Barrett Reduction과 Montgomery Reduction을 기반으로 한 RSA 비밀키 연산 고속화에 관한 연구

김민지, 김동찬
국민대학교 정보보안암호수학과
{minji0022, dckim}@kookmin.ac.kr

A Study on the Secret Modular Exponentiation for RSA Using Barrett Reduction and Montgomery Reduction

Minji Kim, Dong-Chan Kim
Kookmin University

요 약

RSA는 인수분해 기반 트랩도어로 공개키 암호와 전자서명으로 활용 가능한 암호 프리미티브이다. 현재는 112비트 안전성을 가지는 RSA-2048을 사용하지만 한국인터넷진흥원의 권고에 따라 2030년 이후에는 128비트 이상의 보안 강도를 가지는 RSA를 사용해야 한다. RSA는 키 크기가 클수록 비밀키 연산인 모듈러 지수승 속도가 느려지기 때문에 이를 실용 수준으로 구현하는 것이 필요하다. 본 논문에서는 Barrett reduction과 Montgomery reduction 기반으로 모듈러 지수승 알고리듬 중 하나인 Multiply-and-Squaring를 FLINT 라이브러리로 구현한 결과를 제시한다.

Ⅰ. 서 론

RSA는 인수분해 기반 트랩도어로 공개키 암호와 전자서명으로 활용 가능한 암호 프리미티브이다. RSA의 공개키/비밀키 연산은 지수가 각각 공개키, 비밀키인 모듈러 지수승 연산 $x^k \bmod N$이다. 이때 N은 2048비트 이상의 정수로, RSA-XXXX에서 XXXX는 N의 비트 크기를 말한다.

모듈러 지수승은 지수가 클수록 연산량이 많아진다. RSA는 일반적으로 키 관리와 서비스 효율성을 위해 짧은 크기의 공개키를 사용한다(예, $2^{16}+1$). 하지만 비밀키는 대체로 N에 가까운 값이기 때문에 비밀키 연산 속도가 공개키 연산 속도보다 현저히 느리다.

현재 KCMVP 검증대상 암호에는 112비트 보안강도의 RSA-2048과 128비트 보안 강도의 RSA-3072만 포함되어 있다. 그 이상은 키 생성과 비밀키 연산 성능이 실용 수준에 미치지 못하여 배제하였다. 하지만 한국인터넷진흥원이 2030년부터 128비트 이상의 보안 강도를 가지는 알고리듬 사용을 권고하였기 때문에 RSA-3072 이상으로의 전환을 준비해야 한다[4].

본 논문에서는 모듈러 지수승 알고리듬인 Multiply-and-Squaring 알고리듬에서 나눗셈 연산을 곱셈 연산으로 대체하는 Barrett reduction, Montgomery reduction을 적용하여 구현한 결과를 소개한다. 사용한 라이브러리는 FLINT이며 측정 대상 RSA 파라미터는 RSA-3072/4096/7680/15360이다.

Ⅱ. Multiply-and-Squaring 알고리듬

알고리듬 1은 Multiply-and-Squaring 알고리듬의 유사부호를 보여준다.

알고리듬 1. Multiply-and-Squaring	
입력:	$m,\ d=d_{l-1}d_{l-2}\dots d_0$ and N
출력:	$m^d \bmod N$
1.	$t_0, t_1 \leftarrow 1, m$
2.	**for** $i \leftarrow l-1$ downto 0 **do**
3.	$\quad t_{1-d_i} \leftarrow t_0 \times t_1 \bmod N$
4.	$\quad t_{d_i} \leftarrow t_{d_i}^2 \bmod N$
5.	**end for**
6.	**return** t_0

Multiply-and-Squaring 알고리듬은 지수 d의 비트 길이만큼 모듈러 곱셈과 모듈러 제곱 연산을 수행한다. 하지만 더 적은 연산 횟수로 모듈러 지수승 연산을 수행하는 알고리듬도 존재한다. Right-to-Left 알고리듬은 지수 d의 해밍무게, 즉 d의 0이 아닌 비트 수만큼의 곱셈 연산을, d의 비트 길이만큼의 제곱 연산을 수행한다. Multiply-and-Squaring 알고리듬보다 연산 횟수가 적으므로 빠르지만, RSA 비밀키 d의 해밍무게에 따라 연산 속도가 달라지기 때문에 시간차 공격과 같은 부채널 분석을 통해 키가 노출될 수 있다. 그래서 RSA는 일반적으로 부채널 분석으로부터 안전하기 위해, 지수 d의 해밍무게와 관련없이 균등한 모듈러 곱셈 연산을 수행하는 Multiply-and-Squaring 알고리듬으로 구현한다.

Ⅲ. 나눗셈 연산

Multiply-and-Sqaring 알고리듬은 모듈러 곱셈과 모듈러 제곱 연산을 사용한다. 이때 수행되는 모듈러 연산은 나눗셈 연산을 사용하지만, Barrett reduction과 Montgomery reduction은 N이 고정된 경우 사전계산과 근사치를 이용해 이를 곱셈 연산으로 대체한다. 곱셈과 나눗셈 연산의 계산복잡도는 둘 다 $\mathcal{O}(n^2)$이지만, $\mathcal{O}(n^{1.46})$의 계산복잡도를 가지는 3-Way Toom-Cook 등 고속 곱셈 알고리듬을 활용하면 나눗셈 연산을 사용하는 모듈러 연산보다 빠르게 계산할 수 있다.

3.1 Barrett Reduction

Barrett reduction은 다음 정리를 이용한 음이 아닌 $2n$ 이하 워드 정수 A와 n워드 정수 N에 대해 $A \bmod N$을 고속 계산하는 알고리듬이다[1]. 이때, 본 논문의 1워드는 64비트 정수 ($W=2^{64}$)이며, $\lfloor r \rfloor$은 r보다 작거나 같은 수 중 가장 큰 정수를 의미한다. $d \in [a,b)$는 $a \le d < b$를 만족하는 정수를 의미한다.

정리 1. $A \in [0, W^{2n})$를 $N \in [W^{n-1}, W^n)$으로 나누었을 때, 몫 Q는 다음을 만족한다.

$$Q \in \{\hat{Q}, \hat{Q}+1, \hat{Q}+2\},\ (\hat{Q} := \left\lfloor \frac{\left\lfloor \frac{A}{W^{n-1}} \right\rfloor \left\lfloor \frac{W^{2n}}{N} \right\rfloor}{W^{n+1}} \right\rfloor).$$

정리 1로부터 몫이 $\hat{Q}, \hat{Q}+1, \hat{Q}+2$일때, 얻게 되는 나머지 R은 각각 $A-N\hat{Q}, A-N\hat{Q}-N, A-N\hat{Q}-2N$이다. 따라서 $A-N\hat{Q}$가 N보다 작은 값이 될 때까지 N을 빼주면 나머지 R을 얻게 된다. $\hat{Q}$을 계산할 때, N이 고정값이기 때문에 $T=\left\lfloor \frac{W^{2n}}{N} \right\rfloor$를 사전계산할 수 있다. W^{n-1}과 W^{n+1}로 나누는 연산은 워드 이동으로 처리할 수 있으므로, 1회 곱셈으로 $\hat{Q}$을 계산할 수 있다. 알고리듬 2는 Barrett reduction 알고리듬의 유사부호이다.

알고리듬 2. Barrett Reduction
입력: $A \in [0, W^{2n})$, $N \in [W^{n-1}, W^n)$ and $T = \left\lfloor \frac{W^{2n}}{N} \right\rfloor$
출력: $R = A \bmod N \in [0, W^n)$
1. $\hat{Q} \leftarrow \left\lfloor \left\lfloor \frac{A}{W^{n-1}} \right\rfloor \cdot T \cdot \frac{1}{W^{n+1}} \right\rfloor$
2. $R \leftarrow A - N\hat{Q}$
3. **while** $R \geq N$ **do**
4. $\quad R \leftarrow R - N$
5. **end while**
6. **return** R

3.2 Montgomery Reduction

Montgomery reduction은 N보다 큰 정수 M을 선택하여 $A \bmod N$이 아닌 $xM' \bmod N$을 고속 계산하는 알고리듬이다[3]. 정수 M은 다음을 만족한다.

(1) W^s 형태 (s워드 정수)
(2) N과 서로소, 즉, $\gcd(N, M) = 1$.

조건 (2) $\gcd(N, M) = 1$이므로 확장 유클리드 알고리듬에 의해 $MX + NY = 1$을 만족하는 두 정수 X와 Y를 계산할 수 있다. 상기 식을 각각 N과 M에 대해 모듈러 연산한 결과는 다음과 같다.

$$MX + NY \equiv 1 \pmod{N} \rightarrow MX \equiv 1 \pmod{N},$$
$$MX + NY \equiv 1 \pmod{M} \rightarrow NY \equiv 1 \pmod{M}.$$

다음과 같이 두 수 M'과 N'을 정의한다.

$$M' := X \bmod N, \qquad N' := Y \bmod M.$$

M, M', N, N'은 다음 성질을 가진다.

$$MM' \equiv 1 \pmod{N}, \qquad 0 < M' < N,$$
$$NN' \equiv 1 \pmod{M}, \qquad 0 < N' < M.$$

N과 M에 대한 함수 $MontRed_{N,M}: [0, NM) \rightarrow [0, N)$을 다음과 같이 정의한다.

$$MontRed_{N,M}(x) := xM' \bmod N.$$

M'을 사전계산하면 함수 $MontRed_{N,M}(x)$ 계산 시 곱셈 연산 1회와 나눗셈 연산 1회가 발생한다. 이때, N'은 $-M < -N' < 0$이므로 $\widehat{N} = (-N') \bmod M = M - N'$을 사전계산하면 2회 곱셈으로 $xM' \bmod N$을 계산할 수 있다. 알고리듬 3은 Montgomery reduction의 유사부호이다.

알고리듬 3. Montgomery Reduction
입력: $x \in [0, NM)$, N, $(N <)M = W^s$ with $\gcd(N, M) = 1$ and $\widehat{N} = (-N') \bmod M$
출력: $MontRed_{N,M}(x) = xM' \bmod N$
1. $r \leftarrow (x \bmod M)\widehat{N} \bmod M$
2. $t \leftarrow (x + rN)/M$
3. **if** $t \geq N$ **then**
4. $\quad t \leftarrow t - N$
5. **end if**
6. **return** t

(line 1)의 r은 다음 성질을 가진다.

$$rN \equiv (x \bmod M)\widehat{N}N \equiv x\widehat{N}N \equiv x(-N')N \equiv -x \pmod{M}$$
$$\Rightarrow M \mid x + rN.$$

$x + rN$이 M의 배수이므로 (line 2)의 t는 음이 아닌 정수이다. $tM = x + rN \equiv x \pmod{N}$ 이고, $\gcd(N, M) = 1$ 이기 때문에 반환값 $t \equiv xM' \pmod{N}$이다. (line 2)로부터 $r < M$이기 때문에 다음이 성립한다.

$$0 \leq tM = x + rN < NM + NM = 2NM \Rightarrow 0 \leq t < 2N.$$

따라서 계산값 t가 N보다 크거나 같으면, $t - N$을 계산하여 $xM' \bmod N$을 얻게된다. 이때, (line 1)의 $\bmod M$ 연산과 (line 2)의 $/M$ 연산은 M이 t 워드 W^t 형태이기 때문에 비트 이동 및 절삭 연산으로 처리할 수 있다. 따라서 (line 2)와 (line 3)의 2회 곱셈으로 $MontRed_{N,M}(x)$를 계산한다.

지수승 알고리듬에 Montgomery reduction을 결합하기 위해, 함수 $\varphi: \mathbb{Z} \rightarrow [0, N)$을 다음과 같이 정의한다. $\mathbb{Z}$는 정수 집합이다.

$$\varphi(x) := xM \bmod N.$$

정의에 의해 $MontRed_{N,M}(\varphi(x)) = x \bmod N$이 성립하므로, 함수 φ에 대한 곱셈 $*$을 $\varphi(x) * \varphi(y) := \varphi(xy)$로 정의하면 다음 관계가 유도된다.

$$\varphi(x) * \varphi(y) = MontRed_{N,M}(\varphi(x)\varphi(y)) \in [0, N).$$

정의로부터 $\varphi(x)^d = \underbrace{\varphi(x) * \cdots * \varphi(x)}_{d \text{ times}} = \varphi(x^d) \in [0, N)$임을 알 수 있다. 따라서 $\varphi(x)^d$을 계산한 후 결과값을 함수 $MontRed_{N,M}$의 입력으로 넣으면 $x^d \bmod N$이 반환된다.

$$MontRed_{N,M}(\varphi(x)^d) \equiv \varphi(x)^d M' \equiv \varphi(x^d)M' \equiv x^d \pmod{N}.$$

IV. 구현 및 속도 성능 비교

본 절에서는 RSA-3072/4096/7680/15360 입력값에 대해 Mutiply-and-Squaring 알고리듬과 Barrett reduction, Montgomery reduction 기반 Mutiply-and-Squaring 알고리듬의 비밀키 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 [2]

메시지 m은 $0 < m < N$의 임의의 양의 정수이며, 비밀키 d와 N은 RSA 키 길이와 동일하다. Reduction 알고리듬의 워드 W는 2^{64}이며, 각 알고리듬에 사용하는 세부 함수는 FLINT 지원 함수를 사용하였다. [표 1]은 네 가지 RSA 파라미터에 대한 모듈러 지수승 1회 연산 시간을 측정한 결과이며, 시간 단위는 밀리초(ms)이다.

[표 1] 모듈러 지수승 연산 속도 (단위: 밀리초/1회)

알고리듬	RSA-3072	RSA-4096	RSA-7680	RSA-15360
M&S	12.085	26.848	137.161	847.443
Barr. 기반 M&S	13.452	26.782	143.542	837.287
Mont. 기반 M&S	12.868	26.828	142.351	832.378

측정 결과, RSA-3072/7680 입력값에 대해 Multiply-and-Squaring 알고리듬이 가장 빨랐고, RSA-4096/15360 파라미터에 대해 Montgomery reduction 기반 Multiply-and-Squaring 알고리듬이 가장 빨랐다. 최대 차이는 RSA-15360 파라미터에 대해 13밀리초이며, 네 가지 파라미터에 대해 세 알고리듬 모두 큰 속도 차이가 나지 않았다.

V. 결론

본 논문에서는 모듈러 지수승 알고리듬인 Multiply-and-Squaring 알고리듬과 Barrett reduction, Montgomery reduction 알고리듬을 소개하고, 이 알고리듬들을 구현하여 RSA 파라미터에 대한 비밀키 연산 속도를 비교하였다. 그 결과 RSA-3072/4096/7680/15360 파라미터 모두 Reduction 알고리듬을 적용하여 고속화한 알고리듬을 포함하여도 큰 속도 차이는 나지 않았다. 이를 활용하여 2030년 이후 사용될 RSA 알고리듬 별 실용 수준을 예상할 수 있다.

ACKNOWLEDGMENT

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

참 고 문 헌

[1] Barrett, P. "Implementing the rivest shamir and adleman public key encryption algorithm on a standard digital signal processor". *Advances in Cryptology — CRYPTO' 86*. p. 311-323. Springer Berlin Heidelberg, (1987).
[2] FLINT: Fast Library for Number Theory. https://www.flintlib.org/
[3] Montgomery, P. "Modular multiplication without trial division". *Mathematics of Computation*. pp. 519-521. American Mathematical Society (April 1985).
[4] 한국인터넷진흥원 "암호 알고리즘 및 키 길이 이용 안내서" (2018).