

UNIVERSITY OF MASSACHUSETTS
Dept. of Electrical & Computer Engineering

Introduction to Cryptography
ECE 597XX/697XX

Part 7

**Introduction to
the RSA Cryptosystem**

Israel Koren

ECE597/697 Koren Part.7 .1

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Content of this Chapter

- **The RSA Cryptosystem**
- **Implementation aspects**
- **Finding Large Primes**
- **Attacks and Countermeasures**

ECE597/697 Koren Part.7 .2

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

The RSA Cryptosystem

- Hellman and Diffie published their landmark public-key paper in 1976
- Rivest, Shamir and Adleman proposed the asymmetric RSA cryptosystem in 1977
- Until now, RSA is the most widely used asymmetric cryptosystem although elliptic curve cryptography (ECC) becomes increasingly popular
- RSA is mainly used for two applications
 - Transmission of (i.e., symmetric) keys
 - Digital signatures

ECE597/697 Koren Part.7.3

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Encryption and Decryption

- RSA operations are done over the integer ring Z_n (i.e., arithmetic modulo n), where $n = p * q$, with p, q large primes
- Encryption and decryption are exponentiations in the ring

Definition: Given the public key $(n, e) = k_{pub}$ and the private key $d = k_{pr}$ we write

$$y = e_{k_{pub}}(x) \equiv x^e \bmod n$$

$$x = d_{k_{pr}}(y) \equiv y^d \bmod n$$

where $x, y \in Z_n$.

We call $e_{k_{pub}}()$ the encryption and $d_{k_{pr}}()$ the decryption operation.

- In practice x, y, n and d are very long integers (≥ 1024 bits)
- The security of the scheme relies on the fact that it is hard to derive the „private exponent“ d given the public-key (n, e)

ECE597/697 Koren Part.7.4

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Key Generation

- Like all asymmetric schemes, RSA has set-up phase during which the private and public keys are computed

RSA
key
gener-
ation

Output: public key: $k_{pub} = (n, e)$ and private key $k_{pr} = d$

1. Choose two large primes p, q
2. Compute $n = p * q$
3. Compute $\Phi(n) = (p-1) * (q-1)$
4. Select the public exponent $e \in \{1, 2, \dots, \Phi(n)-1\}$ such that $\gcd(e, \Phi(n)) = 1$
5. Compute the private key d such that $d * e \equiv 1 \mod \Phi(n)$
6. RETURN $k_{pub} = (n, e), k_{pr} = d$

Remarks:

- Choosing two large primes p, q (in Step 1) is non-trivial
- $\gcd(e, \Phi(n)) = 1$ ensures that e has an inverse and, thus, there is always a private key d

ECE597/697 Koren Part.7 .5

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Example: RSA with small numbers

ALICE

Message $x = 9$

BOB

1. Choose $p = 7$ and $q = 11$
2. Compute $n = p * q = 77$
3. $\Phi(n) = (7-1) * (11-1) = 60$
4. Choose $e = 7; \gcd(7, 60) = 1$
5. $d \equiv e^{-1} \equiv 43 \mod 60$
 $7 \times 43 = 301 \equiv 1 \mod 60$
(use EEA to get d)

$K_{pub} = (77, 7)$

$$y = x^e = 9^7 = 4782969 \equiv 37 \mod 77$$

$y = 37$

$$y^d = 37^{43} \equiv 9 \mod 77$$

ECE597/697 Koren Part.7 .6

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Proof

- Show that $x = y^d \bmod n = (x^e)^d \bmod n$
- Case 1: $\gcd(x, n) = 1$
- 1. Since $d \cdot e \equiv 1 \bmod \phi(n)$ we can write $d \cdot e = 1 + t \cdot \phi(n)$
- 2. $(x^e)^d \equiv x^{ed} \equiv x^{1+t\phi(n)} \equiv (x^{\phi(n)})^t \cdot x \bmod n$
- 3. Based on Euler's theorem if $\gcd(x, n) = 1$ then $1 \equiv x^{\phi(n)} \bmod n$
- 4. Thus, $(x^{\phi(n)})^t \equiv 1^t \bmod n \equiv 1 \bmod n$
- Case 2: $\gcd(x, n) \neq 1$ then $x = a \cdot p$ (or $a \cdot q$)
- 1. and 2. above
- 3a. $x^{\phi(q)} = x^{q-1} = 1 \bmod q$ since $\gcd(x, q) = 1$ (Euler's theorem)
- 3b. $(x^{\phi(n)})^t = ((x^{q-1})^{p-1})^t = 1 \bmod q$ thus $(x^{\phi(n)})^t = 1 + b \cdot q$
- 4. $(x^{\phi(n)})^t \cdot x = (1 + b \cdot q) \cdot x = x + x \cdot b \cdot q = x + a \cdot p \cdot b \cdot q = x \bmod n$

ECE597/697 Koren Part.7 .7

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Implementation aspects

- RSA uses only one arithmetic operation (modular exponentiation) which makes it conceptually simple
- But, due to the use of very long numbers, RSA is orders of magnitude slower than symmetric schemes, e.g., DES, AES
- When implementing RSA (esp. on a constrained device such as smartcards or cell phones) close attention has to be paid to the correct choice of implementation algorithm
- The **square-and-multiply** algorithm allows fast exponentiation, even with very long numbers

ECE597/697 Koren Part.7 .8

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Square-and-Multiply

- **Basic principle:** Scan exponent bits from left to right and square/multiply operand accordingly

Input: Exponent H , base element x , Modulus n

Initialization: $y=x$; Output: $y = x^H \bmod n$

1. Determine binary representation $H = (h_t, h_{t-1}, \dots, h_0)_2$
2. FOR $i = t-1$ TO 0
3. $y = y^2 \bmod n$
4. IF $h_i = 1$ THEN
5. $y = y * x \bmod n$
6. RETURN y

- **Rule:** Square in every iteration (Step 3) and multiply current result by x if the exponent bit $h_i = 1$ (Step 5)
- Modulo reduction after each step keeps the operand y small

ECE597/697 Koren Part.7 .9

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Example: Square-and-Multiply

- Computes x^{26} without modulo reduction
- Binary rep. of exponent: $26 = (1, 1, 0, 1, 0)_2 = (h_4, h_3, h_2, h_1, h_0)_2$

Step		Binary exponent	Op	Comment
1	$x = x^1$	$(1)_2$		Initial setting, h_4 processed
1a	$(x^1)^2 = x^2$	$(10)_2$	SQ	Processing h_3
1b	$x^2 * x = x^3$	$(11)_2$	MUL	$h_3 = 1$
2a	$(x^3)^2 = x^6$	$(110)_2$	SQ	Processing h_2
2b	-	$(110)_2$	-	$h_2 = 0$
3a	$(x^6)^2 = x^{12}$	$(1100)_2$	SQ	Processing h_1
3b	$x^{12} * x = x^{13}$	$(1101)_2$	MUL	$h_1 = 1$
4a	$(x^{13})^2 = x^{26}$	$(11010)_2$	SQ	Processing h_0
4b	-	$(11010)_2$	-	$h_0 = 0$

- Observe how the exponent evolves into $x^{26} = x^{11010}$
- 6 operations instead of 25 multiplications

ECE597/697 Koren Part.7 .10

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Complexity of Square-and-Multiply Alg.

- The square-and-multiply algorithm has a logarithmic complexity, i.e., its run time is proportional to the bit length (rather than the absolute value) of the exponent
- Given an exponent with $t+1$ bits
$$H = (h_t, h_{t-1}, \dots, h_0)_2$$
with $h_t = 1$, we need the following operations
 - # Squarings $= t$
 - Average # multiplications $= 0.5 t$
 - Total complexity: $\#SQ + \#MUL = 1.5 t$
- Exponents are often randomly chosen, so $1.5 t$ is a good estimate for the average number of operations
- Note that each squaring and each multiplication is an operation with very long numbers, e.g., 2048 bit integers.

ECE597/697 Koren Part.7 .11

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Speed-Up Techniques

- Modular exponentiation is computationally intensive
- Even with the square-and-multiply algorithm, RSA can be quite slow on constrained devices
- Some important tricks:
 - Short public exponent e
 - Chinese Remainder Theorem (CRT)
 - Exponentiation with pre-computation (*not covered here*)

ECE597/697 Koren Part.7 .12

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Fast encryption with small public exponent

- Choosing a small public exponent e does not weaken the security of RSA
- A small public exponent improves the speed of the RSA encryption significantly

Public Key	e as binary string	#MUL + #SQ
$2^1 + 1 = 3$	$(11)_2$	$1 + 1 = 2$
$2^4 + 1 = 17$	$(1\ 0001)_2$	$4 + 1 = 5$
$2^{16} + 1$	$(1\ 0000\ 0000\ 0000\ 0001)_2$	$16 + 1 = 17$

- Small Hamming weight
- Commonly used trick (e.g., SSL/TLS, etc.): makes RSA the fastest asymmetric scheme with regard to encryption

ECE597/697 Koren Part.7 .13

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Fast decryption with CRT

- Choosing a small private key d results in security weakness
 - In fact, d must have at least $0.3t$ bits, where t is the bit length of the modulus n
- However, the Chinese Remainder Theorem (CRT) can be used to accelerate exponentiation with the private key d
- Based on the CRT we can replace the computation of

$$x^d \bmod \varphi(n) \bmod n$$

by two computations

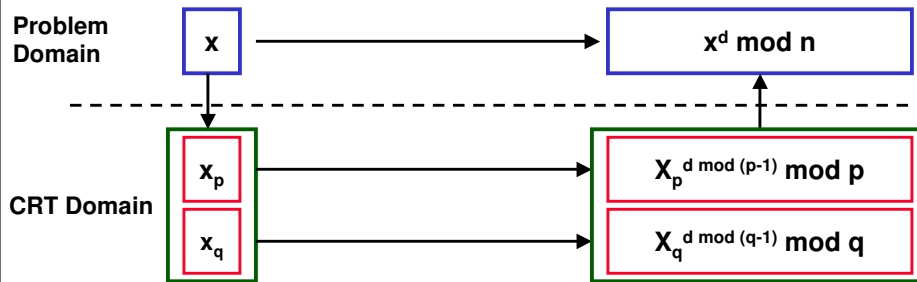
$$x^d \bmod (p-1) \bmod p \quad \text{and} \quad x^d \bmod (q-1) \bmod q$$

where q and p are „small“ compared to n

ECE597/697 Koren Part.7 .14

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Basic principle of CRT-based exponentiation



- CRT-based exponentiation involves three distinct steps
 - (1) Transformation of operand into the CRT domain
 - (2) Modular exponentiation in the CRT domain
 - (3) Inverse transformation into the problem domain
- These steps are equivalent to one modular exponentiation in the problem domain

ECE597/697 Koren Part.7 .15

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

CRT: Step 1 - Transformation

- Transformation into the CRT domain requires the knowledge of p and q
- p and q are only known to the owner of the private key, hence CRT cannot be applied to speed up encryption
- The transformation computes (x_p, x_q) which is the representation of x in the CRT domain. They can be found easily by computing

$$x_p \equiv x \bmod p \quad \text{and} \quad x_q \equiv x \bmod q$$

ECE597/697 Koren Part.7 .16

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

CRT: Step 2 – Exponentiation

- Given d_p and d_q such that

$$d_p \equiv d \bmod (p-1) \quad \text{and} \quad d_q \equiv d \bmod (q-1)$$

one exponentiation in the problem domain requires two exponentiations in the CRT domain

$$y_p \equiv x_p^{d_p} \bmod p \quad \text{and} \quad y_q \equiv x_q^{d_q} \bmod q$$

- In practice, p and q are chosen to have half the bit length of n , i.e., $|p| \approx |q| \approx |n|/2$

ECE597/697 Koren Part.7 .17

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

CRT: Step 3 – Inverse Transformation

- Inverse transformation requires modular inversion twice, which is computationally expensive

$$c_p \equiv q^{-1} \bmod p \quad \text{and} \quad c_q \equiv p^{-1} \bmod q$$

- Inverse transformation assembles y_p, y_q to the final result $y \bmod n$ in the problem domain

$$y \equiv [q * c_p] * y_p + [p * c_q] * y_q \bmod n$$

$$y \equiv [q * q^{-1} \bmod p] * y_p + [p * p^{-1} \bmod q] * y_q \bmod n$$

- The primes p and q typically change infrequently, therefore the cost of inversion can be neglected because the two expressions

$$[q * c_p] \quad \text{and} \quad [p * c_q]$$

can be precomputed and stored

ECE597/697 Koren Part.7 .18

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Example

$p=13, q=11, n=pq=143; \phi(n) = (p-1)(q-1)=120=8 \cdot 3 \cdot 5$

Select $e=77$ as $\gcd(77,120)=1$; $d=53 \bmod 120$ as
 $77 \cdot 53 \equiv 1 \bmod 120$

Encryption: Given $x=101$ then $y = x^e = 101^{77} = 101^{1001101} =$
 $(((((101^2)^2)^2 101)^2 101)^2)^2 101 \bmod 143 = 95 \bmod 143$

Decryption:

$$x_p = y^{d \bmod (p-1)} \bmod p = 95^{53 \bmod 12} \bmod 13 = 4^5 \bmod 13 = 10 \bmod 13$$

$$x_q = y^{d \bmod (q-1)} \bmod q = 95^{53 \bmod 10} \bmod 11 = 7^3 \bmod 11 = 2 \bmod 11$$

$$c_p \equiv q^{-1} \bmod p \equiv 11^{-1} \bmod 13 = 6 \bmod 13 \text{ since } 11 \cdot 6 \equiv 1 \bmod 13$$

$$c_q \equiv p^{-1} \bmod q \equiv 13^{-1} \bmod 11 = 6 \bmod 11 \text{ since } 13 \cdot 6 \equiv 1 \bmod 11$$

$$x = 11 \cdot 6 \cdot 10 + 13 \cdot 6 \cdot 2 = 66 \cdot 10 + 78 \cdot 2 \bmod 143 = 88 + 13 \bmod 143 = 101$$

ECE597/697 Koren Part.7 .19

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Complexity of CRT

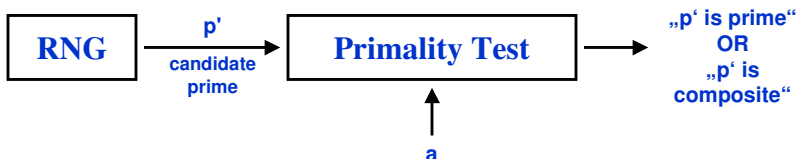
- We ignore the transformation and inverse transformation steps since their costs can be neglected under reasonable assumptions
- Assuming that n has $t+1$ bits, p and q are about $t/2$ bits long
- The complexity is determined by the two exponentiations (using the square-and-multiply algorithm) in the CRT domain. The operands are only $t/2$ bits long:
 - # squarings (one exp.): $\#SQ = 0.5 t$
 - # aver. multiplications (one exp.): $\#MUL = 0.25t$
 - Total complexity: $2 * (\#MUL + \#SQ) = 1.5t$
- This looks the same as regular exponentiations, but since the operands have half the bit length compared to regular exponent, each operation (i.e., multiplying and squaring) is 4 times faster
- Hence CRT is 4 times faster than straightforward exponentiation

ECE597/697 Koren Part.7 .20

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Finding Large Primes

- Generating keys for RSA requires finding two large primes p and q such that $n = p * q$ is sufficiently large
- Size of p and q is typically half the desired size of n
- To find primes, random integers are generated and tested for primality:



- The random number generator (RNG) should be non-predictable otherwise an attacker could guess the factorization of n

ECE597/697 Koren Part.7 .21

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Primality Tests

- Factoring p and q to test for primality is hard
- However, we are not interested in the factorization, we only want to know whether p and q are composite
- Typical primality tests are probabilistic, i.e., they are not 100% accurate but their output is correct with very high probability
- A probabilistic test has two outputs:
 - „p' is composite" - always true
 - „p' is a prime" - only true with a certain probability
- Among the well-known primality tests are the following
 - Fermat Primality-Test
 - Miller-Rabin Primality-Test

ECE597/697 Koren Part.7 .22

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Fermat Primality-Test

- Basic idea: Fermat's Little Theorem holds for all primes, i.e., if a number p' is found for which $a^{p'-1} \not\equiv 1 \pmod{p'}$, it is not a prime

Input: Prime candidate p' , security parameter s

Output: p' is composite or is likely a prime

1. FOR $i = 1$ TO s
2. choose random $a \in \{2, 3, \dots, p'-2\}$
3. IF $a^{p'-1} \not\equiv 1 \pmod{p'}$ THEN
4. RETURN „ p' is composite“
5. RETURN „ p' is likely a prime“

- For certain numbers („Carmichael numbers“) this test often returns „ p is likely a prime“ – although they are composite
- Therefore, the Miller-Rabin Test is preferred
- Example: $561 = 3 \cdot 11 \cdot 17$; $a^{560} \equiv 1 \pmod{561}$ for all $\gcd(a, 561) = 1$

ECE597/697 Koren Part.7 .23

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Theorem for Miller-Rabin's test

- The more powerful Miller-Rabin Test is based on the following theorem

Theorem: Given the decomposition of an odd prime candidate p'

$$p' - 1 = 2^u \cdot r$$

where r is odd. If we can find an integer a such that

$$a^r \not\equiv 1 \pmod{p'} \quad \text{and} \quad a^{r \cdot 2^j} \not\equiv p' - 1 \pmod{p'}$$

For all $j = \{0, 1, \dots, u-1\}$, then p' is composite.

Otherwise it is probably a prime.

- This theorem can be turned into an algorithm

ECE597/697 Koren Part.7 .24

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Miller-Rabin Primality-Test

Input: Prime candidate p' with $p'-1 = 2^u \cdot r$ & security parameter s

Output: „ p' is composite” or „ p' is likely a prime”

```

FOR  $i = 1$  TO  $s$ 
  choose random  $a \in \{2, 3, \dots, p'-2\}$ 
   $z \equiv a^r \pmod{p'}$ 
  IF  $z \neq 1$  AND  $z \neq p'-1$  THEN
    FOR  $j = 1$  TO  $u-1$ 
       $z \equiv z^2 \pmod{p'}$ 
      IF  $z = 1$  THEN
        RETURN „ $p'$  is composite”
      IF  $z \neq p'-1$  THEN
        RETURN „ $p'$  is composite”
    RETURN „ $p'$  is likely a prime”

```

Bit lengths of $\tilde{p}$	Security parameter s
250	11
300	9
400	6
500	5
600	3

ECE597/697 Koren Part.7 .25

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Attacks and Countermeasures

- There are two distinct types of attacks on cryptosystems
 - **Analytical attacks** try to break the mathematical structure of the underlying problem of RSA
 - **Implementation attacks** try to attack a real-world implementation by exploiting inherent weaknesses in the way RSA is realized in software or hardware

Analytical Attacks: (1) Mathematical attacks

- The best known attack is factoring of n to obtain $\mathcal{E}(n)$
- Can be prevented using a sufficiently large n - current factoring record is 664 bits. Thus, n should have between 1024 and 3072 bits

ECE597/697 Koren Part.7 .26

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Analytical Attacks

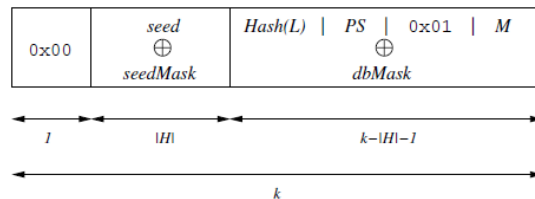
(2) Protocol attacks

- Exploit the malleability of RSA, e.g., a ciphertext can be modified without knowing the private key - Oscar can replace ciphertext y by $y \cdot s^e$ with an integer s resulting in the decrypted message $x \cdot s$
- RSA is deterministic: identical plaintexts $\Rightarrow$ same ciphertext
- $x=0,1,-1$ produce $y=0,1,-1$
- Can be prevented by proper padding (using random numbers)
- Standard padding: Optimal Asymmetric Encryption Padding (OAEP)
- Force all messages to be encrypted to have the maximum length, i.e., k bytes where k is the length of the modulus n in bytes
- Include a random byte string

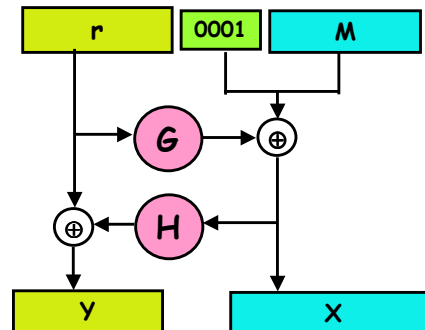
ECE597/697 Koren Part.7 .27

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Optimal Asymmetric Encryption Padding



- Hash(L) | zero bytes | 01 | M where L = optional message label
- Hash(L) has a fixed length
- G expands the bits of the random number to the required number
- Decode:
- Recover $r = Y \oplus H(x)$
- Data = $X \oplus G(r)$

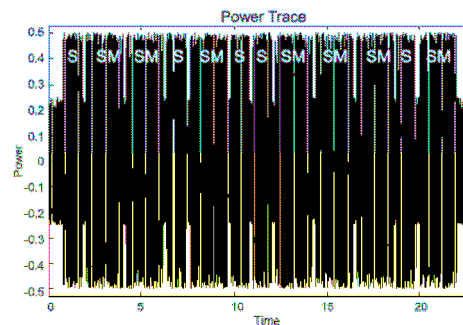


ECE597/697 Koren Part.7 .28

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Implementation Attacks

- Implementation attacks can be one of the following
- Side-channel analysis:
- Exploit physical leakage of RSA implementation (e.g., power consumption, EM emanation, etc.)
- Fault-injection attacks:
- Inducing faults in the device while CRT is executed can lead to a complete leakage of the private key



ECE597/697 Koren Part.7 .29

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Lessons Learned

- RSA is still the most widely used public-key cryptosystem
- RSA is mainly used for key transport and digital signatures
- The public key e can be a short integer, the private key d needs to have almost the full length of the modulus n
- RSA relies on the fact that it is hard to factorize n
- Currently 1024-bit cannot be factored, but progress in factorization could bring this into reach within 10-15 years. Hence, RSA with a 2048 or 3076 bit modulus should be used for long-term security
- A naive implementation of RSA allows several attacks, and in practice RSA should be used together with padding

ECE597/697 Koren Part.7 .30

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources