3-Way Toom-Cook 을 이용한 7680/15360비트 정수 곱셈에 관한 연구

김민지*, 김영효*, 전창열**, 김동찬***

국민대학교(*학부생, **대학원생, ***교수)

A Study on 7680/15360-bit Integer Multiplication Using 3-Way Toom-Cook

Minji Kim*, Young Hyo Kim*, Chang Yeol Jeon**, Dong-Chan Kim***

Kookmin University(*UnderGraduate student, **Graduate student, ***Professor)

요 약

RSA는 가장 대표적인 인수분해 기반 공개키 암호이다. 192/256비트의 보안 강도를 제공할 때, RSA의 키 길이는 7680/15360비트이다. 이때, RSA 암/복호화에서 곱셈 연산의 횟수는 키 길이의 두 배이다. RSA의 키 길이가 길어질수록 암/복호 연산 속도가 느려지므로 빠른 곱셈 알고리듬이 필요하다. 본 논문에서는 곱셈 알고리듬인 3-Way Toom-Cook 알고리듬을 소개하고, 이를 FLINT 라이브러리로 구현한 결과를 제시한다.

I. 서론

RSA는 가장 대표적인 인수분해 기반 공개키 암호이다. 192/256비트의 보안 강도를 가지는 RSA의 키 길이는 7680/15360비트이다[1]. 이는 RSA-7680, RSA-15360의 키 길이에 해당한다. RSA의 암/복호화 과정에서는 모듈러 지수 승을 사용하며, 모듈러 지수 승은 지수가 n비트일 때 $2n$번의 곱셈 연산을 수행한다. 암호화는 공개키, 복호화는 비밀키 제곱 연산을 사용한다. 일반적으로 RSA에서는 17비트 공개키 $pk(=2^{16}+1)$를 사용하기 때문에 암호화의 제곱 연산 시 34회의 곱셈 연산을 수행한다. 하지만 비밀키의 길이는 RSA의 키 길이와 동일하므로 복호화에서 제곱 연산 시 15360회, 30720회의 곱셈 연산이 필요하다. 따라서 키 길이가 길어질수록 곱셈 연산의 속도가 느려지고 횟수도 많아진다. 전체 RSA 암/복호 연산의 속도가 느려지므로, 빠른 곱셈 알고리듬이 필요하다.

본 논문에서는 곱셈 알고리듬인 3-Way Toom-Cook 알고리듬을 소개하고, 이 알고리듬을 구현하여 두 가지 파라미터의 세 가지 상황에 대해 연산 속도를 비교한다. 구현 시 FLINT 라이브러리를 이용한다[3]. 입력값은 두 정수가 7680비트인 경우와 15360비트인 경우로 설정하며, 세 가지 상황은 다음과 같다.

(상황 1) 1회
(상황 2) 15360회
(상황 3) 30720회

(상황 2)는 RSA-7680의 복호화 시 곱셈 횟수이고, (상황 3)은 RSA-15360의 복호화 시 곱셈 횟수이다.

결과적으로 1회 곱셈 연산을 수행하는데 두 정수가 7680비트인 경우 약 0.006 밀리초, 15360비트인 경우 약 0.011 밀리초가 소요되었다. 두 정수가 7680비트인 경우는 15360비트인 경우보다 최대 1.86배 빨랐다.

논문의 구성은 다음과 같다. II절에서는 본 논문에서 사용하는 기호를 정의한다. III절에서는 3-Way Toom-Cook 알고리듬을 소개한다. IV절에서는 입력값을 7680, 15360비트 두 정수로 설정하여 세 가지 상황에 대해 3-Way Toom-Cook 알고리듬의 연산 속도 측정 결과를 나타낸다.

II. 기호

본 논문에서는 다음의 기호를 사용한다.

- AB 정수 A, B의 곱셈
- $a_1 \parallel a_0$ a_1와 a_0의 연접
- $a_{[x:y]}$ $a_{y-1} \parallel a_{y-2} \parallel \cdots \parallel a_{x+1} \parallel a_x$. 즉, a의 x이상 y미만 비트 연접
- $\mathcal{M}^T$ 행렬 $\mathcal{M}$의 전치 행렬
- $len(N)$ 정수 N의 워드 길이를 반환하는 함수

- $\mathbb{N}$ 자연수 집합
- $\lceil n \rceil$ n보다 같거나 큰 수 중 가장 작은 정수
- $\lfloor n \rfloor$ n보다 같거나 작은 수 중 가장 큰 정수
- 0_l l비트가 모두 0으로 채워진 비트열

III. 3-Way Toom-Cook 알고리듬

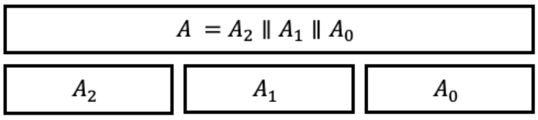
Toom-Cook 알고리듬은 n비트 두 정수를 $\lceil n/k \rceil$비트씩 분할하여 연산하는 곱셈 알고리듬이다[2,5]. 이때 k는 설정할 수 있으며, k가 클수록 연산 횟수는 증가하고 계산 복잡도는 감소한다[4]. 본 논문에서는 k를 3으로 설정하였다.

3-Way Toom-Cook 알고리듬은 크게 5단계로 진행한다. 분할(splitting) 단계와 평가(evaluation), 재귀적 곱셈(recursive multiplication) 단계, 보간(interpolation), 재구성(recomposition) 단계이다.

1. 분할(splitting) 단계

분할 단계는 n비트 두 정수를 세 부분으로 분할하고 분할한 수는 다음과 같이 표현할 수 있다.

$$\begin{cases} A = A_2 2^{2\lceil n/3 \rceil} + A_1 2^{\lceil n/3 \rceil} + A_0 = A_2 \parallel A_1 \parallel A_0, \\ B = B_2 2^{2\lceil n/3 \rceil} + B_1 2^{\lceil n/3 \rceil} + B_0 = B_2 \parallel B_1 \parallel B_0. \end{cases}$$



2. 평가(evaluation) 단계

평가 단계에서는 5개의 점에 대해 정의한다.

$$p(i) \coloneqq \begin{cases} A_2 i^2 + A_1 i + A_0, & i = -2, -1, 0, 1, \\ A_2, & i = \infty. \end{cases},$$

$$q(i) \coloneqq \begin{cases} B_2 i^2 + B_1 i + B_0, & i = -2, -1, 0, 1, \\ B_2, & i = \infty. \end{cases}.$$

3. 재귀적 곱셈(recursive multiplication) 단계

재귀적 곱셈 단계에서는 $p(i)$와 $q(i)$의 곱셈을 수행한다. 이때 $p(i)$와 $q(i)$의 곱셈은 일반 곱셈이 아닌 3-Way Toom-Cook 을 재귀적으로 호출한다.

$$T(i) = p(i) \times q(i), \qquad for\ i = -2, -1, 0, 1, \infty.$$

4. 보간(interpolation) 단계

보간 단계는 5×5 보간 행렬 M을 곱하는 행렬 곱 연산이며, 연산 결과 $r_i (i = 0,1,2,3,4)$는 다음과 같다.

$$\begin{pmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \\ r_4 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{3} & -1 & \frac{1}{6} & -2 \\ -1 & \frac{1}{2} & \frac{1}{2} & 0 & -1 \\ -\frac{1}{2} & \frac{1}{6} & \frac{1}{2} & -\frac{1}{6} & 2 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}}_{\mathcal{M}} \begin{pmatrix} T(0) \\ T(1) \\ T(-1) \\ T(-2) \\ T(\infty) \end{pmatrix}.$$

$$\begin{aligned} r_0 &= A_0 B_0, \\ r_1 &= A_1 B_0 + A_0 B_1, \\ r_2 &= A_2 B_0 + A_1 B_1 + A_0 B_2, \\ r_3 &= A_2 B_1 + A_1 B_2, \\ r_4 &= A_2 B_2. \end{aligned}$$

5. 재구성(recomposition) 단계

재구성 단계는 비트 이동 연산을 하여 더하는 단계이며, 이 결과는 정수 A, B의 곱셈 결과와 동일하다.

$$\begin{aligned} & r_4 2^{4\lceil n/3 \rceil} + r_3 2^{3\lceil n/3 \rceil} + r_2 2^{2\lceil n/3 \rceil} + r_1 2^{\lceil n/3 \rceil} + r_0 \\ &= A_2 B_2 \cdot 2^{4\lceil n/3 \rceil} + (A_2 B_1 + A_1 B_2) 2^{3\lceil n/3 \rceil} \\ &\qquad + (A_2 B_0 + A_1 B_1 + A_0 B_2) 2^{2\lceil n/3 \rceil} \\ &\qquad + (A_1 B_0 + A_0 B_1) 2^{\lceil n/3 \rceil} + A_0 B_0 \\ &= \left(A_2 2^{2\lceil n/3 \rceil} + A_1 2^{\lceil n/3 \rceil} + A_0\right)\left(B_2 2^{2\lceil n/3 \rceil} + B_1 2^{\lceil n/3 \rceil} + B_0\right) \\ &= AB. \end{aligned}$$

알고리듬 1은 3-Way Toom-Cook 알고리듬의 유사부호이다.

알고리듬 1. 3-Way Toom-Cook 알고리듬 $MUL^{3ToomCook}(A, B)$
입력: $flag$, n비트 정수 A, B
출력: AB
1. procedure $MUL^{3ToomCook}(A, B)$
2. If $flag \geq min(len(A), len(B))$ then
3. return AB $\triangleright$ AB:일반곱셈
4. end if
5. $l \leftarrow max(len(A), len(B))$ / 3
6. $A_2,\ A_1,\ A_0 \leftarrow A_{[n:2l]},\ A_{[2l:l]},\ A_{[l:0]}$
7. $B_2,\ B_1,\ B_0 \leftarrow B_{[n:2l]},\ B_{[2l:l]},\ B_{[l:0]}$
8. $cnt \leftarrow 0$
9. $T \leftarrow [\,]$
10. for i in $[-2, -1, 0, 1, \infty]$ do
11. $T[cnt] \leftarrow MUL^{3ToomCook}(p(i), q(i))$
12. $cnt \leftarrow cnt + 1$
13 $r \leftarrow T \cdot \mathcal{M}^T$
14. $R_{1[6l:4l]}, R_{1[4l:2l]}, R_{1[2l:0]} \leftarrow r[4], r[2], r[0]$
15. $R_{0[5l:3l]}, R_{0[3l:l]}, R_{0[l:0]} \leftarrow r[3], r[1], 0_l$
16. $R \leftarrow R_1 + R_0$
17. return R
18. end procedure

1워드를 32비트로 설정하여 두 정수 A, B 의 워드길이가 둘 중 하나라도 $flag$보다 작아지는 경우, $MUL^{3ToomCook}(A, B)$ 함수의 재귀 호출을 멈추고 일반 곱셈 AB를 진행한다. 본 논문에서는 $flag$를 10으로 설정하여, 입력값인 두 정수 A, B가 10워드 보다 작은 경우 일반 곱셈 연산 결과를 반환하였다. Line 5-7은 분할 단계이다. Line 8-12에서는 평가와 재귀적 곱셈 단계를 동시에 수행하며, $p(i)$와 $q(i)$의 곱셈을 열벡터 T 에 저장한다. 이때 $p(i)$와 $q(i)$는 위에서 정의한 함수이다. Line 13에서는 열벡터 T와 보간 행렬 $\mathcal{M}$의 전치 행렬을 곱한다. Line 14-16은 재구성 단계이다.

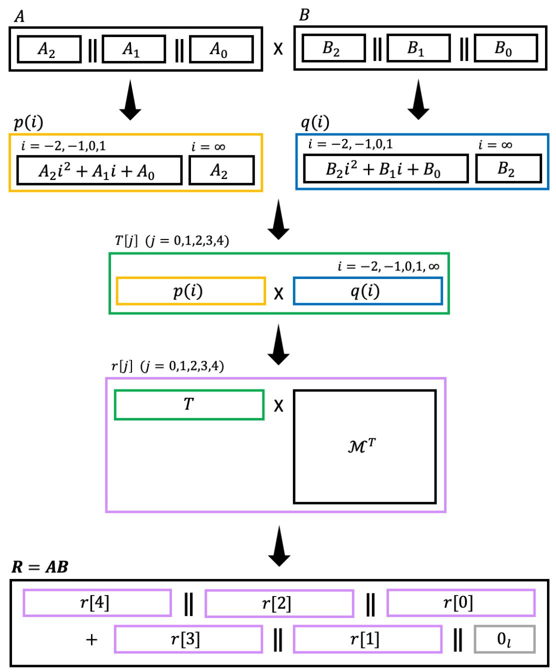
n비트 정수 A, B 곱셈 3-Way Toom-Cook 알고리듬의 계산복잡도는 다음과 같이 유도할 수 있다.

3-Way Toom-Cook 알고리듬의 n비트 두 정수 곱셈 비트 연산 횟수를 $T(n)$로 표기한다. 3-Way Toom-Cook 알고리듬은 n비트 두 정수를 세 부분으로 분할하여 재귀적 곱셈 단계에서 5번의 $MUL^{3ToomCook}$곱셈 연산을 하므로 $5 \times T\left(\frac{n}{3}\right)$로 표현할 수 있다. 5번의 재귀적 곱셈 이외에도 평가 단계의 덧셈과 뺄셈 연산, 보간 단계의 일반 곱셈 연산, 재구성 단계의 비트 이동과 덧셈 연산은 어떤 자연수 c_1에 대한 $c_1 n$으로 표

현할 수 있다. 이 과정을 반복하여 수행하게 되면, $\lfloor log_3 n \rfloor$번 반복하였을 때 $T\left(\frac{n}{3^{\lfloor log_3 n \rfloor}}\right)$가 $T(1) = 1$에 근사한다. 이 근사식의 계산 복잡도는 3-Way Toom-Cook 알고리듬의 계산복잡도와 같다. 따라서 n비트 정수 A, B 곱셈 3-Way Toom-Cook 알고리듬의 계산복잡도는 $\mathcal{O}(n^{\log_3 5})$이다.

$$
\begin{aligned}
T(n) &= 5 \times T\left(\frac{n}{3}\right) + c_1 n \text{ for some } c_1 \in \mathbb{N} \\
&= 5^2 \times T\left(\frac{n}{3^2}\right) + c_2 n \text{ for some } c_2 \in \mathbb{N} \\
&= 5^3 \times T\left(\frac{n}{3^3}\right) + c_3 n \text{ for some } c_3 \in \mathbb{N} \\
&= \cdots \\
&= 5^{\lfloor log_3 n \rfloor} \times T\left(\frac{n}{3^{\lfloor log_3 n \rfloor}}\right) + c_{\lfloor log_3 n \rfloor} n \\
&\qquad\qquad \text{for some } c_{\lfloor log_3 n \rfloor} \in \mathbb{N} \\
&\approx n^{log_3 5} \times T(1) + c_{log_3 n} n \text{ for some } c_{log_3 n} \in \mathbb{N} \\
&= \mathcal{O}(n^{log_3 5}).
\end{aligned}
$$

알고리듬 1의 흐름도는 다음과 같다.



IV. 3-Way Toom-Cook의 연산 속도

본 절에서는 알고리듬 1의 입력으로 두 정수가 7680비트일 때와 15360비트일 때 세 가지 상황에 대한 연산 속도를 측정한다. 구현 환경은 다음과 같다.

하드웨어	MacBook Air. Apple M2. 8GB RAM.
컴파일러	gcc 13.1.6 (-O2)
정수 연산 라이브러리	FLINT 2.9.0

RSA의 키 길이를 파라미터로 설정하여 세 가지 상황에 대해 연산 속도를 비교하였다. 192비트의 보안 강도를 가지는 RSA의 키 길이 경우 7680비트, 256비트의 경우 15360비트의 키에 대해 곱셈 연산을 수행한다. (상황 2)는 RSA-7680의 복호화 시 곱셈 횟수인 15360회이고, (상황 3)은 RSA-15360의 복호화 시 곱셈 횟수인 30720회이다.

[표 1]은 두 가지 파라미터의 세 가지 상황에 대한 3-Way Toom-Cook 연산 시간을 측정한 결과이며, 시간 단위는 밀리초(ms)이다.

[표 1] 3-Way Toom-Cook 알고리듬 연산 시간

(단위: ms)

입력 정수의 비트 길이	7,680	15,360
(상황 1)	0.006	0.011
(상황 2)	104.084	182.981
(상황 3)	184.456	344.708

측정 결과, (상황 1)인 곱셈 연산 1회 수행에서는 입력값 7680/15360비트에 대해 0.006/0.011밀리초가 소요되었다. 입력 파라미터가 7680비트인 경우, 15360비트인 경우와 비교하여 최대 1.86배 빨랐다.

V. 결론

본 논문에서는 곱셈 알고리듬인 3-Way Toom-Cook을 소개하고, 이 알고리듬을 구현하여 두 가지 입력값에 대해 연산 속도를 측정하였다. 그 결과 RSA-15360의 복호화 곱셈 연산 횟수 기준, 입력값이 7680비트일 때 15360비트보다 최대 1.86배 빨랐다. 이를 사용하여 RSA의 복호화를 고속화할 수 있다. 추후 곱셈 알고리듬인 Schönhage-Strassen 알고리듬에 관한 연구를 진행할 계획이다.

ACKNOWLEDGMENT

이 성과는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No.NRF-2021R1F1A1062305).

[참고문헌]

[1] Barker, Elaine, and Quynh Dang. *"Nist special publication 800-57 part 1, revision 4."* NIST, Tech. Rep 16 (2016).

[2] S. A. Cook and S. O. Aanderaa, *"On the minimum computation time of functions,"* Transaction of the American Mathematical Society, vol. 142, pp. 291-314, Aug. 1969.

[3] W. Hart, F. Johansson and S. Pancratz. FLINT: Fast Library for Number Theory, 2013. version 2.4.0, http://flintlib.org.

[4] J. M. B. Mera, A. Karmakar, and I. Verbauwhed, *"Time-memory trade-off in Toom-Cook multiplication: an application to module-lattice based cryptography,"* International Association for Cryptologic Research (IACR) Transactions on Cryptographic Hardware and Embedded Systems, vol. 2020, no. 2, pp. 222-244, Mar. 2020. DOI: 10.13154/tches. v2020.i2.222-244.

[5] A. L. Toom, *"The complexity of a scheme of functional elements realizing the multiplication of integers,"* Soviet Math. Doklady, vol. 3, no. 4, pp. 714-716, 1963.