

The RSA Cryptosystem

PROOF OF RSA

Mar-25

The RSA Cryptosystem

11

11

RSA consistency: preliminaries [→]



- **Definition - Euler's Phi Function**
 - The number of integers in \mathbb{Z}_m relatively prime to m is denoted by $\Phi(m)$.
- **Theorem - Let m have the following canonical factorization**

$$m = p_1^{e_1} \cdot p_2^{e_2} \cdot \dots \cdot p_n^{e_n}$$

where the p_i are distinct prime numbers and e_i are positive integers, then

$$\Phi(m) = \prod_{i=1}^n (p_i^{e_i} - p_i^{e_i-1})$$

Mar-25

The RSA Cryptosystem

12

¹² If you consider \mathbb{Z}_p , so you consider $[0, p-1]$. What are the numbers coprime with p ? $p-1$. If you apply $\Phi(m) = p-1$ otherwise, $m=p \cdot q$ in RSA. $\Phi(m) = (p^e - p^0)(q^e - q^0)$



RSA consistency: preliminaries

- Theorem - Fermat's Little Theorem
 - Let a be an integer and p be a prime, then $a^p \equiv a \pmod{p}$.
- Theorem - Euler's Theorem
 - Let a and m be integers with $\gcd(a, m) = 1$, then $a^{\Phi(m)} \equiv 1 \pmod{m}$.
- Euler's theorem is a generalization of the Fermat's little theorem

Mar-25

The RSA Cryptosystem

13

13

RSA consistency: proof



- We need to prove that decryption is the inverse operation of encryption, $D_{\text{privK}}(E_{\text{pubK}}(x)) = x$
- Step 1
 - $d \cdot e = 1 \pmod{\Phi(n)}$
 - By definition of mod operator $d \cdot e = 1 + t \cdot \Phi(n)$ for some integer t
 - Insert this expression in the decryption: $y^d \equiv x^{ed} \equiv x^{1+t \cdot \Phi(n)} \equiv x \cdot x^{t \cdot \Phi(n)} \equiv x \cdot (x^{\Phi(n)})^t \pmod{n}$

Mar-25

The RSA Cryptosystem

14

14



RSA Consistency: proof

- Step 2: prove that $x \equiv x \cdot (x^{\Phi(n)})^t \pmod{n}$

- Recall**

- Euler's Theorem: if $\gcd(x, n) = 1$ then $1 \equiv x^{\Phi(n)} \pmod{n}$

- Minor generalization $1 \equiv 1^t \equiv (x^{\Phi(n)})^t \pmod{n}$

- Case 1:** $\gcd(x, n) = 1$

- Euler's theorem holds $\rightarrow x \cdot (x^{\Phi(n)})^t \equiv x \cdot 1 \equiv x \pmod{n}$

Q.E.D.

$$(a \cdot b) \pmod{m} = [(a \pmod{m})(b \pmod{m})] \pmod{m}$$

\equiv

Mar-25

The RSA Cryptosystem

15

15

RSA Consistency: proof

- Case 2:** $\gcd(x, n) \neq 1$

x cannot be $p \cdot q$ because
 $x < m$

- Since p and q are primes (and $x < n$) \rightarrow
either $x = r \cdot p$ or $x = s \cdot q$ with $r < p$ and $s < q$

- Assume $x = r \cdot p \rightarrow \gcd(x, q) = 1 \rightarrow$ Euler's Theorem holds in
this form $1 \equiv (x^{\Phi(n)})^t \pmod{q}$ \rightarrow by construction

- Proof: $(x^{\Phi(n)})^t \equiv (x^{(p-1)(q-1)})^t \equiv ((x^{\Phi(q)})^t)^{p-1} \equiv 1^{(p-1)} \equiv 1 \pmod{q}$

- $(x^{\Phi(n)})^t = 1 + u \cdot q$, for some integer $u \rightarrow$ (1 + a multiple of q for some coeff.)
 $x \cdot (x^{\Phi(n)})^t = x + x \cdot u \cdot q = x + (r \cdot p) \cdot u \cdot q = x + r \cdot u \cdot (q \cdot p) = x + r \cdot u \cdot n$

$\rightarrow x \cdot (x^{\Phi(n)})^t \equiv x \pmod{n}$ **Q.E.D.**

WOW!



Mar-25

The RSA Cryptosystem

16

RSA encryption and decryption



- Comments
 - RSA proof is based on Euler's theorem
 - The proof becomes simpler by using the Chinese Remainder Theorem

Mar-25

The RSA Cryptosystem

17

17

The RSA Cryptosystem

RSA SECURITY

Mar-25

The RSA Cryptosystem

18

18

Attacks

Algorithm can be attacked in several ways



UNIVERSITÀ DI PISA

- Protocol attacks
- Mathematical attacks (cryptanalysis attacks)
- Side-channel attacks (attack the implementations)

Mar-25

The RSA Cryptosystem

19

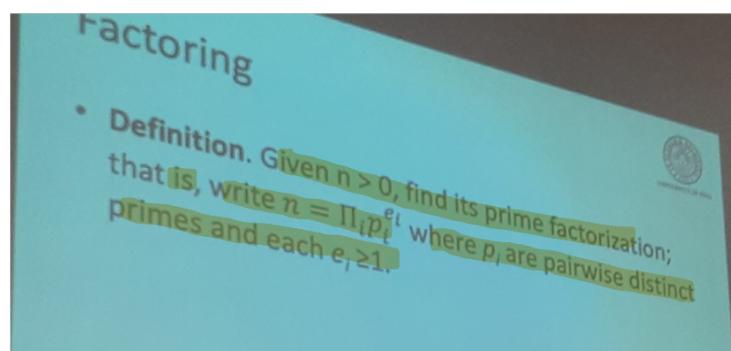
19

Protocol attacks



UNIVERSITÀ DI PISA

- Based on malleability of RSA : bad news: RSA is malleable
- Avoidable by padding



Mar-25

The RSA Cryptosystem

20

20

Mathematical attacks



- The RSA Problem (RSAP)
 - Recovering plaintext x from ciphertext y , given the public key (n, e) . Adversary can grab all ct. And knows PK.
- RSA VS FACTORING
 - If p and q are known, RSAP can be easily solved
 - $\text{RSAP} \leq_p \text{FACTORING}$ i.e., FACTORING is at least as difficult as RSAP or, equivalently, RSAP is not harder than FACTORING
 - It is widely believed that RSAP and Factoring are computationally equivalent, although no proof of this is known.

Mar-25

The RSA Cryptosystem

21

- 21
- For the moment nobody has proven that breaking RSA necessarily requires the ability of factoring n , although this conjecture is considered very plausible

Mathematical Attacks



- Theorem: $\lceil \rightarrow \text{No Proof}$
- THM (FACT 1) Computing the decryption exponent d from the public key (n, e) is computationally equivalent to factoring n
 - Proof
 - If factorization of n is known, then it is possible to compute the private key d efficiently
 - (It can be proven that) if d known, then it is possible to factor n efficiently

\hookrightarrow If I use one d value, I can factorize and decrypt their received messages.

Mar-25

The RSA Cryptosystem

22

22

Mathematical Attacks



- RSAP vs e-th root
 - A possible way to decrypt $y = x^e \pmod{n}$ is to compute the modular e-th root of y , i.e., $x = \sqrt[e]{y} \pmod{n}$
- THM (FACT 2) Computing the e-th root is a computationally easy problem iff n is prime
- THM (FACT 3) If n is composite the problem of computing the e-th root is equivalent to factoring

Mar-25

The RSA Cryptosystem

23

23

Mathematical Attacks



- THM - Knowing ϕ is computationally equivalent to factoring
 - PROOF.
 - Given p and q , s.t. $n = pq$
 - Computing ϕ is immediate.
 - Given ϕ
 - From $\phi = (p-1)(q-1) = n - (p+q) + 1$, determine $x_1 = (p+q)$.
 - From $(p - q)^2 = (p + q)^2 - 4n = x_1^2 - 4n$, determine $x_2 = (p - q)$.
 - Finally, $p = (x_1 + x_2)/2$ and $q = (x_1 - x_2)/2$.

Mar-25

The RSA Cryptosystem

24

24

Mathematical Attacks



- **Exhaustive Private Key Search**
 - This attack must be more difficult than factoring n
 - The bit length of private exponent d must be the same as the bit length of n
 - $\text{sizeof}(p) \approx \text{sizeof}(q)$ *sizeof(d) ≈ sizeof(m)* There are attacks that can exploit this properties
 - $\text{sizeof}(d) >> \text{sizeof}(p)$ AND $\text{sizeof}(d) >> \text{sizeof}(q)$

Libraries take care of these requirements

Mar-25

The RSA Cryptosystem

25

25

Factoring



- Primality testing vs. factoring
 - FACT 5 – To decide whether an integer is composite or prime seems to be, in general, much easier than the factoring problem
 - * *Primality Testing*

I can say in an efficient way if number is prime or not

Mar-25

The RSA Cryptosystem

26

26



Factoring: There exist different families of fact. alg.

- Factoring algorithms

- General purpose algorithms (any number)

- Running time depends on n

- Examples

- Quadratic sieve and general number field sieve

- Special purpose algorithms

- Tailored to perform better when the integer n being factored is of special form

- Running time depends on certain properties of factors of n

- Examples

- Trial division, Pollard's rho alg., Pollard's p – 1 alg., elliptic curve alg., and special number sieve

Mar-25

The RSA Cryptosystem

27

27

Factoring

- Factoring algorithms

- No algorithm can factor all integers in polynomial time

- Neither the existence nor non-existence of such algorithms has been proven, but it is generally suspected that they do not exist

- Peter Shor discovered a quantum algorithm that is polynomial (1994)

- There are sub-exponential algorithms (best known general purpose, *)

- For computers, the best algorithm is General Number Field Sieve (GNFS)

* power of a function of the number of bits of key as complexity

Ex: 2^m is exponential, $\sqrt{2^m}$ or $2^{f(m)}$ are subexponential

Mar-25

The RSA Cryptosystem

28

28

Unless a new algorithm (polynomial) or new cryptanalytic attacks is found, RSA is secure. But we are migrating to lattice, that can not be broken with quantum comp.

Factoring



UNIVERSITÀ DI PISA

- Length of the modulus
 - RSA sparked much interest in the old problem of integer factorization
 - Factoring methods improved considerably during '80s and '90s
 - Advisable modulus length
 - Until recently, 1024-bit was a default
 - Nowadays factorization within 10-15 years or even earlier
 - Modulus in the range 2048-4096 bit for long term security

Mar-25

The RSA Cryptosystem

29

29

The RSA Cryptosystem

RSA IN PRACTICE

Mar-25

The RSA Cryptosystem

30

30



RSA in practice

- Schoolbook/plain RSA is insecure (the one in the original paper)
 - RSA is deterministic
 - A given pt is always mapped into a specific ct
 - PT values 0 and 1 produce CT equal to 0 and 1
 - Small exponent and small pt might be subject to attacks
 - RSA is malleable
- Padding is a solution to all these problems
 - Never use plain RSA

(d must be large to avoid
bruteforce. We are talking
about e)

Mar-25

The RSA Cryptosystem

31

e can be small for efficiency
but you can be subject to attacks

RSA malleability [→]

- Malleability
 - A crypto scheme is said to be malleable if the attacker is capable of transforming the ciphertext into another ciphertext which leads to a known transformation of the plaintext
 - The attacker does not decrypt the ciphertext, but (s)he is able to manipulate the plaintext in a predictable manner



Mar-25

The RSA Cryptosystem

32

32



RSA Malleability [→]

- The sender
 - Transmits $y = x^e \text{ mod } n$
- The adversary
 - Intercepts y
 - Chooses s s.t. $\gcd(s, n) = 1$
 - Computes and forwards $y' = s^e \cdot y \text{ mod } n$
- The receiver
 - Decrypts y' , $x' = y'^d = (s^e \cdot y)^d = s^{ed} \cdot y^d = s \cdot x \text{ mod } n$
 - The attacker manages to multiply the ct x by a factor s

Mar-25

The RSA Cryptosystem

33

³³ ex of bad padding: $x||x$. With the attack I get $y' = (x||x) \cdot s \text{ mod } n$
 It is very unlikely that
 $y' = a||a$ for some a .
 Thus solution is unaffected b/w.



RSA Padding

- Padding intuition
 - It embeds a random structure into the plaintext before encryption
 - Padding in RSA *There are standards implemented by libraries*
 - Optimal Asymmetric Encryption Padding (**OAEP**)
 - Specified and standardized in **PKCS#1** (Public Key Cryptography Standard #1)
- Intuition: put redundancy in PT and you want to find it still after decryption*

Mar-25

The RSA Cryptosystem

34

34

Homomorphism



- More in general, RSA malleability descends from the homomorphic property
 - Let x_1 and x_2 two plaintext messages
 - Let y_1 and y_2 their respective encryptions
 - Then, $y \equiv (x_1 \cdot x_2)^e \equiv x_1^e x_2^e \equiv y_1 \cdot y_2 \pmod{n}$
 - That is, the CT of the product is the product of the CTs

Mar-25

The RSA Cryptosystem

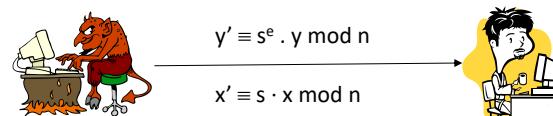
35

35

Adaptive chosen-ciphertext attack



- The problem
 - Assume that Bob decrypts any ciphertext except a given ciphertext y
 - The attacker wants to determine the plaintext corresponding to y



Mar-25

The RSA Cryptosystem

36

36

Adaptive chosen-ciphertext attack



- The attack
 - The adversary selects an integer s , s.t. $\gcd(s, n) = 1$, and sends Bob the quantity $y' \equiv s^e \cdot y \pmod{n}$
 - Upon receiving y' , as $y' \neq y$, Bob decrypts y' , producing $x' \equiv s \cdot x \pmod{n}$, and returns x' to the adversary
 - The adversary determines x , by computing $x \equiv x' \cdot s^{-1} \pmod{n}$
- Countermeasure
 - The attack can be contrasted by using padding
 - Bob returns x' iff it has a structure coherent with padding

Mar-25

The RSA Cryptosystem

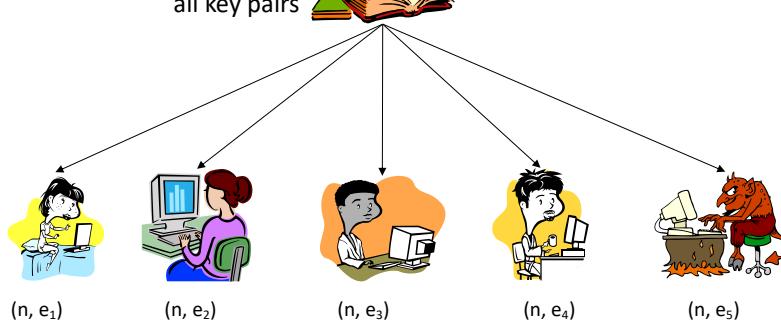
37

37

Common modulus attack



The server uses a common modulus n for all key pairs



Mr Lou Cipher can efficiently factor n from d_5 (FACT 1) and then compute all d 's

Mar-25

The RSA Cryptosystem

38

38



Small message attack

- Let x be a cleartext message, (e, n) a public key, and $y = x^e \bmod n$ a ciphertext message with $x, y \in [0, n-1]$
- Let x be «small», i.e., $x^e < n$, then $y = x^e$ and thus $x = \sqrt[e]{y}$ which is a “normal” e -th root operation and therefore “easy”.

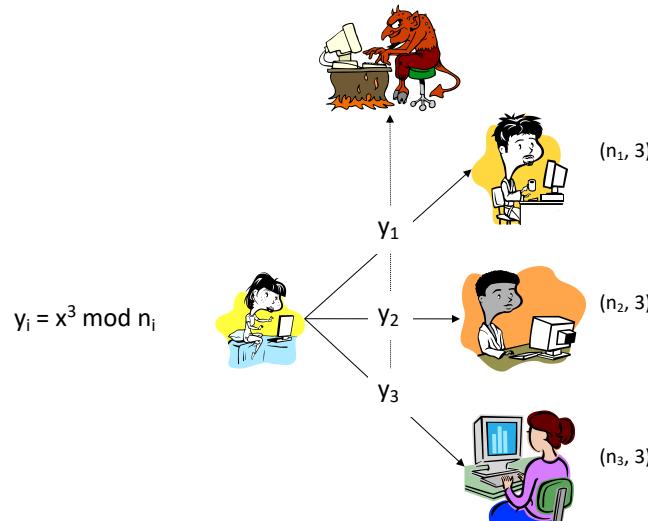
Mar-25

The RSA Cryptosystem

39

39

Low Exponent Attack



Mar-25

The RSA Cryptosystem

40

40

Cinese Remainder Theorem [→]



- CHINESE REMAINDER THEOREM

- If the integers n_1, n_2, \dots, n_k are pairwise relatively prime, then the system of simultaneous congruences
 - $x \equiv a_1 \pmod{n_1}$
 - $x \equiv a_2 \pmod{n_2}$
 - ...
 - $x \equiv a_k \pmod{n_k}$
- has a unique solution modulo $n = n_1 \cdot n_2 \cdots n_k$.

Mar-25

The RSA Cryptosystem

41

41

Cinese Remainder Theorem [→]



- GAUSS' ALGORITHM

- The solution x to the simultaneous congruences in the Chinese remainder theorem may be computed as

$$x = \sum_{i=1}^k a_i N_i M_i \pmod{n}$$

where $N_i = n / n_i \pmod{n_i}$ and $M_i = N_i^{-1} \pmod{n}$

- These computations can be performed in $O((\lg n)^2)$ bit operations.
- A CRT solver can be found here:
<https://www.dcode.fr/chinese-remainder>

Mar-25

The RSA Cryptosystem

42

42

Low Exponent Attack



- If n_i are pairwise coprime, use CRT to compute $z = x^3 \pmod{n_1 n_2 n_3}$ that solves

$$\begin{cases} z \equiv y_1 \pmod{n_1} \\ z \equiv y_2 \pmod{n_2} \\ z \equiv y_3 \pmod{n_3} \end{cases}$$
- According to RSA encryption definition $x < n_i$ then $x^3 < n_1 n_2 n_3$ and thus $z = x^3 \rightarrow x$ is the integer cube root of z , $x = \sqrt[3]{z}$
 - This is not a modular root \rightarrow it is “easy”

Mar-25

The RSA Cryptosystem

43

43

Low Exponent Attack



- COUNTERMEASURES
- Salting
 - A different salt for each receiver: $x \parallel \text{salt}_i$
- Use large exponent
 - E.g., $e = 2^{16} + 1$

Mar-25

The RSA Cryptosystem

44

44

Selecting primes p and q – hints



- Primes p and q should be selected so that factoring $n = p \cdot q$ is computationally infeasible, therefore
- p and q should be sufficiently large and about the same bit length (to avoid the elliptic curve factoring algorithm)
- $p - q$ should be not too small
- $(p - 1)/2$ and $(q - 1)/2$ should be relatively prime

Mar-25

The RSA Cryptosystem

45

45

The RSA Cryptosystem

PERFORMANCE

Mar-25

The RSA Cryptosystem

46

46

RSA



UNIVERSITÀ DI PISA

- RSA algorithms for key generation, encryption and decryption are “easy”
- They involve the following operations
 - Discrete exponentiation
 - Generation of large primes
 - Solving diophantine equations

Mar-25

The RSA Cryptosystem

47

47

Computation of e and d (refined)



UNIVERSITÀ DI PISA

1. Select $e \in (1, \varphi(n))$
2. Apply EEA with input parameters n and e
 - $\gcd(\Phi(n), e) = s \cdot \varphi(n) + t \cdot e$ (Diophantine equation)
 - If $\gcd(e, \varphi(n)) = 1$ then
 - Parameter e is a valid public key
 - Unknown $t = e^{-1} \bmod \Phi(n)$, i.e., $t = d \bmod \Phi(n)$
 - If $\gcd(e, \Phi(n)) \neq 1$ then
 - Select another value for e and repeat the process
 - Efficiency
 - Number of steps is close to the number of digit of the input parameter (\approx logarithmic)

Mar-25

The RSA Cryptosystem

48

48

Finding large primes



- Algorithm

```

repeat
    p ← RNG(x);    // secure random generator
    until isPrime(p); // primality test

```

- Comment

- RNG must be secure, i.e., unpredictable

- Problems

- How many random numbers we must test before we have a prime?
- How fast can we check whether a random integer is prime?
- It turns out that both steps are reasonably fast

Mar-25

The RSA Cryptosystem

49

49

How common are primes?



- Let $\pi(x)$ be the number of prime less than x

- Prime Numbers Theorem

- For a very large x , $\pi(x)$ tends to $x/\ln(x)$
- Furthermore, primes are distributed approximately uniformly over $[2, x]$

- Probability to find a prime in $[0, x] \approx 2/(\ln x)$

- As we test only odd numbers: $P = (x/\ln x)/(x/2) = 2/\ln x$
- Expected number of trials to find a prime in $[0, x]$ is $(\ln x)/2$

Mar-25

The RSA Cryptosystem

50

50

Primality tests



UNIVERSITÀ DI PISA

- Primality tests are computationally much easier than factorization
- Practical primality tests are probabilistic
 - At the question: “is p^* prime?” they answer
 - p^* is composed which is always a true statement
 - p^* is prime, which is only true with a high probability
 - Primality test: Fermat test, Miller-Rabin test
- True primality test are more computationally intensive than probabilistic ones
 - Apply true tests to a candidate that passes through probabilistic tests

Mar-25

The RSA Cryptosystem

51

51

Modular ops - complexity



UNIVERSITÀ DI PISA

- Bit complexity of basic operations in \mathbb{Z}_n
 - Let n be on k bits ($n < 2^k$)
 - Let a and b be two integers in \mathbb{Z}_n (on k -bits)
 - Addition $a + b$ can be done in time $O(k)$
 - Subtraction $a - b$ can be done in time $O(k)$
 - Multiplication $a \times b$ can be done in $O(k^2)$
 - Division $b \times a^{-1}$ can be done in time $O(k^2)$
 - Inverse a^{-1} can be done in $O(k)$
 - Modular exponentiation can be done in $O(k^3)$

Mar-25

The RSA Cryptosystem

52

52

Fast exponentiation



- How many multiplications to compute 2^{20} ?
- Grade-school Algorithm requires
 - $2 \times 2 \times 2 \times \dots \times 2 \rightarrow 19$ multiplications
- Square-and-Multiply Algorithm
 - $((2 \times (2^2)^2)^2)^2 \rightarrow 1$ multiplication + 4 squares $\rightarrow 5$ multiplications

Mar-25

The RSA Cryptosystem

53

53

Fast exponentiation



- RSA computes modular exponentiation
 - $a^x \bmod n$, where n is on k bits (i.e., $n \leq 2^k$)
- Grade-school Algorithm
 - requires $(x - 1)$ modular multiplications
 - If x is as large as n , which is exponentially large in k , the Grade-school Algorithm is inefficient
- Square-and-multiply Algorithm
 - requires up to $2k$ multiplications ($2 \times \log_2 x$)
 - Overall, can be done in $O(k^3)$

Mar-25

The RSA Cryptosystem

54

54



UNIVERSITÀ DI PISA

Fast exponentiation

- Square and multiply
 - Exponentiation by repeated squaring and multiplication
 - The exponentiation $a^x \bmod n$ requires at most
 - $\log_2(x)$ multiplications and
 - $\log_2(x)$ squares
 - Proof
 - See next slide

Mar-25

The RSA Cryptosystem

55

55

Fast exponentiation

$$\begin{aligned}
 a^x \bmod n &= a^{(x_{k-1}2^{k-1} + x_{k-2}2^{k-2} + \dots + x_22^2 + x_12 + x_0)} \bmod n \equiv \\
 &a^{x_{k-1}2^{k-1}} a^{x_{k-2}2^{k-2}} \dots a^{x_22^2} a^{x_12} a^{x_0} \bmod n \equiv \\
 &\left(a^{x_{k-1}2^{k-2}} a^{x_{k-2}2^{k-3}} \dots a^{x_22} a^{x_1} \right)^2 a^{x_0} \bmod n \equiv \\
 &\left(\left(a^{x_{k-1}2^{k-3}} a^{x_{k-2}2^{k-4}} \dots a^{x_2} \right)^2 a^{x_1} \right)^2 a^{x_0} \bmod n \equiv \\
 &\dots \\
 &\left(\left(\left(\left(a^{x_{k-1}} \right)^2 a^{x_{k-2}} \right)^2 \dots a^{x_2} \right)^2 a^{x_1} \right)^2 a^{x_0} \bmod n
 \end{aligned}$$

ALGORITHM

```

c ← 1
for (i = k-1; i >= 0; i--) {
    c ← c2 mod n;
    if (xi == 1)
        c ← c × a mod n;
}

```

COMMENT

- always k square operations
- at most k multiplications
 - equal to the number of 1 in the binary representation of x
- Modulo reduction is performed at each round in order to keep the intermediate results small.

Mar-25

The RSA Cryptosystem

56

56

Fast exponentiation – exercise



- Compute $r = a^{20}$
 - $x = 20 = 10100_2$
 - Step 0 (10100)
 - $r_0 = a^1$
 - Step 1 (10100)
 - $r_1 = (a^1)^2 = a^2 = a^{[10]}_2$
 - Step 2 (10100)
 - $r_2 = (r_1)^2 = a^4 = a^{[100]}_2$
 - $r_2 = r_2 \cdot a = x^5 = a^{[101]}_2$
 - Step 3 (10100)
 - $r_3 = (r_2)^2 = a^{10} = a^{[1010]}_2$
 - Step 4 (10100)
 - $r_4 = (r_3)^2 = a^{20} = a^{[10100]}_2$

Mar-25

The RSA Cryptosystem

57

57

Fast exponentiation



- Let $k = 1024$
- #MUL in the Grade-School Algorithm
 - #MUL = 2^{1024} multiplications
- #Ops in the Square-and-Multiply Algorithm
 - #SQ = k
 - #MUL = #(1's in the binary representation)
 - On average #MUL = 0.5K
 - #Ops = $1.5k = 1536$ (multiplications)
 - Each multiplication is on 1024 bits

Mar-25

The RSA Cryptosystem

58

58

RSA fast encryption with short public exponent



UNIVERSITÀ DI PISA

- RSA ops with public exponent e can be speeded-up
 - Encryption, Digital signature verification
 - The public key e can be chosen to be a very small value and RSA is still secure
 - Values
 - $e = 3$ $\#MUL + \#SQ = 2$ (commonly used in practice)
 - $e = 17$ $\#MUL + \#SQ = 5$
 - $e = 2^{16}+1$ $\#MUL + \#SQ = 17$ (avoid small exp attack)
 - $\gcd(e, \Phi(n)) = 1$ must hold.

Mar-25

The RSA Cryptosystem

59

59

RSA decryption



UNIVERSITÀ DI PISA

- Assume a 2048-bit modulus and a 32-bit CPU and determine decryption computing overhead
 - On average $\#MUL+\#SQ = 1.5 \times 2048 = 3072$ long multiplications each of which involves 2018-bit operands
 - Single long-number multiplication
 - Each operand requires $2048/32 = 64$ registers
 - Each long-number multiplication requires $64^2 = 4096$ integer multiplications
 - Modulo reduction requires $64^2 = 4096$ integer multiplications
 - In total $4096 + 4096 = 8192$ integer multiplications for a single long multiplication
 - In total, $3072 \times 8192 = 25.165.824$ integer multiplications

Mar-25

The RSA Cryptosystem

60

60

RSA decryption



UNIVERSITÀ DI PISA

- '70s-'80s: only hardware implementation
- Today, an RSA decryption takes $\approx 100 \mu\text{s}$ on high-speed hw
- End '80s, software implementation becomes possible
- Today, 2048-bit RSA takes $\approx 10 \text{ ms}$ on a 2 GHz CPU
 - Throughput = $2048 \times 100 = 204.800 \text{ bit/s}$
 - ≈ 3 orders of magnitude slower than symmetric encryption

Mar-25

The RSA Cryptosystem

61

61

RSA Fast decryption



UNIVERSITÀ DI PISA

- There is no easy way to accelerate RSA when the private exponent d is involved
 - $\text{sizeof}(d) = \text{sizeof}(n)$ to discourage brute force attack
 - It can be shown that $\text{sizeof}(d) \geq 0.3 \text{ sizeof}(n)$
- One possible approach is based on the Chinese Remainder Theorem (CRT)
 - We do not prove the theorem
 - We just apply it

Mar-25

The RSA Cryptosystem

62

62

Fast RSA decryption by CRT [→]



- Problem: Compute $y \equiv x^d \pmod{n}$ efficiently
- The method
 1. Transformation of the problem in the CRT domain
 1. Compute $x_p \equiv x \pmod{p}$
 2. Compute $x_q \equiv x \pmod{q}$
 2. Exponentiation in the CRT domain
 1. $y_p \equiv x_p^{d_p} \pmod{p}$, where $d_p \equiv d \pmod{p-1}$
 2. $y_q \equiv x_q^{d_q} \pmod{q}$, where $d_q \equiv d \pmod{q-1}$

Mar-25

The RSA Cryptosystem

63

63

Fast RSA decryption by CRT [→]



- The method (cont.ed)
- 3. Inverse transformation in the problem domain
 1. $y \equiv [q \cdot c_p]y_p + [p \cdot c_q]y_q \pmod{n}$ where
 - $c_p \equiv q^{-1} \pmod{p}$ and
 - $c_q \equiv p^{-1} \pmod{q}$

Mar-25

The RSA Cryptosystem

64

64

Fast RSA decryption by CRT [→]



- Comments

- With reference to step 2, as $\text{sizeof}(p) = \text{sizeof}(q)$, d_p , d_q , y_p , y_q have about half the bit length of n
 - This leads to a speedup = 4
- With reference to step 3, expressions in square brackets can be precomputed
 - Then, the inverse transformation requires two modular multiplications and one modular addition

Mar-25

The RSA Cryptosystem

65

65

Fast RSA decryption by CRT



- Complexity of CRT-based RSA decryption

- Step 1 and step 3 are negligible
- Step 2
 - Let n length is t bits, then all quantities in step 2 are on $t/2$ bits
 - By applying the Square-and-multiply algorithm
 - #OPS = #SQ + #MUL = $2 \times (1.5 t/2) = 1.5 t$
 - The #OPS is the same as without CRT, however, each operation involve $t/2$ -bit operands instead of t -bit operand, so its time is $(t/2)^2$
 - It follows that the total speed up is a factor of 4

- The method is subject to fault-injection attack

Mar-25

The RSA Cryptosystem

66

66