Asymmetric Encryption

By Prof. Hesham Abusaimeh

MEU

Chapter 9 – Public Key Cryptography and RSA

Every Egyptian received two names, which were known respectively as the true name and the good name, or the great name and the little name; and while the good or little name was made public, the true or great name appears to have been carefully concealed.

—*The Golden Bough*, Sir James George Frazer

Private-Key Cryptography

- traditional **private/secret/single key** cryptography uses **one** key
- shared by both sender and receiver
- if this key is disclosed communications are compromised
- also is **symmetric**, parties are equal
- hence does not protect sender from receiver forging a message & claiming is sent by sender

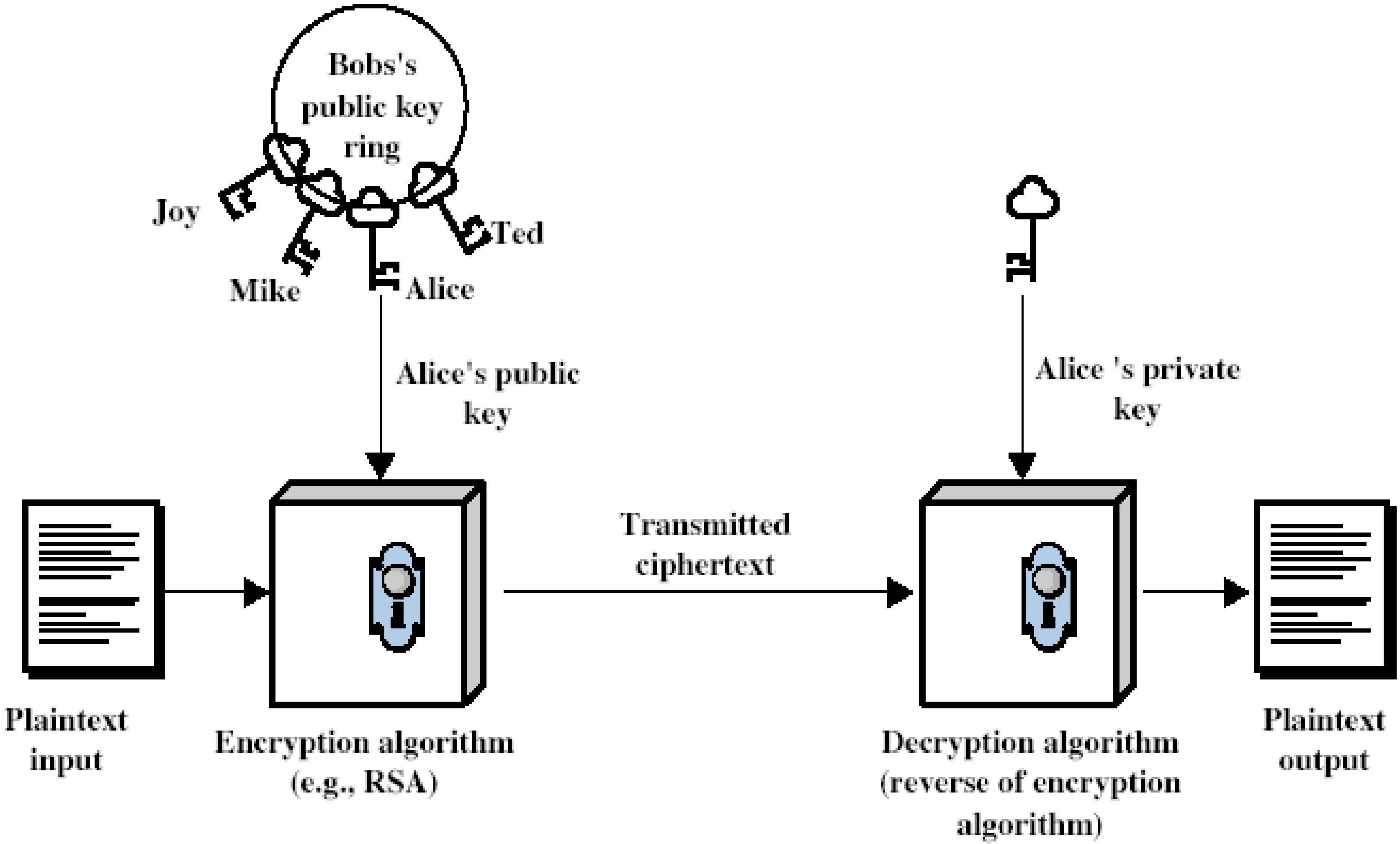
Public-Key Cryptography

- probably most significant advance in the 3000 year history of cryptography
- uses **two** keys – a public & a private key
- **asymmetric** since parties are **not** equal
- uses clever application of number theoretic concepts to function
- complements **rather than** replaces private key crypto

Public-Key Cryptography

- **public-key/two-key/asymmetric** cryptography involves the use of **two** keys:
  - a **public-key**, which may be known by anybody, and can be used to **encrypt messages**, and **verify signatures**
  - a **private-key**, known only to the recipient, used to **decrypt messages**, and **sign** (create) **signatures**
- is **asymmetric** because
  - those who encrypt messages or verify signatures **cannot** decrypt messages or create signatures

Public-Key Cryptography



Why Public-Key Cryptography?

- developed to address two key issues:
  - **key distribution** – how to have secure communications in general without having to trust a KDC with your key
  - **digital signatures** – how to verify a message comes intact from the claimed sender
- public invention due to Whitfield Diffie & Martin Hellman at Stanford Uni in 1976
  - known earlier in classified community

Public-Key Characteristics

- Public-Key algorithms rely on two keys with the characteristics that it is:
  - computationally infeasible to find decryption key knowing only algorithm & encryption key
  - computationally easy to en/decrypt messages when the relevant (en/decrypt) key is known
  - either of the two related keys can be used for encryption, with the other used for decryption (in some schemes)

Public-Key Cryptosystems

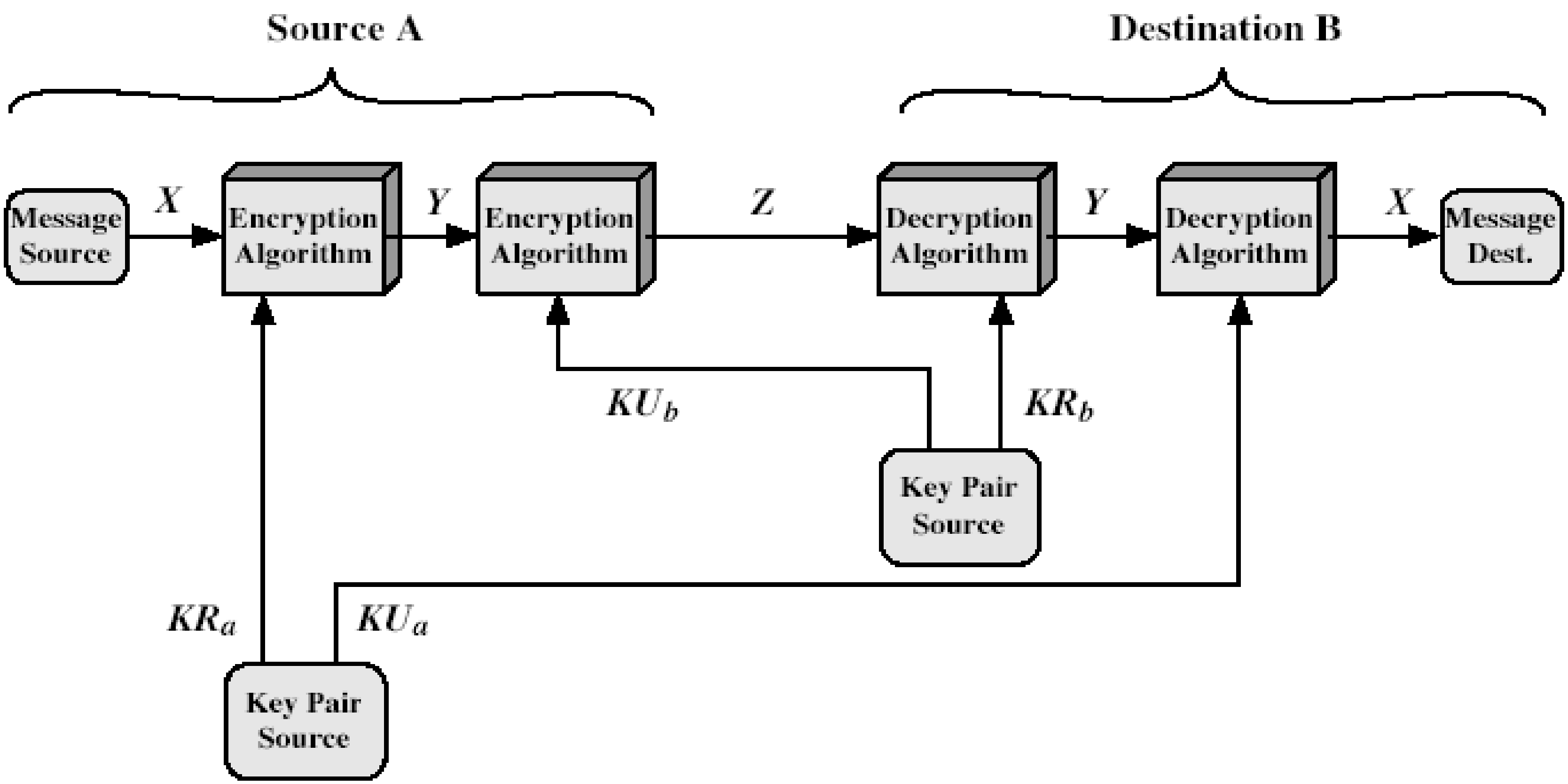


Figure 9.4 Public-Key Cryptosystem: Secrecy and Authentication

Public-Key Applications

- can classify uses into 3 categories:
  - **encryption/decryption** (provide secrecy)
  - **digital signatures** (provide authentication)
  - **key exchange** (of session keys)
- some algorithms are suitable for all uses, others are specific to one

Security of Public Key Schemes

- like private key schemes brute force **exhaustive search** attack is always theoretically possible
- but keys used are too large (>512bits)
- security relies on a **large enough** difference in difficulty between **easy** (en/decrypt) and **hard** (cryptanalyse) problems
- more generally the **hard** problem is known, its just made too hard to do in practise
- requires the use of **very large numbers**
- hence is **slow** compared to private key schemes

RSA

- by Rivest, Shamir & Adleman of MIT in 1977
- best known & widely used public-key scheme
- based on exponentiation in a finite (Galois) field over integers modulo a prime
  - nb. exponentiation takes $O((\log n)^3)$ operations (easy)
- uses large integers (eg. 1024 bits)
- security due to cost of factoring large numbers
  - nb. factorization takes $O(e^{\log n \log \log n})$ operations (hard)

RSA Key Setup

- each user generates a public/private key pair by:
- selecting two large primes at random - p, q
- computing their system modulus $N=p.q$
  - note $ø(N)=(p-1)(q-1)$
- selecting at random the encryption key e
  - where $1<e<ø(N)$, $\gcd(e,ø(N))=1$
- solve following equation to find decryption key d
  - $e.d=1 \bmod ø(N)$ and $0\leq d\leq N$

  (what is the number if multiplied by e the product will divide ø(N) and remainder=1)
- publish their public encryption key: $KU=\{e,N\}$
- keep secret private decryption key: $KR=\{d,p,q\}$

RSA Use

- to encrypt a message M the sender:
  - obtains **public key** of recipient $KU=\{e,N\}$
  - computes: $C=M^e \bmod N$, where $0 \leq M < N$
- to decrypt the ciphertext C the owner:
  - uses their private key $KR=\{d,p,q\}$
  - computes: $M=C^d \bmod N$
- note that the message M must be smaller than the modulus N (block if needed)

gcd (e,Φ)=1

- gcd (e,Φ)=(47,152)
- 152 mod 47 = 11
- 47 mod 11 = 3
- 11 mod 3 =2
- 3 mod 2 =1
- gcd(47,152)=1, or
- gcd(152,47)= 152/47=47/11=11/3=3/2=1

Φ?

- Φ called Euler function
- suppose that Z_n is a ring of +ve integers and e ε Z_n has a multiplicative Inverse e^{-1}
- If gcd(e,n)=1 then e has an inverse e^{-1}
- Φ (n); +ve integer number $1<n<\Phi$
- Examples:
  - gcd(25,5)=0 is not ok
  - gcd(25,2)=1 is ok
  - gcd(25,3)=1 is ok

Gcd $(e,\Phi)=1$

- gcd $(e,\Phi)=(47,152)$
- 152 mod 47 = 11
- 47 mod 11 = 3
- 11 mod 3 =2
- 3 mod 2 =1
- gcd(47,152)=1, or
- gcd(152,47)= 152/47=47/11=11/3=3/2=1

Algorithm to calculate Φ?

1. If n is prime then $\Phi(n)=n-1$
2. If $n = p*q$; p and q are primes then
   1. $\Phi(n)= (p-1)*(q-1)$
3. If $n = p*q$; p and q are primes and $p=q$ then

   $\Phi(n)= (p-1)*(q)$

It is recommended to use same length of p and q to make it hard to analyse ($p=107$, $q=101$);the 2nd form is more common

Example $p=5$, $q=7$ both primes; $\Phi(n)= (p-1)*(q-1) = 4*6=24$

Algorithm

1. Choose two primes p and q
2. Compute the Modulo(n); $n=p*q$
3. compute $\Phi(n)= (p-1)*(q-1)$
4. Choose the public key (e) such that:
    1. $1<e<\Phi$
    2. $\gcd(e, \Phi)=1$
5. Compute the Inverse $e^{-1} \bmod \Phi$ as follows:
    1. $Ed \equiv 1 \bmod \Phi$

Example;

Let P=5,q=7

Compute the Modulo(n); $n=p*q$; $5*7=35$

compute $\Phi(n)= (p-1)*(q-1)$; $4*6=24$

Choose the public key (e) such that:

$1<e<\Phi$; let e=5 where $1<5<24$

$\gcd(e, \Phi)=1$ $\gcd(5,24)=1$

Compute the Inverse $e^{-1} \bmod \Phi$ as follows:

$ed \equiv 1 \bmod \Phi$; $5*d \equiv 1 \bmod 24 \rightarrow d=5$ (what is the number if multiplied by 5 the product will divide 24 and remainder=1)

Why RSA Works

- because of Euler's Theorem:
- $a^{\varnothing(n)} \bmod N = 1$
  - where $\gcd(a,N)=1$
- in RSA have:
  - $N=p.q$
  - $\varnothing(N)=(p-1)(q-1)$
  - carefully chosen e & d to be inverses $\bmod \varnothing(N)$
  - hence $e.d=1+k.\varnothing(N)$ for some k
- hence :
  $C^d = (M^e)^d = M^{1+k.\varnothing(N)} = M^1.(M^{\varnothing(N)})^q = M^1.(1)^q = M^1 = M \bmod N$

RSA Example

1. Select primes: p=17 & q=11
2. Compute $n = pq$ =17×11=187
3. Compute $\varnothing(n)=(p-1)(q-1)=16\times10=160$
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 KU={7,187}
7. Keep secret private key KR={23,17,11}

RSA Example cont

- sample RSA encryption/decryption is:
- given message $M = 88$ (nb. $88<187$)
- encryption:

  $C = 88^7 \bmod 187 = 11$
- decryption:

  $M = 11^{23} \bmod 187 = 88$

Exponentiation

- can use the Square and Multiply Algorithm
- a fast, efficient algorithm for exponentiation
- concept is based on repeatedly squaring base
- and multiplying in the ones that are needed to compute the result
- look at binary representation of exponent
- only takes $O(\log_2 n)$ multiples for number n
  - eg. $7^5 = 7^4.7^1 = 3.7 = 10 \bmod 11$
  - eg. $3^{129} = 3^{128}.3^1 = 5.3 = 4 \bmod 11$

Exponentiation

```
c ← 0; d ← 1
for i ← k downto 0
    do  c ← 2 × c
        d ← (d × d) mod n
        if  b_i = 1
            then  c ← c + 1
                  d ← (d × a) mod n
return d
```

RSA Key Generation

- users of RSA must:
  - determine two primes at random - `p, q`
  - select either `e` or `d` and compute the other
- primes `p,q` must not be easily derived from modulus `N=p.q`
  - means must be sufficiently large
  - typically guess and use probabilistic test
- exponents `e, d` are inverses, so use Inverse algorithm to compute the other

RSA Security

- three approaches to attacking RSA:
  - brute force key search (infeasible given size of numbers)
  - mathematical attacks (based on difficulty of computing ø(N), by factoring modulus N)
  - timing attacks (on running of decryption)

Factoring Problem

  - Trial Division:
    - N=667=23*29; √667≈25 the closes prime is 23, so 667/23=69; 23*29=667
    - N=1403=23*61;√1403≈37 ; 1403/37=61, so 1403=23*61
- mathematical approach takes 3 forms:
  - factor $N=p.q$, hence find $\varnothing(N)$ and then d
  - determine $\varnothing(N)$ directly and find d
  - find d directly
- currently believe all equivalent to factoring
  - have seen slow improvements over the years
    - as of Aug-99 best is 130 decimal digits (512) bit with GNFS
  - biggest improvement comes from improved algorithm
    - cf “Quadratic Sieve” to “Generalized Number Field Sieve”
  - barring dramatic breakthrough 1024+ bit RSA secure
    - ensure p, q of similar size and matching other constraints

Timing Attacks

- developed in mid-1990’s
- exploit timing variations in operations
  - eg. multiplying by small vs large number
  - or IF's varying which instructions executed
- infer operand size based on time taken
- RSA exploits time taken in exponentiation
- countermeasures
  - use constant exponentiation time
  - add random delays
  - blind values used in calculations

Summary

- have considered:
  - principles of public-key cryptography
  - RSA algorithm, implementation, security