Other Public-Key Cryptosystems

Topics

- Diffie-Hellman key exchange
- Elgamal Cryptographic systems

Diffie-Hellman key exchange

§ First public-key type scheme proposed by Diffie & Hellman in 1976 for key distribution only.

§ Purpose of the algorithm is to enable two users to exchange a secret key securely that then can be used for subsequent encryption of messages.

§ The algorithm itself is limited to the exchange of the keys.

The Algorithm

§ There are two publicly known numbers: a **prime number q** and an **integer α** that is a primitive root of q.

§ Suppose the users A and B wish to create a shared key.

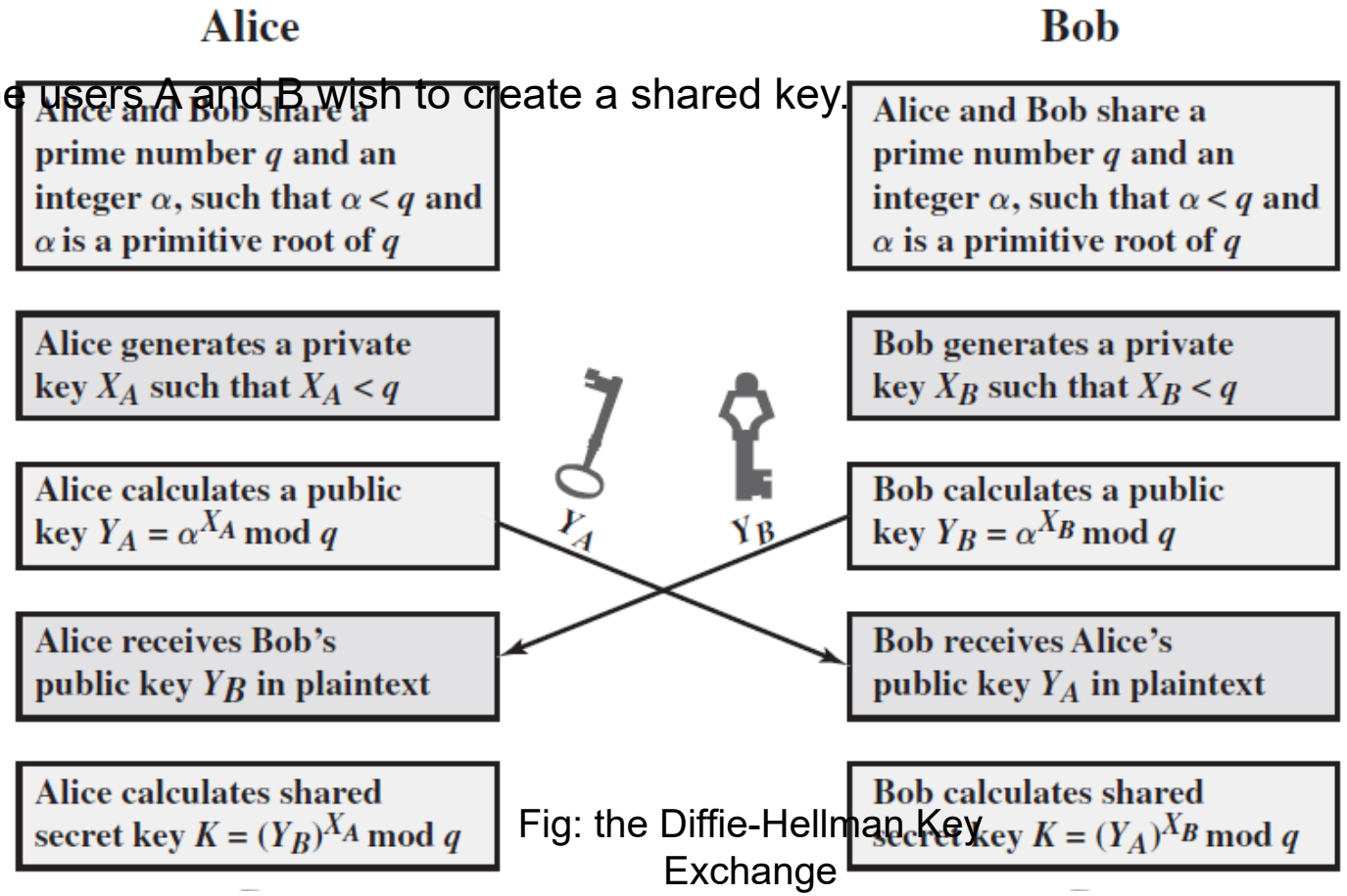


Fig: the Diffie-Hellman Key Exchange

Example:

- Lets take q = 353 and a primitive root of 353, α= 3.
- A and B select private keys $X_A = 97$ and $X_B = 233$, respectively.
- Each computes its public key:

  A computes $Y_A = 3^{97} \bmod 353 = 40$.

  B computes $Y_B = 3^{233} \bmod 353 = 248$.
- After they exchange public keys, each can compute the common secret key:

  A computes $K = (Y_B)^{XA} \bmod 353 = 248^{97} \bmod 353 = 160$.

  B computes $K = (Y_A)^{XB} \bmod 353 = 40^{233} \bmod 353 = 160$.
- An attacker would have available the following information:

  $q = 353$; $\alpha = 3$; $Y_A = 40$; $Y_B = 248$.

Man-in-the-Middle Attack

§ Suppose Alice and Bob wish to exchange keys, and Darth is the adversary.

§ The attack proceeds as follows

1. Darth prepares for the attack by generating two random private keys X_{D1} and X_{D2} and then computing the corresponding public keys Y_{D1} and Y_{D2}.
2. Alice transmits Y_A to Bob.
3. Darth intercepts Y_A and transmits Y_{D1} to Bob. Darth also calculates $K2 = (YA)^{X\,D2} \bmod q$.
4. Bob receives Y_{D1} and calculates $K1 = (Y_{D1})^{XB} \bmod q$.
5. Bob transmits Y_B to Alice.
6. Darth intercepts Y_B and transmits Y_{D2} to Alice. Darth calculates $K1 = (Y^B)^{XD1} \bmod q$.
7. Alice receives Y_{D2} and calculates $K2 = (Y_{D2})^{XA} \bmod q$.

- At this point, Bob and Alice think that they share a secret key, but instead Bob and Darth share secret key *K*1 and Alice and Darth share secret key *K*2.
- All future communication between Bob and Alice is compromised in the following way.
  1. Alice sends an encrypted message *M*: E(*K*2, *M*).
  2. Darth intercepts the encrypted message and decrypts it to recover *M*.
  3. Darth sends Bob *E*(*K*1, *M*) or E(*K*1, *M*′), where *M*′ is any message.
- In the first case, Darth simply wants to eavesdrop on the communication without altering it.
- In the second case, Darth wants to modify the message going to Bob.

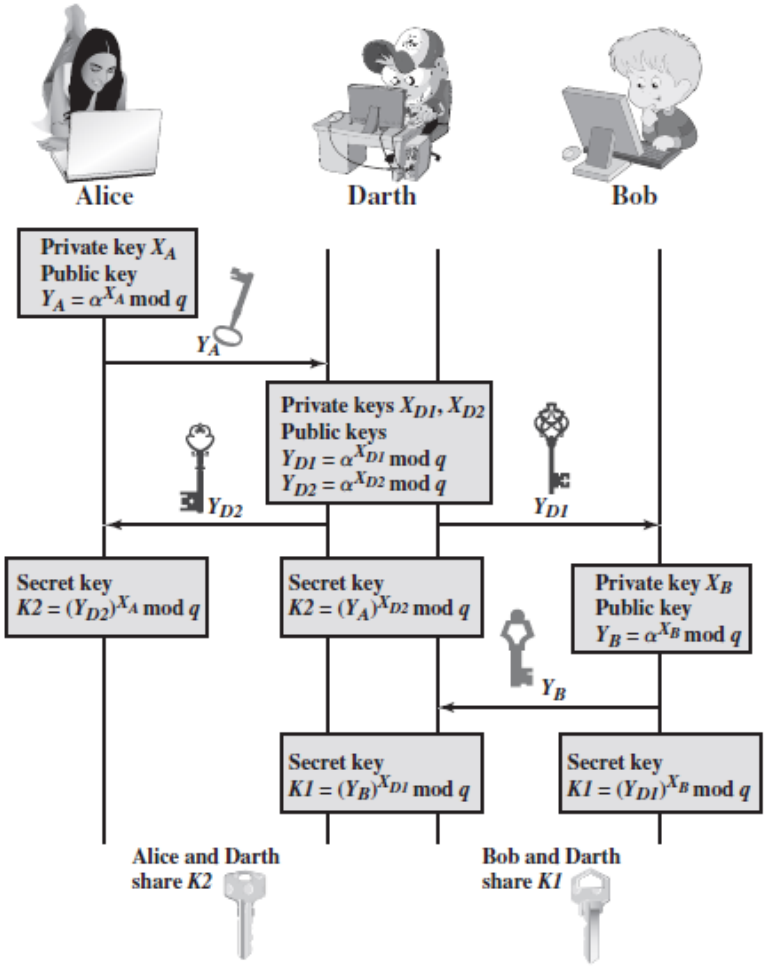


Fig: Man-in-the-Middle Attack

Problems on Diffie-Hellman algorithm

1. Users A and B use the Diffie-Hellman key exchange technique with a common prime q = 71 and a primitive root α = 7.

   a. If user A has private key X_A = 5, what is A's public key Y_A?

   b. If user B has private key X_B = 12, what is B's public key Y_B?

   c. What is the shared secret key?

2. Consider a Diffie-Hellman scheme with a common prime q = 11 and a primitive root α = 2.

   a. If user A has public key Y_A = 9, what is A's private key X_A?

   b. If user B has public key Y_B = 3, what is the secret key K shared with A?

Applications of Diffie-Hellman algorithm

- Secure Shell (SSH)
- Transport Layer Security (TLS) / Secure Sockets Layer (SSL)
- Public Key Infrastructure (PKI)
- Internet Key Exchange (IKE)
- Internet Protocol Security (IPSec)

Elgamal Cryptographic systems

- Presented in 1984 by Tather Elgamal
- Key aspects:
  - Based on the Discrete Logarithm problem
  - Randomized encryption

§ User A generates a private/public key pair as follows:

1. Generate a random integer X_A, such that $1 < X_A < q - 1$.
2. Compute $Y_A = \alpha^{XA} \bmod q$.
3. A's private key is X_A and A's public key is $\{q, \alpha, Y_A\}$.

§ Any user B that has access to A's public key can encrypt a message as follows:

1. Represent the message as an integer M in the range $0 \le M \le q - 1$.
2. Choose a random integer k such that $1 \le k \le q - 1$.
3. Compute a one-time key $K = (Y_A)^k \bmod q$.
4. Encrypt M as the pair of integers (C1, C2) where

$$C1 = \alpha^k \bmod q; \qquad C2 = KM \bmod q$$

§ User A recovers the plaintext as follows:

1. Recover the key by computing $K = (C_1)^{XA} \bmod q$.
2. Compute $M = (C_2 K^{-1}) \bmod q$.

$K = (Y_A)^k \bmod q$	K is defined during the encryption process
$K = (\alpha^{X_A} \bmod q)^k \bmod q$	substitute using $Y_A = \alpha^{X_A} \bmod q$
$K = \alpha^{kX_A} \bmod q$	by the rules of modular arithmetic
$K = (C_1)^{X_A} \bmod q$	substitute using $C_1 = \alpha^k \bmod q$

Next, using K, we recover the plaintext as

$$C_2 = KM \bmod q$$
$$(C_2 K^{-1}) \bmod q = KMK^{-1} \bmod q = M \bmod q = M$$

Global Public Elements	
q	prime number
α	$\alpha < q$ and α a primitive root of q

Key Generation by Alice	
Select private X_A	$X_A < q - 1$
Calculate Y_A	$Y_A = \alpha^{X_A} \bmod q$
Public key	$\{q, \alpha, Y_A\}$
Private key	X_A

Encryption by Bob with Alice's Public Key	
Plaintext:	$M < q$
Select random integer k	$k < q$
Calculate K	$K = (Y_A)^k \bmod q$
Calculate C_1	$C_1 = \alpha^k \bmod q$
Calculate C_2	$C_2 = KM \bmod q$
Ciphertext:	(C_1, C_2)

Decryption by Alice with Alice's Private Key	
Ciphertext:	(C_1, C_2)
Calculate K	$K = (C_1)^{X_A} \bmod q$
Plaintext:	$M = (C_2 K^{-1}) \bmod q$

§ We can restate the Elgamal process as follows:

1. Bob generates a random integer *k*.
2. Bob generates a one-time key *K* using Alice's public-key components Y_A, *q*, and *k*.
3. Bob encrypts *k* using the public-key component a, yielding *C*1. *C*1 provides sufficient information for Alice to recover *K*.
4. Bob encrypts the plaintext message *M* using *K*.
5. Alice recovers *K* from *C*1 using her private key.
6. Alice uses K^{-1} to recover the plaintext message from *C*2.

§ Thus, *K* functions as a one-time key, used to encrypt and decrypt the message.

- Example, let us consider q = 19, primitive roots {2, 3, 10, 13, 14, 15}. We choose a = 10.Alice generates a key pair as follows:

  1.Alice chooses X_A = 5.

  2.Then $Y_A = \alpha^{XA} \bmod q = \alpha^5 \bmod 19 = 3$

  3.Alice's private key is 5 and Alice's public key is $\{q, a, Y_A\} = \{19, 10, 3\}$.

- Suppose Bob wants to send the message with the value M = 17. Then:

  1.Bob chooses k = 6.

  2. Then $K = (Y_A)^k \bmod q = 3^6 \bmod 19 = 729 \bmod 19 = 7$.

  3. So

  $C1 = a^k \bmod q = \alpha^6 \bmod 19 = 11$

  $C2 = KM \bmod q = 7 * 17 \bmod 19 = 119 \bmod 19 = 5$

  4. Bob sends the ciphertext (11, 5).

- For decryption:
  1. Alice calculates $K = (C_1)^{XA} \bmod q = 11^5 \bmod 19 = 161051 \bmod 19 = 7$
  2. Then K^{-1} in GF(19) is $7^{-1} \bmod 19 = 11$
  3. Finally, $M = (C_2 K^{-1}) \bmod q = 5 * 11 \bmod 19 = 55 \bmod 19 = 17$
- If a message must be broken up into blocks and sent as a sequence of encrypted blocks, a unique value of k should be used for each block.
- If k is used for more than one block, knowledge of one block $M1$ of the message enables the user to compute other blocks as follows. Let

$$C_{1,1} = \alpha^k \bmod q; C_{2,1} = KM_1 \bmod q$$
$$C_{1,2} = \alpha^k \bmod q; C_{2,2} = KM_2 \bmod q$$

Then,

$$\frac{C_{2,1}}{C_{2,2}} = \frac{KM_1 \bmod q}{KM_2 \bmod q} = \frac{M_1 \bmod q}{M_2 \bmod q}$$

If M_1 is known, then M_2 is easily computed as

$$M_2 = (C_{2,1})^{-1} C_{2,2} M_1 \bmod q$$

§ Application:

- Establishing a secure channel for key sharing
- Encrypting messages
- ElGamal encryption is used in **the free GNU Privacy Guard software**, recent versions of PGP

Problems:

1. Consider an Elgamal scheme with a common prime $q = 71$ and a primitive root $a = 7$.

   a. If B has public key $YB = 3$ and A choose the random integer $k = 2$, what is the ciphertext of $M = 30$?

   b. If A now chooses a different value of k so that the encoding of $M = 30$ is $C = (59, C2)$, what is the integer $C2$?

Table 8.3 Powers of Integers, Modulo 19

a	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}	a^{15}	a^{16}	a^{17}	a^{18}
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	4	8	16	13	7	14	9	18	17	15	11	3	6	12	5	10	1
3	9	8	5	15	7	2	6	18	16	10	11	14	4	12	17	13	1
4	16	7	9	17	11	6	5	1	4	16	7	9	17	11	6	5	1
5	6	11	17	9	7	16	4	1	5	6	11	17	9	7	16	4	1
6	17	7	4	5	11	9	16	1	6	17	7	4	5	11	9	16	1
7	11	1	7	11	1	7	11	1	7	11	1	7	11	1	7	11	1
8	7	18	11	12	1	8	7	18	11	12	1	8	7	18	11	12	1
9	5	7	6	16	11	4	17	1	9	5	7	6	16	11	4	17	1
10	5	12	6	3	11	15	17	18	9	14	7	13	16	8	4	2	1
11	7	1	11	7	1	11	7	1	11	7	1	11	7	1	11	7	1
12	11	18	7	8	1	12	11	18	7	8	1	12	11	18	7	8	1
13	17	12	4	14	11	10	16	18	6	2	7	15	5	8	9	3	1
14	6	8	17	10	7	3	4	18	5	13	11	2	9	12	16	15	1
15	16	12	9	2	11	13	5	18	4	3	7	10	17	8	6	14	1
16	9	11	5	4	7	17	6	1	16	9	11	5	4	7	17	6	1
17	4	11	16	6	7	5	9	1	17	4	11	16	6	7	5	9	1
18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1

Module-2

Public-Key Cryptography and RSA

Topics Covered

Ø Principles of public-key cryptosystems

Ø Public-key cryptosystems

Ø Applications for public-key cryptosystems

Ø Requirements for public-key cryptosystems

Ø Public-key cryptanalysis.

Ø The RSA algorithm: description of the algorithm, computational aspects, the security of RSA.

Principles of Public-Key Cryptosystems

- Developed to address two key issues:

1. Key distribution – how to have secure communications in general without having to trust a KDC with your key

2. Digital signatures – how to verify a message comes intact from the claimed sender

Public-Key Cryptosystems

§ Asymmetric algorithms rely on one key for encryption and a different but related key for decryption. These algorithms have the following important characteristic.

- It is computationally infeasible to determine the decryption key given only knowledge of the

  cryptographic algorithm and the encryption key.

- In addition, some algorithms, such as RSA, also exhibit the following characteristic.

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

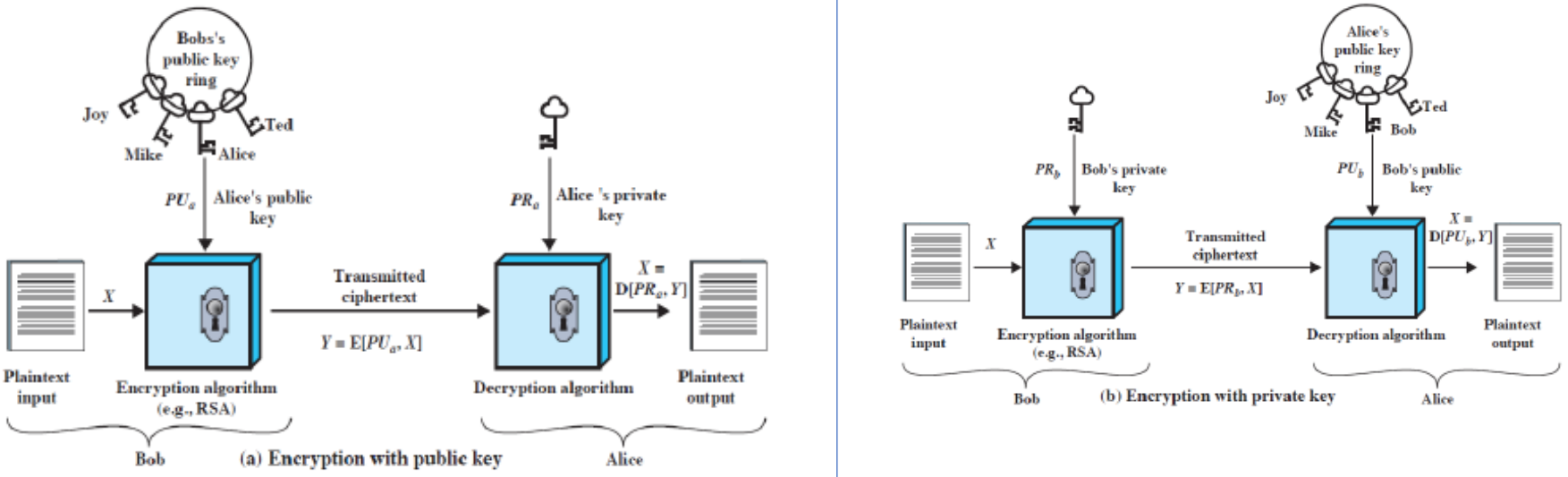


Fig: Public key cryptography

§ A **public-key encryption** scheme has six ingredients

1. Plaintext: This is the readable message or data that is fed into the algorithm as input.
2. Encryption algorithm: The encryption algorithm performs various transformations on the plaintext.
3. Public and private keys: This is a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption. The exact transformations performed by the algorithm depend on the public or private key that is provided as input.
4. Ciphertext: This is the scrambled message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different ciphertexts.
5. Decryption algorithm: This algorithm accepts the ciphertext and the matching key and produces the original plaintext.

§ The essential steps are the following.

1. Each user generates a pair of keys to be used for the encryption and decryption of messages.

2. Each user places one of the two keys in a public register or other accessible file. This is the public key. The companion key is kept private, each user maintains a collection of public keys obtained from others.

3. If Bob wishes to send a confidential message to Alice, Bob encrypts the message using Alice's public key.

4. When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message because only Alice knows Alice's private key.

Conventional Encryption	Public-Key Encryption
Needed to Work: 1. The same algorithm with the same key is used for encryption and decryption. 2. The sender and receiver must share the algorithm and the key. *Needed for Security:* 1. The key must be kept secret. 2. It must be impossible or at least impractical to decipher a message if the key is kept secret. 3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key.	*Needed to Work:* 1. One algorithm is used for encryption and a related algorithm for decryption with a pair of keys, one for encryption and one for decryption. 2. The sender and receiver must each have one of the matched pair of keys (not the same one). *Needed for Security:* 1. One of the two keys must be kept secret. 2. It must be impossible or at least impractical to decipher a message if one of the keys is kept secret. 3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.

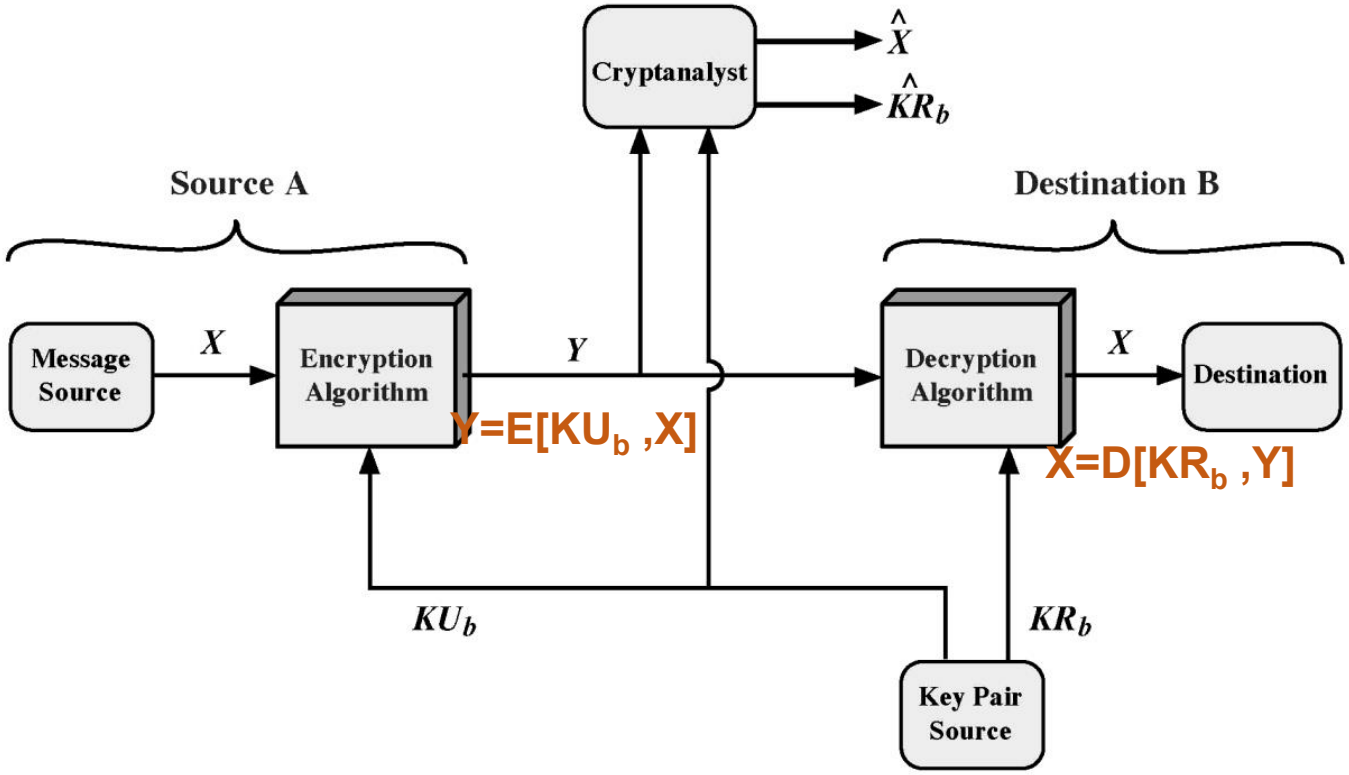


Fig: Public-Key Cryptosystem: Secrecy

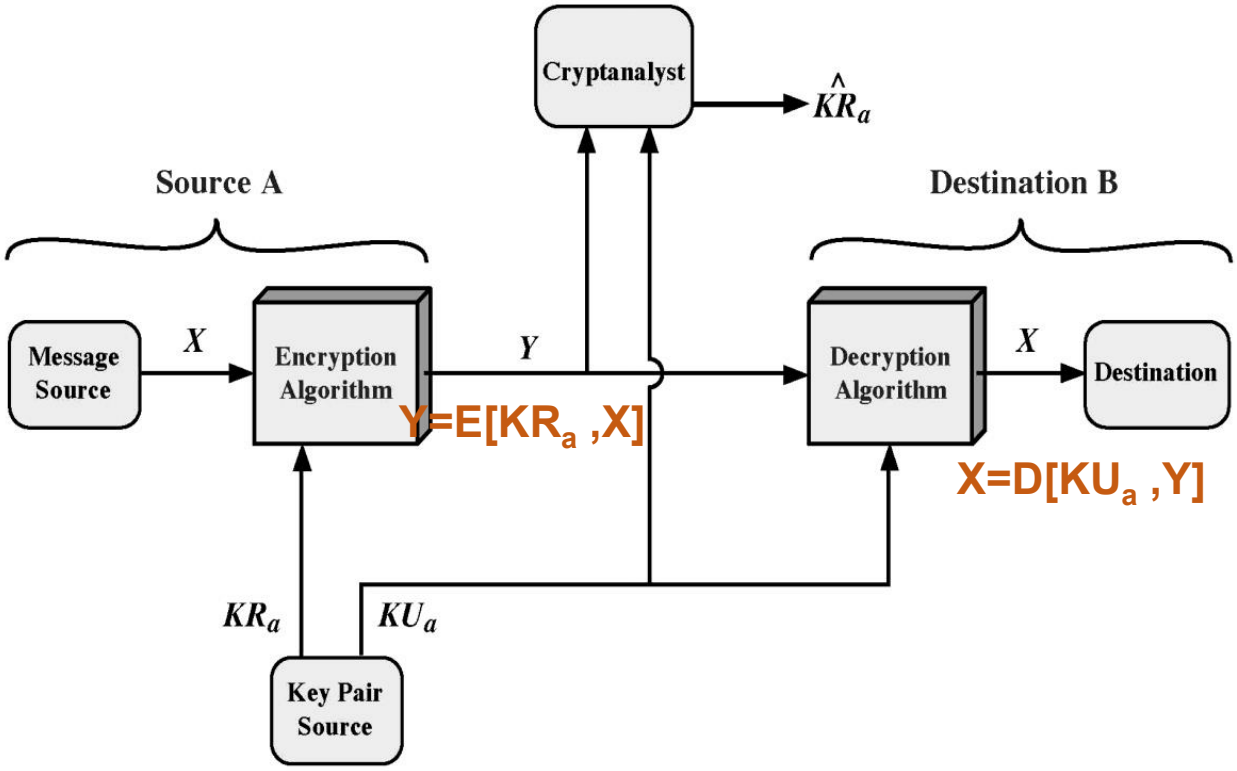


Fig: Public-Key Cryptosystem: Authentication

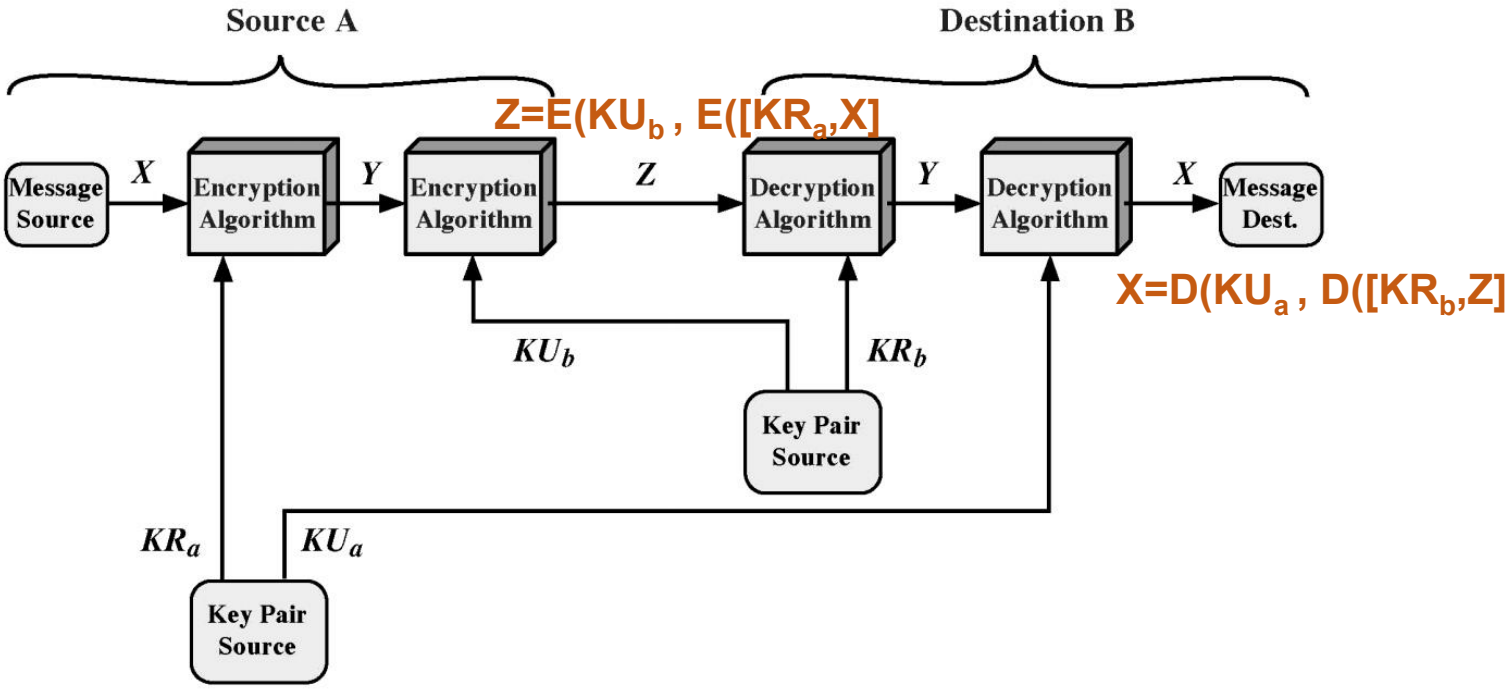


Fig: Public-Key Cryptosystem: Authentication and Secrecy

Applications for Public-Key Cryptosystems

§ The use of public-key cryptosystems can be classified into three categories

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

   −Provide secrecy

2. **Digital signature:** The sender "signs" a message with its private key.

   − Provide authentication

3. **Key exchange:** Two sides cooperate to exchange a session key.

Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Elliptic Curve	Yes	Yes	Yes
Diffie-Hellman	No	No	Yes
DSS	No	Yes	No

Table: Applications for Public-Key Cryptosystems

Requirements for Public-Key Cryptography

1. It is computationally easy for a party B to generate a pair (public key PU_b, private key PR_b).

2. It is computationally easy for a sender A, knowing the public key and the message to be encrypted, M, to generate the corresponding ciphertext:

$$C = E(PU_b, M)$$

3. It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message:

$$M = D(PR_b, C) = D[PR_b, E(PU_b, M)]$$

4. It is computationally infeasible for an adversary, knowing the public key, PU_b, to determine the private key, PR_b.

5. It is computationally infeasible for an adversary, knowing the public key, PU_b, and a ciphertext, C, to recover the original message, M.

6. The two keys can be applied in either order:

$$M = D[PU_b, E(PR_b, M)] = D[PR_b, E(PU_b, M)]$$

§ need a trapdoor one-way function

§ one-way function has

$Y = f(X)$ easy

$X = f^{-1}(Y)$ infeasible

§ A trap-door one-way function has

$Y = f_k(X)$ easy, if k and X are known

$X = f_k^{-1}(Y)$ easy, if k and Y are known

$X = f_k^{-1}(Y)$ infeasible,, if Y known but k not known

§ A practical public-key scheme depends on a suitable trap-door one-way function

Public-Key Cryptanalysis

§ Vulnerable to a brute-force attack

§ To find some way to compute the private key given the public key

The RSA Algorithm

§ best known & widely used public-key scheme

§ invented by Rivest, Shamir and Adleman of MIT in year 1977 and hence name **RSA** algorithm.

§ plaintext and ciphertext are integers between 0 and n - 1 for some n.

§ RSA makes use of an expression with exponentials. Plaintext is encrypted in blocks, with each block having a binary value less than some number n.

- block size must be less than or equal to $\log2(n) + 1$

§ For some plaintext block M and ciphertext block C.

$$C = M^e \bmod n$$

$$M = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$$

- Both sender and receiver must 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 *PU* = {*e*, *n*} and a private key of *PR* = {*d*, *n*}.
- Ingredients of RSA:

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

Key Generation by 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:	$M = C^d \bmod n$

Example:

1. Select two prime numbers, $p = 17$ and $q = 11$.
2. Calculate $n = pq = 17 * 11 = 187$.
3. Calculate $f(n) = (p - 1)(q - 1) = 16 * 10 = 160$.
4. Select e: GCD(e,160) = 1. we choose $e = 7$.
5. Determine d such that $de = 1 \pmod{160}$ and $d < 160$. The correct value is $d = 23$, because $23 * 7 = 161 = (1 * 160) + 1$; d can be calculated using the extended Euclid's algorithm
6. Publish public key: PU = {7,187}
7. Keep private key secret: PR = {23,187}

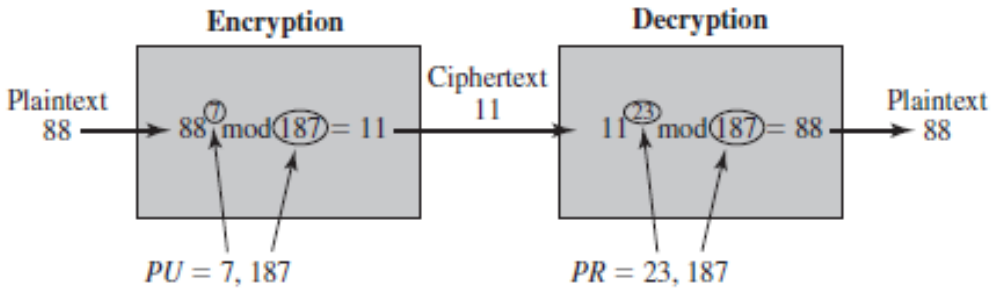
§ Given message M = 88 (88<167)

- Encryption:

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

- Decryption:

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



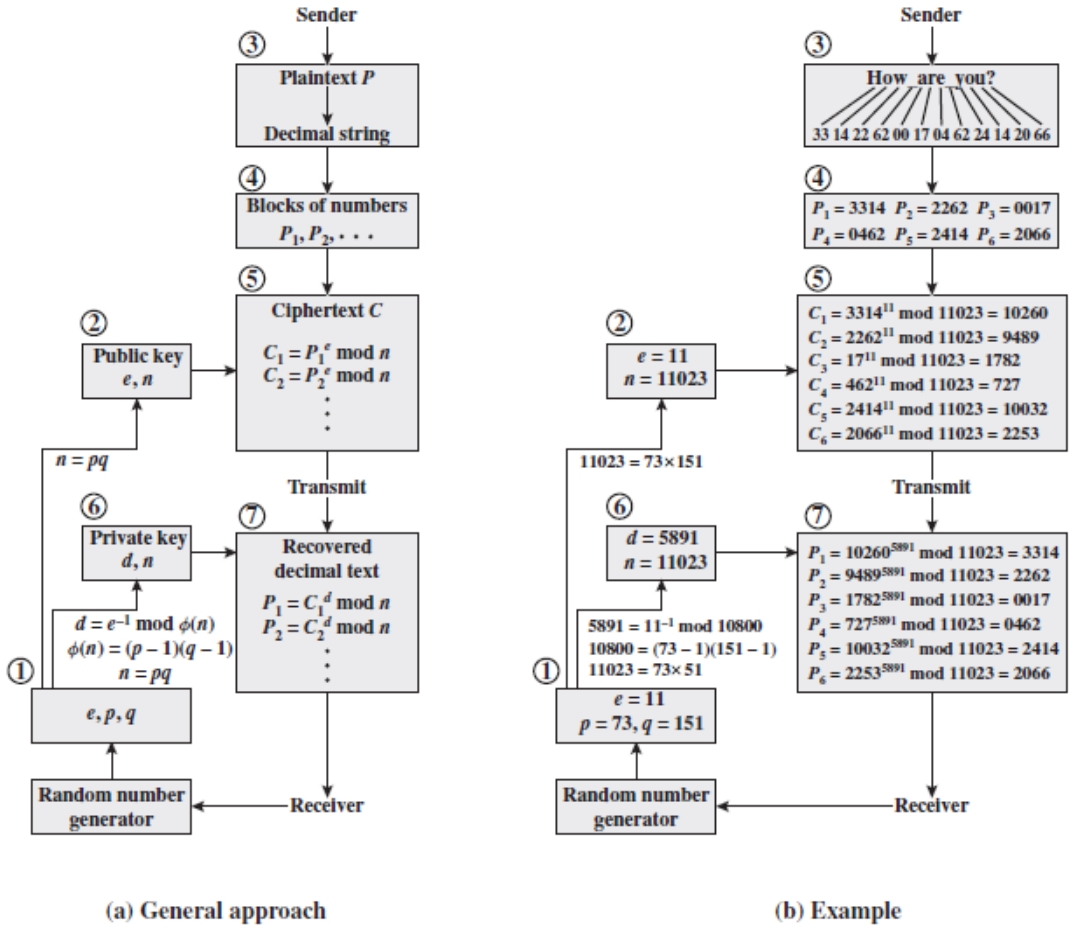


Fig: RSA Processing of Multiple Blocks

Computational Aspects

- There are two issues to consider:

  1.Encryption/decryption

  2. Key generation

1. Encryption/decryption

- Both encryption and decryption in RSA involve raising an integer to an integer power, mod n.
- we can make use of a property of modular arithmetic:

$$[(a \bmod n) * (b \bmod n)] \bmod n = (a * b) \bmod n$$

- Another consideration is the efficiency of exponentiation, because with RSA, we are dealing with potentially large exponents
- Ex: we wish to calculate $x^{11} \bmod n$ for some integers x and n.

$$x^{11} = x^{1+2+8} = (x)\,(x^2)\,(x^8).$$

$$= [(x \bmod n) * (x^2 \bmod n) * (x^8 \bmod n)] \bmod n.$$

- encryption uses exponentiation to power e
- hence if e small, this will be faster
  - often choose e = 65537 (2^{16} - 1)
  - also see choices of e = 3 or e = 17
- but if e too small (eg. e = 3) can attack
- if e fixed must ensure GCD(e,ø(n)) = 1
  - ie reject any p or q where p-1 or q-1 are not relatively prime to e

- Decryption uses exponentiation to power d
  - this is likely large, insecure if not
- can use the Chinese Remainder Theorem (CRT) to compute mod p and mod q separately; then combine to get answer
  - approx 4 times faster than doing it directly
- only owner of private key who knows values of p and q can use this technique

2. 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 p, q must be sufficiently large

– typically guess and use probabilistic test

§ exponents e, d are inverses, so use Inverse algorithm to compute the other

§ The procedure for picking a prime number is as follows.

1. Pick an odd integer *n* at random (e.g., using a pseudorandom number generator).
2. Pick an integer $a < n$ at random.
3. Perform the probabilistic primality test, such as Miller-Rabin, with *a* as a parameter. If *n* fails the test, reject the value *n* and go to step 1.
4. If *n* has passed a sufficient number of tests, accept *n*; otherwise, go to step 2.

The Security of RSA

§ Five possible approaches to attacking the RSA algorithm are

1. **Brute force:** This involves trying all possible private keys.
2. **Mathematical attacks:** There are several approaches, all equivalent in effort to factoring the product of two primes.
3. **Timing attacks:** These depend on the running time of the decryption algorithm.
4. **Hardware fault-based attack:** This involves inducing hardware faults in the processor that is generating digital signatures.
5. **Chosen ciphertext attacks:** This type of attack exploits properties of the RSA algorithm.

The Factoring Problem

§ We can identify three approaches to attacking RSA mathematically.

1. Factor n into its two prime factors. This enables calculation of $\Phi(n) = (p - 1) * (q - 1)$, which in turn enables determination of $d \equiv K\ e^{-1} \pmod{\Phi(n)}$.
2. Determine $\Phi(n)$ directly, without first determining p and q. Again, this enables determination of $d \equiv K\ e^{-1} \pmod{\Phi(n)}$.
3. Determine d directly, without first determining $\Phi(n)$.

§ currently assume 1024-2048 bit RSA is secure

- ensure p, q of similar size and matching other constraints

Timing Attacks

§ developed by Paul Kocher in mid-1990's

§ exploit timing variations in operations

– eg. multiplying by small vs.. large number

– or varying which instructions executed

§ infer operand size based on time taken

§ For RSA, exploits time taken for exponentiation

§ countermeasures

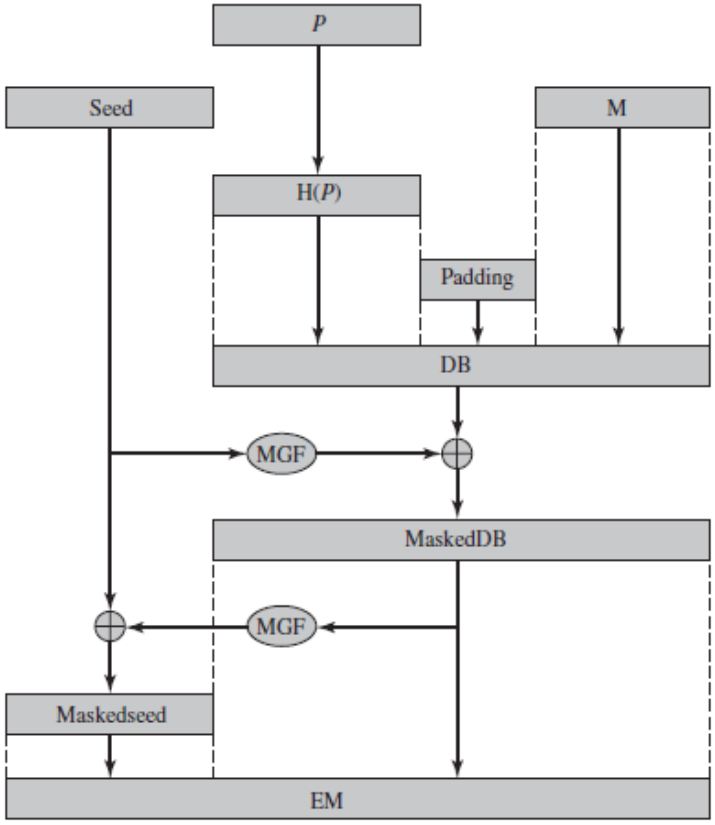
– use constant exponentiation time

– add random delays

– blind values used in calculations

Chosen Ciphertext Attacks

- RSA is vulnerable to a Chosen Ciphertext Attack (CCA)
- based on C(P1 x P2) = C(P1) x C(P2)
- attacker chooses ciphertexts and gets decrypted plaintext back
- choose ciphertext to exploit properties of RSA to provide info to help cryptanalysis
- can counter with random pad off plaintext
- or use Optimal Asymmetric Encryption Padding (OASP)



P = encoding parameters
M = message to be encoded
H = hash function
DB = data block
MGF = mask generating function
EM = encoded message

Problems on RSA:

1. Perform encryption and decryption using the RSA algorithm for the following:

   i. $p = 3$; $q = 11$, $e = 7$; $M = 5$

   ii. $p = 5$; $q = 11$, $e = 3$; $M = 9$

   iii. $p = 7$; $q = 11$, $e = 17$; $M = 8$

   iv. $p = 11$; $q = 13$, $e = 11$; $M = 7$

   v. $p = 17$; $q = 31$, $e = 7$; $M = 2$

2. In a public-key system using RSA, you intercept the ciphertext $C = 10$ sent to a user whose public key is $e = 5$, $n = 35$. What is the plaintext M?

Simple method to compute d

d=((Φ(n)*i)+1)/e should produce an integer result

d=integer result

Try substituting i=1,2,3,……

solution

1. a. $n = 33$; f(n) = 20; $d = 3$; C = 26.

b. $n = 55$; f(n) = 40; $d = 27$; C = 14.

c. $n = 77$; f(n) = 60; $d = 53$; C = 57.

d. $n = 143$; f(n) = 120; $d = 11$; C = 106.

e. $n = 527$; f(n) = 480; $d = 343$; C = 128.

For decryption, we have

$$128343 \bmod 527 = 128^{256} * 128^{64} * 128^{16} * 128^{4} * 128^{2} * 128^{1} \bmod 527$$

$$= 35 * 256 * 35 * 101 * 47 * 128 = 2 \bmod 527$$

$$= 2 \bmod 257$$

2. 5

Applications of RSA

§ **Banking**

- RSA algorithm is commonly used by banks to protect their data, like customer information and transaction record. Some scenarios are credit card and office computers.

§ **Telecommunications**

- RSA algorithm is useful to encrypt the call data as a concern for privacy issues.

§ **Ecommerce**

- RSA algorithm is useful in protecting user identity for transactions.