1.3 PUBLIC-KEY CRYPTOGRAPHY

ASYMMETRIC KEY CIPHERS

Difficulties in Symmetric encryption

According to Diffie-Hellman

i) Key distribution is a serious issue.

ii) Symmetric encryption is not applicable for Digital signatures

Public key encryption scheme:

Asymmetric algorithms rely on one key for encryption and a different but **related key** for decryption. Characteristics

i) It is computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption key.

ii) Either of the two related keys can be used for encryption, with the other used for decryption.

Conventional / symmetric / private	Asymmetric / public key encryption
The same algorithm with the same key is used for encryption and decryption.	One algorithm is used for encryption and decryption with a pair of keys, one for
The sender and receiver must share the algorithm and the key.	The sender and receiver must each have one of the matched pair of keys
The key must be kept secret.	One of the two keys must be kept secret.

Applications for Public-Key Cryptosystems

We can classify the use of public-key cryptosystems into three categories

1. Encryption /decryption: The sender encrypts a message with the recipient‘s public key.

2. Digital signature: The sender ―signs" a message with its private key. Signing is achieved by a cryptographic algorithm applied to the message or to a small block of data that is a function of the message.

3. Key exchange: Two sides cooperate to exchange a session key. Several different approaches are possible, involving the private key(s) of one or both parties.

RSA – Algorithm

- It was developed by Rivest, Shamir and Adleman. This algorithm makes use of an expression with exponentials. Plaintext is encrypted in blocks, with each block having a binary value less than some number n.
- The RSA Algorithm: It is a public key cryptography algorithm. RSA can be used for key exchange, digital signatures and the encryption of small blocks of data.
- The RSA scheme is a cipher in which the plaintext and cipher text are integers between 0 and n - 1 for some n. A typical size for n is 1024 bits, or 309 decimal digits. That is, n is less than 2^{1024}

Encryption and decryption are of the following form, for some plaintext block M and cipher text block C:

$$\mathbf{C = M^e \bmod n}$$

$$\mathbf{M = C^d \bmod n}$$

Both the sender and receiver know the value of n. the sender knows the value of e and only the receiver knows the value of d. thus, this is a public key encryption algorithm with a public key of KU = {e, n) and a private key of KR = {d, n}.

For this algorithm to be satisfactory for public key encryption, the following requirements must be met:

1. It is possible to find values of e, d, n such that Med = M mod n for all M < n.
2. It is relatively easy to calculate Me and Cd for all values of M < n.
3. It is infeasible to determine d given e and n.

- It is the best known & widely used public-key scheme and based on exponentiation in a finite (Galois) field over integers modulo a prime.
- Security due to cost of factoring large numbers
- Plaintext and cipher text are integers between 0 and n – 1 for some n. (eg . 1024 bits) Ingredients of RSA Algorithm

The ingredients are the following:

p, q, two prime numbers	(private, chosen)
$n = pq$	(public, calculated)
e, with gcd(ϕ (n),e) = 1; $1 < e < \phi(n)$	(public, chosen)
$d \equiv e^{-1} (\mathrm{mod}\ \phi\ (n))$	(private, calculated)

RSA Key Setup

- This key setup is done once (rarely) when a user establishes (or replaces) their public key.
- Each user generates a public/private key pair by:
- Selecting two large primes at random - p, q
- Computing their system modulus N=p .q
    - 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 mod ø(N) and 0 ≤ d ≤ N

$$\boxed{ed = 1 \bmod \text{Ø}(n)}$$

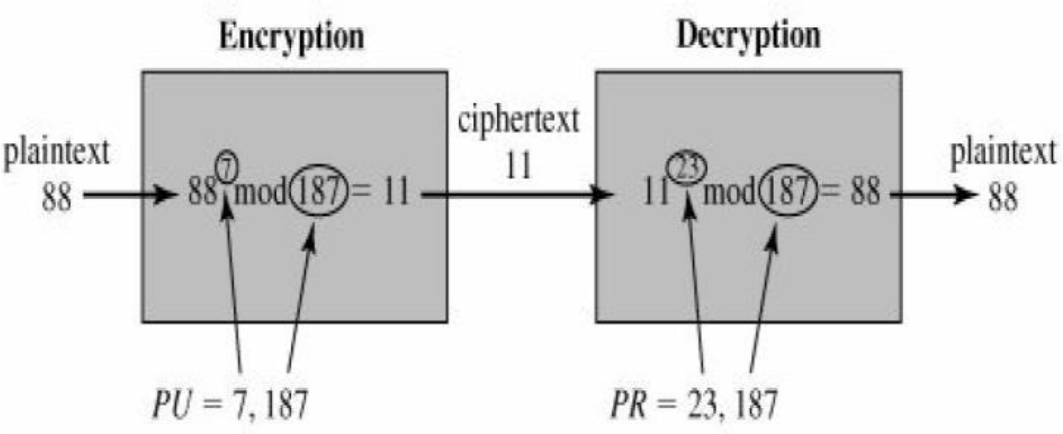
OR

$$\boxed{d = \frac{1 + k\,\text{Ø}(n)}{e}}$$

- 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 **$C=M^e$ mod N**
- To decrypt the ciphertext C the owner:
  - uses their private key KR={d,p,q}
  - computes: **$M=C^d$ mod N**
- note that the message M must be smaller than the modulus N (block if needed)

RSA Example

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
6. Publish public key KU={7,187}
7. Keep secret private key KR={23,17,11}
8. Given message M = 88 (88<187)
9. Encryption: C = 88^7 mod 187 = 11



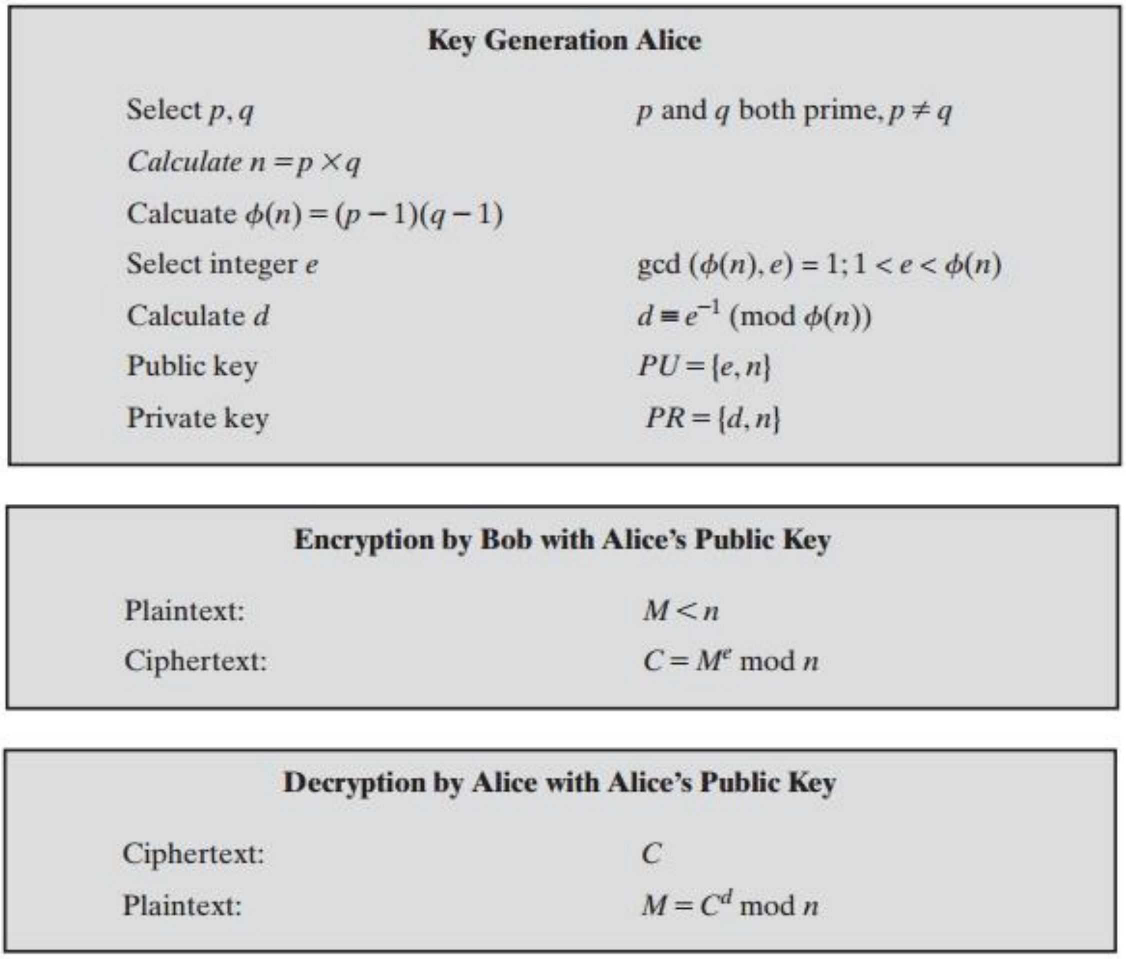
88^7 mod 187 = [(88^4 mod 187) x (88^2 mod 187) x (88^1 mod 187)] mod 187

88^1 mod 187 = 8888^2 mod 187 = 7744 mod 187 = 77

88^4 mod 187 = 59969536 mod 187 = 132

88^7 mod 187 = (88 x 77 x 132) mod 187= 894432 mod 187 = 11

10. Decryption: M = 11^{23} mod 187 = 88



Key Generation Alice

Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calcuate $\phi(n) = (p-1)(q-1)$	
Select integer e	$\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \pmod{\phi(n)}$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$

Encryption by Bob with Alice's Public Key

Plaintext:	$M < n$
Ciphertext:	$C = M^e \bmod n$

Decryption by Alice with Alice's Public Key

Ciphertext:	C
Plaintext:	$M = C^d \bmod n$

Figure 9.5 The RSA Algorithm

The security of RSA:

Five possible approaches to attacking the RSA algorithm are

■ Brute force: This involves trying all possible private keys.

■ Mathematical attacks: There are several approaches, all equivalent in effort to factoring the product of two primes.

■ Timing attacks: These depend on the running time of the decryption algorithm.

■ Hardware fault-based attack: This involves inducing hardware faults in the processor that is generating digital signatures.

■ Chosen ciphertext attacks: This type of attack exploits properties of the RSA algorithm.

Note : Finding private key d (ie) multiplicative inverse of e^{-1} using extended Euclidean algorithm) ie) $d \equiv e^{-1} \bmod \phi(n)$

$d * e \equiv 1 \bmod \phi(n)$ Here $d * 3 \equiv 1 \bmod 160$

According Extended Euclidean algorithm initial values

A1 = 1 A2 = 0 A3 = 160

B1 = 0 B2 = 1 B3 = 7

Find Q = └ A3/B3┘ (take lowest nearest integer)

Then A1 = B1 ; A2= B2; A3 = B3

B1 = A1+QB1 ; B2 = A2+QB2; B3 = A3-QB3

Q	A1	A2	A3	B1	B2	B3
	1	0	160	0	1	7
22	0	1	7	1	22	6
1	1	22	6	1	23	1

Since B3 = 1 ; Multiplicative inverse

B2 = 23 $d * 3 \equiv 1 \bmod 160$

$23 * 7 \equiv 1 \bmod 160$ d = 23

Security of RSA / RSA Attacks:

Four possible approaches to attacking the RSA algorithm are as follows:

RSA Attacks	Counter-measures
i)Brute force: This involves trying all possible private keys.	The defense against the brute-force approach is the same for RSA as for other cryptosystems, namely, use a large

ii)Mathematical attacks: There are several approaches, all equivalent in effort to factoring the product of two primes. **The Factoring Problem** Three approaches to attack RSA mathematically: • Factor *n* into its two prime factors. This enables calculation of $\phi(n) = (p-1) \text{ x } (q-1)$, which, in turn, enables determination of $d \equiv e^{-1} (\text{mod } \phi(n))$. □ Determine $\phi(n)$ directly, without first determining *p* and *q*. Again, this enables determination of $d \equiv e^{-1} (\text{mod } \phi(n))$. □ Determine *d* directly, without first determining $\phi(n)$.	To avoid values of *n* that may be factored more easily, the algorithm's inventors suggest the following constraints on *p* and *q*: □ *p* and *q* should differ in length by only a few digits. Thus, for a 1024-bit key (309 decimal digits), both *p* and *q* should be on the order of magnitude of 10^{75} to 10^{100}. □ Both (*p*- 1) and (*q*- 1) should contain a large prime factor. □ gcd (p- 1, q - 1) should be small.
iii)Timing attacks: • These depend on the running time of the decryption algorithm. A timing attack is somewhat analogous to a burglar guessing the combination of a safe by observing how long it takes for someone to turn the dial from number to number. • If modular exponentiation is accomplished bit by bit, with one modular multiplication performed at each iteration and an additional modular	i)**Constant exponentiation time**: Ensure that all exponentiations take the same amount of time before returning a result. ii)**Random delay:** Better performance could be achieved by adding a random delay to the exponentiation algorithm to confuse the timing attack. iii)**Blinding:** Multiply the cipher text by a random number before performing exponentiation. This process prevents the attacker from knowing what cipher text bits are being processed inside the computer and therefore prevents the bit-by-bit analysis essential to the timing

iv)Chosen cipher text attacks: This type of attack exploits properties of the RSA algorithm. The basic RSA algorithm is vulnerable to a chosen cipher text attack (CCA). CCA is defined as an attack in which adversary chooses a number of cipher texts and is then given the corresponding plaintexts, decrypted with the target's private key.	**Optimal asymmetric encryption padding** • Message M to be encrypted is padded. A set of optional parameters P is passed through a hash function H. • The output is then padded with zeros to get the desired length in the overall data block (DB). Next, a random seed is generated and passed through another hash function, called the mask generating function (MGF).