Asymmetric Cryptographic Systems based on Trapdoor Functions

Dr. Basel Halak

Learning Outcomes

At the end of this lecture you should be able to:

1. Describe the principles of public key crypto systems.
2. Define trapdoor functions
3. Outline the principles of RSA algorithm
4. Evaluate the security of RSA-based Ciphers

Learning Outcomes

At the end of this lecture you should be able to:

1. Describe the principles of public key crypto systems.
2. Define trapdoor functions
3. Outline the principles of RSA algorithm
4. Evaluate the security of RSA-based Ciphers

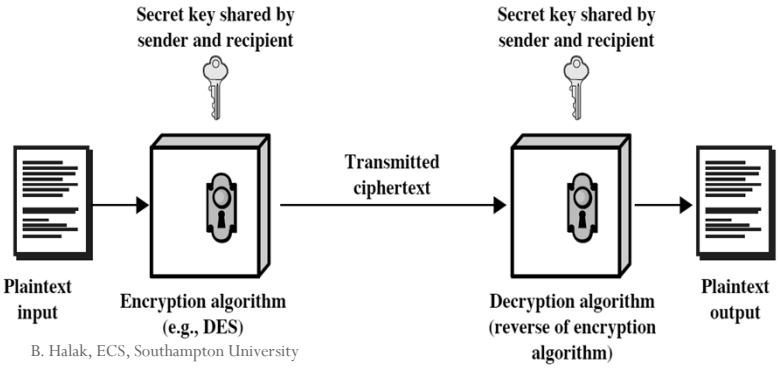
Principles of Cryptographic Systems

- Cryptographic can be classified into two distinct categories:

1. Shared Key Systems
2. Public Key Systems

Principles of Cryptographic Systems

- **Shared Key Systems :** Both sender and receiver use the same key which must remain private. These systems are also called **symmetric**
- **Examples**: DES, Triple-DES, Twofish, Rijndael

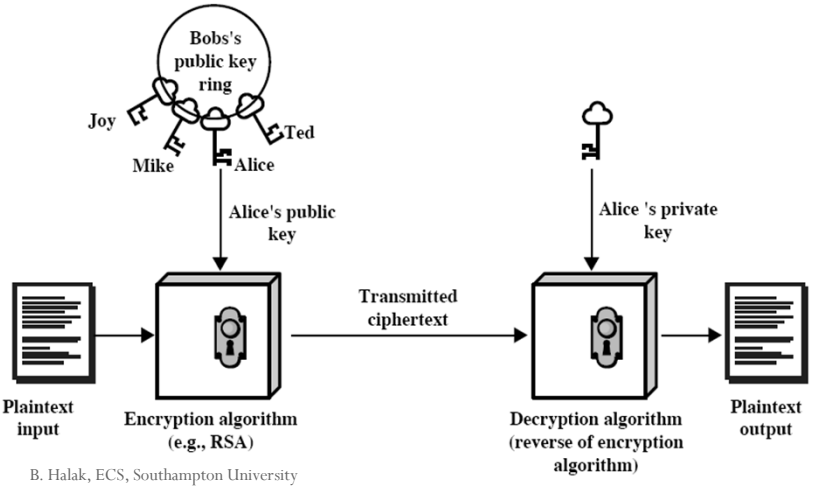


Problems with Shared Key Systems

1. Compromised key means interceptors can decrypt any ciphertext they have acquired. Keys can be changed frequently to limit damage.
2. Distribution of keys is problematic: keys must be transmitted securely e.g. distribute key in pieces over separate channels.

Principles of Cryptographic Systems

- **Public Key Systems**



Principles of Cryptographic Systems

- **Public Key Systems:** also known as **asymmetric-key** systems. Each user has a pair of keys: a public key and a private key. The public key is used for encryption. The key is known to the public(i.e. other users of the systems). The private key is used for decryption. The key is only known to the owner.

Learning Outcomes

At the end of this lecture you should be able to:

1. Describe the principles of public key crypto systems.
2. Define trapdoor functions
3. Outlines the principles of RSA algorithm
4. Evaluate the security of RSA-based Ciphers

Trapdoor Functions (TDF)

- Construct a pair of functions f, f^{-1}
- Given x, evaluation of f(x) is trivial
- Given only f and y, evaluation of $f^{-1}(y)$ computationally difficult
- But, given f^{-1} & y, evaluation of $f^{-1}(y)$ is trivial

Trapdoor Functions (TDF)

- **Definition**: a trapdoor function. $X \longrightarrow Y$ is a triple of efficient algorithms (G, F, F^{-1}):

1. A Key generation algorithm(G): randomized algorithm outputs a key pair (pk, sk)
2. An Encryption function $(pk,\cdot)$: deterministic algorithm that defines a function $X \longrightarrow Y$
3. A Decryption Function: $F^{-1}(sk,\cdot)$: defines a function $Y \longrightarrow X$ that inverts $F(pk,\cdot)$

In other words : $\forall$(pk, sk) output by G $\forall x \in X$: $F^{-1}(sk, F(pk, x)) = x$

Trapdoor Functions (TDFs)

What is a secure TDF?

- **Intuitively** : (G, F, F^{-1}) is secure if $F(pk, \cdot)$ is a "one-way" function: i.e. TDF can be evaluated, but cannot be inverted without sk

- **It is assumed that the adversary has access to:**

1. An Encrypted message (C)
2. The public key of the intended recipient(pk)
3. The trapdoor function F.

He can try to decrypt the message by finding the inverse function F^{-1} and applying it on the cipher text

A(C, F, pk) → p'

- **Mathematically**: (G, F, F^{-1}) is a secure TDF if for all efficient A:

$$Adv[A,F] = \mathbf{Pr[\ p = p'\ ]} < \text{negligible}$$

How not to use a Trapdoor Function (TDF)

- **Never encrypt by applying F directly to plaintext:**

E(pk, m) :	**D(sk, c) :**
output $c \leftarrow F(pk, m)$	output $F^{-1}(sk, c)$

- **Problems**

1. Deterministic: cannot be semantically secure !!
2. Many attacks exist

Learning Outcomes

At the end of this lecture you should be able to:

1. Describe the principles of public key crypto systems.
2. Define trapdoor functions
3. Outline the principles of RSA algorithm
4. Evaluate the security of RSA-based Ciphers
5. Use RSA algorithm to build a digital signature scheme

Number Theory Review

- Let $N = p \cdot q$ where p,q are prime numbers
- $Z_N = \{0,1,2,\ldots,N-1\}$; $(Z_N)^* = \{\text{invertible elements in } Z_N\}$
- $x \in Z_N$ is invertible $\Leftrightarrow$ $\gcd(x, N) = 1$
- Number of elements in $(Z_N)^*$ is $\varphi(N) = (p-1)(q-1) = N-p-q+1$

Number Theory Review

- **How to Compute Euler Totient Function φ(N) :**
  - For N= p (p prime) **φ(N) = N-1**
  - For N=p.q (p,q prime) **φ(N) = (p-1)(q-1)**
- **Examples:**
  φ(37) = 36
  φ(21) = (3−1)×(7−1) = 2×6 = 12
- **Euler's Theorem**: a generalisation of Fermat's Theorem

$$\forall\, x \in (Z_N)^* \;:\; x^{\varphi(N)} \bmod N = 1$$

The RSA as TDF Function

Basic Key Generation Algorithm:

1. Choose random primes p,q
2. Set **N=pq**.
3. Choose integers **e , d** s.t. **e·d = 1 (mod φ(N))**
4. Output **pk = (N, e) , sk = (p,q , d)**

The RSA as TDF Function

Encryption Algorithm

$$F(pk, x): (\boldsymbol{Z_N})^* \rightarrow (\boldsymbol{Z_N})^*: RSA(x, e) = x^e$$

Decryption Algorithm

$$F(sk, y): (\boldsymbol{Z_N})^* \rightarrow (\boldsymbol{Z_N})^*: RSA(y, d) = y^d$$

Quiz: Prove that $y^d = x$

The RSA as TDF Function

Encryption Algorithm

$$F(pk, x): (\boldsymbol{Z_N})^* \rightarrow (\boldsymbol{Z_N})^*: RSA(x, e) = x^e$$

Decryption Algorithm

$$F(sk, y): (\boldsymbol{Z_N})^* \rightarrow (\boldsymbol{Z_N})^*: RSA(y, d) = y^d$$

Quiz: Prove that $y^d = x$

$$y^d = (x^e)^d = x^{ed} = x^{(k\varphi(N)+1)} = \left(x^{(\varphi(N))}\right)^k \cdot x = x$$

RSA Example

- **Key Generation:**

1. Select primes: p=17 & q=11
2. Compute $n = pq$ =17×11=187
3. Compute ø(n)=(p–1)(q-1)=16×10=160
4. Select e : gcd(e,160)=1; choose e=7
5. Determine d: de=1 mod 160 and d < 160 Value is d=23 since 23×7=161= 10×160+1
6. Publish public key : pk = (N, e) =(7,187)
7. Keep secret private key : **sk = (p,q , d)**=(17,11,23)

- **Exercis**e:

Encrypt the message M=88, using the above scheme

RSA Example

- **Sample RSA encryption/decryption:**
- Given a message M = 88 (88<187)
- Encryption:
  $C = 88^7 \bmod 187 = 11$
- Decryption:
  $M = 11^{23} \bmod 187 = 88$

Learning Outcomes

At the end of this lecture you should be able to:

1. Describe the principles of public key crypto systems.
2. Define trapdoor functions
3. Outlines the principles of RSA algorithm
4. Evaluate the security of RSA-based Ciphers

RSA Security

To invert the RSA one-way function (without d) attacker must compute,

$$y = (x^e) \pmod N \Rightarrow x = \sqrt[e]{y} \pmod N$$

RSA Security

To invert the RSA one-way function (without d) attacker must compute,

$$y = (x^e) \pmod N \Rightarrow x = \sqrt[e]{y} \pmod N$$

- **However**: if an adversary could factorize **N** she would know **p** and **q** and hence also Ø(n). Since **e** is public knowledge she could then find **d**, and easily compute $x = y^d$
- RSA security depends on the fact that it is so far very difficult to factorize large integers

RSA Security

- **Brute Force Approach for finding p** and **q** :

1. Find $\sqrt{n}$
2. It is clear that either $p \leq \sqrt{n}$ *or* $q \leq \sqrt{n}$
3. Divide n by each of the primes until a factor is found

- **However the number of primes** $\leq \sqrt{n}$ **are about** $\frac{2\sqrt{n}}{\ln(n)}$

for large n very inefficient methods

RSA Security

- **A Fermat number is a positive integer of the form:**

$$F_i = 2^{(2^i)} + 1 \quad i \geq 1$$

Examples: $F_1 = 5$, $F_2 = 17$, $F_3 = 257$, $F_4 = 65537$.

- **Factorization of Fermat numbers can show the progress that has been made in the area of factorization:**

1732 L. Euler found that $F_5 = 4294967297 = 641 \cdot 6700417$

1880 Landry+LeLasser found that

$F_6 = 18446744073709551617 = 274177 \cdot 67280421310721$

1970 Morrison+Brillhart found factorization for F_7 =(39 digits)

1980 Brent+Pollard found factorization for F_8 (78 digits)

As of 2015 all numbers up to F_{11} have been factored

RSA Security

- Best Known Algorithm is number field sieve: with a run time of the order: $e^{O(\sqrt[3]{N})}$
- Current Record: **RSA-768** (232 digits) (two years and 100s of machines)
- Factorizing a 1024-bit integer ?

Choosing the primes p and q

- **There are some precautions which need to be taken in choosing the primes p and q**. For instance, they should not be too close together, for otherwise it might be possible to factorize n by Fermat factorization,
- **Fermat's factorization** method is based on the representation of an odd integer as the difference of two squares:

$$N = i^2 - j^2 = (i - j)(i + j)$$

Fermat's Factorization Method

- **Fermat's factorization** method is based on the representation of an odd integer as the difference of two squares:

$$N = i^2 - j^2 = (i - j)(i + j)$$

- Indeed, if N can be factored:

$$N = p.q$$

- This can be written as:

$$N = \left(\frac{p+q}{2}\right)^2 - \left(\frac{p-q}{2}\right)^2$$

- **So if p and q are very close:**

$$\frac{p-q}{2} \approx 0 \text{ and so } \sqrt{n} \approx \frac{p+q}{2}$$

Fermat's Factorization Method

- So if p and q are very close:

$$\frac{p-q}{2} \approx 0 \text{ and so } \sqrt{n} \approx \frac{p+q}{2}$$

- **In this case, factorising N will be too easy as follows:**

1. Choose integers i slightly larger than $\sqrt{n}$
2. Test whether $i^2 - N$ is a perfect square j^2 for some integer j
3. If i can be found than: $N = i^2 - j^2 = (i - j)(i + j)$

Fermat's Factorization Method

- Example: Factorise N= 11413

Fermat's Factorization Method

- Example: Factorise N= 11413

$107\ is\ sligtly\ smaller\ than \sqrt{N}$

So we try $\mathrm{i} = 107, therfore\ \ i^2 - N = 11449 - 11413 = 36 = 6^2$

Thus:

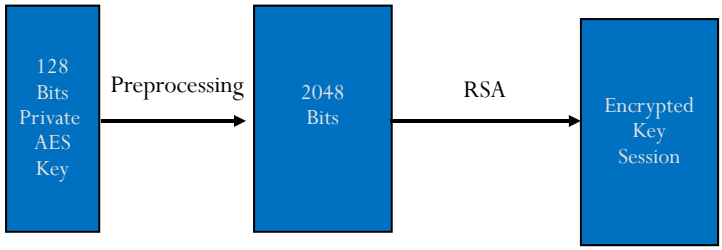
$N = 117^2 - 6^2 = (107 - 6)(107 + 6) = 101 * 113$

Textbook RSA

- **Textbook RSA encryption:**
  - public key: **(N,e)** Encrypt: $c \leftarrow m^e$ (in Z_N)
  - secret key: **(N,d)** Decrypt: $c^d \rightarrow m$
- **Is this a semantically secure encryption system ?**

How to use RSA encryption in practice

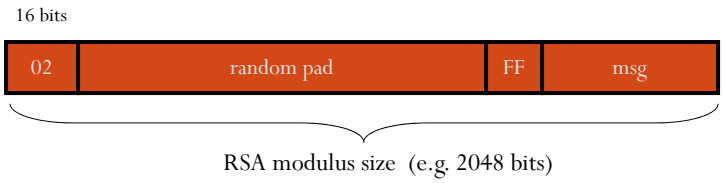
- Never use textbook RSA.
- Example: RSA in practice (RSA-2048 bits encryption of 128 bit AES Private key)



Main questions:
- How should the preprocessing be done?
- Can we argue about security of resulting system?

PKCS1 v1.5

PKCS1 mode 2: (encryption)



- Resulting value is RSA encrypted
- Widely deployed, e.g. in HTTPS

Implementation attacks

Timing Attacks:

- It is developed in mid-1990's to exploit timing variations in operations eg. multiplying by small vs large number. It can infer operand size based on time taken.
- In an RSA based system, the time it takes to compute $y^d \pmod N$ can expose d
- Countermeasures are based on masking the times taken by operations for example by adding random operation to a loop or random delays

Implementation attacks

Power attack:

- It is developed late 1990's by Kocher et al..
- It exploits the variation of power consumption of a cipher hardware under different input values to infer the key. For example by analyzing the power consumption of a smartcard while it is computing $c^d \pmod N$ one can expose d.
- Countermeasures include masking the actual power consumption of the circuit during computation through the use of redundant operations. Other methods are based on the use of power balanced logic which consume the same amount of power regardless of inputs.

Exercise

In a public-key system using RSA, you intercept the ciphertext y = 9 sent to a user whose public key is e = 5, n = 35. What is the plaintext x?

Answer:

x=4

Exercise

In a public-key system using RSA, you intercept the ciphertext $y = 30$ sent to a user whose public key is $e = 13$, $n = 77$. What is the plaintext x?

Answer: x=2