情報保護學會論文誌
第20卷 第4號, 2010. 8

연산자 조작 공격에 대한 CRT-RSA 알고리듬의 안전성 재분석*

하 재 철[1†], 백 이 루[1], 박 제 훈[2], 문 상 재[2‡]
[1]호서대학교, [2]경북대학교

Security Reconsideration on CRT-RSA Algorithm Against Fault Attacks using Opcode Modification*

Jae-Cheol Ha[1†], Yi-Roo Baek[1], JeaHoon Park[2], SangJae Moon[2‡]
[1]Hoseo University, [2]Kyungpook National University

요 약

최근 오류주입 공격은 중국인의 나머지 정리(Chinese Remainder Theorem)를 이용한 RSA 암호 시스템(CRT-RSA)의 안전성에 심각한 위협이 되고 있다. 따라서 오류주입 공격에 강인한 대응 암호 알고리듬들이 개발 되었고 다양한 오류 검사 방법이나 오류 확산을 이용한 대응책이 제시된 바 있다. 그러나 최근 Hur 등은 연산자 조작을 이용한 오류주입 공격 시 Kim 등이나 Ha 등이 제안한 알고리듬이 공격될 수 있다고 분석하였다. 본 논문에서는 이 연산자 조작 공격이 다정도(multi-precision) 연산을 하는 CRT-RSA 알고리듬에는 적용할 수 없으며 Kim 등이나 Ha 등이 제안한 알고리듬도 여전히 안전함을 밝히고자 한다.

ABSTRACT

Since the RSA cryptosystem based on Chinese Remainder Theorem is vulnerable to many fault insertion attacks, some countermeasures against them were proposed. Recently, Kim et al. or Ha et al. respectively proposed each countermeasure scheme based on fault propagation method. Unfortunately, Hur et al. insist that these countermeasures are vulnerable to their opcode modification fault attack. In this paper, we show that the proposed attack can not apply to almost CRT-RSA countermeasures which use multi-precision operations in long bit computation. Therefore, the countermeasure against fault attack proposed by Kim et al. or Ha et al. are still secure.

Keywords: CRT-RSA, Fault attack, Opcode modification, Multi-precision operation

I. 서 론

RSA 암호 시스템은 현재 보안과 관련된 많은 응용 분야에 사용되고 있는 공개키 암호 알고리듬이다[1]. RSA 암호 시스템을 구현할 경우에는 계산 효율성을 고려하여 중국인의 나머지 정리(Chinese Remainder Theorem)를 이용한 방법(CRT-RSA)이 널리 사용되고 있다[2]. 하지만 CRT-RSA는 단 한 번의 오류주입만으로도 저장된 비밀키를 쉽게 노출될 수 있음이 밝혀졌다[3, 4]. 따라서 Shamir에 의해 오류주입 공격 대응 방법[5]이 제안된 이후 많은 대응 기법들이 연구되었다. 이 중에서 2008년 Ha 등은 한국정보보호학회 논문에서 오류 확산 기법을 이용한 오류주

* 접수일(2010년 1월 29일), 수정일(2010년 3월 19일), 게재확정일(2010년 4월 14일)
* 이 논문은 2009학년도 경북대학교 연구교수 연구비에 의하여 연구되었음.
† 주저자, jcha@hoseo.edu
‡ 교신저자, sjmoon@ee.knu.ac.kr

입 공격 대응책을 제시한 바 있다[6]. 또한 Kim 등도 Yen 등의 기법[7]을 개선한 대응책을 제시하였다[8]. 이 대응책들은 CRT-RSA 연산시 오류가 주입되더라도 이 오류가 서명문 전체에 확산되어 비밀키를 찾아내지 못하도록 하는 기법을 사용하였다. 그러나 허순행 등은 연산 도중 특정 연산자를 건너뛰는 연산자 조작 오류공격(opcode modification fault attack)을 가정하면서 이 두 알고리듬을 비롯한 많은 CRT-RSA 알고리듬들의 비밀키를 추출할 수 있다고 주장하였다[9]. 여기에서는 이들이 주장한 공격을 "Hur 등의 공격"이라 약칭하기로 한다. 본 논문에서는이들이 제시한 연산자 오류 공격이 실질적으로 다정도(multi-precision) 연산을 하는 CRT-RSA 알고리듬에는 적용이 불가능하며 저자들이 비밀키를 찾을 수 있다고 주장하는 대부분의 알고리듬들이 오류 주입 공격에 여전히 안전함을 보이고자 한다.

II. Kim 등의 CRT-RSA 오류주입 공격 대응 방식

2.1 CRT-RSA 암호 알고리듬과 오류주입 공격

RSA 암호 시스템에서는 두 큰 소수 p와 q를 선택하여 합성 수 $n=p \cdot q$를 구하고, $\gcd(\phi(n), e)=1$이 되는 공개키 e를 선택하고 $e \cdot d \equiv 1 \bmod \phi(n)$을 만족하는 비밀키 d를 구한다. RSA 암호 시스템에서는 입력 메시지 $m \in Z_n$에 대한 서명이나 복호 연산은 다음과 같은 멱승을 수행한다. 편의상 논문에서는 이 연산을 간단히 서명 과정이라고 칭하고 최종 출력을 서명 값이라 한다.

$$s = m^d \bmod n$$

서명문을 구하는데 있어 연산 효율성을 높이기 위해 CRT 기법을 적용하게 되는데 CRT-RSA 연산 방법은 아래와 같이 정리된다.

$$s_p = m^{d_p} \bmod p, \quad s_q = m^{d_q} \bmod q$$
$$s = CRT(s_p, s_q)$$

단, $d_p = d \bmod (p-1)$이고 $d_q = d \bmod (q-1)$이다. Garner 방법을 사용한 CRT 재결합(recombination) 단계는 다음과 같다.

$$s = CRT(s_p, s_q)$$
$$= s_q + q \cdot ((s_p - s_q) \cdot i_q \bmod p)$$

단, 여기서 $i_q = q^{-1} \bmod p$이다.

하지만 상기한 CRT-RSA 서명 시스템은 1997년 Boneh 등에 의해 오류 주입 기술을 적용하면 비밀키가 노출될 수 있음이 밝혀졌다[3]. 이들이 제안한 공격 방법은 CRT-RSA 연산 과정에서 s_p나 s_q 중 어느 하나에 오류를 주입하여 오류가 주입된 서명 s'을 얻을 수 있을 경우 비밀키를 계산해 내는 공격이다. 즉, 정상 서명 값이 s이고, s_p를 연산할 때 오류가 주입되어 생성된 값을 s_p'라고 하면, $GCD(s-s', n)$을 계산함으로써 비밀 소수 q를 구할 수 있다. 또한 Lenstra는 하나의 오류 서명만으로도 $GCD(s'^e - m, n)$와 같이 비밀 키를 찾아낼 수 있음을 보였다[4]. 이 Lenstra의 공격을 흔히 Bellcore 공격이라고 부른다.

2.2 Kim 등의 오류주입 공격 대응 방법

2008년 Kim 등에 의해 Yen 의 방식[7]을 개선한 오류주입 공격 대응책이 제시되었다[8]. 본 논문에서는 개선된 두 가지 방법을 CRT-1과 CRT-2라고 부르고 안전성과 공격 방법이 두 방법에 모두 동일하게 적용되므로 편의상 CRT-1을 중심으로 설명하고자 한다. Kim 등의 대응 알고리듬 CRT-1은 [그림 1]과 같으며 여기에 사용되는 정의는 다음과 같고 r은 작은 랜덤한 값이다.

$$d_r = d - r, \quad e_r = d_r^{-1} \bmod \phi(N).$$

처음 Yen이 제시한 대응책에서는 [그림 1]의 단계 3에서 $\tilde{m}$의 연산은 아래 식과 같았다.

$$\tilde{m} = (s_q^{e_r} \bmod q + k_q \cdot q) \quad (1)$$

Input : A message $m \in Z_n$
Output : $Sig := m^d \bmod n$

1. Compute both
   $k_p = \left\lfloor \frac{p}{m} \right\rfloor$ and $k_q = \left\lfloor \frac{q}{m} \right\rfloor$
2. Compute $m^{d_r} \bmod n$ via a CRT speedup as
   $s_p = m^{d_r \bmod (p-1)} \bmod p$
   $s_q = \tilde{m}^{d_r \bmod (q-1)} \bmod q$
   where $\tilde{m} = ((s_p^{e_r} \bmod p) + k_p \cdot p) \bmod q$
3. Compute the required signature as
   $s = CRT(s_p, s_q) \cdot \tilde{m}^r \bmod n$
   where $\tilde{m} = m \wedge (s_q^{e_r} \bmod q + k_q \cdot q)$
4. Output s

[그림 1]. Kim 등이 제안한 CRT-1 알고리듬

그러나 원래 Yen의 방식에서 k_q 연산 과정에서 오류가 주입되어 비밀 키가 노출되는 공격[10]을 방지하기 위해 Kim 등의 개선안에서는 다음과 같은 수식으로 바뀌게 되었다.

$$\tilde{m}=m\wedge(s_q^{e_r} \bmod q+k_q \cdot q) \quad (2)$$

즉, 기존의 $\tilde{m}$값에 원본 메시지 m으로 논리적 AND 연산을 추가하였다. 이 대응 기법은 AND 연산의 특징 중 $m\wedge m=m$이 되지만 $m\wedge x=R$이 된다는 성질을 이용한 것이다. 이렇게 함으로써 k_q에 오류가 주입되어도 비밀키를 알 수 없도록 개선한 것이다.

III. Hur 등의 연산자 조작 공격

최근 Hur 등은 한국정보보호학회 논문에서 Kim 등이나 Ha 등이 제안한 알고리듬이 연산자 오류 공격에 취약하다고 분석하였다. Hur 등은 AND나 XOR와 같은 연산자를 사용하는 경우 어셈블리 언어를 분석하여 이를 건너뛸 수 있는 오류주입 공격이 가능하다고 제안하였다. 그리고 그 공격을 Kim 등의 방식이나 Ha 등이 제시한 방식에 적용하여 취약한 특성이 있다고 주장하였다.

여기서 저자들은 Kim 등이 제안한 방식에서 단계 3의 $\tilde{m}=m\wedge(s_q^{e_r} \bmod q)+k_q \cdot q)$연산을 어셈블리 코드로 다음과 같이 간단히 나타낼 수 있다고 하였다.

Step 1. MOVE eax DWORD PTR $(s_q^{e_r} \bmod q+k_q q)$
Step 2. MOVE ebx DWORD PTR m
Step 3. AND eax, ebx
Step 4. MOVE DWORD PTR $\tilde{m}$, eax

여기서 eax, ebx는 범용 레지스터로서 일반적으로 4바이트 크기를 가진다. 위 어셈블리 코드에 따르면 Step 1에서 eax에 $(s_q^{e_r} \bmod q+k_q q)$값이 저장되고 Step 2에서 ebx에 m값이 저장되며 Step 3에서 논리적 AND 연산을 한다고 가정하였다. 그리고 Step 3을 오류 주입을 통해 건너뛰면 바로 Step 4의 저장 과정에서 $\tilde{m}$에 $(s_q^{e_r} \bmod q+k_q q)$이 저장되게 되고 이렇게 하면 기존 Yen 등이 제안한 방식과 동일한 취약점을 가진다고 분석하였다. Yen 등의 방식에서의 공격은 k_q에 대한 피연산자 오류를 주입하는 것이므로 Hur 등의 공격에서는 k_q에 대한 피연산자 오류와 AND 연산자를 건너뛰는 2차 오류주입 공격을 가정하였다.

IV. 연산자 조작 공격에 대한 재분석

그러나 위에서 제시한 연산자 오류 공격은 중대한 분석 오류를 가지고 있다. 이 원인은 RSA 또는 CRT-RSA에서 사용하는 메시지나 파라미터의 길이로 인해 연산 과정이 다정도(multi-precision) 연산을 해야 한다는 점이다. 즉, 일반적으로 RSA에서는 최소의 안전도를 위해 소수 p나 q는 512비트 이상을 사용하고 메시지 m도 약 1024 비트를 사용한다. 그러므로 m에 대한 실제 연산을 위해서는 m을 $m=\sum_{i=0}^{k} m_i b^i$와 같이 여러 개의 디지트로 나누어 처리한다. 예를 들어 m이 1024비트(l비트)이고 데이터의 처리 단위가 16비트($n=16, b=2^n$)라면 64개의 자리수($k=l/n=64$)를 갖게 된다. 따라서 1024비트의 메시지에 대한 AND 연산을 해야 하므로 실제로는 아래 수식에서 표현한 것처럼 연산 단위에 따라 수십 번의 단정도(single-precision) AND 연산의 반복으로 이루어지게 된다.

$$\tilde{m}=m\wedge(s_q^{e_r} \bmod q+k_q \cdot q) \quad (3)$$

따라서 CRT-RSA 시스템의 다정도 연산을 고려하면, Hur 등의 공격이 성공하기 위해서는 m의 디지트 개수 길이만큼의 논리적 AND 연산을 모두 건너뛰어야 한다.

위의 식 (3)의 다정도 연산을 소프트웨어로 구성해 보면 아래와 같은 단정도 연산의 반복문으로 구성할 수 있다. 여기서 *Temp*는 $s_q^{e_r} \bmod q+k_q \cdot q$ 값을 의미하며 *Temp*[*i*]는 *Temp*의 각 자리 값이다.

```
for(i=0 ; i<k ; i++) {
    m̃[i] = m[i] ∧ Temp[i]
}
```

이와 같이 실제로 CRT-RSA에 사용하는 큰 정수에 대한 AND 연산은 Hur 등이 분석한 것처럼 하나의 디지트에 단 한번의 AND 연산으로 처리할 수 없으며 수십 번의 단정도 AND 연산의 반복으로 처리된다.

본 논문에서는 위의 for문을 이용한 다정도 계산 과정을 어셈블리어로 작성해 분석하였으며 자세한 코

드는 [그림 2]에 나타내었다. [그림 2]의 어셈블리 코드는 식 (3)의 다정도 연산을 C-언어로 구현한 후 컴파일한 것으로 Hur 등이 제시한 컴파일러의 어셈블리 구조와 거의 유사하다. 따라서 [그림 2]에서 음영으로 표시된 4줄의 어셈블리어 코드는 Hur 등의 분석 코드 Step 1에서 Step 4까지에 해당한다고 볼 수 있다.

[그림 2]를 분석해 보면 연산을 위해 for문이 인덱스 i값의 증가에 따라 반복되는 것을 볼 수 있다. 여기에서는 m이 1024비트(l비트)이고 데이터의 처리 단위가 32비트이고 32개의 자리 수로 이루어져 있다고 가정하였다. 실제로 어셈블리 코드는 맨 마지막 줄(명령어 주소 004010DD)에서 for구문의 인덱스 증가를 위한 위치(명령어 주소 004010b3)로 반복적으로 이동하는 것을 것을 볼 수 있다.

그러나 Hur 등이 제시한 어셈블리 코드를 보면 데이터 처리 단위가 DWORD 형태(32비트 혹은 64비트)의 디지트만 처리하도록 되어 있다. 그러므로 그들이 제안한 한 번의 AND 연산은 한 디지트에 대한 처리밖에 할 수 없으므로 Step 3에 오류를 주입한다고 해서 약 1024비트의 $\tilde{m}$에 $(s_q^{e_r} \bmod q + k_q q)$을 그대로 저장한다는 것은 불가능하다. 즉, Hur 등의 공격의 단계 1이나 2에서와 같이 하나의 레지스터 eax나 ebx에 1024 비트 정도의 $(s_q^{e_r} \bmod q + k_q q)$나 m을 저장할 수 없으므로 다정도 연산을 해야 하지만 모든 다정도 AND 연산을 오류 주입에 의해 건너뛸 수는 없다. 즉, [그림 2]의 어셈블리 코드에서 명령어 주소 004010D4를 수십 번(예제에서는 32번)의 규칙적인 AND 연산을 모두 건너뛰는 것도 불가능하다. 최악의 경우를 가정하여 [그림 2]의 어셈블리 코드 중 004010D4 번지 코드를 손상시켜 건너뛴다 하더라도 변수 $Temp[i]$와 $m[i]$의 사용 순서를 바꾸면 $\tilde{m}$에는 $(s_q^{e_r} \bmod q + k_q q)$가 아닌 m이 저장되도록 변수를 조정할 수 있다. 이 경우에는 [그림 1]에서 사용된 r값을 모르는 이상 오류 주입 공격은 불가능하다.

결국, Hur 등이 제안한 연산자 오류 공격은 16비트 혹은 32비트와 같은 단정도 AND 연산에만 오류 주입이 가능하므로 실제로 1024 비트 데이터를 사용하는 CRT-RSA에는 적용할 수 없다. 따라서, Kim 등 혹은 Ha 등의 방식을 비롯한 저자들이 공격 가능하다는 주장하던 대부분의 CRT-RSA 알고리듬들은 연산자 오류 공격에 모두 안전하다.

```
■ for(i=0 ; i<k ; i++)                          // C 언어 실제코드

명령어 주소 어셈블리어
004010AA   mov dword ptr [ebp-28h],0                    // i를 0으로 고정
004010B1   jmp  main+0ACh (004010bc)          // 004010bc로 이동
004010B3   mov eax,dword ptr [ebp-28h]        // 인덱스 i값 eax 로 이동
004010B6   add  eax,1                         // i= i+1
004010B9   mov dword ptr [ebp-28h],eax        // 증가된 i값을 이동
004010BC   cmp dword ptr [ebp-28h],20h        // 0x20(십진수 32)과 비교
004010C0   jge  main+0CFh (004010df)          // 004010df(for 구문 밖)번지로 이동

■   {
■       m̃[i] = m[i] ∧ Temp[i] ;                  // C-언어 실제코드
■   }

004010C2   mov ecx,dword ptr [ebp-28h]        //ebp-28h에는 i값이 저장되어 있음 m[i]
004010C5   xor  edx,edx                       //edx 레지스터를 0으로 초기화
004010C7   mov dl,byte ptr [ebp+ecx-0Ch]      //레지스터 edx의 오른쪽(최하위)에 m[i] 이동
004010CB   mov eax,dword ptr [ebp-28h]        //ebp-28h에는 i값이 저장되어 있음 temp[i]
004010CE   xor  ecx,ecx                       //ecx 레지스터를 0으로 초기화
004010D0   mov cl,byte ptr [ebp+eax-18h]      //레지스터 ecx의 오른쪽(최하위)에 temp[i] 이동
004010D4   and  edx,ecx                       //m[i] & temp[i]를 edx에 저장
004010D6   mov eax,dword ptr [ebp-28h]        //ebp-28h에는 i값이 저장되어 있음 m̃[i]
004010D9   mov byte ptr [ebp+eax-24h],dl      //edx에 저장된 결과 값을 m̃[i]로 이동
004010DD   jmp main+0A3h (004010b3)           //004010b3(for 구문 처음)로 이동
```

[그림 2] 다정도 AND 연산에 대한 어셈블리 코드

V. 결론

본 논문에서는 Hur 등이 제안한 연산자 오류주입 공격에 대한 가능성을 재분석하였다. 아주 정밀도를 요하는 오류 주입 장비로 피연산자 공격과 연산자 공격을 가정하였지만 연산자 오류 공격은 CRT-RSA가 큰 소수에 대한 다정도 연산을 수행하는 점을 고려하면 한번의 단정도 연산자 동작을 건너뛰어 공격한다는 것은 불가능하다. 따라서 Kim 등이나 Ha 등이 제안한 CRT-RSA 방식을 비롯한 대부분의 알고리듬은 연산자 조작 공격에 안전성을 유지하고 있다.

참고문헌

[1] R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-key Cryptosystems," ACM, Vol. 21, pp. 120-126. 1978.

[2] C. Couvreur, J. Quisquater, "Fast Decipherment Algorithm for RSA Public-Key Cryptosystem," Electronics Letters, Vol. 18, pp. 905-907, 1982.

[3] D. Boneh, R. DeMillo, and R. Lipton, "On the Importance of Checking Cryptographic Protocols for Fault," EUROCRYPT'97, LNCS 1233, pp. 37-51, Springer-Verlag, 1997.

[4] A. K. Lenstra, "Memo on RSA signature generation in the presence of faults," Sept. 1996.

[5] A. Shamir, "Method and Apparatus for Protecting Public Key Schemes from Timing and Fault attacks," US Patent 5991415, 23, Nov. 1999.

[6] 하재철, 박제훈, 문상재, "오류 확산 기법을 이용한 CRT-RSA 오류주입 공격 대응방안," 한국정보보호학회 논문지, 18(2), pp. 75-84, 2008년 4월.

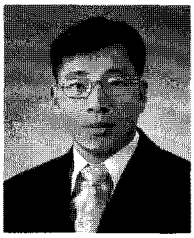
[7] S. Yen, S. Kim, S. Lim, and S. Moon, "RSA speedup with Chinese Remainder Theorem Immune Against Hardware Fault Cryptanalysis," IEEE Transaction on Computer, Special issue on CHES, Vol. 52, No. 4, pp. 461-472, 2003.

[8] 김성경, 김태현, 한동국, 박영호, 홍석희, "비교연산을 사용하지 않는 오류주입 공격에 안전한 CRT 기반의 RSA," 한국정보보호학회 논문지, 18(4), pp. 17-25, 2008년 8월.

[9] 허순행, 이형섭, 이현승, 최동현, 원동호, 김승주, "연산자 조작 공격과 피 연산자 조작 공격에 대한 기존 CRT-RSA Scheme의 안전성 분석," 한국정보보호학회 논문지, 19(6), pp. 185-190, 2009년 12월.

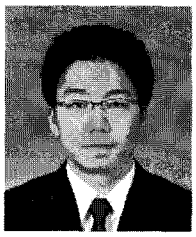
[10] S. Yen, D. Kim, and S. Moon, "Cryptanalysis of Two Protocols for RSA with CRT Based on Fault Infection," FDTC'06, LNCS 4236, pp. 53-61, Springer-Verlag, 2006.

〈著者紹介〉

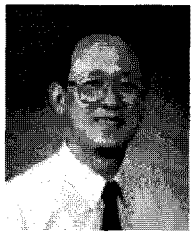


하 재 철 (Jae Cheol Ha) 종신회원
1989년 2월: 경북대학교 전자공학과 졸업
1993년 8월: 경북대학교 전자공학과 석사
1998년 2월: 경북대학교 전자공학과 박사
1998년 3월~2006년 1월: 나사렛대학교 전자계산소장, 학술정보관장, 입시학생처장
1998년 3월~2007년 2월: 나사렛대학교 정보통신학과 부교수
2006년 7월~2006년 12월: QUT in Australia 연구 교수
2007년 3월~현재: 호서대학교 정보보호학과 부교수
2002년 3월~현재: 한국정보보호학회 이사, 논문지 편집위원
2009년 1월~현재: 한국산학기술학회 이사
〈관심분야〉 정보보호, 네트워크 보안, 스마트카드 보안

백 이 루 (Yi Roo Baek) 학생회원
2008년 8월: 호서대학교 정보보호학과 졸업
2008년 9월~현재: 호서대학교 정보보호학과 석사과정
〈관심분야〉 네트워크 보안, 프로토콜, 스마트 카드 보안



박 제 훈 (JeaHoon Park) 학생회원
2004년 2월: 경북대학교 전자·전기공학부 졸업
2006년 2월: 경북대학교 전자공학과 석사
2006년 3월~현재: 경북대학교 전자공학과 박사과정
〈관심분야〉 정보보호, 네트워크 보안, 스마트카드 보안



문 상 재 (SangJae Moon) 종신회원
1972년 2월: 서울대학교 공업교육(전자전공)과 학사
1974년 2월: 서울대학교 전자공학과 석사
1984년 6월: 미국 UCLA 전기공학과 박사
1984년 7월~1985년 6월: UCLA Postdoctor 근무
1984년 7월~1985년 6월: 미국 OMNET 컨설턴트
1997년 9월~1998년 8월: 경북대학교 전자전기공학부 학부장
1974년 12월~현재: 경북대학교 전자전기컴퓨터공학부 교수
2000년 8월~현재: 경북대학교 이동네트워크 정보보호기술 연구센터장
2002년 2월~현재: 한국정보보호학회 명예회장
〈관심분야〉 정보보호, 디지털 통신, 이동 네트워크