NTT에서의 Montgomery 곱셈수 최적화

서은영, 김광식, 김영식*

조선대학교

* 교신저자: iamyskim@chosun.ac.kr

Optimizing the number of Montgomery Multiplication in Number Theoretic Transform

Eunyoung Seo, Kwang-Sik Kim, Young-Sik Kim

Chosun University

요 약

오늘날 Kyber와 Dilithium과 같은 표준 양자내성암호는 다항식 기반 알고리즘으로 Number Theoretic Transform(NTT)을 통해서 효율적 구현이 가능하다. NTT는 정수 계수를 사용하는 Fast Fourier Transform (FFT)의 일종으로, 이 논문에서는 NTT의 연산 효율화를 위해 Montgomery 곱셈의 수를 효율화하여 전체 연산 성능을 높일 방법을 제안한다. 이를 통해 기존 $k2^{k-1}$의 Montgomery 곱셈이 $(k+1)2^{k-2}$개의 Montogmery 곱셈과 $(k-1)2^{k-2}$개의 일반 곱셈으로 대체할 수 있었다.

Ⅰ. 서 론

최근 양자컴퓨터에 대한 저항성을 갖는 양자내성암호의 중요성이 커지고 있다. NIST에서는 2022년 7월에 4개의 표준양자내성암호를 선정하였다. 표준화된 양자내성암호 중 Kyber, Dilithium에서는 효율적 다항식 연산을 위해 Number Theoretic Transform (NTT)를 사용한다 [1],[2]. NTT는 정수 계수를 사용하는 Fast Fourier Transform (FFT) 라고 설명할 수 있으며[3], $f(x)=\Pi_{i=1}^{m} f_i(x)$ 일 때, 다항식 환 $Z_q[x]/f(x)$ 에 속하는 다항식을 $\Pi_{i=1}^{m}(Z_q[x]/f_i(x))$ 상의 원소로 대응시키는 ring homomorphism 이라고 정의할 수 있다 [4],[5].

이 논문에서는 NTT의 연산 효율화를 위해 Montgomery 곱셈의 수를 효율화하여 전체 연산 성능을 높이는 방법을 제안한다. 이를 통해 기존 $k2^{k-1}$의 Montgomery 곱셈이 $(k+1)2^{k-2}$개의 Montogmery 곱셈과 $(k-1)2^{k-2}$개의 일반 곱셈으로 대체할 수 있었다.

Ⅱ. 본론

다항식을 NTT를 이용하여 다른 환(ring)상의 원소들로 대응시키고, 그 환에서 곱셈 후, 그 결과를 Inverse NTT (INTT)를 이용하여 원래 환의 원소로 다시 대응을 시키면, 처음에 구하고자 했던 다항식들의 곱셈에 대한 결과를 얻을 수 있다. 이 과정에서 필요한 계수들의 곱셈의 횟수가 $O(nlog(n))$ 으로, NTT 를 사용하지 않을시 필요한 $O(n^2)$ 에 비해서 줄어들어 속도면에서 이득을 얻을 수 있다.

이런 NTT는 다항식 환의 원소들 간의 곱셈이 필요한 다양한 분야에 이용되지만, 특히, 양자내성암호에서 격자 이론을 바탕으로 만들어진 공개키 암호와 전자 서명에서 효율적인 계산을 위해 필수적으로 사용된다. 다항식 환에 속한 다항식들에 사칙 연산을 하거나 주어진 다항식의 NTT를 계산할 때, 최종 결과는 항상 modulo q 가 되어야 한다. 즉, q 보다 큰 정수에 대해서는 q 로 나눈 나머지 값을 계산해야 하는데, 두 정수의 합이나 차에 대해서는 입력 값에 상관없이 일정한 시간에 modulo q 한 값을 계산하는 것이 쉬우나, 곱셈의 결과에 대해서는 실수 연산을 사용하지 않고 일정한 시간에 modulo q 한 값을 얻는 것이 간단하지 않다. 이 때, 사용하는 방법이 Montgomery 곱셈이다.

$a \bmod q$ 와 $b \bmod q$ 의 곱셈을 결과 $ab \bmod q$ 를 얻고자 할 때, $a \bmod q$ 와 $b \bmod q$ 를 Montgomery 도메인의 값인 $aR \bmod q$ 와 $bR \bmod q$ 로 바꾸면, $abR \bmod q$ 의 값을 입력에 상관없이 일정한 시간에 실수 연산을 하지 않고 얻을 수 있게 해주는 것이 Montgomery 곱셈 방법이다. 여기서, R 은 2^k 형태의 값을 사용하여 적절한 때에 곱셈 연산을 bitwise shift 연산으로 대신하는 것이 가능하도록 하며, $\gcd(q,R)=1$ 과 $R \geq q$를 만족시켜야한다.

서로 소인 두 자연수 q 와 R 에 대해서 extended Euclidean 알고리즘을 적용하면 $R \cdot R' + q \cdot q' = 1$ 을 만족하는 정수 R' 와 q' 가 존재한다. 여기서 R' 와 q' 는 R 과 q 의 modulo q 와 modulo R 에 대한 곱셈에 대한 역원이 된다. 위와 같이 계산된 값은 항상 $-q$ 와 q 사이에 있기 때문에, 이런 방식으로 두 수의 곱셈에 대해 modulo q 된 값을 구할 수 있다.

여기서 R 이 2^k 형태이기 때문에 주어진 수에 대해서 modulo R 을 취하거나, R 을 나눈 값을 구하는 것은 bitwise and 연산과 shift 연산으로 간단하게 구현할 수 있다. 하지만, 이런 Montgomery 곱셈에 대해서도 소프트웨어로 구현을 하기 위해서는 세 번의 곱셈, 두 번의 bitwise and 연산, 두 번의 shift 연산, 세 번의 덧셈, 한 번의 -1 을 곱하는 연산이 필요하게 된다. 즉, 두 자연수의 곱셈에 대해서 modulo q 된 값을 얻기 위해서 추가적으로 비용을 지불해야한다.

Montgomery 곱셈에서 $R > q^2$ 이라는 조건을 만족시키는 R 을 선택할 경우, 한 번의 곱셈에서는 Montgomery 곱셈을 하지 않더라도, 두 번째 곱셈에서 Montgomery 곱셈을 통해 한 번에 modulo q 가 된 값을 얻을 수 있게 된다. 이것을 일반적으로 $R > q^t$ 의 조건으로 확장시키면 총 t 번의 곱셈을 할 때, $t-1$ 번의 곱셈에 대해서는 Montgomery 곱셈을 하지 않고, 마지막 한 번의 곱셈에만 Montgomery 곱셈을 적용하는 방식으로 확장시킬 수 있다.

다항식의 NTT를 계산시 가장 기본적인 구조는 Radix 2 기반 방식이다. 예를 들어, $Z_q[x]/f(x)$ 에서 $f(x)$ 가 8차인 다항식이라고 할 때, 이 다항식 환에 속하는 원소 $g(x)$가 있다고 하자. 이 다항식을 Radix 2인 NTT로 표현하면 다음과 같다.

$$g(x) = ((g_0 + g_4x^4) + x^2(g_2 + g_6x^4)) + x((g_1 + g_5x^4) + x^2(g_3 + g_7x^4)$$

이 때 첫 번째 단계에서 $A_0=(g_0+g_4x^4)$, $A_1=(g_1+g_5x^4)$, $A_2=(g_2+g_6x^4)$, $A_3=(g_3+g_7x^4)$ 을 계산한다. 두 번째 단계에서 $B_0=(A_0+A_2x^2)$, $B_1=(A_1+A_3x^2)$ 을 계산하고, 세 번째 단계에서 $C_0=(B_0+B_1x)$ 를 계산한다.

$g(x)$ 의 NTT 를 구한다는 것은 $g(\alpha^{2i+1})$, $0\le i\le 7$, 의 값을 구한다는 것이고, 여기서 $\alpha\in Z_q$ 이고, $\alpha^8=1$, $\alpha^4=-1$ 의 조건을 만족시킨다. 그러므로 x, x^2, x^4 이 될 수 있는 값들은 다음의 표로 나타낼 수 있다.

	α	α^3	α^5	α^7	$-\alpha$	$-\alpha^3$	$-\alpha^5$	$-\alpha^7$
x^4	α^4	$-\alpha^4$	α^4	$-\alpha^4$	α^4	$-\alpha^4$	α^4	$-\alpha^4$
x^2	α^2	α^6	$-\alpha^2$	$-\alpha^6$	α^2	α^6	$-\alpha^2$	$-\alpha^6$
x	α	α^3	α^5	α^7	$-\alpha$	$-\alpha^3$	$-\alpha^5$	$-\alpha^7$

표에 의하면 x 를 곱한다는 것은 $\alpha, \alpha^3, \alpha^5, \alpha^7$ 의 네 가지 값들을 곱한다는 것을 의미하므로 네 번의 곱셈을 해야한다. x^2의 경우 α^2, α^6 의 두 가지 값에 대응되므로 두 번의 곱셈을 해야한다. 마찬가지로 x^4의 경우는 α^4 의 값을 곱한다는 것을 의미하므로 한 번의 곱셈을 해야한다. A_0 를 구하는 과정에서 발생하는 곱셈인 g_4x^4 는 NTT 의 마지막까지 더 이상 곱이 없으므로 Montgomery 곱셈으로 계산을 해줘야 modulo q 가 된다. 하지만, A_2 을 구하는 과정에서 발생하는 g_6x^4 의 곱셈에서는 Montgomery 곱셈을 하지 않더라도 다음 단계에서 x^2 을 곱하는 과정에서 Montgomery 곱셈을 해주면 modulo q 가 된 값을 얻을 수 있다.

마찬가지로, A_1 을 구하는 과정에서 발생하는 g_5x^4 의 곱셈에서는 Montgomery 곱셈을 하지 않더라도 다음 단계에서 x 을 곱하는 과정에서 Montgomery 곱셈을 해주면 modulo q 가 된 값을 얻을 수 있다. 그리고, B_1 을 구하는 과정에서 발생하는 A_3x^2 의 곱셈에서는 Montgomery 곱셈을 하지 않더라도 다음 단계에서 x 를 곱하는 과정에서 Montgomery 곱셈을 해주면 modulo q 가 된 값을 얻을 수 있다. 여기서 x^2 을 곱한다는 것은 두 번의 곱셈을 내포하고 있다.

이렇게 하면, 전체 12번의 곱셈 중 4번의 곱셈에 대해서 Montgomery 곱셈을 하지 않아도 된다. Radix 2 의 NTT 를 계산할 때, 곱셈의 결과에 대해 더 이상의 곱셈이 필요하지 않으면 Montgomery 곱셈을 하고, 마지막 곱셈에서부터 그 전의 곱셈에 대해서는 Montgomery 곱셈을 하지 않으며, 그 전의 곱셈에 대해서는 Montgomery 곱셈을 하는 방식으로 번갈아가며 Montgomery 곱셈을 적용하여 Montgomery 곱셈의 횟수를 줄일 수 있다.

이 방식을 일반화시키면 $f(x)$ 의 차수가 2^k 인 경우 Radix 2 로 구현한 NTT 의 경우 $k\cdot 2^{k-1}$ 번의 Montgomery 곱셈을 하게 되는데, 이 중에 $(k-1)\cdot 2^{k-2}$ 번의 곱셈에 대해서는 Montgomery 곱셈 대신 일반 곱셈으로 대체할 수 있다. 이 때 k 개의 단계가 필요한데, 각 단계마다 2^{k-1} 번의 Montgomery 곱셈을 하고 있기 때문에 총 $k\cdot 2^{k-1}$ 번의 곱셈이 필요했는데, 여기서 제시한 방법에 따르면 마지막 단계를 제외한 $k-1$ 개의 단계에서 기존 곱셈의 반만 Montgomery 곱셈을 하므로 단계마다 2^{k-2} 개의 곱셈은 일반 곱셈으로 대치할 수 있다. 그리하여 총 $(k-1)\cdot 2^{k-2}$ 개의 곱셈은 일반 곱셈으로 할 수 있게 된다.

Ⅲ. 결론

본 논문에서는 Radix 2 기반의 NTT 알고리즘에서 Montgomery 곱셈을 더 단순한 일반 곱셈으로 수행하여 연산 성능을 개선하는 방법을 제안하였다. 이를 통해 기존 $k2^{k-1}$의 Montgomery 곱셈이 $(k+1)2^{k-2}$개의 Montogmery 곱셈과 $(k-1)2^{k-2}$개의 일반 곱셈으로 대체할 수 있었다. 일반 곱셈과 달리 곱셈 후 Reduction을 위한 추가 연산이 생략되어 SW 구현 속도를 높일 수 있게 되었다.

ACKNOWLEDGMENT

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government (MSIT) (No. NRF-2021R1A2C2011082)..

참 고 문 헌

[1] NIST PQC Standard Kyber, 2023 (https://pq-crystals.org).

[2] NIST PQC Standard Dilithium, 2023 (https://pq-crystals.org).

[3] Martin Fürer, "Faster Integer Multiplication", Proc. STOC 2007 Proceedings, pp. 57 - 66.

[4] Hwang, V., Liu, J., Seiler, G., Shi, X., Tsai, M.-H., Wang, B.-Y., & Yang, B.-Y. (2022). Verified NTT Multiplications for NISTPQC KEM Lattice Finalists: Kyber, SABER, and NTRU. IACR Trans CHES, 2022(4), 718 - 750.

[5] Becker, H., Hwang, V., Kannwischer, M. J., Yang, B.-Y., & Yang, S.-Y. (2021). Neon NTT: Faster Dilithium, Kyber, and Saber on Cortex-A72 and Apple M1. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2022(1), 221 - 244.