

Public Key Cryptography and the RSA Algorithm

Cryptography and Network Security
by William Stallings

Lecture slides by Lawrie Brown

Edited by Dick Steflik

Private-Key Cryptography

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

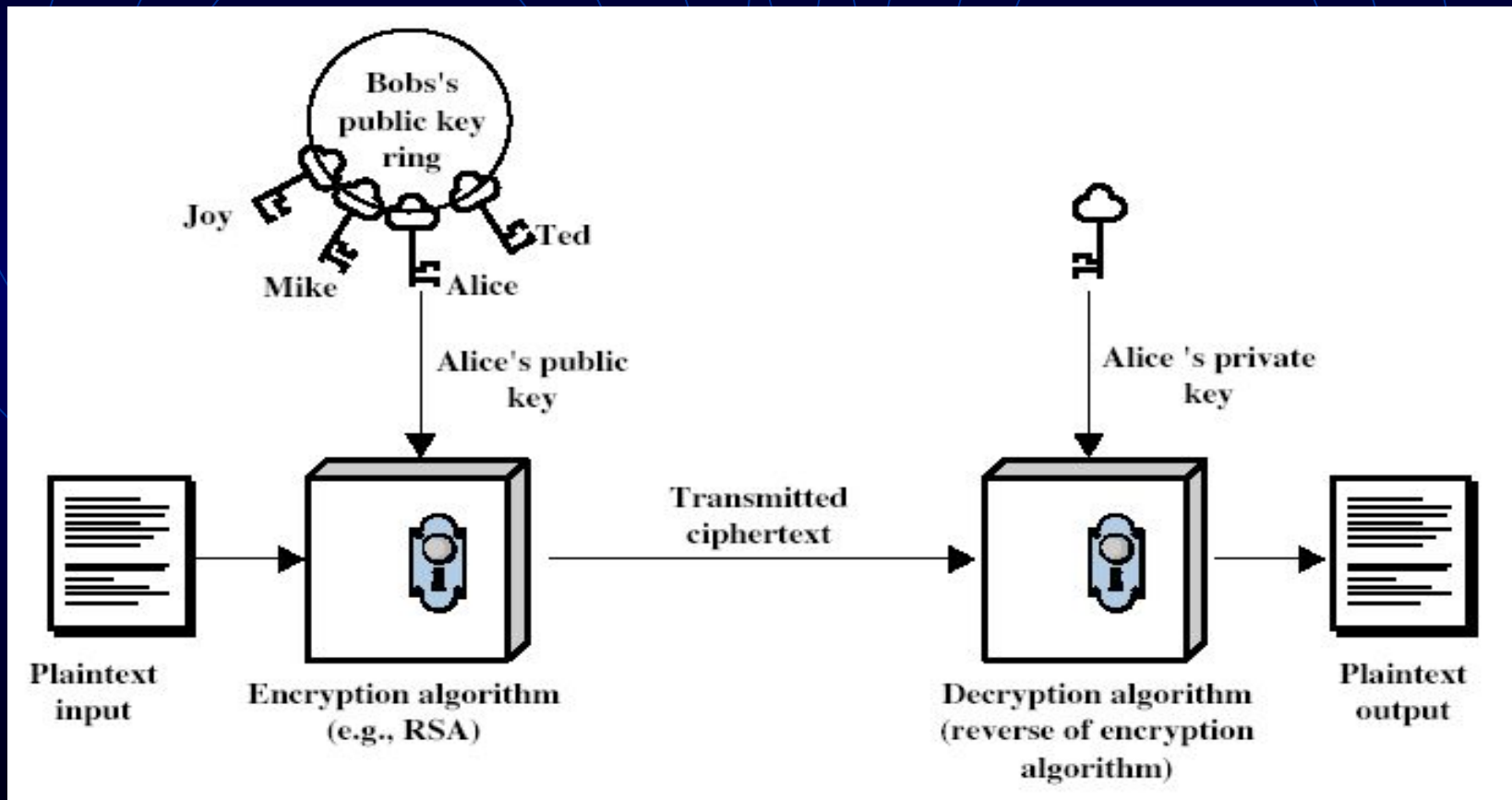
Public-Key Cryptography

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

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



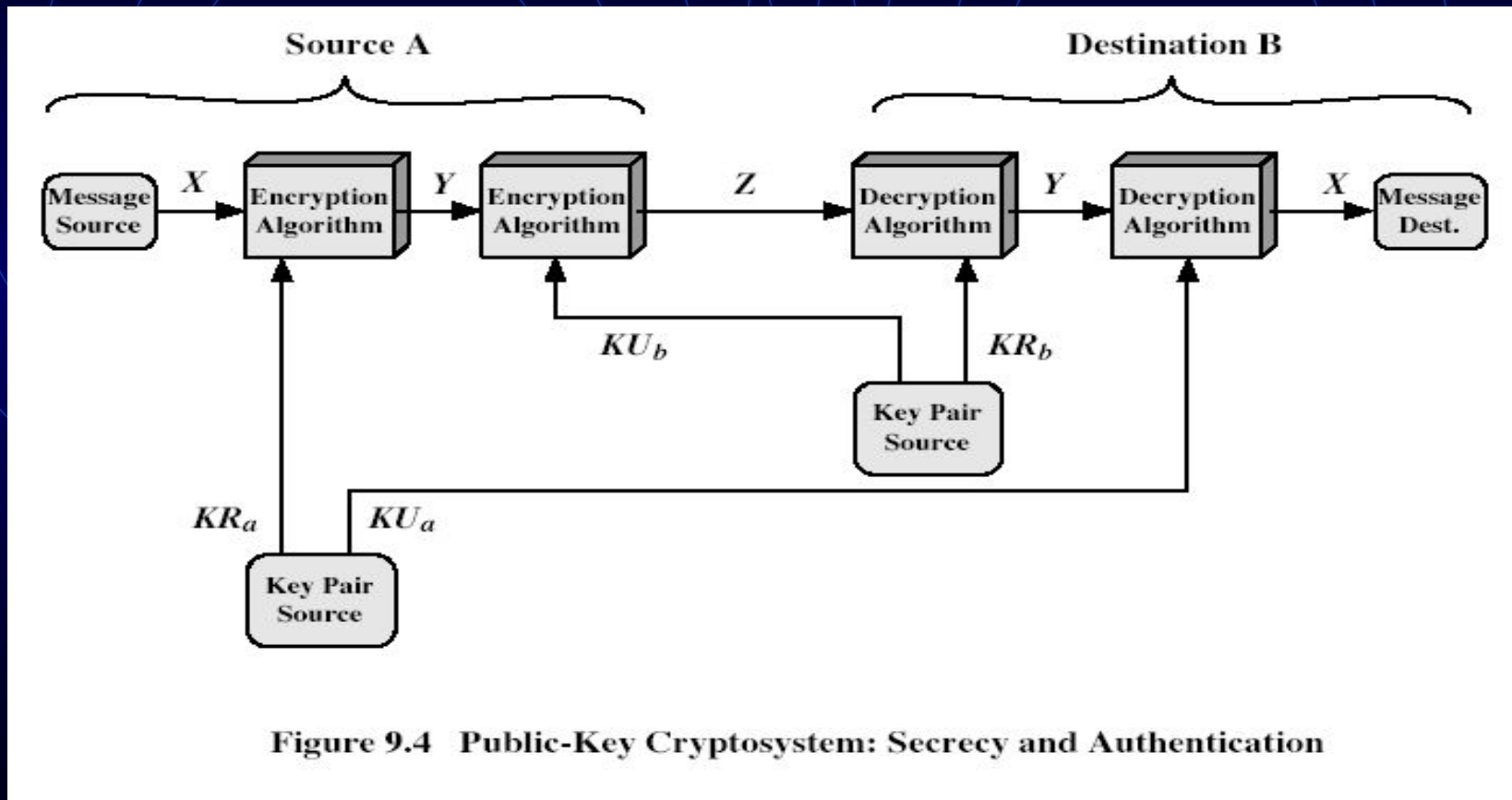
Why Public-Key Cryptography?

- 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 U. 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



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 (>512 bits)
- 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 $N=p \cdot q$
 - note $\phi(N) = (p-1)(q-1)$
- selecting at random the encryption key e
 - where $1 < e < \phi(N)$, $\gcd(e, \phi(N)) = 1$
- solve following equation to find decryption key d
 - $e \cdot d = 1 \pmod{\phi(N)}$ and $0 \leq d \leq N$
- 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)

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 mod $\phi(N)$
- hence $e \cdot d = 1 + k \cdot \phi(N)$ for some k

- hence :

$$\begin{aligned} C^d &= (M^e)^d = M^{1+k \cdot \phi(N)} = M^1 \cdot (M^{\phi(N)})^k = M^1 \cdot (1)^k \\ &= M^1 = M \bmod N \end{aligned}$$

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 $KR = \{23, 17, 11\}$

Multiplicative inverse

- $160 = 7 * 22 + 6$
- $7 = 6 * 1 + 1$
- $1 = 7 * 1 - 6 * 1$
- $= 7 * 1 - 160 * 1 + 7 * 22$
- $= 7 * 23 - 160 * 1$

RSA Example cont

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

$$C = 88^7 \bmod 187 = 11$$

$$88^2 \bmod 187 = 7744 \bmod 187 = 77$$

$$88^4 \bmod 187 = (88^2)^2 \bmod 187 = 77^2 = 5929 - 5797 = 132$$

$$88^6 \bmod 187 = 88^4 * 88^2 = 132 * 77 \bmod 187 = (10164 - 10098) \bmod 187 = 66$$

- $88^7 \bmod 187 = 88^6 * 88^1 \bmod 187$
- $= 66 * 88 \bmod 187$
- $= 5808 - 5797$
- Cipher Text = 11

- decryption:

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

decryption:

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

- $C = M^e \bmod N$
- $M = C^d \bmod N$
- $M = 11^{23} \bmod 187$
- $11^4 \bmod 187 = 55$
- $11^8 \bmod 187 = 33$
- $11^{16} \bmod 187 = 33 * 33 \bmod 187$
 $= 1089 - 935 \bmod 187 = 154$
- $11^{23} \bmod 187 = 88$

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. $7^5 = 7^4 \cdot 7^1 = 3 \cdot 7 = 10 \pmod{11}$
 - eg. $3^{129} = 3^{128} \cdot 3^1 = 5 \cdot 3 = 4 \pmod{11}$

Exponentiation

```
c ← 0; d ← 1
for i ← k downto 0
  do c ← 2 × c
      d ← (d × d) mod n
      if bi = 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 $N=p \cdot q$
 - 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
 - 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