

Security of the RSA

Martin Stanek

Department of Computer Science
Comenius University
`stanek@dcs.fmph.uniba.sk`

Cryptology 1 (2020/21)

Content

Factorization, RSA problem

- Problems with primes

- Small plaintext space, small public/private exponent

- Small public/private key

- Homomorphism of RSA

- Partial decryption oracles

 - Half and parity predicates

 - Bleichenbacher's attack on PKCS#1 v 1.5

 - Manger's attack

- Other implementation attacks

RSA scheme

- ▶ $n = p \cdot q$ (product of two distinct primes)
- ▶ $e \cdot d \equiv 1 \pmod{\varphi(n)}$, where $\varphi(n) = (p-1)(q-1)$
- ▶ public key: (e, n)
- ▶ private key: d
- ▶ public/private transforms $E, D : \mathbb{Z}_n \rightarrow \mathbb{Z}_n$
 - ▶ $E(m) = m^e \bmod n$
 - ▶ $D(c) = c^d \bmod n$

Hybrid encryption

- ▶ encryption of message m for recipient A (his public key is pk_A):

$$\langle E_k(m), E_{pk_A}^{RSA}(k) \rangle$$

- ▶ notation:
 - ▶ E – symmetric cipher (e.g. AES)
 - ▶ k – random symmetric key for E
 - ▶ $E_{pk_A}^{RSA}$ – RSA encryption with A 's public key
- ▶ A can decrypt easily
- ▶ advantages: key management (asymmetric scheme), speed
- ▶ disadvantages: the security depends on both constructions

Real world – key transport

- ▶ usually wrapping symmetric keys, providing confidentiality and integrity
- ▶ key transport
 - ▶ RFC 5990: Use of the RSA-KEM Key Transport Algorithm in the Cryptographic Message Syntax (CMS)
 - ▶ NIST SP 800-56B rev. 2: Recommendation for Pair-Wise Key-Establishment Schemes Using Integer Factorization Cryptography; various schemes, e.g. KTS-OAEP: Key-Transport Using RSA-OAEP

Factorization and RSA

- ▶ factorization \Rightarrow compute the private key \Rightarrow decryption (trivial)
- ▶ decryption (knowing only the public key) $=? \Rightarrow$ factorization (open)
- ▶ knowledge of $\varphi(n)$ is equivalent to factorization
 - \Leftarrow trivial
 - \Rightarrow solving 2 equations with 2 variables:

$$n = p \cdot q$$

$$\varphi(n) = (p - 1)(q - 1)$$

- ▶ knowledge of d is equivalent to factorization
 - \Leftarrow trivial
 - \Rightarrow more complicated procedure needed
- ▶ corollary: do not share n among group of users

RSA problem

- ▶ RSA problem:
given (e, n) and $c \in \mathbb{Z}_n$; compute m such that $m^e \equiv c \pmod{n}$
- ▶ RSA problem is not more difficult than factorization
 - ▶ (open problem) Is the RSA problem as difficult as factorization or easier?

Problems with primes

- ▶ specific algorithms for factorization, when p, q satisfy some properties, for example:
 - ▶ small $|p - q|$,
 - ▶ $p - 1$ (or $q - 1$) without a large prime factor, etc.
- ▶ suspicious methods of generating primes, e.g.
 - ▶ weak or poorly initialized PRNG
 - ▶ primes with some internal structure (“optimization”)
- ▶ Lenstra et al. (2012)
 - ▶ 11.4 million RSA moduli (X.509 certificates, PGP keys)
 - ▶ 26965 (incl. 10 RSA-2048) vulnerable (shared a single common prime factor)

Problems with primes (2)

- ▶ Bernstein et al. (2013)
 - ▶ Taiwan's national "Citizen Digital Certificate" database
 - ▶ generated by government-issued smart cards (certified)
 - ▶ 3.2 million unique RSA moduli
 - ▶ 103 moduli factored by computing the gcd (sharing a non-trivial prime divisor)
 - ▶ observing non-randomness in the primes ... 184 distinct 1024-bit RSA keys factored
- ▶ Nemec et al. (2017)
 - ▶ problem with "FastPrime" method for primes generation implemented in library for particular hardware chips
 - ▶ factor public modulus
 - ▶ ID cards – e.g. Estonia (750.000), Slovakia (300.000)

General factorization algorithms

- ▶ General number field sieve (GNFS)
- ▶ heuristic complexity: $\exp\left((\sqrt[3]{64/9} + o(1))(\ln n)^{1/3}(\ln \ln n)^{2/3}\right)$
- ▶ equivalent key lengths:

symmetric	RSA
80	1024
112	2048
128	3072
192	7680
256	15360

- NIST Recommendations (SP 800-57 part 1 rev. 5) (2020)
- various estimates are compared at www.keylength.com

Small message (plaintext) space

- ▶ RSA scheme is deterministic (the textbook version)
- ▶ small plaintext space:
 - ▶ e.g. {"yes", "no", "maybe"}
 - ▶ attacker can compute $E(m)$ for any m and compare the result with the ciphertext
- ▶ potential plaintexts can be tested regardless of plaintext space
- ▶ randomization with padding



- ▶ is it secure (can you prove it)?
- ▶ see OAEP for provable security

Small public exponent – broadcast

- ▶ small exponent – speed
- ▶ let $e = 3$ for three recipients A, B, C with moduli n_A, n_B, n_C
- ▶ broadcasting m :

$$c_A = m^3 \bmod n_A$$

$$c_B = m^3 \bmod n_B$$

$$c_C = m^3 \bmod n_C$$

- ▶ an attacker solves the system of congruences (CRT):

$$x \equiv c_A \pmod{n_A}$$

$$x \equiv c_B \pmod{n_B}$$

$$x \equiv c_C \pmod{n_C}$$

Small public exponent – broadcast (2)

- ▶ solution x (obtained from CRT) and m^3 satisfy the system of congruences, thus

$$x \equiv m^3 \pmod{n_A n_B n_C}$$

- ▶ $x = m^3$, since $m < n_A, n_B, n_C$
- ▶ m can be computed as a cube root of x
- ▶ padding as a prevention

Small public exponent – related messages

- ▶ m_1, m_2 linearly dependent messages; $c_1 = E(m_1)$, $c_2 = E(m_2)$
- ▶ $\exists a, b \in \mathbb{Z}$: $m_2 = am_1 + b$, the attacker knows a, b
- ▶ variable z (m_1 is a root of the following polynomials):

$$\begin{aligned}z^e - c_1 &\equiv 0 \pmod{n} \\(az + b)^e - c_2 &\equiv 0 \pmod{n}\end{aligned}$$

- ▶ $(z - m_1)$ divides both polynomials; $(z^e - c_1)/(z - m_1)$ is irreducible
- ▶ $\gcd(z^e - c_1, (az + b)^e - c_2)$ reveals m_1 and m_2
 - ▶ Example: $n = 91$, $e = 5$. Let $c_1 = 45$, $c_2 = 28$, and $m_2 = 30 \cdot m_1 + 11$.

$$\begin{aligned}\gcd(z^5 - 45, (30z + 11)^5 - 28) &= \\&= \gcd(z^5 + 46, 88z^5 + 40z^4 + 90z^3 + 33z^2 + 47z + 44) = z + 37 = z - 54\end{aligned}$$

Thus $m_1 = 54$ and $m_2 = 30 \cdot 54 + 11 = 84$.

- ▶ easy to generalize for any known polynomial relation
- ▶ prevention: suitable padding
 - ▶ not every padding is secure (see Coppersmith's attack)

Small private exponent

- ▶ motivation: fast decryption
- ▶ implementation: choose d first, e computed afterward
- ▶ results – d can be computed from a public key:
 - ▶ Wiener (1990): $d < \frac{1}{3}n^{0.25}$ (continued fraction)
 - ▶ Boneh, Durfee (1999): $d < n^{0.292}$ (Coppersmith, LLL)
 - ▶ some other improvements exist
- ▶ do not “optimize” d (!)

Some applications of Coppersmith's theorem

- ▶ Coppersmith's theorem – finding all small solutions of modular polynomial equation
- ▶ computing plaintext when using short/improper padding (and small e)
- ▶ computing primes given some fraction of their bits
- ▶ reconstructing d given some fraction of its bits

Using homomorphism of RSA

- ▶ $E(m_1 \cdot m_2) = E(m_1) \cdot E(m_2)$, computations are mod n
- ▶ let's assume, that l -bit symmetric key k is encrypted, i.e. $k < 2^l$
- ▶ the attacker pre-computes $E(1), E(2), E(3), \dots, E(2^{l/2})$, and stores the values $\langle E(i), i \rangle$ in a hash table
- ▶ if $k = k_1 \cdot k_2$, for $k_i \leq 2^{l/2}$:
 - ▶ the attacker tries $k_1 = 1, 2, 3, \dots, 2^{l/2}$, and searches $c/E(k_1) = E(k/k_1)$ in the table
 - ▶ a match yields k_1, k_2 , i.e. k
- ▶ time complexity $O(2^{l/2})$
- ▶ increasing the number of pre-computed values \Rightarrow higher probability of success
- ▶ (!) for small e , e.g. $e = 3$, the attacker can compute $\sqrt[3]{c}$ directly (if $k^3 < n$)

Half predicate

- ▶ Knowing a ciphertext – can anything be computed about the plaintext?
- ▶ (textbook) RSA is not semantically secure (e.g. testing any plaintext)
- ▶ oracle $\text{half}(c) = 0$ if $0 \leq m < n/2$, or 1 otherwise
- ▶ we decrypt any c using predicate $\text{half}()$

$$\text{half}(c) = 0 \quad \Leftrightarrow \quad m \in \{0, \dots, \lfloor n/2 \rfloor\}$$

$$\text{half}(c \cdot E(2)) = 0 \quad \Leftrightarrow \quad m \in \{0, \dots, \lfloor n/4 \rfloor\} \cup \{\lceil 2n/4 \rceil, \dots, \lfloor 3n/4 \rfloor\}$$

$$\text{half}(c \cdot E(2^2)) = 0 \quad \Leftrightarrow \quad m \in \{0, \dots, \lfloor n/8 \rfloor\} \cup \dots$$

- ▶ we can compute m by binary search ($c \cdot E(2^l) = E(m \cdot 2^l)$)
- ▶ remark: d is not used nor computed in this attack

Parity predicate

- ▶ similarly to `half()`, we can use the predicate `parity()`
 - ▶ $\text{parity}(c) = m \& 0x1$
- ▶ relation between predicates: $\text{half}(c) = \text{parity}(c \cdot E(2))$
 - ▶ if $0 \leq m < n/2$:
then $0 \leq 2m < n$ and the plaintext corresponding to $c \cdot E(2)$ is even
 - ▶ if $n/2 < m < n$:
then $n \leq 2m < 2n \Rightarrow 2m \bmod n = 2m - n$,
i.e. the plaintext corresponding to $c \cdot E(2)$ is odd

Bleichenbacher's attack on PKCS#1 v1.5 (1)

- ▶ chosen ciphertext attack (1998)
- ▶ PKCS#1 v1.5 oracle (error message, timing, etc.) \Rightarrow decryption of arbitrary ciphertext
- ▶ PKCS#1 v1.5 padding:

00	02	≥ 8 random non-zero bytes	00	message
----	----	--------------------------------	----	---------

- ▶ k – byte length of n ; $2^{8(k-1)} \leq n < 2^{8k}$
- ▶ PKCS conforming block:
 1. starts with bytes 00 02
 2. bytes 3 ... 10 are non-zero
 3. there is some 00 byte later (bytes 11 ... k)
- ▶ let's denote $B = 2^{8(k-2)}$, i.e. PKCS conforming block: $2B \leq m < 3B$
- ▶ ciphertext is called PKCS conforming if its decryption is PKCS conf.

Bleichenbacher's attack on PKCS#1 v1.5 (2)

- ▶ given $c \in \mathbb{Z}_n$ the attacker wants to compute $m = c^d \bmod n$
- ▶ Step 1 (blinding)
 - ▶ not used if c is PKCS conforming
 - ▶ generate random $s_0 \in \mathbb{Z}_n$ until $c \cdot s_0^e \bmod n$ is PKCS conforming
 - ▶ homomorphic property of RSA: $m_0 = m \cdot s_0 \bmod n$
 - ▶ probability of random ciphertext being PKCS conf. is $> 0.58 \cdot 2^{-16}$ (for 2048 bit n)
 - ▶ we know that $2B \leq m_0 \leq 3B - 1$
 - ▶ set of intervals for m_0 : $M_0 = \{\langle 2B, 3B - 1 \rangle\}$
- ▶ Step 2 (searching): 2a, 2b, 2c
- ▶ Step 2a (for the first iteration $i = 1$)
 - ▶ find the smallest possible $s_1 \geq n/3B$: $c_0 \cdot s_1^e \bmod n$ is PKCS conf.
 - ▶ for $s_1 < n/3B$ the result cannot be PKCS conf.

Bleichenbacher's attack on PKCS#1 v1.5 (3)

- ▶ Step 2b (more than 1 interval in M_{i-1})
 - ▶ rare situation (if ever, then for $i = 2$)
 - ▶ find the smallest possible $s_i > s_{i-1}$: $c_0 \cdot s_i^e \bmod b$ is PKCS conf.
- ▶ Step 3c ($M_{i-1} = \{\langle a, b \rangle\}$ contains a single interval)
 - ▶ find small $r_i \geq 2 \cdot (bs_{i-1} - 2B)/n$ (approx. halving the interval) and s_i such that $c_0 \cdot s_i^e \bmod n$ is PKCS conf.
 - ▶ thus, $2B \leq m_0 s_i - r_i n \leq 3B - 1$ and we get:

$$\frac{2B + r_i n}{m_0} \leq s_i \leq \frac{3B - 1 + r_i n}{m_0}$$
$$\frac{2B + r_i n}{b} \leq s_i \leq \frac{3B - 1 + r_i n}{a}$$

- ▶ the last derivation used the fact that $m_0 \in \langle a, b \rangle$

Bleichenbacher's attack on PKCS#1 v1.5 (4)

- ▶ Step 3 (narrowing after s_i was found)
 - ▶ compute M_i by updating all intervals $\langle a, b \rangle \in M_{i-1}$:

$$M_i = \bigcup_{(a,b,r)} \left\{ \left\langle \max \left(a, \left\lceil \frac{2B + rn}{s_i} \right\rceil \right), \min \left(b, \left\lfloor \frac{3B - 1 + rn}{s_i} \right\rfloor \right) \right\rangle \right\}$$

- ▶ the union must consider all potential r values intersecting with $\langle a, b \rangle$:

$$\begin{aligned} 2B &\leq m_0 s_i - rn \leq 3B - 1 & (\star) \\ \frac{m_0 s_i - 3B + 1}{n} &\leq r \leq \frac{m_0 s_i - 2B}{n} \\ \frac{a s_i - 3B + 1}{n} &\leq r \leq \frac{b s_i - 2B}{n} \end{aligned}$$

- ▶ new intervals for M_i are obtained by rearranging (\star)

Bleichenbacher's attack on PKCS#1 v1.5 (5)

- ▶ Final step:
 - ▶ if $M_i = \langle a, a \rangle$ we have $m_0 = a$ and $m = m_0 \cdot s_0^{-1} \bmod n$
- ▶ Impact:
 - ▶ SSL/TLS RSA key exchange method: client sends *pre-master secret* encrypted with server's public key (PKCS#1 v1.5)
 - ▶ decryption of the pre-master secret yields the session keys
 - ▶ careful implementation needed, see TLS 1.2 (RFC 5246)
 - ▶ when relevant, the attack allows to create a PKCS#1 v1.5 signature of arbitrary message (using server's private key)
- ▶ ROBOT (Return Of Bleichenbacher's Oracle Threat)
 - ▶ recent attack on TLS after 19 years (2018)
 - ▶ advice: disable all TLS_RSA ciphersuits
 - ▶ non-standard message flow (shortened)
 - ▶ different responses: different alert codes, TCP FIN, TCP timeout, TCP reset, two alerts ...

Manger's attack

- ▶ Does OAEP help (it is almost impossible to generate a valid ciphertext)?
- ▶ Manger's attack (2001): compute $m = c^d \bmod n$ for any c
- ▶ assumption: access to the following oracle:
 - ▶ Given c' , is the first byte of $(c')^d \bmod n$ zero?
 - ▶ let k be the byte length of n , and $B = 2^{8(k-1)}$
 - ▶ oracle: $(c')^d \bmod n < B$
- ▶ recognizing bad first byte vs. bad internal integrity of decrypted block
- ▶ gradually reduce an interval of possible m values
- ▶ can be adapted to PKCS#1 v1.5
- ▶ there are also improvements to Bleichenbacher's attack

Combining various attack ideas

- ▶ *The 9 Lives of Bleichenbacher's CAT: New Cache Attacks on TLS Implementations* (2018)
- ▶ cache-based attack techniques for side channel
- ▶ ...leading to Manger's oracle, Bleichenbacher's oracle and several other types of oracles
- ▶ optimizations to speed up the attacks
- ▶ most TLS implementations were vulnerable

Other implementation attacks – examples

▶ Timing attacks

- ▶ straightforward implementation of modular exponentiation
- ▶ computation time of $D(c)$ depends on c , d , and n
- ▶ statistical correlation analysis to recover d from many samples (c_i, time_i)
- ▶ variant used to attack SSL implementation (2003) with approx. million queries for extracting private key and factoring 1024 bit modulus n
- ▶ prevention: blinding

▶ Fault attacks

- ▶ induce faults while executing sensitive operations
- ▶ heat, power spikes, clock glitches, etc.
- ▶ example: fault in a single value/computation in RSA CRT (signature computation) – correct and fault signatures yield the factorization of n