Attacks on RSA

Gaspare FERRARO
CyberSecNatLab
Matteo ROSSI
Politecnico di Torino

CYBER CHALLENGE.IT

CYBERSECURITY NATIONAL LABORATORY

https://cybersecnatlab.it

License & Disclaimer

License Information

This presentation is licensed under the Creative Commons BY-NC License

To view a copy of the license, visit:

http://creativecommons.org/licenses/by-nc/3.0/legalcode

Disclaimer

- We disclaim any warranties or representations as to the accuracy or completeness of this material.
- Materials are provided “as is” without warranty of any kind, either express or implied, including without limitation, warranties of merchantability, fitness for a particular purpose, and non-infringement.
- Under no circumstances shall we be liable for any loss, damage, liability or expense incurred or suffered which is claimed to have resulted from use of this material.

Goal

- Present different methods to exploit RSA algorithm based on:
  - Modulo
  - Oracles
  - Public/private exponents
  - External information (time and error)

Prerequisites

- Lectures:
  - *CR_0.1 - Number Theory and modular arithmetic*
  - *CR_2.3 – RSA*

Outline

- Attacks on the modulus
- Oracles
- Attacks on small public exponents
- Attacks on small private exponents (sketch)
- Implementation attacks (sketch)

Outline

- Attacks on the modulus
- Oracles
- Attacks on small public exponent
- Attacks on small private exponent (sketch)
- Implementation attacks (sketch)

Attacks on the modulus

- Recall that the security of RSA is based on the hardness of factoring integers
- The most natural way to attack it, is to directly factorize the modulus
- In this section, we describe the problem of factoring and some particularly weak instances regarding RSA

Factoring

- Direct factorization of composite numbers is still considered hard
- In general there is no chance to factor composite numbers with 1024 or more bits
- There are particular cases for which it is feasible, for example:
  - If the primes are reused
  - If all but (at most) one prime are small
  - If two primes are very close

Direct Factorization

- One of the best known algorithms is the *General Number Field Sieve* (GNFS), but it is still NP
- An implementation can be found in the CADO-NFS software (http://cado-nfs.gforge.inria.fr/)
- Installation and use of CADO-NFS can be a little bit tedious and in general not required in CTFs

Direct Factorization

- Easier-to-use alternatives to CADO-NFS are:
  - Factordb.com - database of numbers and their factorization
    - just a database, not a factoring software
  - The factor function in SageMath
    - Available also online (with limited execution time)
  - The YAFU software
    - Available for Linux and Windows (https://github.com/DarkenCode/yafu)

A particularly easy case

- Let's write $p = a + b$ and $q = a - b$ for integers a, b
- The RSA modulus becomes $N = a^2 - b^2$, that is $a^2 - N = b^2$
- If b is small enough, we can actually bruteforce on a or b to get the factorization
- This is called *Fermat's factorization method*, and it is implemented in YAFU as the "fermat" function

Common Prime Attack

- Lenstra et al. In 2012 published the paper *"Ron was wrong, Whit is right"*, stating*"two out of every one thousand RSA moduli that we collected offer no security"*
- They showed that in real life it is common that two RSA keys share one of the prime factors

Common Prime Attack

- Why is this a problem?
  - Given $(N_1 = p \times q, N_2 = p \times r)$ we can factorize the two modules as $p = GCD(N_1, N_2)$
  - The Euclidean Algorithm runs in polynomial time!
  - Once we have p, the two RSA keys are completely broken

Common Modulus Attack

- What happens when the entire modulus is reused?
- In practice, since the exponent is fixed to 65537, this is not an issue
- But what if the two exponents are different?

Bézout's Identity

- In order to understand the solution, we need the following:
  - Bézout's Identity: *Let a and b be integers with greatest common divisor d. Then there exist integers x and y such that* $ax + by = d$

Common Modulus Attack

- Suppose to have two keys $(N, e_1), (N, e_2)$, for simplicity with $GCD(e_1, e_2) = 1$, and two ciphertexts coming from the same plaintext c_1, c_2
  - By Bézout's identity, we can compute u and v such that:
    - $u \times e_1 + v \times e_2 = 1$
  - Computing $c_1^u + c_2^v \ mod\ N$ reveals the plaintext!
  - Notice that this can be an issue even with $GCD(e_1, e_2) > 1$ but small! (More on this in the *Attacks on small public exponent* section)

Outline

- Attacks on the modulus
- Oracles
- Attacks on small public exponent
- Attacks on small private exponent (sketch)
- Implementation attacks (sketch)

Oracles

- In this section we analyze the security of RSA in the presence of oracles.
- This happens in particular with:
  - Encryption oracles
  - Decryption oracles
  - Padding oracles

Modulus Recovery

- Scenario 1: we have an encryption oracle that encrypt with a fixed secret key.
- Can we recover the modulus?
  - Encrypt two chosen messages x and y
  - Use the fact that the standard RSA exponent is 65537 to compute x^e and y^2 (without modular reduction)
  - Notice that $x^e - Enc(x)$ is a multiple of N
  - Compute $GCD(x^e - Enc(x), y^e - Enc(y))$ to find small multiple of N
  - Repeat with different values if necessary

Homomorphic properties of RSA

- Scenario 2: we have a decryption oracle that does not allow to decrypt messages containing sensible data.
- Can we decrypt an arbitrary ciphertext c?
  - Encrypt a chosen plaintext x (locally!)
  - Ask the server to decrypt $x^e \times c$
  - Notice that $D(x^e \times c) = D(x^e) \times D(c) = x \times D(x)$ and recover the message
- This is called *Homomorphic Property* of RSA

LSB Oracles

- Scenario 3: we have a decryption oracle that gives a partial decryption: only the least significant bit.
- Can we recover the plaintext?
  - The idea is to "binary search" the answer using the fact that N is odd
  - Decrypt $2^e \times c$: if the result is 1, then $2m > N$ and so $N/2 < m < N$, otherwise $0 < m < N/2$
  - Repeat with $4^e \times c, 8^e \times c$ and so on
  - We can recover the message in $O(\log N)$ steps
- This technique can be easily extended to leakages of the k least significant bits

Padding oracles (sketch)

- As CBC mode, also RSA padding schemes have been exploited in the past.
- The most famous attacks were:
  - Coppermith's attack against short padding schemes (1997)
  - Bleichenbacher's attack against PKCS#1 v1.5 (1998)
  - Manger's attack against PKCS#1 v2.0 (2001)

Outline

- Attacks on the modulus
- Oracles
- Attacks on small public exponent
- Attacks on small private exponent (sketch)
- Implementation attacks (sketch)

Attacks on small public exponent

- In this section we show why the choice of $e = 3$, while being the best computationally speaking, has been replaced by 65537

Direct cube root

- Scenario 1: we have an unpadded implementation of RSA with $e = 3$ and an encrypted message that comes from a "short message"
- Can we decrypt it?
  - If the number of bits in the message is less than $\log_2 N /3$, the modulus has no effect!
  - We can simply compute an integer cube root to find the message

Hastad's broadcast attack

- Scenario 2: 3 parties have 3 distinct RSA keys $(N_1, 3), (N_2, 3), (N_3, 3)$
- The same message is sent to them without padding
- Knowing the 3 ciphertexts can we decrypt them?
  - This follows from a direct application of the Chinese Remainder Theorem: we can get $m^3 mod\ N_1 \times N_2 \times N_3$
  - Since $m < \min(N_1, N_2, N_3)$ this boils down again to an integer cube root
  - This works also for exponents different from 3!

Outline

- Attacks on the modulus
- Oracles
- Attacks on small public exponent
- Attacks on small private exponent (sketch)
- Implementation attacks (sketch)

Attacks on small private exponent

- To reduce decryption time, we may wish to have small private exponents
  - Wiener showed that if $3 \times d < N^{\frac{1}{4}}$ then d can be efficiently recovered
  - Boneh and Durfee improved this bound to $d < N^{0.292}$
  - The best bound is believed to be $d < N^{0.5}$, but no one proved it yet (it is an open research problem!)
  - The details on these attacks are tedious and outside the scope of this presentation

Outline

- Attacks on the modulus
- Oracles
- Attacks on small public exponent
- Attacks on small private exponent (sketch)
- Implementation attacks (sketch)

Implementation Attacks (Sketch)

- In this last section we turn our attention to an entirely different class of attacks: impementation attacks
- Rather than attacking the underlying structure of RSA, we attack its implementation

Timing Attacks (Sketch)

- P. Kocher in 1996 showed that it is possible to recover RSA private exponents by measuring the timing of the decryption
- The main problem is that the "repeated squaring" algorithm to compute exponentiations is slower when the exponent has more ones in its binary representation
- Kocher's exploitation technique is based on correlation between a locally generated sample set and observed values

Fault Injections (Sketch)

- Recall that exponentiation is usually done in two parts via Chinese Remainder Theorem
- Boneh et al. In 1997 showed that if there's a glitch on the decrypting computer that miscalculates *exactly one* of the two parts, then the modulus can be factored efficiently

Gaspare FERRARO

CyberSecNatLab

Matteo ROSSI

Politecnico di Torino

Attacks on RSA

CYBER CHALLENGE.IT

CYBERSECURITY NATIONAL LABORATORY

https://cybersecnatlab.it