

Cryptography and Network Security

MODULE-2

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

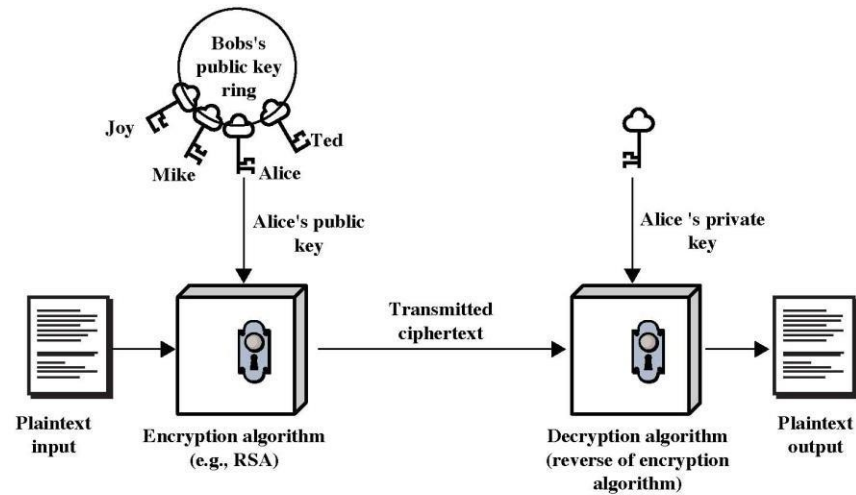
- probably most significant advance in the 3000 year history of cryptography
- uses **two** keys – a public & a private key
 - Anyone knowing the public key can encrypt messages or verify signatures
 - **But cannot** decrypt messages or create signatures
- **asymmetric** since parties are **not** equal
- complements **rather than** replaces private key

crypto

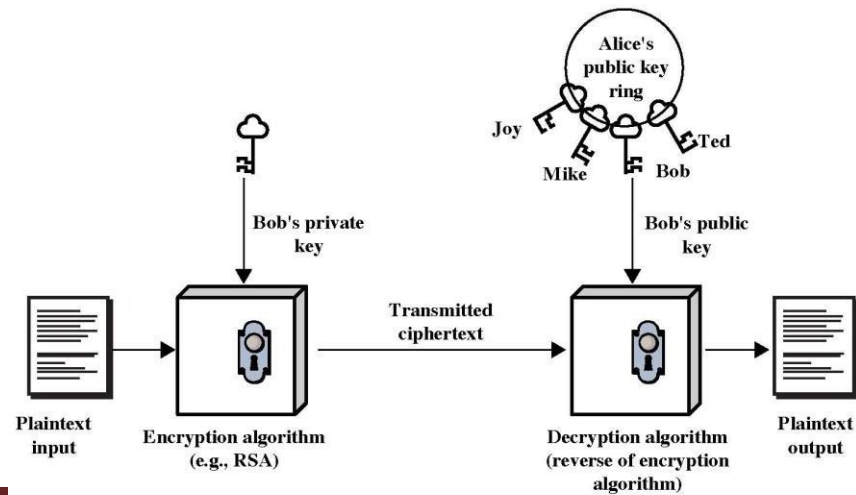
Public-Key Cryptography

- **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



(a) Encryption



(b) Authentication

Figure 9.1 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
 - No need for secure key delivery
 - No one else needs to know your private key
 - **digital signatures** – how to verify a message comes intact from the claimed sender

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
 - Oneway-ness is desirable: exp/log, mul/fac
 - either of the two related keys can be used for

encryption, with the other used for decryption
(in some schemes)

Public-Key Cryptosystems: Secrecy

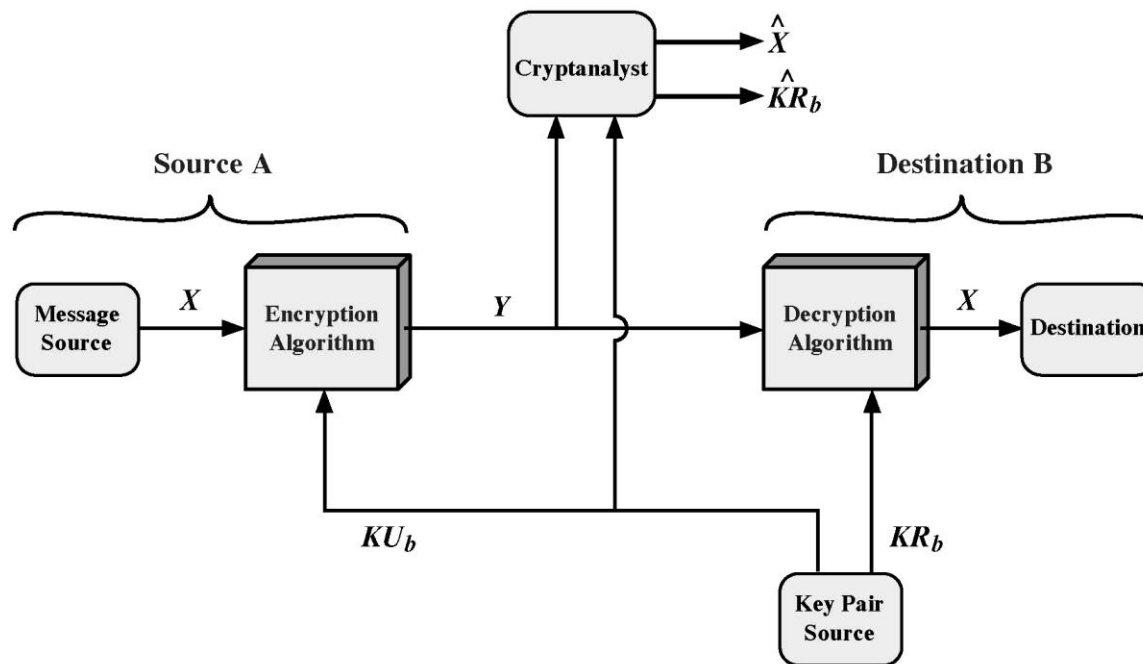


Figure 9.2 Public-Key Cryptosystem: Secrecy

Public-Key Cryptosystems: Authentication

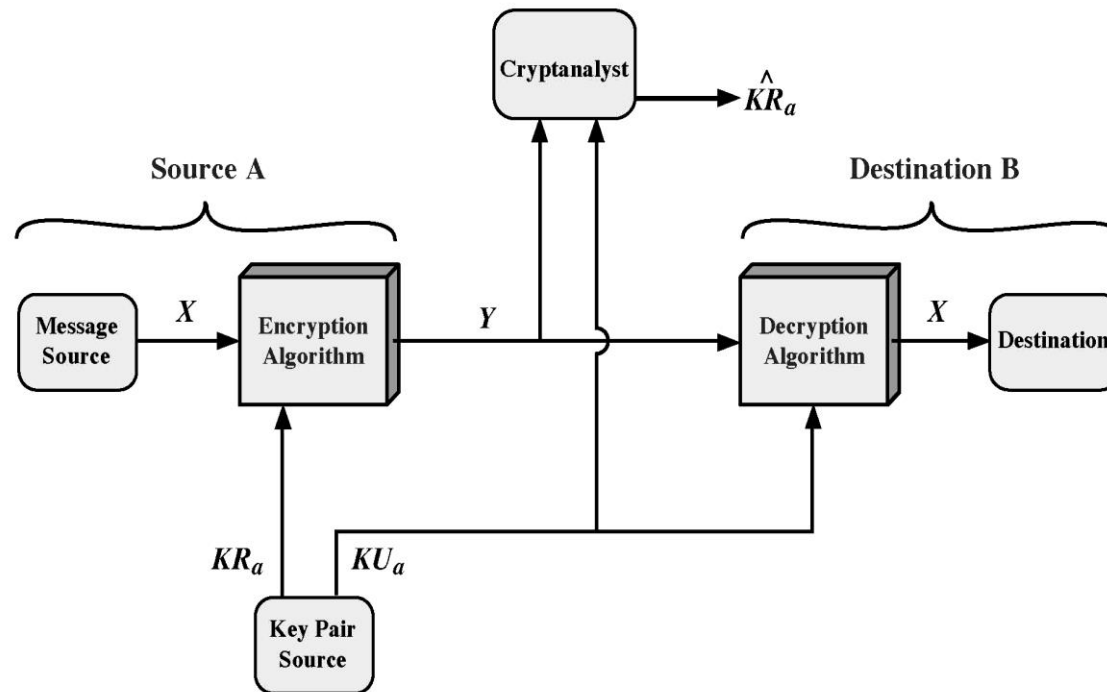


Figure 9.3 Public-Key Cryptosystem: Authentication

Public-Key Cryptosystems: Secrecy and Authentication

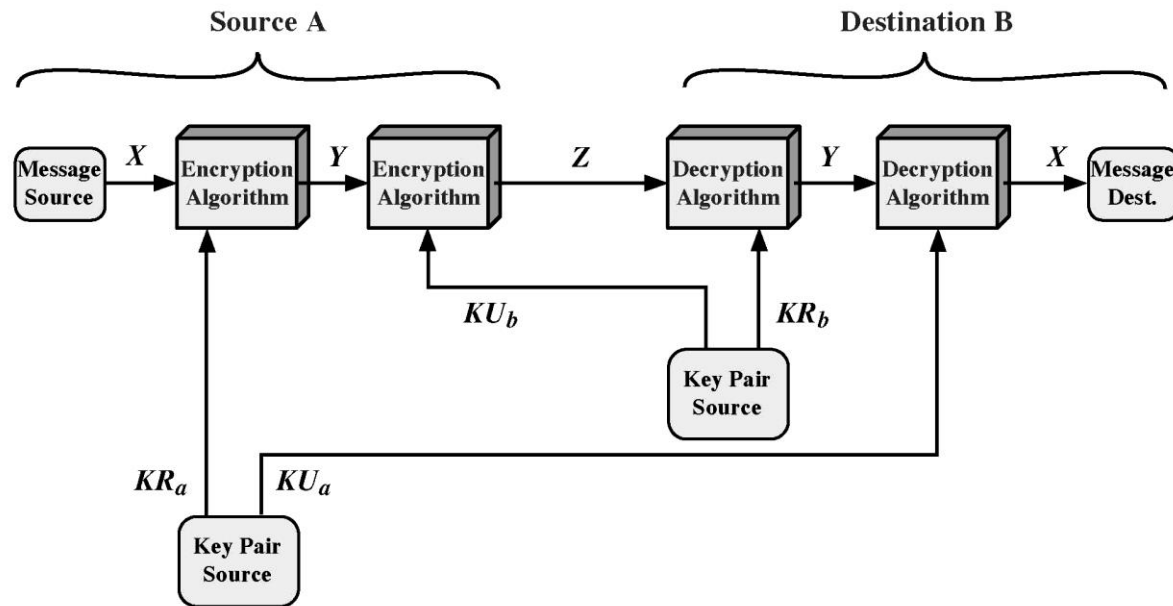


Figure 9.4 Public-Key Cryptosystem: Secrecy and Authentication

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
- 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 of integers in a finite (Galois) field
 - Defined over integers modulo a prime
 - exponentiation takes $O((\log n)^3)$ operations (easy)
- uses large integers (eg. 1024 bits)

- security due to cost of factoring large numbers
 - factorization takes $O(e^{\log n \log \log n})$ operations (hard)

RSA Key Setup

- each user generates a public/private key pair by:
 1. selecting two large primes at random - p, q (secret)
 2. computing their system modulus $N=p \cdot q$ (public)
 - note $\phi(N) = (p-1)(q-1)$ (secret)
 3. selecting at random the encryption key e (public)
 - where $1 < e < \phi(N)$, $\gcd(e, \phi(N)) = 1$
 4. solve following equation to find decryption key d (secret)
 - $e \cdot d = 1 \pmod{\phi(N)}$ and $0 \leq d \leq N$
 - Use the extended Euclid's algorithm to find the multiplicative inverse of $e \pmod{\phi(N)}$

- publish their public encryption key: $KU=\{e,N\}$
- keep secret private decryption key: $KR=\{d,p,q\}$

Block size of RSA

- Each block is represented as an integer number
- Each block has a value M less than N
- The block size is $\leq \log_2(N)$ bits
- If the block size is k bits then

$$2^k \leq N \leq 2^{K+1}$$

RSA Use

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

than the modulus N (block if needed)

Why RSA Works

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

- 1. $\gcd(M, N) = 1$
- 2. $\gcd(M, N) > 1$, see equation (8.6) in P.243

RSA Example

1. Select primes: $p=17$ & $q=11$
2. Compute $n = pq = 17 \times 11 = 187$
3. Compute $\phi(n) = (p-1)(q-1) = 16 \times 10 = 160$
4. Select e : $\gcd(e, 160) = 1$; **choose** $e=7$
5. Determine d : $de=1 \pmod{160}$ **and** $d < 160$
Value is $d=23$ **since** $23 \times 7 = 161 = 10 \times 160 + 1$
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 $M = 88$ ($88 < 187$)

- 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

Exponentiation

$c \leftarrow 0; d \leftarrow 1$

for $i \leftarrow k$ **downto** 0

do $c \leftarrow 2 \times c$

$d \leftarrow (d \times d) \bmod n$

if $b_i = 1$

then $c \leftarrow c + 1$

$d \leftarrow (d \times a) \bmod n$

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 $N=p \cdot q$
 - 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 $\phi(N)$, by factoring modulus N)
 - timing attacks (on running of decryption)

Factoring Problem

- mathematical approach takes 3 forms:
 - factor $N = p \cdot q$, hence find $\phi(N)$ and then d
 - determine $\phi(N)$ 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
 - infer bits of d based on time taken
- countermeasures
 - use constant exponentiation time
 - add random delays
 - blind values used in calculations

- $C' = (Mr)^e$, $M' = (C')^d$, $M = M'r^{-1}$

Summary

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