Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.26, NO.6, Dec. 2016 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2016.26.6.1401

비트 일부로부터 Multi-Prime RSA와 Prime Power RSA의 개인키를 복구하는 알고리즘

백 유 진†‡
우석대학교

Key Recovery Algorithm from Randomly-Given Bits of Multi-Prime RSA and Prime Power RSA

Yoo-Jin Baek†‡
Woosuk University

요 약

Multi-Prime RSA와 Prime Power RSA는 변형 RSA 시스템의 일종이며, 이 중에서 Multi-Prime RSA는 서로 다른 $r(r>2)$개의 소수 $p_1, p_2, \ldots, p_r$에 대하여 $N=p_1p_2\cdots p_r$을, Prime Power RSA는 서로 다른 소수 p, q와 양의 정수 $r(>1)$에 대하여 $N=p^rq$를 각각 모듈러스로 사용한다. 본 논문에서는 Heninger와 Shacham에 의해 제안된 방법을 사용하여 이 시스템들에 대한 안전성을 분석하며 구체적으로, 만약 $p_1, p_2, \ldots, p_r$의 전체 비트 중 $2-2^{1/r}$의 비율에 해당하는 비트가 랜덤하게 주어지면 $N=p_1p_2\cdots p_r$이 다항식 시간 안에 소인수분해될 수 있음을, 그리고 p, q의 전체 비트 중 $2-\sqrt{2}$의 비율에 해당하는 비트가 랜덤하게 주어지면 $N=p^rq$가 다항식 시간 안에 소인수분해될 수 있음을 각각 보인다. 또한 $N=p_1p_2p_3$, $N=p^2q$, $N=p^3q$에 적용한 실험 결과를 통해 본 논문의 결과를 검증한다.

ABSTRACT

The Multi-Prime RSA and the Prime Power RSA are the variants of the RSA cryptosystem, where the Multi-Prime RSA uses the modulus $N=p_1p_2\cdots p_r$ for distinct primes $p_1, p_2, \ldots, p_r (r>2)$ and the Prime Power RSA uses the modulus $N=p^rq$ for two distinct primes p, q and a positive integer $r(>1)$. This paper analyzes the security of these systems by using the technique given by Heninger and Shacham. More specifically, this paper shows that if the $2-2^{1/r}$ random portion of bits of $p_1, p_2, \ldots, p_r$ is given, then $N=p_1p_2\cdots p_r$ can be factorized in the expected polynomial time and if the $2-\sqrt{2}$ random fraction of bits of p, q is given, then $N=p^rq$ can be factorized in the expected polynomial time. The analysis is then validated with experimental results for $N=p_1p_2p_3$, $N=p^2q$ and $N=p^3q$.

Keywords: Multi-prime RSA, Prime Power RSA, Key Recovery, Factorization

I. 서 론

RSA는 서로 다른 두 소수 p, q와 $ed=1 \bmod (p-1)(q-1)$을 만족시키는 지수 e, d가 주어졌을 때, $N=pq$와 임의의 정수 m에 대하여 $(m^e)^d = m \bmod N$ 가 성립한다는 성질을 이용하여 디지털 정보에 대한 암·복호화 및 전자서명·검증을 수행하는 암호시스템으로 $N=pq$를 RSA 시스템의 모듈러스(modulus)라고 한다[1]. RSA 시스템은 현재 가장 널리 사용되는 암호시스템 중의

Received(09. 05. 2016), Modified(11. 16. 2016), Accepted(11. 17. 2016)
† 주저자, yoojin.baek@gmail.com
‡ 교신저자, yoojin.baek@gmail.com(Corresponding author)

하나로 그 안전성은 모듈러스 N의 소인수분해가 어렵다는 가정에 기반을 둔다고 알려져 있다. 하지만 컴퓨팅 기술이 발전하면서 충분한 안전성을 보장할 수 있는 N의 크기가 점점 커지고 있고, 이에 따라 RSA시스템의 암·복호화에 필요한 연산 시간 역시 점점 증가하고 있다. 따라서 이러한 문제를 해결하기 위해서 RSA를 대체하는 시스템에 대한 연구가 활발히 진행되고 있으며 이러한 연구 방향의 하나가 바로 RSA 변형 시스템에 대한 연구이다. 지금까지 제안된 RSA 변형 시스템으로는 먼저 2개의 소수가 아닌 3개 이상의 서로 다른 소수의 곱을 모듈러스로 사용하는 Multi-Prime RSA[2]와 서로 다른 두 소수 p, q와 1보다 큰 자연수 r에 대하여 $N = p^r q$ 을 모듈러스로 사용하는 Prime Power RSA[3,4,11]가 있다.

RSA 시스템의 암호문이 주어지는 경우 이에 대응하는 평문을 복구하는 문제는 일반적으로 RSA 모듈러스를 소인수분해하는 것만큼 어려울 것으로 예상이 되고 있지만, 만약 암·복호문 이외에 다른 부가정보가 주어지는 경우에는 소인수분해 문제보다 쉬울 수 있음이 알려져 있다. 예를 들어, Coppersmith는 $N = pq$에 대하여 p 또는 q의 최상위 혹은 최하위 비트의 절반 이상이 주어지면 다항식 시간 안에 N을 소인수분해할 수 있음을 보였다[5]. 또한 Boneh 등은 Coppersmith의 방법을 확장하여, $N = p^r q$일 때 p의 최상위 비트 중 $\frac{1}{r+1}$의 비율에 해당하는 비트가 주어지면 역시 다항식 시간 안에 N을 소인수분해할 수 있음을 보였다[6]. 이러한 격자 기반의 분석 방법과는 달리, Heninger와 Shacham은 $N = pq$에 대하여 p 와 q의 전체 비트 중 57%에 해당하는 비트가 랜덤하게 주어진 경우 N을 소인수분해할 수 있음을 보였다[7]. Heninger와 Shacham의 분석 방법은 이전의 Coppersmith, Boneh 등의 방법과 매우 다른데, 예를 들어 Coppersmith 등은 소수의 알려진 비트가 최상위 또는 최하위에 위치한 연속적인 비트여야 함을 가정하지만, Heninger와 Shacham은 알려진 소수 비트의 위치가 랜덤 해야 함을 가정한다.

본 논문에서는 Heninger와 Shacham의 분석방법과 유사한 방법을 적용하여 Multi-Prime RSA와 Prime Power RSA의 모듈러스를 구성하는 소수의 비트 중 어느 정도의 비트가 주어져야 해당 모듈러스를 다항식 시간 안에 인수분해할 수 있는지를 이론적으로 분석하고 이를 실험적으로 검증한다. 구체적으로 본 논문은, 만약 $p_1, p_2, \ldots, p_r$의 전체 비트 중 $2 - 2^{1/r}$의 비율에 해당하는 비트가 랜덤하게 주어지면 $N = p_1 p_2 \ldots p_r$이 다항식시간 안에 인수분해될 수 있음을, 그리고 p, q의 전체 비트 중 $2 - \sqrt{2}$의 비율에 해당하는 비트가 랜덤하게 주어지면 $N = p^r q$이 다항식시간 안에 인수분해될 수 있음을 보이고, $N = p_1 p_2 p_3$, $N = p^2 q$, $N = p^3 q$에 대한 구체적인 실험 결과를 제시한다.

II. 사전 지식

2.1 Multi-Prime RSA와 Prime Power RSA

암·복호화 속도를 높이고 안전성을 향상시키려는 노력의 일환으로 많은 변형 RSA 암호시스템이 제안되었다. 그 중에서 Multi-prime RSA는 2개의 소수가 아닌 3개 이상의 서로 다른 소수의 곱을 모듈러스로 사용하며 특히 3개의 소수를 사용하는 시스템이 많이 연구되고 있다[2]. 이러한 시스템의 안전성과 관련하여 Hinek은, 가령 3개의 소수가 사용된 경우 어떤 한 소수의 최상위 또는 최하위 비트의 2/3가 알려지고 다른 한 소수의 최상위 또는 최하위 비트의 1/2이 알려진 경우 시스템의 모듈러스가 다항식 시간 안에 소인수분해될 수 있음을 보였다[8].

또 다른 변형 RSA 시스템으로는 Prime Power RSA가 있는데, 이 시스템은 모듈러스로 $N = p^r q$를 사용하여 암·복호화 연산을 수행한다. 여기서 p, q는 서로 다른 소수이고 r은 1보다 큰 양의 정수로 임의로 선택 가능하지만 $r = 2$인 경우가 가장 많이 연구되고 있다. Prime Power RSA는 2가지의 서로 다른 변형이 가능한데, 그 중의 하나는 $ed = 1 \bmod (p-1)(q-1)$ 을 만족시키는 지수 e, d를 사용하여 암·복호화 연산을 수행하고, 또 다른 변형은 $ed = 1 \bmod p^{r-1}(p-1)(q-1)$ 을 만족시키는 e, d를 사용한다[3,4]. Prime Power RSA에 대하여 Boneh 등은 Coppersmith의 방법을 확장하여, $N = p^r q$일 때 p의 최상위 비트 중 $\frac{1}{r+1}$의 비율에 해당하는 비트가 주어지면 다항식 시간 안에 N을 소인수분해할 수 있음을 보였다[6].

2.2 Heninger와 Shacham 알고리즘

본 논문에서 사용하는 분석방법은 [7]에서 제시한 분석기술에 기반을 두기 때문에 본 절에서는 이에 대한 간략한 소개를 하며 이를 위해 본 논문에서는 다음과 같은 표기법을 사용한다: 양의 정수 x에 대하여 $x[i]$는 x의 i번째 비트를 나타내며 따라서 $x[0]$는 x의 최하위비트를 표시한다.

RSA 시스템의 복호화는 비밀 지수 d를 이용하여 $c \mapsto c^d \bmod N$과 같이 진행될 수도 있지만, PKCS(Public-Key Cryptography Standard) 1번 문서에서는 중국인 나머지 정리를 이용한 빠른 RSA 복호화를 위해 다음을 만족시키는 $(p, q, d, d_p, d_q, q^{-1} \bmod p)$ 형태의 개인키 사용을 권장한다[9]:

$$\begin{aligned} N &= pq \\ ed &= 1 \bmod (p-1)(q-1) \\ ed_p &= 1 \bmod (p-1) \\ ed_q &= 1 \bmod (q-1). \end{aligned} \tag{1}$$

이중에서 $q^{-1} \bmod p$는 이후의 분석에서 사용이 되지 않기 때문에, 본 논문에서는 RSA 개인키가 (p, q, d, d_p, d_q)의 형태임을 가정한다.

일반적으로 RSA 개인키 (p, q, d, d_p, d_q)의 한 구성 요소가 주어지면 N의 소인수분해가 가능하며, Heninger와 Shacham은 이 사실을 이용해서, 만약 (p, q, d, d_p, d_q)의 전체 비트 중 (랜덤한 위치의) 일부 비트가 주어졌을 때 이를 이용해 RSA 개인키 전체를 복구하는 알고리즘을 제시하였다. 이를 위해 그들은 먼저 공개 지수 e가 매우 작다고 가정하였으며 (만약 RSA 개인키 중 (p, q)에 대해서만 분석을 진행하는 경우 e가 작다는 가정은 불필요하다), 이 가정 하에서 다음을 만족시키는 정수 k, k_p, k_q의 정확한 값은 기껏해야 $O(e)$번의 계산을 통해 알아낼 수 있음을 보였다.

$$\begin{aligned} ed &= 1 + k(p-1)(q-1) \\ ed_p &= 1 + k_p(p-1) \\ ed_q &= 1 + k_q(q-1) \end{aligned} \tag{2}$$

또한 그들은 (1)과 (2)로부터 다음과 같은 모듈라 방정식이 유도될 수 있음을 보였다: $i \geq 1$에 대하여 만약 p', q', d', d_p', d_q'가

$$\begin{aligned} N &= p'q' \bmod 2^i \\ ed' &= 1 + k(p'-1)(q'-1) \bmod 2^i \\ ed_p' &= 1 + k_p(p'-1) \bmod 2^i \\ ed_q' &= 1 + k_q(q'-1) \bmod 2^i \end{aligned} \tag{3}$$

을 만족시키면, $p[i], q[i], d[i], d_p[i], d_q[i]$는 다음 식을 만족시킨다.

$$\begin{aligned} &p[i] + q[i] = (N - p'q')[i] \bmod 2 \\ &d[i] + p[i] + q[i] = \\ &\quad (k(p'-1)(q'-1) + 1 - ed')[i] \bmod 2 \\ &d_p[i] + p[i] = \\ &\quad (k_p(p'-1) + 1 - ed_p')[i] \bmod 2 \\ &d_q[i] + q[i] = \\ &\quad (k_q(q'-1) + 1 - ed_q')[i] \bmod 2. \end{aligned} \tag{4}$$

특히 식 (4)는 미지수가 5개이지만 방정식은 4개이기 때문에 별다른 조건이 주어지지 않으면 해 $(p[i], q[i], d[i], d_p[i], d_q[i]) \in \{0,1\}^5$가 2개 존재함을 알 수 있다. 또한 논문 [7]에서는 식 (4)에서 $d[i], d_p[i], d_q[i]$ 대신에 $d[i+\tau(k)], d_p[i+\tau(k_p)], d_q[i+\tau(k_q)]$를 사용하지만 ($\tau(\,)$에 대한 정의는 [7]을 참조), $\tau(k) = \tau(k_p) = \tau(k_q) = 0$을 가정해도 분석의 복잡도와 유효성에는 영향을 미치지 않기 때문에 본 논문에서는 $\tau(k) = \tau(k_p) = \tau(k_q) = 0$임을 가정한다. 이제 $i \geq 0$에 대하여

$$\mathit{Slice}[i] = (p[i], q[i], d[i], d_p[i], d_q[i])$$

라고 정의하면 수식 (3)과 (4)가 의미하는 것은, 만약 $\mathit{Slice}[0], ..., \mathit{Slice}[i-1]$이 주어지면 (4)를 통해 $\mathit{Slice}[i]$를 계산할 수 있다는 것이다. 또한 p와 q는 홀수이기 때문에 $\mathit{Slice}[0] = (1, 1, 1, 1, 1)$ 임을 알 수 있고, 따라서 식 (3)과 (4)를 이용하면, 비트 위치를 최하위에서 시작하여 상위로 한 비트씩 차례로 옮겨가면서 p, q, d, d_p, d_q의 모든 비트를 복구할 수 있음을 알

수 있다. 그리고 Heninger와 Shacham은, 만약 p, q, d, d_p, d_q 비트 중 일부가 주어진다면 그 주어진 비트와 복구한 비트가 일치하는지 여부를 조사함으로써 복구된 비트의 유효성을 검사할 수 있고, 따라서 주어진 비트의 개수가 충분히 많으면 다항식 시간 안에 RSA 개인키 전체를 알아낼 수 있음을 증명하였다. 구체적으로, 만약 (p, q, d, d_p, d_q)의 비트 중 27%가 랜덤하게 주어지거나, (p, q, d)의 비트 중 42%가 랜덤하게 주어지거나, 또는 (p, q)의 비트 중 57%가 랜덤하게 주어지면 N은 다항식 시간 안에 소인수분해될 수 있음이 증명되었다.

III. Multi-Prime RSA와 Prime Power RSA에 대한 키 복구 알고리즘

이번 장에서는 Multi-Prime RSA와 Prime Power RSA에 대하여 시스템을 구성하는 소수의 비트 중 일정 부분이 주어진 경우 해당 모듈러스를 인수분해하는 알고리즘을 제시하고, 이러한 알고리즘이 효율적으로 동작할 수 있는 이론적인 조건을 제시한다. 그리고 이러한 이론적인 분석 결과는 4장의 실험 결과에 의해 검증된다. 특히 본 논문의 분석 방법은 시스템의 소수 비트가 주어진 경우뿐만 아니라 비밀 지수인 d, d_p 또는 d_q의 비트가 주어진 경우에도 쉽게 확장이 가능하지만 이에 대한 구체적인 논의는 생략한다.

3.1 Multi-Prime RSA에 대한 키 복구 알고리즘

서로 다른 (홀수) 소수 $p_1, p_2, ..., p_r (r > 2)$에 대하여, Multi-Prime RSA는 $N = p_1 p_2 ... p_r$을 모듈러스로 사용한다. 그리고 이 시스템에 대한 안전성 분석을 위해서는 (3), (4)와 유사한 방정식을 유도해야 하는데, 이를 위해 먼저 $i \geq 1$에 대하여 다음을 만족시키는 $p_1', p_2', ..., p_r'$이 주어졌다고 가정하자:

$$N = \prod_{k=1}^{r} p_k' \mod 2^i.$$

이제 비트 $a_1, a_2, ..., a_r \in \{0,1\}$에 대하여

$$N = \prod_{k=1}^{r} (a_k 2^i + p_k') \mod 2^{i+1}$$

이 만족되기 위해서는

$$N = (a_1 + ... + a_r) 2^i + \prod_{k=1}^{r} p_k' \mod 2^{i+1}$$

가 만족되어야 하며, 이로부터 다음과 같은 방정식이 유도된다:

$$a_1 + ... + a_r = (N - \prod_{k=1}^{r} p_k')[i] \mod 2 \quad (5)$$

여기서 주목할 점은, 방정식 (5)는 미지수가 r개이기 때문에 이를 만족시키는 해 $(a_1, a_2, ..., a_r)$는 일반적으로 2^{r-1}개가 존재한다는 점이다. 이제 (5)를 이용하면 다음과 같은 알고리즘을 통해 $p_1, p_2, ..., p_r$을 복구할 수 있게 되며 알고리즘에서 n은 $p_1, p_2, ..., p_r$의 비트 길이 중 최댓값을 의미한다.

Algorithm 1 (Multi-Prime RSA의 키 복구)

입력: $p_1, p_2, ..., p_r$의 전체 비트들 중 δ 비율에 해당하는 비트, $N (= p_1 p_2 ... p_r)$

출력: $p_1, p_2, ..., p_r$

1) $W_1 \leftarrow \{(1,1,...,1)\}$

2) For $i = 1$ to $n-1$

   a) $W_{i+1} \leftarrow \varnothing$;

   b) W_i의 각 원소 $(x_1, x_2, ..., x_r)$에 대하여, $a_1 + ... + a_r = (N - \prod_{k=1}^{r} x_k)[i] \mod 2$를 만족시키는 해 $(a_1, a_2, ..., a_r) \in \{0,1\}^r$을 구한다. 그리고 만약 $p_j[i]$가 알려져 있는 모든 $j \in \{1, 2, ..., r\}$에 대하여 $p_j[i]$가 a_j와 같으면 $(a_1 2^i + x_1, ..., a_r 2^i + x_r)$를 W_{i+1}에 추가한다.

3) 만약 $x_1 x_2 \cdots x_r = N$을 만족시키는 $(x_1, x_2, ..., x_r) \in W_n$이 존재하면

$(x_1, x_2, ..., x_r)$을 반환하고 알고리즘을 종료한다.

Algorithm 1에서 주목할 점은, 이 알고리즘은 항상 N의 소인수 $p_1, p_2, ..., p_r$을 성공적으로 출력하지만 δ 값에 따라 알고리즘의 연산 시간은 많이 달라질 수 있다는 것이다. 즉 δ가 매우 작으면 알고리즘에서 사용하는 변수인 W_i가 너무 많은 원소를 가지게 되어 알고리즘의 연산 시간 (및 메모리 사용량)이 매우 늘어나게 된다. 그리고 알고리즘의 연산 시간을 측정하기 위해서는 δ값에 따른 W_i의 크기를 예측할 수 있어야 하며 W_i의 크기는 $p_1, p_2, ..., p_r$의 i-번째 비트가 주어져 있는 지 여부에 따라서 달라진다. 구체적으로 Algorithm 1을 분석하기 위해서 본 논문은 [7]에 주어진 다음과 같은 표기법과 가정을 사용한다: $(x_1, x_2, ..., x_r) \in W_i$가 모든 $k = 1, ..., r$에 대하여 $x_k = p_k \bmod 2^i$ 를 만족시키면 $(x_1, x_2, ..., x_r)$를 'good solution'이라고 하고, 그렇지 않으면 'bad solution'이라고 한다.

가정 1 $(x_1, x_2, ..., x_r) \in W_i$가 bad solution일 때 $a_1 + ... + a_r = (N - \prod_{k=1}^{r} x_k)[i] \bmod 2$이 성립할 확률은 1/2이다. 여기서 확률은 $(x_1, x_2, ..., x_r)$와 $(a_1, a_2, ..., a_r) \in \{0,1\}^r$의 랜덤한 선택에 따른다.

이제 Algorithm 1의 W_i의 크기를 S_i라 표기하면 S_i는 다음을 바탕으로 계산될 수 있다.

1) $S_1 = 1$
2) $i \geq 1$과 $(x_1, x_2, ..., x_r) \in W_i$에 대하여
   a) (δ^r의 확률로) r개의 비트 $p_1[i], p_2[i], ..., p_r[i]$가 모두 주어져 있고, $(x_1, x_2, ..., x_r)$가 'good solution'이면 정확히 1개의 원소 $(p_1[i]2^i + x_1, ..., p_r[i]2^i + x_r)$가 W_{i+1}에 추가된다. 그러나 $p_1[i], p_2[i], ..., p_r[i]$가 모두 주어져 있으나 $(x_1, x_2, ..., x_r)$가 'bad solution'이면 가정 1에 의해 1/2의 확률로 $(p_1[i]2^i + x_1, ..., p_r[i]2^i + x_r)$가 W_{i+1}에 추가된다.
   b) ($\delta^k(1-\delta)^{r-k}$의 확률로) r개의 비트 $p_1[i], p_2[i], ..., p_r[i]$중 $k (0 \leq k < r)$개의 비트가 주어져 있으면, $a_1 + ... + a_r = (N - \prod_{k=1}^{r} x_k)[i] \bmod 2$를 만족시키는 해 $(a_1, a_2, ..., a_r) \in \{0,1\}^r$가 정확히 2^{r-k-1}개가 있고, 따라서 2^{r-k-1}개의 원소가 W_{i+1}에 추가된다.

이를 바탕으로 다음의 식이 유도된다: $i \geq 1$에 대하여

$$\begin{aligned} S_{i+1} &= [(\frac{1}{S_i} + \frac{S_i - 1}{S_i} \times \frac{1}{2})\delta^r \\ &\quad + \sum_{k=0}^{r-1} \binom{r}{k} 2^{r-k-1} \delta^k (1-\delta)^{r-k}] S_i \\ &= \sum_{k=0}^{r} \binom{r}{k} 2^{r-k-1} \delta^k (1-\delta)^{r-k} S_i + \frac{1}{2}\delta^r \\ &= \frac{1}{2}(2-\delta)^r S_i + \frac{1}{2}\delta^r. \end{aligned}$$

그리고 $S_1 = 1$이라는 사실을 이용하면 S_i는 다음과 같이 계산된다: $i \geq 1$, $\alpha = \frac{1}{2}(2-\delta)^r$, $\beta = \frac{1}{2}\delta^r$에 대하여

$$S_i = \begin{cases} (\alpha + \beta - 1)\frac{1 - \alpha^{i-1}}{1 - \alpha} + 1, & \alpha \neq 1 \\ 1 + (i-1)\beta & \alpha = 1 \end{cases}. \quad (6)$$

(6)에 의하면 S_i는 α값에 따라서 그 값이 증가하는 정도가 달라지는데, 가령 $\alpha \leq 1$이면 S_i는 (평균적으로) 선형적으로 증가하고, $\alpha > 1$이면 S_i는 (평균적으로) 기하급수적으로 증가하게 된다. 그리고 Algorithm 1의 연산 시간은 S_i에 주로 의존하기 때문에, $\alpha \leq 1$이면 Algorithm 1 은 효과적으로 N을 소인수분해할 수 있다고 말할 수 있게 된다.

좀 더 자세하게 우리는 다음 정리를 증명할 수 있다.

정리 1. 가정 1이 성립하면 다음 역시 성립한다. 서로 다른 $r(>2)$개의 (홀수) 소수 $p_1, p_2, ..., p_r$에 대하여 $N=p_1p_2...p_r$이라 하고 $p_1, p_2, ..., p_r$의 전체 비트 중 δ에 해당하는 비율만큼의 비트가 주어졌다고 가정하자. 이 때, 만약 $\delta \geq 2-2^{1/r}$이면 Algorithm 1 은 (평균적으로) $\log_2 N$에 대한 다항식시간 안에 $p_1, p_2, ..., p_r$를 반환한다.

증명) Algorithm 1의 연산 시간은 2단계의 반복회수 $n=O(\log_2 N)$과 S_i에 의해 결정되며, 구체적으로는 $O(\sum_{i=1}^{n} S_i)$로 표현 가능하다. 그리고 $\alpha \leq 1$이면 $O(\sum_{i=1}^{n} S_i)$는 기껏해야 $O(n^2)$이다. 마지막으로 $\alpha \leq 1$일 필요충분조건은 $\delta \geq 2-2^{1/r}$이기 때문에 정리가 증명된다. ■

Table 1은 정리 1의 $2-2^{1/r}$값에 대한 근삿값을 여러 가지 r값에 대하여 보여준다.

Table 1. Approximate values of $2-2^{1/r}$for various r

r	Approximate Value of $2-2^{1/r}$
3	0.74
4	0.81
5	0.85
6	0.88

3.2 Prime Power RSA

서로 다른 두 (홀수) 소수 p, q와 양의 정수 $r(>1)$에 대하여 Prime Power RSA는 $N=p^r q$를 모듈러스로 사용하며, 이번 절에서는 Prime Power RSA에 대한 안전성을 짝수 r과 홀수 r의 2가지 경우로 나누어서 분석한다.

먼저, r을 짝수라고 가정하고 $r=2k$라고 표기하자. 그리고 $i \geq 1$에 대하여 다음을 만족시키는 p', q'가 주어졌다고 가정하자:

$$N=p'^{2k}q' \bmod 2^i.$$

그러면 $a_1, a_2 \in \{0,1\}$가

$$\begin{aligned} N &= (a_1 2^i + p')^{2k}(a_2 2^i + q') \bmod 2^{i+1} \\ &= a_2 2^i + p'^{2k}q' \bmod 2^{i+1} \end{aligned}$$

을 만족시키기 위해서는 다음이 만족되어야 한다:

$$a_2 = (N - p'^{2k}q')[i] \bmod 2. \qquad (7)$$

이제 (7)를 이용하면 다음과 같은 알고리즘을 통해 p, q을 복구할 수 있게 되며 알고리즘에서 n은 p, q의 비트 길이 중 최댓값을 의미한다.

Algorithm 2 (짝수 r에 대한 Prime Power RSA의 키 복구)

입력: p, q의 전체 비트들 중 δ의 비율에 해당하는 비트, 짝수 $r(>1)$에 대하여 $N=p^r q$

출력: p, q

1) $W_1 \leftarrow (1,1)$

2) For $i=1$ to $n-1$

a) $W_{i+1} \leftarrow \varnothing$;

b) W_i의 각 원소 (x, y)에 대하여, $a_2 = (N - x^r y)[i] \bmod 2$를 만족시키는 $a_2 \in \{0,1\}$을 계산한다. 그리고 만약 $p[i], q[i]$가 모두 주어져 있으면 $(x, a_2 2^i + y)$와 $(2^i + x, a_2 2^i + y)$를 W_{i+1}에 추가하고, 만약 $p[i]$는 주어져 있지 않고 $q[i]$만 주어져 있으며 주어진 $q[i]$가 a_2와 같으면 $(x, a_2 2^i + y)$와 $(2^i + x, a_2 2^i + y)$를 W_{i+1}에 추가한다. 또한 $p[i]$는 주어져 있지만 $q[i]$는 주어져 있지 않으면 $(p[i]2^i + x, a_2 2^i + y)$를 W_{i+1}에 추가한다. 마지막으로 $p[i], q[i]$가 모두 주어져 있고 $q[i]$가 a_2와 같으면, $(p[i]2^i + x, a_2 2^i + y)$를 W_{i+1}에 추가한다.

3) 만약 $x^r y = N$을 만족시키는 $(x, y) \in W_n$이 존재하면 (x, y)을 반환하고 알고리즘을 종료한다.

이번 절에서는 앞 절과 유사하게 다음과 같은 표기법과 가정을 사용한다: $(x,y)\in W_i$가 $x=p \bmod 2^i$, $y=q \bmod 2^i$를 만족시키면 (x,y)를 'good solution'이라고 하고, 그렇지 않으면 'bad solution'이라고 한다.

가정 2 $r(>1)$이 짝수이고 $(x,y)\in W_i$가 'bad solution'일 때, a_2가 $a_2=(N-x^r y)[i] \bmod 2$를 만족시킬 확률은 1/2다. 여기서의 확률은 (x,y)와 $(a_1,a_2)\in\{0,1\}^2$의 랜덤한 선택에 따른다.

이제 Algorithm 2의 W_i의 크기를 S_i라 표기하면 S_i는 다음을 바탕으로 계산될 수 있다.

1) $S_1=1$
2) $i\geq 1$과 $(x,y)\in W_i$에 대하여
   a) (δ^2의 확률로) $p[i],q[i]$가 모두 주어지고 (x,y)가 'good solution'이면 $(p[i]2^i+x, q[i]2^i+y)$가 W_{i+1}에 추가된다. 그러나 $p[i],q[i]$가 모두 주어졌으나 (x,y)가 'bad solution'이면 가정 2에 의하여 $q[i]=(N-x^r y)[i] \bmod 2$가 성립할 확률은 1/2이다. 따라서 이 경우에는 1/2의 확률로 $(p[i]2^i+x, q[i]2^i+y)$가 W_{i+1}에 추가된다.
   b) ($\delta(1-\delta)$의 확률로) $p[i]$만 주어지면 $a_2=(N-x^r y)[i] \bmod 2$를 만족시키는 a_2에 대하여 $(p[i]2^i+x, a_2 2^i+y)$가 W_{i+1}에 추가된다.
   c) ($\delta(1-\delta)$의 확률로) $q[i]$만 주어지고 (x,y)가 'good solution'이면, $(x, q[i]2^i+y)$와 $(2^i+x, q[i]2^i+y)$가 W_{i+1}에 추가된다. 그러나 $q[i]$만 주어졌을 때 (x,y)가 'bad solution'이면, 가정 2에 의하여 $q[i]=(N-x^r y)[i] \bmod 2$가 성립할 확률은 1/2이다. 따라서 이 경우 1/2의 확률로 $(x, q[i]2^i+y)$와 $(2^i+x, q[i]2^i+y)$가 W_{i+1}에 추가된다.
   d) ($(1-\delta)^2$의 확률로) $p[i],q[i]$ 모두가 주어져 있지 않으면 $a_2=(N-x^r y)[i] \bmod 2$를 만족시키는 a_2에 대하여 $(x, a_2 2^i+y)$와 $(2^i+x, a_2 2^i+y)$가 W_{i+1}에 추가된다.

이를 바탕으로 다음 식이 유도된다: $i\geq 1$에 대하여

$$\begin{aligned} S_{i+1} &= [(\frac{1}{S_i}+\frac{S_i-1}{S_i}\times\frac{1}{2})\delta^2 \\ &\quad +\delta(1-\delta)(1+\frac{1}{S_i}\times 2+\frac{S_i-1}{S_i}\times\frac{1}{2}\times 2) \\ &\quad +2(1-\delta)^2]S_i \\ &=(\frac{1}{2}\delta^2-2\delta+2)S_i+\delta-\frac{1}{2}\delta^2. \end{aligned}$$

따라서 $S_1=1$라는 사실을 이용하면, $i\geq 1$에 대하여 다음이 성립함을 알 수 있다: $\alpha=\frac{1}{2}\delta^2-2\delta+2, \beta=\delta-\frac{1}{2}\delta^2$에 대하여

$$S_i=\begin{cases}(\alpha+\beta-1)\dfrac{1-\alpha^{i-1}}{1-\alpha}+1, & \alpha\neq 1\\ 1+(i-1)\beta & \alpha=1\end{cases}.$$

마지막으로 r이 홀수이면 $r=2k+1$로 표기할 수 있다. 이 때 $i\geq 1$에 대하여 다음을 만족시키는 p', q'가 주어졌다고 가정하자:

$$N=p'^{2k+1}q' \bmod 2^i.$$

그러면 $a_1,a_2\in\{0,1\}$가

$$\begin{aligned} N &= (a_1 2^i+p')^{2k+1}(a_2 2^i+q') \bmod 2^{i+1} \\ &= (a_1+a_2)2^i+p'^{2k+1}q' \bmod 2^{i+1} \end{aligned}$$

을 만족시키기 위해서는 다음이 성립하여야 한다:

$$a_1+a_2=(N-p'^{2k+1}q')[i] \bmod 2. \qquad (8)$$

이제 (8)를 이용하면 다음과 같은 알고리즘을 통

해 p, q을 복구할 수 있게 되며 알고리즘에서 n은 p, q의 비트 길이 중 최댓값을 의미한다.

Algorithm 3 (홀수 r에 대한 Prime Power RSA의 키 복구)

입력: p, q의 전체 비트들 중 δ의 비율에 해당하는 비트, 홀수 $r(>1)$에 대하여 $N=p^r q$

출력: p, q

1) $W_1 \leftarrow (1,1)$

2) For $i=1$ to $n-1$

a) $W_{i+1} \leftarrow \varnothing$;

b) W_i의 각각의 원소 (x, y)에 대하여, $a_1+a_2 = (N-x^r y)[i] \bmod 2$를 만족시키는 $(a_1, a_2) \in \{0,1\}^2$을 계산한다. 만약 $p[i], q[i]$가 모두 주어져 있지 않으면 $(a_1 2^i + x, a_2 2^i + y)$를 W_{i+1}에 추가한다. 만약 $p[i]$가 주어져 있을 때 그 값이 a_1과 같고, $q[i]$가 주어져 있을 때 그 값이 a_2와 같으면 $(a_1 2^i + x, a_2 2^i + y)$를 W_{i+1}에 추가한다.

3) 만약 $x^r y = N$을 만족시키는 $(x, y) \in W_n$이 존재하면 (x, y)을 반환하고 알고리즘을 종료한다.

Algorithm 3 역시 다음과 같은 표기법과 가정을 바탕으로 분석이 진행된다: $(x, y) \in W_i$가 $x = p \bmod 2^i$, $y = q \bmod 2^i$를 만족시키면 (x, y)를 'good solution'이라고 하고, 그렇지 않으면 'bad solution'이라고 한다.

가정 3 $r(>1)$이 홀수이고 $(x, y) \in W_i$가 'bad solution'일 때, a_1, a_2가 $a_1+a_2 = (N-x^r y)[i] \bmod 2$를 만족시킬 확률은 1/2이다. 여기서의 확률은 (x, y)와 $(a_1, a_2) \in \{0,1\}^2$의 랜덤한 선택에 따른다.

이제 Algorithm 3의 W_i의 크기를 S_i라 표기하면 S_i는 다음을 바탕으로 계산될 수 있다.

1) $S_1 = 1$

2) $i \geq 1$과 $(x, y) \in W_i$에 대하여

a) (δ^2의 확률로) $p[i], q[i]$가 모두 주어지고 (x, y)가 'good solution'이면 $(p[i]2^i + x, q[i]2^i + y)$가 W_{i+1}에 추가된다. 그러나 $p[i], q[i]$가 모두 주어져 있지만 (x, y)가 'bad solution'이면 가정 3에 의하여 $p[i]+q[i] = (N-x^r y)[i] \bmod 2$가 성립할 확률은 1/2이다. 따라서 이 경우에는 1/2의 확률로 $(p[i]2^i + x, q[i]2^i + y)$가 W_{i+1}에 추가된다.

b) ($\delta^k(1-\delta)^{2-k}$의 확률로) $p[i], q[i]$ 중 $k\,(0 \leq k < 2)$개가 주어지면, $a_1+a_2 = (N-x^r y)[i] \bmod 2$를 만족시키는 2^{1-k}개의 a_1, a_2에 대하여 $(a_1 2^i + x, a_2 2^i + y)$가 W_{i+1}에 추가된다.

이를 바탕으로 다음 식이 유도된다.

$$\begin{aligned} S_{i+1} &= [(\frac{1}{S_i} + \frac{S_i - 1}{S_i} \times \frac{1}{2})\delta^2 \\ &\quad + \sum_{k=0}^{1} \binom{2}{k} 2^{1-k} \delta^k (1-\delta)^{2-k}] S_i \\ &= \sum_{k=0}^{2} \binom{2}{k} 2^{1-k} \delta^k (1-\delta)^{2-k} S_i + \frac{1}{2}\delta^2 \\ &= \frac{1}{2}(2-\delta)^2 S_i + \frac{1}{2}\delta^2 \end{aligned}$$

따라서 $S_1 = 1$라는 사실을 이용하면 $i \geq 1$에 대하여 다음이 성립함을 알 수 있다: $\alpha = \frac{1}{2}(2-\delta)^2, \beta = \frac{1}{2}\delta^2$에 대하여

$$S_i = \begin{cases} (\alpha+\beta-1)\dfrac{1-\alpha^{i-1}}{1-\alpha} + 1, & \alpha \neq 1 \\ 1 + (i-1)\beta & \alpha = 1 \end{cases}.$$

마지막으로, 지금까지의 논의를 종합하면 다음 정리를 얻을 수 있다.

정리 2. 가정 2와 3이 성립하면 다음 역시 성립한다. 서로 다른 소수 p, q와 양의 정수 $r(>2)$에 대하여 $N=p^r q$이라 하고 p, q의 전체 비트 중 δ에

해당하는 비율만큼의 비트가 주어졌다고 가정하자. 이 때, 만약 $\delta \geq 2-\sqrt{2} \approx 0.586$이면 Algorithm 2와 3은 (평균적으로) $\log_2 N$에 대한 다항식시간 안에 p, q를 반환한다.

증명) Algorithm 2와 3의 연산 시간은 2단계의 반복회수 $n = O(\log_2 N)$과 S_i에 이해 결정되며 구체적으로 $O(\sum_{i=1}^{n} S_i)$와 같이 계산된다. 그리고 $\alpha \leq 1$이면 $O(\sum_{i=1}^{n} S_i)$는 기껏해야 $O(n^2)$이다. 마지막으로 $\alpha \leq 1$일 필요충분조건은 $\delta \geq 2-\sqrt{2}$이기 때문에 정리가 증명된다. ■

IV. 실험결과

3장의 결과를 검증하기 위하여 본 논문은 다음의 실험 1, 2, 3을 수행하였으며, 각 실험에 대한 실험 환경은 다음과 같다.

- Celeron(R) Dual-Core CPU T3300@2.00 GHz 2.00 GHz
- 6 GB RAM
- Windows 10 Pro
- Visual Studio 2015 Community

또한 실험에서 사용하는 RSA 연산 및 Big Number 연산은 NTL 라이브러리를 사용하였다 [10].

실험 1 (모듈러스 $N = pqr$를 사용하는 Multi-Prime RSA에 대한 RSA의 키 복구)

1) 3개의 서로 다른 512-비트 소수 p_1, p_2, p_3을 생성하고 $N = p_1 p_2 p_3$을 계산한다.
2) 주어진 δ값에 대하여, p_1, p_2, p_3를 나타내는 (1,536-비트 크기의) 비트열에 대해서 $\lfloor 1536\delta \rfloor$개의 비트 위치를 랜덤하게 선택하고 이 위치에 해당하는 p_1, p_2 또는 p_3의 비트가 주어져 있다고 가정한다.
3) Algorithm 1을 수행하고 S_{512}를 출력한다.

실험 2 (모듈러스 $N = p^2 q$를 사용하는 Prime Power RSA에 대한 RSA의 키 복구)

1) 2개의 서로 다른 512-비트 소수 p, q을 생성하고 $N = p^2 q$을 계산한다.
2) 주어진 δ값에 대하여, p, q를 나타내는 (1,024-비트 크기의) 비트열에 대해서 $\lfloor 1024\delta \rfloor$개의 비트 위치를 랜덤하게 선택하고 이 위치에 해당하는 p 또는 q의 비트가 주어져 있다고 가정한다.
3) Algorithm 2를 수행하고 S_{512}를 출력한다.

실험 3 (모듈러스 $N = p^3 q$를 사용하는 Prime Power RSA에 대한 RSA의 키 복구)

1) 2개의 서로 다른 512-비트 소수 p, q을 생성하고 $N = p^3 q$을 계산한다.
2) 주어진 δ값에 대하여, p, q를 나타내는 1,024-비트 크기의 이진 벡터에 대해서 $\lfloor 1024\delta \rfloor$개의 비트 위치를 랜덤하게 선택하고 이 위치에 해당하는 p 또는 q의 비트가 주어져 있다고 가정한다.
3) Algorithm 3를 수행하고 S_{512}를 출력한다.

각 실험은 주어진 δ에 대하여 100회 반복되었으며 실험 결과는 Table 2, 3, 4에 주어진다. 각 실

Table 2. Result of Experiment 1

δ	0.76	0.75	0.74	0.73	0.72
Maximum	4,247	65,590	4,349	666,150	232,748
3rd Quartile	28.5	32	25	69	175
Median	4	6	4	10	19
1st Quartile	2	1	1	3	3
Minimum	1	1	1	1	1
# of failures	0	5	3	10	14

Table 3. Result of Experiment 2

δ	0.59	0.58	0.57	0.56	0.55
Maximum	28,688	1,076	33,792	168,864	139,177
3rd Quartile	10.5	17	33.25	64	64
Median	4	4	6	15	8
1st Quartile	2	2	2	2	2
Minimum	1	1	1	1	1
# of failures	0	0	0	1	2

Table 4. Result of Experiment 3

δ	0.59	0.58	0.57	0.56	0.55
Maximum	3,840	44,080	11,508	65,553	100,013
3rd Quartile	25.25	27.25	27.5	88.5	96
Median	7.5	7.5	8	12.5	24
1st Quartile	2	2	2.5	3.75	4
Minimum	1	1	1	1	1
# of failures	0	0	1	0	1

험의 평균 수행 시간은 수 분 이내였으며, Table에서 'Maximum', '3rd Quartile', 'Median', '1st Quartile', 'Minimum'는 100회 측정된 S_{512}값의 최댓값, 3/4 분위수, 중간값, 1/4분위수, 최솟값을 나타내고, '# of failures'는 메모리 부족 등의 이유로 계산이 갑자기 중단된 회수를 나타낸다. Table 2, 3, 4의 결과가 보여주고 또한 정리 1과 2가 예측하듯이, 각 실험에서 δ값이 작아지면 S_{512}값과 '# of failures'는 점점 증가함을 알 수 있다.

V. 결 론

본 논문에서는 $N=p_1p_2\cdots p_r$과 $N=p^rq$를 인수분해하기 위해 각 소수의 얼마나 많은 비트들이 필요한지에 대한 분석을 수행하였다. 그리고 그 분석에 대한 검증을 위해 다양한 실험을 수행하고 그 결과를 제시하였다.

References

[1] R.L. Rivest, A. Shamir, and L.M. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Communications of the ACM, vol. 21, no. 2, pp. 120 - 126, Feb. 1978.

[2] R.L. Rivest, A. Shamir, and L.M. Adleman, "Cryptographic Communications System and Method," US Patent 4,405,829, 1983.

[3] T. Takagi, "Fast RSA-type Cryptosystem Modulo p^kq," Advances in Cryptology, CRYPTO '98, LNCS 1462, pp. 318-326, 1998.

[4] T. Takagi, "A Fast RSA-type Public-Key Primitive Modulo p^kq Using Hensel Lifting," IEICE Transactions, vol. 87-A, no. 1, pp. 94-101, Jan. 2004.

[5] D. Coppersmith, "Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities," J. Cryptology, vol. 10, no. 4, pp. 233 - 260, Sep. 1997.

[6] D. Boneh, G. Durfee, and N. Howgrave-Graham, "Factoring $n=p^rq$ for Large r," Advances in Cryptology,

CRYPTO '99, LNCS 1666, pp. 326-337, 1999.
〔7〕 N. Heninger and H. Shacham, "Reconstructing RSA Private Keys from Random Key Bits," Advances in Cryptology, CRYPTO '09, LNCS 5677, pp. 1 - 17, 2009.
〔8〕 M.J. Hinek, "On the Security of Some Variants of RSA," Ph. D Thesis, University of Waterloo, 2007.
〔9〕 RSA Security Inc., "Public-Key Cryptography Standards (PKCS) #1 v2.1: RSA Cryptography Standard," 2002.
〔10〕 NTL: A Library for Doing Number Theory, available at www.shoup.net/ntl
〔11〕 S. Sarkar, "Revisiting Prime Power RSA," Discrete Applied Mathematics vol. 203, pp. 127-133, Apr. 2016.

〈저자소개〉

백 유 진 (Yoo-Jin Baek) 종신회원
1997년 2월: 서울대학교 수학과 졸업
1999년 2월: 서울대학교 수학과 이학석사
2003년 2월: 서울대학교 수리과학부 이학박사
2003년 3월~2003년 6월: KAIST 박사후 연구원
2003년 7월~2013년 3월: 삼성전자 책임연구원
2013년 3월~현재: 우석대학교 정보보안학과 조교수
〈관심분야〉 부채널 공격, 정보 보안