Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.26, NO.5, Oct. 2016 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2016.26.5.1089

일반화된 확률 측도를 이용하여 에러가 있는 RSA 개인키를 복구하는 알고리즘

백 유 진†‡
우석대학교

Key Recovery Algorithm of Erroneous RSA Private Key Bits Using Generalized Probabilistic Measure

Yoo-Jin Baek†‡
Woosuk University

요 약

RSA 시스템에서 암·복호문 이외의 부가 정보가 주어졌을 때 개인키를 알아내는 것은 소인수분해보다 더 쉬울 수 있음이 잘 알려져 있다. 예를 들어, Coppersmith는 RSA 시스템을 구성하는 소수 중 하나의 최상위 또는 최하위 비트의 절반 이상이 주어지면 RSA 모듈러스가 다항식 시간 안에 인수분해될 수 있음을 보였다. 또한 Henecka 등은 (p, q, d, d_p, d_q) 형태의 RSA 개인키 비트 중 23.7%에 해당하는 비트에 에러가 삽입되더라도 원래의 RSA 개인키를 복구할 수 있는 알고리즘을 제안하였고, 이를 위해 후보 키 비트와 에러가 삽입된 RSA 개인키 비트 사이의 서로 매칭이 되는 비트들의 개수를 사용할 것을 제안하였다. 본 논문에서는 Henecka 등의 방법을 확장하여, 후보 키 비트와 에러가 삽입된 개인키 비트 사이의 일치되는 정도를 보여주는 좀 더 일반화된 확률 측도의 사용과 이 측도를 사용한 RSA 개인키 복구 알고리즘을 제시한다.

ABSTRACT

It is well-known that, if additional information other than a plaintext-ciphertext pair is available, breaking the RSA cryptosystem may be much easier than factorizing the RSA modulus. For example, Coppersmith showed that, given the 1/2 fraction of the least or most significant bits of one of two RSA primes, the RSA modulus can be factorized in a polynomial time. More recently, Henecka et. al showed that the RSA private key of the form (p, q, d, d_p, d_q) can efficiently be recovered whenever the bits of the private key are erroneous with error rate less than 23.7%. It is notable that their algorithm is based on counting the matching bits between the candidate key bit string and the given decayed RSA private key bit string. And, extending the algorithm, this paper proposes a new RSA private key recovery algorithm using a generalized probabilistic measure for measuring the consistency between the candidate key bits and the given decayed RSA private key bits.

Keywords: RSA, Key Recovery, Side-Channel Attack, Generalized Hoeffding Bound

I. 서 론

RSA는 서로 다른 두 소수 p, q와 $ed = 1 \bmod (p-1)(q-1)$를 만족시키는 e, d가 주어졌을 때, 임의의 정수 m에 대하여 $(m^e)^d = m \bmod N$ 이 성립한다는 사실을 이용하여 디지털 정보에 대한 암·복호화를 수행하는 암호시스템으로, $N = pq$를 RSA 시스템의 모듈러스라고 한

Received(07. 06. 2016), Modified(08. 30. 2016), Accepted(08. 31. 2016)

† 주저자, yoojin.baek@gmail.com
‡ 교신저자, yoojin.baek@gmail.com(Corresponding author)

다[1]. RSA 시스템의 복호화는 일반적으로 비밀 지수 d를 이용하여 $c \mapsto c^d \bmod N$과 같이 수행될 수도 있지만, PKCS(Public-Key Cryptography Standard) 1번 문서에서는 중국인 나머지 정리를 이용한 빠른 RSA 복호화를 위해 $(p, q, d, d_p, d_q, q^{-1} \bmod p)$의 형태를 가지는 개인키의 사용을 권장한다[2].

RSA 시스템의 암호문이 주어지는 경우 이에 대응하는 평문을 계산하는 것은 일반적으로 RSA 모듈러스를 소인수분해하는 것만큼 어려울 것으로 예상이 되고 있지만, 만약 암·복호문 이외에 다른 부가정보가 주어지면 소인수분해보다 더 쉬울 수 있음 역시 잘 알려져 있다. 예를 들어 Coppersmith는 p 또는 q의 최상위 혹은 최하위 비트의 절반 이상이 주어지면 다항식 시간 안에 N을 소인수분해할 수 있음을 보였다[3]. 또한 Boneh 등은 Coppersmith의 방법을 확장하여, $N=p^r q$에 대하여 p의 최상위 $\frac{1}{r+1}$비트가 주어지면 다항식 시간 안에 N을 소인수분해할 수 있음을 보였다[4]. 이러한 격자 기반의 분석 방법과는 달리 Heninger와 Shacham은 RSA 개인키에 해당하는 (p, q, d, d_p, d_q)의 전체 비트 중 27%에 해당하는 비트가 랜덤하게 주어지면 N을 소인수분해할 수 있음을 보였고[5], Henecka 등은 (p, q, d, d_p, d_q)의 비트들 중 23.7%에 해당하는 (랜덤한) 비트들에 에러가 삽입되더라도 원래의 RSA 개인키를 복구할 수 있는 알고리즘을 제시하였다[6]. Heninger 등의 알고리즘은 이전 Coppersmith 등의 방법에 비해 몇 가지 점에서 다른데, 예를 들어 Coppersmith 등이 제안한 격자 기반의 RSA 안전성 분석에서는 RSA를 구성하는 소수의 비트가 최상위 또는 최하위 비트 위치에서 연속적으로 주어져야 함을 가정하지만, Heninger 등의 방법은 RSA 개인키 비트가 주어지는 위치에 대한 사전 정보를 필요로 하지 않는다.

본 논문에서는 Henecka 등의 알고리즘을 확장하고 이를 이용해 랜덤하게 변형된 RSA 개인키로부터 원래의 개인키를 복구하는 새로운 알고리즘을 제안한다. 즉, Henecka 등이 제안한 알고리즘은 복구된 후보키 비트들과 에러가 삽입된 RSA 개인키 비트들 사이의 서로 매칭이 되는 비트들의 개수를 계산하고 이를 이용해서 복구된 후보 키 비트들의 유효성을 검사하는데 반해, 본 논문에서는 이러한 키 유효성 검사를 위한 확률 측도로서 매칭 비트들의 개수가 아닌 좀 더 일반화된 방법을 제시한다. 이렇게 제시된 확률 측도는 실제 RSA 개인키 복구에 독자적으로 사용이 될 수도 있지만 또한 기존에 사용되는 측도와 함께 사용이 될 수도 있으며, 만약 이전 측도와 함께 사용이 되면 하나의 측도를 사용하는 것에 비해 개인키 복구 알고리즘의 성능을 향상시킬 수 있음을 실험 결과와 함께 제시한다.

II. HS 및 HMM 알고리즘

본 논문에서 사용하는 방법은 [5]와 [6]에서 제시된 분석기술에 기반을 두기 때문에 이번 장에서는 이에 대한 간략한 소개를 한다.

RSA 시스템의 복호화는 비밀 지수 d를 이용하여 $c \mapsto c^d \bmod N$과 같이 진행될 수도 있지만, PKCS 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[2]:

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

그리고 본 논문의 분석에서는 $q^{-1} \bmod p$ 값이 사용되지 않기 때문에, 이후부터는 RSA 개인키가 (p, q, d, d_p, d_q) 형태로 주어졌다고 가정한다.

(p, q, d, d_p, d_q) 형태의 RSA 개인키의 특징 중의 하나는, p, q, d, d_p 또는 d_q 중 하나를 알면 N을 매우 쉽게 소인수분해할 수 있다는 점이며, 이러한 의미에서 (p, q, d, d_p, d_q) 형태의 RSA 개인키는 매우 많은 여분의 정보(redundancy)를 가지고 있음을 알 수 있다. 그리고 Heninger와 Shacham은 이러한 여분의 정보를 이용해서, p, q, d, d_p, d_q의 전체 비트 중 일부 비트가 주어졌을 때 RSA 개인키 전체를 복구하는 알고리즘을 제시하였다[5]. 이를 위해 그들은 먼저 공개 지수 e가 작다고 가정하였고, 이러한 가정 하에서 다음을 만족시키는 정수 k, k_p, k_q의 정확한 값을 빠른 시간 안에 계산할 수 있음을 보였으며, 본 논문 역시 [5]에서와 마찬가지로 e가 충분히 작음을 가정한다.

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

만약 주어진 양의 정수 x에 대하여 $x[i]$를 x의 i번째 비트라고 표기했을 때 (따라서 $x[0]$은 x의 최하위비트를 나타낸다), Heninger와 Shacham은 식 (1)과 (2)로부터 다음과 같은 모듈라 방정식이 유도될 수 있음을 보였다: $i \geq 1$에 대하여

$$\begin{aligned} &p[i]+q[i]=(N-p'q')[i] \bmod 2 \\ &d[i]+p[i]+q[i]= \\ &\quad (k(p'-1)(q'-1)+1-ed')[i] \bmod 2 \\ &d_p[i]+p[i]=(k_p(p'-1)+1-ed_p')[i] \bmod 2 \\ &d_q[i]+q[i]=(k_q(q'-1)+1-ed_q')[i] \bmod 2 \end{aligned} \quad (3)$$

여기서 p', q', d', d_p', d_q'는 다음을 만족시키는 값을 의미한다:

$$\begin{aligned} N &= p'q' \bmod 2^i \\ ed' &= 1+k(p'-1)(q'-1) \bmod 2^i \\ ed_p' &= 1+k_p(p'-1) \bmod 2^i \\ ed_q' &= 1+k_q(q'-1) \bmod 2^i \end{aligned} \quad (4)$$

그리고 (3)과 (4)를 이용해서 Heninger와 Shacham은, 최하위 비트에서 시작해서 최상위 비트까지 비트 위치를 한 비트씩 차례대로 옮겨가면서 p, q, d, d_p, d_q를 복구하는 다음과 같은 알고리즘을 제시하였으며, 만약 주어진 비트의 개수가 충분히 많으면 제안한 알고리즘은 다항식 시간 안에 RSA 개인키 전체를 알아낼 수 있음을 증명하였다.

Algorithm 1. (HS 알고리즘, [5])

입력: RSA 공개키 (N, e), RSA 개인키 (p, q, d, d_p, d_q)의 전체 비트 중 $\delta(0 < \delta < 1)$에 해당하는 비율만큼의 비트, N의 비트 크기 n

출력: (p, q, d, d_p, d_q)

1. $W_1 \leftarrow \{(1,1,1,1,1)\}$
2. For $i=1$ to $\left\lceil \frac{n}{2} \right\rceil - 1$
   1) $W_{i+1} \leftarrow \varnothing$;
   2) 모든 $(x,y,z,u,v) \in W_i$에 대하여 다음을 만족시키는 해 $(x_i,y_i,z_i,u_i,v_i) \in \{0,1\}^5$ 2개를 모두 계산한다:
      $x_i+y_i=(N-xy)[i] \bmod 2$
      $z_i+x_i+y_i$
      $=(k(x-1)(y-1)+1-ez)[i] \bmod 2$
      $u_i+x_i=(k_p(x-1)+1-eu)[i] \bmod 2$
      $v_i+y_i=(k_q(y-1)+1-ev)[i] \bmod 2$
   3) 2)에서 구한 해 (x_i,y_i,z_i,u_i,v_i) 중에서, $p[i]$가 주어져 있을 때 그 값이 x_i와 같고, $q[i]$가 주어져 있을 때 그 값이 y_i와 같고, $d[i]$가 주어져 있을 때 그 값이 z_i와 같고, $d_p[i]$가 주어져 있을 때 그 값이 u_i와 같고, $d_q[i]$가 주어져 있을 때 그 값이 v_i와 같으면 $(x_i2^i+x, y_i2^i+y, z_i2^i+z, u_i2^i+u, v_i2^i+v)$를 W_{i+1}에 추가한다.
3. $xy=N$을 만족시키는 $(x,y,z,u,v) \in W_{\lceil n/2 \rceil}$가 존재하면, (x,y,z,u,v)를 출력하고 알고리즘을 종료한다.

HS 알고리즘이 RSA 개인키의 일부분이 주어졌을 때 이를 바탕으로 개인키 전체를 복구하는 방법을 제시하였다면, Henecka 등의 알고리즘은 RSA 개인키 비트에 에러가 삽입되었을 때 원래의 개인키를 복구하는 방법을 제시한다[6]. 즉, Henecka 등이 제안한 HMM 알고리즘은, RSA 개인키 $sk=(p, q, d, d_p, d_q)$의 비트 중 $\delta(0 < \delta < 1/2)$의 비율만큼의 비트들이 원래의 값에서 반전된 값으로 변형된 $\widetilde{sk}=(\widetilde{p}, \widetilde{q}, \widetilde{d}, \widetilde{d_p}, \widetilde{d_q})$가 주어진 경우 $\widetilde{sk}$로부터 원래의 sk를 복구하는 방법을 제시한다. 이를 위해 Henecka 등은 1비트씩 RSA 개인키를 복구하는 HS 알고리즘과는 다르게 RSA 개인키를 구성하는 p, q, d, d_p, d_q 각각에 대하여 $t(>1)$ 비트를 한꺼번에 복구하고, 이렇게 복구된 $5t$개의 비트들과 이 비트들의 위치에 대응되는 $\widetilde{sk}$의 $5t$개의 비트들 중 서로 매칭되는 비트들의 개수를 계산한 후 이 매칭비트 수를 이용하여 복구된 후보키 비트의 유효성을 검사하는 방법을 사용한다.

Algorithm 2. (HMM 알고리즘, [6])

입력: 에러율 $\delta(0 < \delta < 1/2)$, 양의 정수 t, 기준값 $C(>0)$, RSA 공개키 (N, e), RSA 개인키 $sk=(p, q, d, d_p, d_q)$의 전체 비트 중 δ의 비율에

해당하는 비트들에 에러가 삽입된 형태로 주어진 $\widetilde{sk}=(\widetilde{p},\widetilde{q},\widetilde{d},\widetilde{d_p},\widetilde{d_q})$, N의 비트 크기 n

출력: $sk=(p,q,d,d_p,d_q)$

1. $W_1 \leftarrow \{(1,1,1,1,1)\}$;
2. For $i=1$ to $\left\lceil \frac{n}{2t} \right\rceil - 1$
   1) $W_{i+1} \leftarrow \varnothing$;
   2) 모든 $(x,y,z,u,v) \in W_i$와 $T \in \{0,1,...,2^t-1\}$에 대하여 다음을 만족시키는 해 $(x_i,y_i,z_i,u_i,v_i) \in \{0,1,...,2^t-1\}^5$를 모두 계산한다:
   $x_i = T$
   $A_1 = x + x_i 2^{1+(i-1)t}$
   $A_2 = y + y_i 2^{1+(i-1)t}$
   $A_3 = z + z_i 2^{1+(i-1)t}$
   $A_4 = u + u_i 2^{1+(i-1)t}$
   $A_5 = v + v_i 2^{1+(i-1)t}$
   $A_1 A_2 = N \bmod 2^{1+it}$
   $eA_3 = 1 + k(A_1-1)(A_2-1) \bmod 2^{1+it}$
   $eA_4 = 1 + k_p(A_1-1) \bmod 2^{1+it}$
   $eA_5 = 1 + k_q(A_2-1) \bmod 2^{1+it}$
   3) 2)에서 구한 해 (x_i,y_i,z_i,u_i,v_i)에 대해서 다음과 같이 $mw_1, mw_2, mw_3, mw_4, mw_5$를 계산한 후 그 합 $mw = \sum_{i=1}^{5} mw_i$가 C보다 크거나 같으면 2)에서 계산한 (A_1,A_2,A_3,A_4,A_5)를 W_{i+1}에 추가한다: MW가 (길이가 t인) 두 입력 비트열 사이의 매칭비트들의 개수를 나타낸다고 하면
   $mw_1 = MW(x_i, p/2^{1+(i-1)t} \bmod 2^t)$
   $mw_2 = MW(y_i, q/2^{1+(i-1)t} \bmod 2^t)$
   $mw_3 = MW(z_i, d/2^{1+(i-1)t} \bmod 2^t)$
   $mw_4 = MW(u_i, d_p/2^{1+(i-1)t} \bmod 2^t)$
   $mw_5 = MW(v_i, d_q/2^{1+(i-1)t} \bmod 2^t)$
3. $xy=N$를 만족시키는 $(x,y,z,u,v) \in W_{\lceil n/2t \rceil}$가 존재하면 (x,y,z,u,v)를 출력하고 알고리즘을 종료한다.

HMM 알고리즘은 이전의 HS 알고리즘과 몇 가지 점에서 상이함을 알 수 있는데, 예를 들어 HS 알고리즘에서 올바른 개인키 sk는 항상 알고리즘의 출력 후보 집합인 $W_{\lceil n/2 \rceil}$에 포함되는 반면, HMM 알고리즘은 에러율 δ에 따라 출력 후보 집합 $W_{\lceil n/2t \rceil}$의 크기가 결정될 뿐만 아니라 sk가 $W_{\lceil n/2t \rceil}$에 포함되지 않을 수도 있다. 즉, δ가 너무 크면 $W_{\lceil n/2t \rceil} = \varnothing$이 될 수도 있고 또한 sk와 $\widetilde{sk}$의 비트들 중에서 서로 매칭되는 비트들의 개수가 특정 구간에서 C보다 작으면 sk가 $W_{\lceil n/2t \rceil}$에 포함되지 않을 수도 있다. 본 논문에서는 HMM 알고리즘이 사용했던 매칭 비트들의 개수에 기반한 확률 측도를 좀 더 일반화하고 이 일반화된 측도를 RSA 개인키 복구 알고리즘에 적용하는 방법을 제안한다.

III. RSA 개인키 복구 알고리즘

2장에서 살펴보았듯이 HMM 알고리즘은 기본적으로 두 비트열 사이에서 서로 매칭되는 비트의 개수를 이용하여 후보 키의 유효성을 검사한다. 그리고 본 논문에서는 이러한 매칭 비트의 수를 일반화한 확률 측도를 사용하여 분석을 진행함으로써 RSA 개인키 복구 알고리즘의 성능을 높이고자 한다. 이를 위해 먼저, 서로 독립이고 $\Pr(X_i=1)=1-\delta$를 만족시키는 베르누이 확률변수 $X_1, X_2, ..., X_n$가 주어졌을 때, n을 나누는 양의 정수 M과 $l=n/M$에 대하여 새로운 확률 변수 $S_k, k=1,...,l$를 다음과 같이 정의하자:

$$S_k = \left(\frac{\sum_{j=1}^{M} X_{(k-1)M+j}}{M} \right)^2$$

이 때 $S=\sum_{k=1}^{l} S_k$라고 하면 다음이 성립함을 쉽게 알 수 있다:

$$E[S] = \frac{l(1-\delta+(M-1)(1-\delta)^2)}{M}$$

여기서 $E[S]$는 확률변수 S의 기대값을 나타낸다.

보조정리 1 위에서 정의된 확률변수 S에 대하여 다음이 성립한다.

1) $\Pr(S \geq E[S]+l\gamma) \leq e^{-2l\gamma^2}$

2) $\Pr(S \leq E[S]-l\gamma) \leq e^{-2l\gamma^2}$

보조정리 1은 일반화된 Hoeffding bound라고 불리며[7], 이후의 분석에서 주된 역할을 하게 된다. 이제 보조정리 1을 바탕으로 하여 본 논문에서는 다음과 같은 RSA 개인키 복구 알고리즘을 제안한다.

Algorithm 3 (RSA 개인키 복구)

입력: 에러율 $\delta(0<\delta<1/2)$, 양의 정수 $t(>1)$, $5t$를 나눌 수 있는 양의 정수 M, 기준값 $C_M(>0)$, RSA 공개키 (N,e), RSA 개인키 $sk=(p,q,d,d_p,d_q)$의 전체 비트 중 δ의 비율에 해당하는 비트들에 에러가 삽입된 형태로 주어진 $\widetilde{sk}=(\tilde{p},\tilde{q},\tilde{d},\tilde{d}_p,\tilde{d}_q)$, N의 비트 크기 n

출력: $sk=(p,q,d,d_p,d_q)$

1. $W_1 \leftarrow \{(1,1,1,1,1)\}$;
2. For $i=1$ to $\left\lceil \frac{n}{2t} \right\rceil -1$
   1) $W_{i+1} \leftarrow \varnothing$;
   2) 모든 $(x,y,z,u,v) \in W_i$와 $T \in \{0,1,...,2^t-1\}$에 대하여 다음을 만족시키는 해 $(x_i,y_i,z_i,u_i,v_i) \in \{0,1,...,2^t-1\}^5$를 모두 계산한다:
   $x_i = T$
   $A_1 = x + x_i 2^{1+(i-1)t}$
   $A_2 = y + y_i 2^{1+(i-1)t}$
   $A_3 = z + z_i 2^{1+(i-1)t}$
   $A_4 = u + u_i 2^{1+(i-1)t}$
   $A_5 = v + v_i 2^{1+(i-1)t}$
   $A_1 A_2 = N \bmod 2^{1+it}$
   $eA_3 = 1 + k(A_1-1)(A_2-1) \bmod 2^{1+it}$
   $eA_4 = 1 + k_p(A_1-1) \bmod 2^{1+it}$
   $eA_5 = 1 + k_q(A_2-1) \bmod 2^{1+it}$
   3) 2)에서 구한 해 (x_i,y_i,z_i,u_i,v_i)와 (p,q,d,d_p,d_q)의 해당 비트들 사이의 매칭 여부(비트가 서로 같으면 1, 다르면 0)를 나타내주는 벡터 $(u_1,u_2,...,u_{5t}) \in \{0,1\}^{5t}$를 계산한다.
   4) $l=5t/M$에 대하여 $s_k = \left(\frac{\sum_{j=1}^{M} u_{(k-1)M+j}}{M} \right)^2$, $k=1,...,l$라 하고 $s=\sum_{k=1}^{l} s_k$라 했을 때 $s \geq C_M$이면 (A_1,A_2,A_3,A_4,A_5)를 W_{i+1}에 추가한다.
3. $xy=N$를 만족시키는 $(x,y,z,u,v) \in W_{\left\lceil \frac{n}{2t} \right\rceil}$가 존재하면 (x,y,z,u,v)를 출력하고 알고리즘을 종료한다.

알고리즘 3의 분석을 위해 본 논문에서는 다음과 같은 용어를 사용한다[6]. $(x,y,z,u,v) \in W_i$과 (p,q,d,d_p,d_q)가 다음을 만족시키면 (x,y,z,u,v)를 '올바른 부분키 후보'라고 하고, 그렇지 않은 경우에는 '틀린 부분키 후보'라고 한다:

$$\begin{aligned} x &= p \bmod 2^{1+(i-1)t} \\ y &= q \bmod 2^{1+(i-1)t} \\ z &= d \bmod 2^{1+(i-1)t} \\ u &= d_p \bmod 2^{1+(i-1)t} \\ v &= d_q \bmod 2^{1+(i-1)t} \end{aligned}$$

또한 알고리즘 3의 유효성을 증명하기 위해서 본 논문은 [6]에서 제시된 다음과 같은 가정을 필요로 한다.

가정 (x,y,z,u,v)가 '틀린 부분키 후보'이면, 알고리즘 3의 2.2)단계에서 (x,y,z,u,v)로부터 계산이 되는 해 (x_i,y_i,z_i,u_i,v_i)를 나타내는 확률변수는 이항분포 $Bin(5t,\frac{1}{2})$를 따른다.

정리 1. 임의의 $\epsilon>0$과 임의의 양의 정수 M에 대하여 다음이 성립한다. n비트 크기를 가지는 N에 대하여 (N,e)를 RSA 공개키라고 하고,

$t = M\left\lceil \frac{\ln(n)}{10\epsilon^2} \right\rceil$, $\gamma_0 = \sqrt{\frac{M(t+1)}{10t}\ln(2)}$,

$C = \frac{5t(M+1)}{4M^2} + \frac{5t}{M}\gamma_0$라 하자. 또한

$U = M - M\epsilon - \frac{M+1}{4} - M\gamma_0$ 에 대하여

$$\delta \le \begin{cases} \frac{2M-1-\sqrt{(2M-1)^2-4(M-1)U}}{2(M-1)}, & M>1 \\ \frac{1}{2}-\gamma_0-\epsilon & M=1 \end{cases}$$

라 하고, $\widetilde{sk} = (\widetilde{p}, \widetilde{q}, \widetilde{d}, \widetilde{d_p}, \widetilde{d_q})$를 δ의 확률로 각 비트에 에러가 삽입된 RSA 개인키라고 하면,

알고리즘 3은 평균적으로 $O(n^{2+\frac{\ln(2)}{5\epsilon^2}})$ 시간 안에, 적어도 $1-(\frac{5\epsilon^2}{\ln(n)}+\frac{1}{n})$의 확률로 $sk = (p, q, d, d_p, d_q)$를 복구할 수 있다. 여기서 $\ln(\,)$은 자연로그를 의미한다.

정리 1은 [6]에서 사용한 증명 방법과 유사한 방식으로 증명될 수 있으며, 이를 위해 먼저 Y_i를 알고리즘 3의 2단계에서 i번째 루프를 통과하는 '틀린 부분키 후보'의 개수를 나타내는 확률변수라고 하고, $\tau = \left\lceil \frac{n}{2t} \right\rceil - 1$에 대하여 $Y = \sum_{i=1}^{\tau} Y_i$라 하자.

보조정리 2 $E[Y_i] < 2^{t+1}$

증명) Z_g는 '올바른 부분키 후보'로부터 유도되는 다음 단계의 '틀린 부분키 후보'의 개수를, Z_b는 한 개의 '틀린 부분키 후보'로부터 유도되는 다음 단계의 '틀린 부분키 후보'의 개수를 나타내는 확률변수라고 하면 다음이 유도될 수 있다[6]:

$$E[Y_i] < \frac{E[Z_g]}{1-E[Z_b]}.$$

이제, 하나의 '틀린 부분키 후보'로부터 유도될 수 있는 2^t개의 예비 후보키에 대하여, 확률변수 $Z_b^i, i = 1, ..., 2^t$를, 만약 i번째 예비 후보키에 대한 (알고리즘 3의 2.4)단계의) 확률 측도값 s가 C_M보다 크거나 같으면 1의 값을, C_M보다 작으면 0의 값을 가지는 확률변수라고 하면

$$E[Z_b] = 2^t E[Z_b^i]$$

임을 알 수 있다. 또한 상기 가정에 의하면, $\Pr(X_i = 1) = 1/2$를 만족시키는 베르누이 확률변수 $X_1, X_2, ..., X_{5t}$와 $l = 5t/M$에 대하여 정의된

$$S_k = \left(\frac{\sum_{j=1}^{M} X_{(k-1)M+j}}{M} \right)^2, k = 1, 2, ..., l$$

와

$$S = \sum_{k=1}^{l} S_k$$

에 대하여

$$E[S] = \frac{5t(M+1)}{4M^2}$$

이고

$$E[Z_b^i] = \Pr(S \ge C_M)$$

임을 역시 알 수 있다. 따라서 보조정리 1을 적용하면

$$\Pr(S \ge C_M) = \Pr(S \ge E[S] + l\gamma_0) \le e^{-2l\gamma_0^2} \le 2^{-(t+1)}$$

이고, 따라서 $E[Z_b] = 2^t E[Z_b^i] \le \frac{1}{2}$이다. 마지막으로 $E[Z_g] \le 2^t - 1$이기 때문에

$$E[Y_i] < \frac{E[Z_g]}{1-E[Z_b]} < 2^{t+1}$$

임을 알 수 있다. ■

보조정리 3 알고리즘 3의 성공 확률은 적어도 $1-(\frac{5\epsilon^2}{\ln(n)}+\frac{1}{n})$이다.

증명) 알고리즘 3에서 '올바른 부분키 후보'가 2.4) 단계를 통과하지 못할 확률은, $\Pr(X_i=1)=1-\delta$를 만족시키는 베르누이 확률변수 $X_1, X_2, ..., X_n$에 대하여 정의된

$$S_k=\left(\frac{\sum_{j=1}^{M} X_{(k-1)M+j}}{M}\right)^2, k=1,2,...,l$$

와

$$S=\sum_{k=1}^{l} S_k$$

에 대하여 $\Pr(S<C_M)$과 같음을 알 수 있다. 따라서

$$E[S]=\frac{l(1-\delta+(M-1)(1-\delta)^2)}{M}$$

라는 사실과 보조정리 1을 적용하면

$$\Pr(S<C_M)=\Pr(S<\frac{5t(M+1)}{4M^2}+\frac{5t}{M}\gamma_0)$$
$$\le \Pr(S<l(\frac{1-\delta+(M-1)(1-\delta)^2}{M}-\epsilon))$$
$$\le e^{-2l\epsilon^2} \le \frac{1}{n}$$

이다. 마지막으로 알고리즘 3의 2단계는 $\tau \le \frac{n}{2t}+1$ 번을 반복하기 때문에 알고리즘 3의 성공확률 $\Pr$은 다음을 만족시킨다:

$$\Pr=(1-\Pr(S<C_M))^\tau$$
$$\ge (1-\frac{1}{n})^\tau \ge 1-\frac{\tau}{n}$$
$$\ge 1-(\frac{1}{2t}+\frac{1}{n})$$
$$=1-(\frac{5\epsilon^2}{\ln(n)}+\frac{1}{n})$$

■

정리1의 증명) [6]에 의하면, 알고리즘 3의 평균 연산 시간은 $O(n^{2+\frac{\ln(2)}{5\epsilon^2}})$임을 알 수 있다. 따라서 보조정리 2와 3을 이용하면 정리가 증명된다. ■

참고 정리 1에서 $M=1$이면 [6]의 정리 3과 동일한 결과를 얻을 수 있다. 따라서 정리 1은 [6]에서 제시한 결과를 포함함을 알 수 있다. 또한, 다음 소절에서 주어지는 실험결과에 의하면, 정리 1의 $M>1$인 경우는 $M=1$인 경우와 확률적으로 서로 독립인 것으로 예측이 된다. 즉, 실험 결과에 의하면, $M=1$에 대한 키후보 유효성 검사 (알고리즘 3의 2.4)단계)의 통과여부는 $M>1$에 대한 키후보 유효성 검사의 통과여부와 확률적으로 서로 독립인 것으로 보인다. 따라서 $M=1$ 만을 이용한 HMM 알고리즘과는 달리, 만약 여러 M 값을 이용해서 후보 키의 유효성을 검사하게 된다면 좀 더 정확하게 RSA 개인키를 복구할 수 있을 것으로 예상이 되며 이러한 점이 정리 1의 가장 큰 장점이 된다.

3.1 실험결과

정리 1을 검증하기 위해서 본 논문은 2가지 실험을 수행하였으며, 이 실험들에 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

실험을 위해서는 기본적으로 RSA 연산을 수행해야 하는데, 이를 위해서는 반드시 Big Number 연산이 필요하며 본 논문에서는 이러한 Big Number 연산을 위해서 NTL 라이브러리를 사용하였다[8]. 또한 RSA 개인키에 랜덤한 에러를 삽입하기 위해서는 난수(random number)를 생성해야 하는데 C 언어에서 사용되는 rand()함수의 랜덤성은 그 수준이 매우 낮다고 알려져 있다. 따라서 이를 해결하기 위해서 본 논문은 먼저 rand()함수를 사용하여 seed 값을 만들고, 이렇게 만들어진 seed 값에 SHA-256 해시함수[9]를 적용하여 실험에 필요한 의사난수열을 생성하는 방법을 사용하였다.

본 논문의 첫 번째 실험은 알고리즘 3의 성공 확률을 알아내는 것이 목적이었으며, 이를 위하여 '올바른 부분키 후보'가 알고리즘의 2.4)단계를 모두 통과하는 확률을 측정하였다. 그리고 이러한 실험을 위해서 먼저 512-비트 크기를 가지는 서로 다른 두 소수 p, q를 생성한 후 $e = 65537$에 해당하는 RSA 개인키 (p, q, d, d_p, d_q)를 계산하였다. 그리고 나서 주어진 δ값에 해당하는 비율로 (p, q, d, d_p, d_q)의 각 비트들을 변형시킨 $\widetilde{sk} = (\widetilde{p}, \widetilde{q}, \widetilde{d}, \widetilde{d_p}, \widetilde{d_q})$를 계산한 후, 원래의 RSA 개인키 (p, q, d, d_p, d_q)의 비트들이 알고리즘 3의 2.4)단계를 통과하는지 여부를 검사하였다. 이러한 과정은 100번 반복되었으며 그 결과는 Table 1에 주어진다. 실험에 사용된 파라미터 δ, t, M, C_M의 선택은, 먼저 $M = 1$인 경우 [6]에서 주어진 파라미터를 참고하였으며, $M > 1$의 경우에는 정리 1을 바탕으로 자체적으로 선택되었다. Table 1에서 M값이, 가령 '1+3'으로 주어진 것은 알고리즘 3의 2.4) 단계에서 $M = 1$인 경우와 $M = 3$인 경우의 s를 모두 계산한 후 그 값들이 각각 $s \geq C_1$과 $s \geq C_3$을 만족하는지를 검사하였음을 의미한다. 흥미롭게도 M값이 '1+3'인 경우의 성공 확률은, $M = 1$인 경우의 성공 확률과 $M = 3$인 경우의 성공 확률의 곱과 거의 비슷하게 나오고 있으며, 이것은 알고리즘 3의 2.4)단계에 주어진 확률측도 s는 M 값에 따라 서로 독립적으로 행동함을 보여준다고 추측할 수 있다.

알고리즘 3은 최종적으로 RSA 개인키 후보를 출력하게 되는데, 두 번째 실험은 알고리즘 3이 출력하는 후보키의 개수에 관한 것이다. 이를 위해 이전 실험과 마찬가지로, 먼저 512-비트의 서로 다른 두 소수 p, q를 생성한 후 $e = 65537$에 해당하는 RSA 개인키 (p, q, d, d_p, d_q)를 계산하고 나서 주어진 δ의 확률로 (p, q, d, d_p, d_q)의 각 비트들을 변형시킨 후 알고리즘 3을 수행하였다. Table 2는 이러한 과정 후에 알고리즘 3이 출력하는 후보 키의 개수에 대한 통계를 보여준다.

Table 1. Success Probabilities of Algorithm 3 for Various δ, t, M and C_M

δ	0.10			0.12		
t	9			10		
M	1	3	1+3	1	2	1+2
C_M	36	99	36+99	39	72	39+72
Succ. Prob. (%)	66	71	45	54	47	38

Table 2. Output Candidate Numbers of Algorithm 3 for various δ, t, M and C_M

δ	0.10		0.12	
t	9		10	
M	1	1+3	1	1+2
C_M	36	36+99	39	39+72
Mean	5.0	4.9	9.1	7.7
St. Dev.	5.3	5.9	10.4	12.8

Table 2가 보여주듯이, 알고리즘 3의 2.4) 단계에서 단일한 M을 사용한 후보키 검사보다는 두 개의 M을 사용한 후보키 검사 방법이 최종 출력된 후보 키의 개수를 줄이는 효과를 주고 있음을 알 수 있으며, 따라서 이러한 성질을 이용하면 알고리즘 3의 연산 시간이나 성능을 최적화할 수 다양한 방안이 도출될 수 있을 것으로 예상이 된다.

IV. 결 론

본 논문에서는 Henecka 등이 제안한 알고리즘에서 키 유효성 검사를 위해 사용된 매칭비트의 개수 방법을 일반화한 방법을 제안하고 이렇게 일반화된 확률 측도를 사용하여 RSA 개인키를 복구하는 알고리즘과 이에 대한 실험 결과를 제시하였다.

References

[1] R. Rivest, A. Shamir, and L. 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] RSA Laboratories, "PKCS #1 v2.2: RSA Cryptography Standard," Oct. 2012.

[3] D. Coppersmith, "Small solutions to polynomial equations, and low exponent RSA vulnerabilities," Journal of Cryptology, vol. 10, no. 4, pp. 233 - 260, Sep. 1997.

[4] D. Boneh, G. Durfee, and N. Howgrave-Graham, "Factoring $n = p^r q$

for large r," Advances in Cryptology, CRYPTO '99, LNCS 1666, pp. 326-337, 1999.
[5] N. Heninger and H. Shacham, "Reconstructing rsa private keys from random key bits," Advances in Cryptology, CRYPTO '09, LNCS 5677, pp. 1 - 17, 2009.
[6] W. Henecka, A. May, and A. Meurer, "Correcting errors in RSA private keys," Advances in Cryptology, CRYPTO '10, LNCS 6223, pp. 351-369, 2010.
[7] W. Hoeffding, "Probability inequalities for sums of bounded random variables," Journal of the American Statistical Association, vol. 58, no. 301, pp. 13 - 30, 1963.
[8] NTL: A library for doing number theory, available at www.shoup.net/ntl
[9] National Institute of Standards and Technology, "Secure Hash Standard (SHS)," FIPS PUB 180-4, Mar. 2012.

〈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