RSA-CRT 의 구현과 비교 분석*

은하수 [1†], 이훈정 [1], 오희국 [1], 김상진 [2‡]
[1] 한양대학교 컴퓨터공학과
[2] 한국기술교육대학교 컴퓨터공학과
e-mail : hseun@infosec.hanyang.ac.kr

Implementation and Comparative Analysis of RSA-CRT*

Hasoo Eun[1†], Hoonjung Lee[1], Heekck Oh[1], Sangjin Kim[2‡]
[1]Dept. of Computer Science, Hanyang University
[2] Dept. of Computer Science, Korea University of Technology and Education

요 약

RSA 는 인수분해의 어려움에 기반한 공개키 암호 시스템으로, 여러 분야에서 사용되고 있지만 연산속도가 느리다는 단점이 있다. 이를 개선하기 위해 RSA-CRT 기법이 제안되었다. 법의 크기를 줄여 연산 속도가 빠르다는 장점이 있는 반면 오류주입공격에 취약하다는 단점이 있다. 이를 보완하기 위한 여러 기법들이 제안되었다. 하지만 제안된 기법들은 연산과정이 상대적으로 복잡하고 부가적인 연산에 따른 부담이 생겼다. 본 논문에서는 RSA-CRT 를 구현하여 사용되는 인자들의 크기에 따른 연산시간을 정량적으로 비교하고, 가상적으로 각 인자에 오류주입공격을 취해봄으로써 소수 추출에 약한 부분을 찾고 향후 연구 방향을 모색한다.

1. 서론

RSA 는 인수분해의 어려움에 기반한 공개키 암호 시스템으로, 1978 년 제안된 이래 여러 분야에서 사용되고 있다(이하 “RSA78”). 하지만 키의 크기가 커질수록 지수 연산이 증가하여 속도가 느려진다는 단점이 있다. 이를 개선하기 위한 여러 기법 중 하나로 중국인 나머지 정리(CRT, Chinese Remainder Theorem)를 이용하여 법의 크기를 줄이기 위한 연구가 진행되고 있다.

1982 년 처음으로 제안된 RSA-CRT(이하 “RSA82”)는 법 $n(=p.q)$에서의 연산을 법 p와 q에서 각각 진행한 후 CRT 연산을 이용하여 결합하는 기법이다. 법의 크기를 줄이면 멱승의 처리가 빨라지므로 연산 속도를 높일 수 있다. 현재 OpenSSL, Java, .Net 등의 라이브러리에서 이를 지원하고 있으며, 주로 서명과 복호화에 사용되고 있다.

RSA82 는 RSA78 에 비해 4 배이상 빠른 연산이 가능한 장점이 있지만, 오류주입공격에 취약하다는 문제가 있어 이후 이를 보완하기 위한 여러 기법들이 제안되었다. 1999 년 Shamir 는 법 p와 q에 공통법 r을 곱하여 서명 값을 구한 후, 비교를 통해 오류가 없는 경우 CRT 로 연산하는 기법(이하 “RSA99”)을 제안하였다. 2003 년 Yen 등은 비교연산이 오류주입공격에 대안이 될 수 없음을 보이고, 오류확산에 기반한 대응기법(이하 “RSA03”)을 제안하였다. 하지만 이 기법도 동일한 저자에 의해 오류주입공격에 취약함이 증명되었으며, 이후 오류주입공격을 방지하기 위한 여러 기법들이 제안되었다.

위와 같은 기법을 제안한 논문들에는 자신들이 제안한 기법이 몇 배 빠르며 안정적이라는 표현만 있을 뿐, 얼마나 빠른지, 공통법 r의 크기가 미치는 영향은 얼마인지 등에 대한 정량적 분석이 제외되어 있다. 본 논문에서는 위와 같은 기법들을 구현하여 각 기법간의 연산시간 차이를 분석하고, 공통법 r이 연산에 미치는 영향에 대해서도 분석한다. 또한 가상적으로 각 인자에 오류주입공격을 취해봄으로써 소수 추출에 약한 부분을 찾고 향후 연구 방향을 모색한다.

본 논문의 구성은 다음과 같다. 먼저 2 장에서 제안된 RSA-CRT 기법들을 3 장에서 이를 구현한 환경과 프로그램의 구조를 설명한다. 4 장에서는 구현한 프로그램을 통해 수치를 조절하며 이들이 연산시간에 미치는 영향을 조사하며, 5 장에서 결론을 맺는다.

* 본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT 연구센터 지원사업의 연구결과로 수행되었음(NIPA-2011-C1090-1111-0010).
* 이 논문은 2011 년도 정부(교육과학기술부)의 재원으로 한국과학재단의 지원을 받아 수행된 연구임(No.2011-0000189).

2. 관련연구

2-1. RSA78

RSA78 은 인수분해 문제에 기반한 암호시스템으로 1978 년 R. Rivest, A. Shamir, L. Adleman 에 의해 제안되었다[1]. 이 기법은 키 생성과정에서 공개키 (e,n) 과 개인키 (d,n)을 생성하며, 메시지에 공개키 또는 개인키로 멱을 취하여 암호화, 복호화, 서명 등을 수행한다. 인수분해 문제에 기반한 만큼 키를 알지 못하면 복호화에 많은 시간이 필요하며, 키의 길이를 통해 이를 조절할 수 있다. 하지만 키의 크기가 커지면 지수의 크기가 증가하여 연산시간이 기하급수적으로 증가하게 되는 문제가 있다.

2-2. RSA82

RSA 의 연산속도를 높이기 위한 연구 중 하나로 CRT 를 사용하는 기법을 제안하였다[2]. 제안된 기법은 기존의 RSA 가 법 n에서 하던 연산을 법 p와 q에서 각각 수행한 후 이를 CRT 를 통해 재결합하는 것으로 법의 크기를 줄여 연산속도를 높였다. 저자는 기존 RSA 에 비하여 4 배정도 빠른 연산이 가능하다고 주장하였다. RSA78 은 키 생성 과정에서 n을 생성한 후 비밀 소수 p와 q를 유지할 필요가 없었으나, 각 소수를 법으로 사용하기 위하여 개인키와 함께 이들을 유지해야 하는 부담이 있다. 이 기법은 단 한번의 오류주입공격에도 쉽게 비밀 소수를 노출한다는 문제점이 있다[3][4].

2-3. RSA99

1999 년 Shamir 는 기존의 RSA82 의 문제점을 보완하기 위하여 공통법 r과 비교연산 통한 기법을 제안하였다[5][6]. 제안된 기법은 각각의 법 pr과 qr에서 연산한 결과를 법 r에서 비교하여 이들이 같은 경우 CRT 재결합을 하여 결과를 낸다. 이 기법은 법의 크기가 r배 증가하여 연산속도에 영향을 미치게 된다. 따라서 적절한 r 의 크기를 선택할 필요하다. 제안된 논문에는 1024bit 의 경우 임의의 32bit 수를 사용할 경우 3%정도 연산시간이 증가한다고 주장하고 있다. 하지만 비교연산을 사용하는 기법은 하드웨어적 특성에 의해 이를 쉽게 우회할 수 있다는 문제가 있다.

2-4. RSA03

2003 년 Yen 등은 비교연산을 사용하는 기법이 오류주입공격에 취약함을 보이고 오류확산에 기반한 기법을 제안하였다[7]. 제안된 기법은 법 p에서 연산된 결과를 법 p의 연산에 사용하고 이를 다시 재결합 연산에 사용하는 형태로 진행된다. 따라서 법 p에 오류가 주입되면 법 q의 연산에도 영향을 주게 되고, 법 q에 오류를 주입하면 결과 값 전체에 영향을 주어 소수 추출을 막을 수 있다. 하지만 각기 다른 법 p와 q에서 연산된 값을 제거하기 위해 사용한 인자들에 오류를 주입하는 경우 기존과 동일하게 소수를 추출할 수 있다는 문제가 있다[8]. 또한 기존의 연산들은 법 p와 q의 연산을 독립적으로 수행할 수 있기 때문에 병렬적으로 연산을 수행할 수 있었으나, 제안된 기법은 이를 순차적으로 수행하므로 연산시간 면에서 기존의 기법들에 비해 느리다는 단점도 있다.

3. 구현

3-1. 구현환경

본 논문은 자바를 통해 RSA-CRT 를 구현하기 위해 암호 라이브러리 제공자로 잘 알려진 BouncyCastle(이하 "BC")의 라이브러리를 분석하여 RSA 모듈을 추출하였고, 이 모듈을 토대로 각각의 알고리즘을 모듈화 하였다. 라이브러리에서 제공되는 RSA 모듈은 RSA82 의 형태를 띠고 있다. 사용되는 모든 수는 java.Math.BigInteger 라이브러리를 통해 표현한다.

구현에 사용된 시스템 제원은 <표 1>과 같다.

<표 1> 구현에 사용된 시스템 제원

구분		제원
H/W	CPU	Intel ® Core™ i5 750 @ 2.67GHz
	RAM	PC3-10700 2Gb × 2
	HDD	WDC WD6400AAKS-00E4A0 ATA
S/W	OS	Windows 7 Enterprise Edition
	JDK	Java™ 6 Standard Edition (1.6.0_24-b07)
	SDK	Eclipse Helios 3.6.2
	Library	BouncyCastle bcprov-jdk16-146

3-2. 키 생성 단계

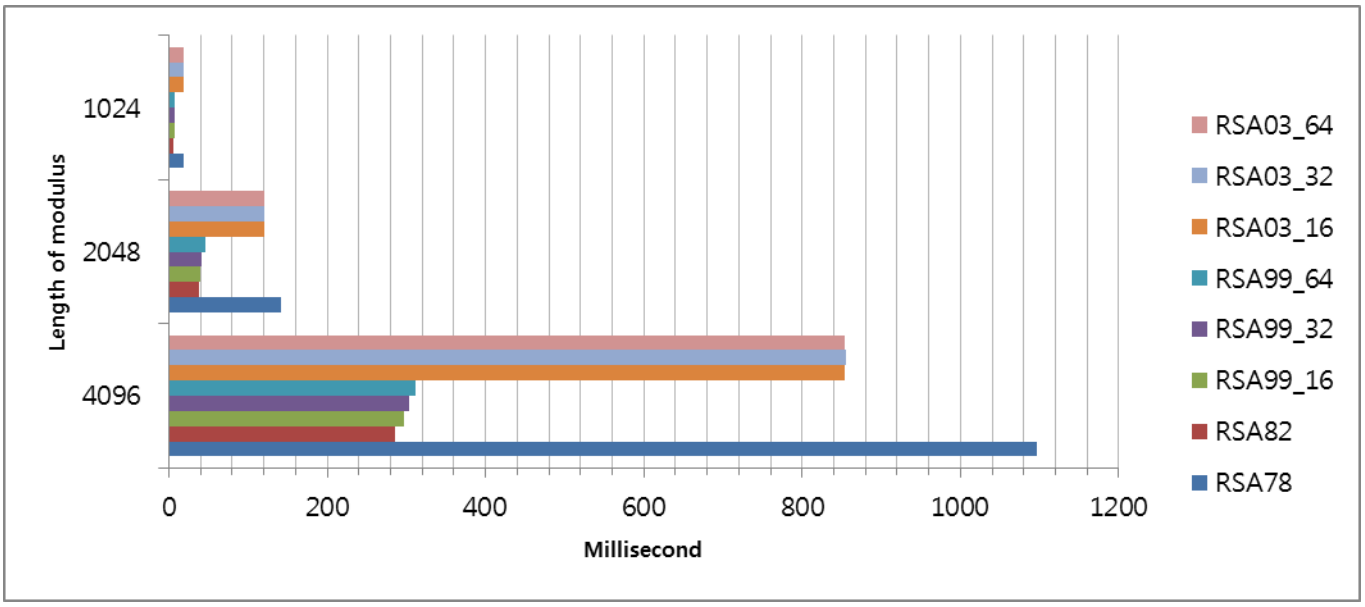
RSA 연산 부분을 최소화 하기 위하여 키 생성 과정에서 미리 계산할 수 있는 값은 모두 연산해 둔다. 각각의 모듈에서 공개키는 (e,n) 만을 저장한다. RSA78 은 개인키로 (d,n)을 저장하는데 비해, 이후의 기법들은 CRT 재결합을 위한 인자들을 추가로 저장해야 한다. 그리고 RSA99 이후의 기법은 RSA82 에 랜덤 값 r을 추가로 저장해야 한다. 각각 개인키 저장 값은 다음과 같다.

RSA78: (d,n)
RSA82: $(d,e,n,p,q,d_p,d_q,q^{-1})$
RSA99: $(d,e,n,p,q,d_p,d_q,q^{-1},r)$
RSA03: $(d_r,e_r,n,p,q,d_p,d_q,q^{-1},r)$

RSA82 이후에 개인키에 e 가 들어가는 것은 BC 의 RSAPrivateCrtKeyParameters 타입에 따랐기 때문이다. RSA03 은 연산 도중에 암호화 복호화가 일어나며, 이때 d와 e 대신 d_r과 e_r을 사용하기 때문에 대치하여 저장했다.

3-3. 복호화 단계

RSA78 에서는 개인키로 지수연산만 하는데 비하여 RSA82 이후의 기법들은 CRT 재결합 연산이 필요하다. BC 의 RSA 라이브러리는 RSA82 의 형태를 띠고 있으며, Garner 의 CRT 재결합 기법을 사용하고 있다. 본 논문에서는 CRT 재결합 부분을 상속하여 모듈간의 차이를 줄였다. 복호화 함수는 인자를 받아오고 입력 값에 대해 연산 후, 재결합 연산을 수행한다.



(그림 2) 키의 크기에 따른 연산시간 비교

4. 분석

4-1. 연산시간 비교

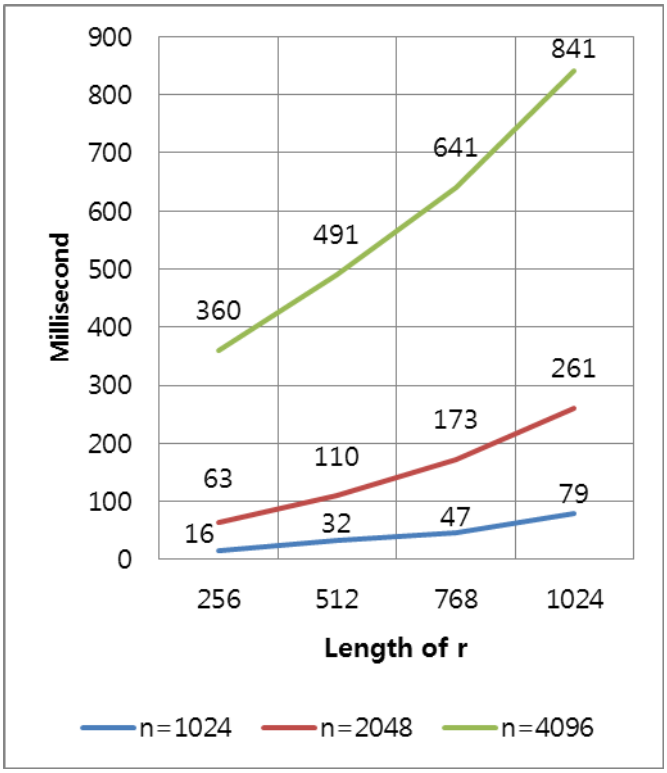
본 논문에서는 각 기법들의 정량적 비교를 위하여 모든 인자의 크기를 동일하게 고정하고, n의 크기를 변화시키며 연산 속도를 관찰하였다. 또한 RSA99, RSA03 에 대해서는 r의 크기를 16bit에서 64bit까지 변화시키며 추가적인 연산시간을 비교하였다.

먼저 법 n의 크기가 1024bit 인 경우 RSA78 과 RSA03 의 연산시간은 동일한 모두 19ms로 CRT 재결합에 따른 성능 향상을 보지 못하였다. 반면 RSA82 는 6ms, RSA99 는 7ms로 3 배 가량 빠른 속도로 복호화가 가능했다.

연산시간의 차이는 법 n의 크기가 2048bit 일 때부터 드러나기 시작한다. 이 경우 RSA78 은 142ms가 소요 되었다. RSA82 는 그의 26.7%인 38ms로 RSA78 에 비해 연산 시간이 대략 1/4 만큼 감소하였다. RSA99 는 r의 크기에 따라 연산시간이 차이가 나는데, RSA82 와 비교해 보았을 때 r이 16bit인 경우 2ms, 32bit인 경우 3ms, 64bit인 경우 8ms 더 걸렸다. 오류확산기법을 사용하는 RSA03 은 RSA78 에 비해 15.5%감소한 120ms가 걸렸다. RSA99 와는 달리 r의 크기 변화에 대해서는 1ms 내외의 차이를 보였다.

법 n의 크기가 4096bit 일 때는 RSA78 의 경우 1097ms가 걸렸고 RSA82 는 2048bit일 때와 마찬가지로 약 1/4만큼 감소한 286ms가 걸렸다. RSA99 의 경우 r의 크기가 16bit 일 때 RSA82 에 비하여 11ms, 32bit일 때 17ms, 64bit 일 때 25ms가 추가적으로 소요되었다. 마지막으로 RSA03 은 r의 크기에는 크게 영향을 받지 않았으며 RSA82 에 비하여 568ms ~ 569ms 가 추가적으로 소요되었다.

RSA82 는 오류주입공격에 대해 어떠한 안전장치도 갖고 있지 않은 만큼 다른 기법들에 비하여 빠른 연산 속도를 보인다. 여기에 비교연산을 통해 오류주입 공격에 대응하고자 했던 RSA99 는 공통법의 크기에 따라 다소 차이를 보이지만, 추가적인 연산이 많지 않은 만큼 RSA82 와 유사한 성능을 보인다. 하지만 RSA99 의 성능은 r의 크기 변화에 민감히 반응하였다. (그림 2)는 r 의 크기 변화에 따른 RSA99 의 연산시간을 비교한 것이다. n 의 크기가 커질수록 r 의 크기 변화에 따른 연산시간이 민감하게 증가하고 있다. 반면 RSA03 은 r 의 크기에는 크게 반응하지 않았다. 하지만 RSA03 은 r의 크기가 $n/4$인 RSA99 와 비슷한



(그림 1) r의 크기에 따른 RSA99 의 연산시간 비교

성능을 보인다.

4-2. 오류주입공격을 통한 소수 추출

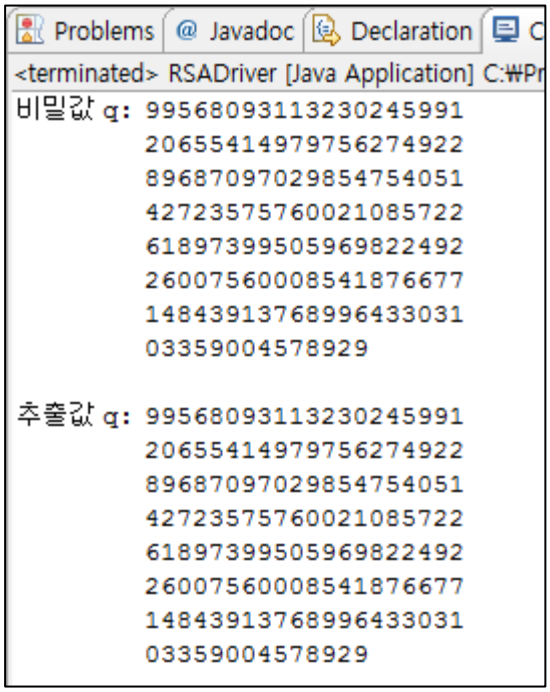
Boneh 등이 제안한 오류주입공격은 CRT 재결합에 사용되는 인자에 오류를 주입하여 오류가 주입된 결과 값을 얻고, 정상적으로 처리된 결과 값과 GCD 연산을 하여 소수를 추출하는 기법이다. 앞서 소개한 RSA-CRT 기법들은 위와 같은 성능 차이를 보이는데 비하여 RSA78 을 제외한 모든 기법이 오류주입공격에 취약하다. 본 논문에서는 취약하다고 알려진 인자에 정수 1 을 오류라 가정하고 더하여 GCD 를 통해 소수가 추출됨을 보이고자 한다.

오류 주입의 결과는 모든 기법이 동일하므로, 이들 중 RSA03 에 오류를 주입하여 소수를 추출하였다. 오류가 주입된 변수는 법 q에서 연산 된 값을 법 n 의 값으로 바꾸기 위해 사용되는 k_q이다. RSA03 에서의 소수 추출은 다음과 같은 순서로 진행된다.

STEP 1. k_q에 오류를 주입하여 서명 값을 구한다.
STEP 2. 오류 서명을 공개키 e로 암호화 한다.
STEP 3. 2 에서 구한 결과 값에서 메시지를 뺀다.
STEP 4. 3 에서 구한 값과 n의 GCD 를 구한다.

이를 실험하기 위하여 k_q에 해당하는 변수인 kq 에 1 을 더하였고, 추출된 값과 비교하기 위하여 소수를 출력하도록 하였다. 메인 함수에서는 decrypt()를 통해 구해진 결과를 encrypt()에 넣는다. encrypt() 함수는 입력 값에 공개키 e 승을 한다. 이를 통해 얻은 결과 값에서 메시지를 빼고 BigInteger 에서 제공하는 GCD 함수를 통해 결과 값을 얻는다.

수식분석의 결과 k_q에 오류를 주입하였을 경우 q 값을 얻을 수 있다. 이를 앞서 구현한 프로그램에서 테스트하여 (그림 3)과 같은 결과를 얻었다.



(그림 3) 소수 추출 결과

5. 결론

최근 국내 논문들에서 보이는 오류주입공격 대응 기법들은 RSA03 의 변형이다[9][10]. 하지만 분석 4-1 에서 보인 바와 같이 RSA03 에서 오류확산을 위해 추가되는 연산의 부담은 다른 기법들에 비해 3 배 가량 크다. 따라서 향후에서는 이와 같은 추가적인 부담을 줄이면서 오류주입에 강건할 수 있는 RSA-CRT 기법 연구가 필요하다.

참고문헌

[1] R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Communications of the ACM, 21(2), pp. 129-126, Feb. 1978.
[2] C. Couvreur, and J. Quisquater, "Fast Decipherment Algorithm for RSA Public-Key Cryptosystem," Electronics Letters, 18(21), pp. 905-907, Oct. 1982.
[3] A. Lenstra, "Memo on RSA Signature Generation in the Presence of Faults," Manuscript, Sep. 1996.
[4] D. Boneh, R. DeMillo, and R. Lipton, "On the Importance of Checking Cryptographic Protocols for Fault," EUROCRYPT '97, LNCS 1233, pp. 37-51, May 1997.
[5] A. Shamir, "How to Check Modular Exponentiation," EUROCRYPT '97 Rump Session, May 1997.
[6] A. Shamir, "Method and Apparatus for Protecting Public Key Schemes from Timing and Fault Attacks," United States Patent 5991415, Nov. 1999.
[7] S. Yen, S. Kim, S. Lim, and S. Moon, "RSA Speedup with Chinese Remainder Theorem Immune Against Hardware Fault Cryptoanalysis," IEEE Transactions on Computers, 52(4), pp. 461-472, Apr. 2003.
[8] S. Yen, D. Kim, and S. Moon, "Cryptoanalysis of Two Protocols for RSA with CRT Based on Fault Infection," Fault Diagnosis and Tolerance in Cryptography 2006, LNCS 4236, pp. 53-61 Oct. 2006.
[9] 김성현, 김태현, 한동국, 박영호, 홍석희, “비교연산을 사용하지 않는 오류주입 공격에 안전한 CRT 기반의 RSA,” 정보보호학회논문지, 제 18 권, 제 4 호, pp. 17-25, 2008 년 8 월.
[10] 백이루, 하재철, “오류 확산 기법에 기반한 RSA-CRT 대응책에 대한 선택 메시지 공격,” 정보보호학회논문지, 제 20 권, 제 3 호, pp. 135-104, 2010 년 6 월.