한 엑셀 환경에서 SHA3-224를 구현하였다. SHA3-224는 224비트의 해시값을 출력하는 알고리즘으로, SHA-3 계열 중 메시지를 흡수하는 영역인 rate 영역이 가장 크다. 연산 효율성을 고려하여 동일한 메시지에 대해 순열 함수의 호출을 최소화할 수 있는 해당 알고리즘을 선택하였다.

3.1 state 표현

SHA-3의 state는 [그림 4]의 64비트 단위 표현과 같이 5×5개의 64비트 Lane으로 구성된다. 엑셀 환경에서 비트 연산자는 48비트 이상의 정수 입력에 대해 처리할 수 없으므로 64비트 Lane을 상위 32비트(High

32-bit)와 하위 32비트(Low 32-bit)의 두 블록으로 나누어 연산을 수행하며, 별도의 시트에서 두 블록을 결합하여 실제 state 표현을 통해 연산 과정의 중간값을 확인할 수 있도록 구성하였다. [그림 4]는 각 시트에서 state의 중간값을 확인한 예시이다.



[그림 4] state 중간값 예시(좌: 32비트 단위 표현, 우: 64비트 단위 표현)

3.2. Step mapping

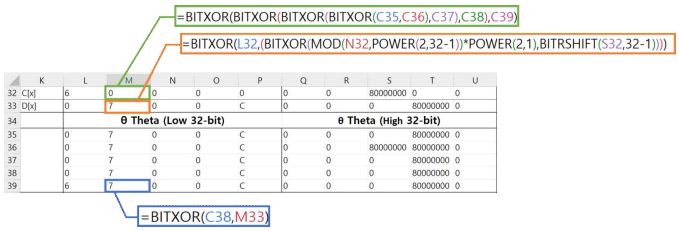
Step mapping 함수는 비트별 독립적으로 수행되는 비트 단위 연산 XOR, AND, NOT과 비트 순환 연산인 ROTL로 구성되며, Lane 단위로 해당 연산을 수행한다. 이때 ROTL 연산은 carry 비트로 인해 상, 하위 블록 간의 비트 이동이 필요하므로, 이에 대한 추가적인 처리가 필요하다. 본 절에서는 ROTL 연산 처리를 포함한 Step mapping의 θ 함수에 대한 엑셀 구현 예시를 제시한다.

θ 함수는 $C[x], D[x]$을 계산하고 이를 이용하여 Lane $L[x, \cdot\,]$을 갱신하며, 자세한 연산 과정은 [그림 5]와 같다. [그림 6]은 1라운드의 θ 함수에서 $L[1, 0]$을 갱신하는 과정을 구현한 예시이다. 여기서 셀 M32, M33, M39는 각각 $C[1]$, $D[1]$, $L[1, 0]$의 Low_32bit를 의미한다.

Steps:

1. For all pairs (x, z) such that $0 \le x < 5$ and $0 \le z < w$, let
   $C[x, z] = \mathbf{A}[x, 0, z] \oplus \mathbf{A}[x, 1, z] \oplus \mathbf{A}[x, 2, z] \oplus \mathbf{A}[x, 3, z] \oplus \mathbf{A}[x, 4, z]$.
2. For all pairs (x, z) such that $0 \le x < 5$ and $0 \le z < w$ let
   $D[x, z] = C[(x-1) \bmod 5, z] \oplus C[(x+1) \bmod 5, (z-1) \bmod w]$.
3. For all triples (x, y, z) such that $0 \le x < 5$, $0 \le y < 5$, and $0 \le z < w$, let
   $\mathbf{A}'[x, y, z] = \mathbf{A}[x, y, z] \oplus D[x, z]$.

[그림 5] θ 함수 알고리즘[2]



[그림 6] θ 함수 엑셀 구현 예시

3.3 입력 매개변수 및 메시지 전처리

해시값 생성을 위해 엑셀 시트에 입력하는 매개변수는 [표 1]과 같다.

[표 1] 입력 매개변수

MSG	16진수 문자열 형식의 입력 메시지
MSG_Len	입력 메시지의 크기(단위: 비트)
TEST_Vector	검증용 해시값으로, 입력값(MSG, MSG_Len)에 대해 올바른 해시값을 생성하는지 비교하는 기준으로 사용

입력 메시지의 길이를 바이트 단위로 제한하지 않고, 비트 단위의 입력을 허용하기 때문에 입력 메시지를 고정된 크기의 블록으로 나누는 과정에서 패딩 처리가 필요하다. 다만, 입력 메시지 전처리 및 패딩을 엑셀에서 지원하는 기본 수식만으로 구현하기에는 기능적 한계가 존재하므로 해당 과정은 VBA(Visual Basic for Applications)를 사용하여 구현 가능하다.

3.4 구현 결과

구현 정확성 검증을 위해 NIST의 CAVP(Cryptographic Algorithm Validation Program)에서 제공하는 데이터 세트를 사용하여 입력 메시지에 대한 해시값이 테스트 벡터와 일치함을 확인하였다[5].

본 연구의 구현물을 통해 [그림 7]과 같이 매개변수를 입력하였을 때 입력 메시지에 대한 해시값이 올바르게 출력되었는지 검증할 수 있으며, [그림 8]과 같이 SHA-3의 모든 라운드에 대한 Step mapping 과정의 중간값을 state 형태로 확인할 수 있다.

[그림 7] 구현 정확성 검증 결과(상: NIST 테스트 벡터, 하: 테스트 벡터 확인 결과)

[그림 8] 전체 라운드 구성(좌: 32비트 단위 표현, 우: 64비트 단위 표현)

본 연구에서 SHA3-224 알고리즘은 Microsoft® Excel® LTSC MSO (16.0.14334.20440) 64비트 환경에서 구현되었으며, 매크로 언어는 VBA 7.1을 사용하였다.

Ⅳ. 결론

본 논문은 엑셀 환경에서 SHA-3 알고리즘을 구현하는 방식을 소개한다. SHA-3의 5×5 Lane 형태의 state를 시각화함으로써 단순히 입력 메시지와 해시값만을 출력하는 것이 아닌, 라운드별 세부 함수를 거쳐 state가 갱신되는 과정을 확인할 수 있도록 구현하였다. 이를 통해 디버깅에 사용 가능한 중간값을 제공하여 구현 정확성을 검증하고, 복잡한 연산 과정을 가시화하여 알고리즘 구조 분석을 돕기 위한 도구로 활용할 수 있을 것으로 기대한다. 향후 연구에서는 SHA3-224 외에도 SHA3-256/384/512 및 확장 가능한 출력 함수인 SHAKE까지 구현 범위를 확대하여, SHA-3에 대한 통합 분석 도구를 구축할 예정이다.

ACKNOWLEDGMENT

이 논문은 서울시 산학연 협력사업 2025년도 양자 기술개발 지원사업 (QR250002, 양자내성암호 Kyber를 이용한 스마트카드 인증기술 개발)의 지원을 받아 수행된 연구임

참 고 문 헌

[1] NIST, "Recommendation for Applications Using Approved Hash Algorithms," NIST Special Publication 800-107 Revision 1, 2012.

[2] NIST, "SHA-3 standard: Permutation-Based Hash and Extendable-Output Functions," FIPS PUB 202, 2015.

[3] Chok, O.-S., Herath, J., and Herath, S. "Computer Security Learning Laboratory: Implementation of DES and AES Algorithms using Spreadsheets," International Journal of Effective Management, 1(2), 2004.

[4] Yoo, D., Bae, M., and Yeom, Y. "A Secure Score Report Implemented in a Spreadsheet without Privacy Concerns," International Journal of Security and Its Applications, 10(3), 2016.

[5] NIST, "CAVP Testing: Secure Hashing," Cryptographic Algorithm Validation Program, 2025, (https://csrc.nist.gov/Projects/Cryptographic-Algorithm-Validation-Program/Secure-Hashing#sha3vsha3vss).