Unit 3

COMPUTER-BASED ASYMMETRIC-KEY CRYPTOGRAPHY ALGORITHMS

MKK

INTRODUCTION

- Symmetric-key cryptography is fast and efficient.
- However, it suffers from a big disadvantage of the problem of key exchange.
- The sender and the receiver of an encrypted message use the same key in symmetric-key cryptography, and it is very tough to agree upon a common key without letting anyone else know about it.
- Asymmetric-key cryptography solves this problem. Here, each communicating party uses two keys to form a key pair—one key (the private key) remains with the party, and the other key (the public key) is shared with everybody.

BRIEF HISTORY OF ASYMMETRIC-KEY CRYPTOGRAPHY

In the mid-1970s, Whitfield Diffie, a student at the Stanford University met with Martin Hellman, his professor, and the two began to think about the problem of key exchange. After some research and complicated mathematical analysis, they came up with the idea of asymmetric-key cryptography. Many experts believe that this development is the first—and perhaps the only—truly revolutionary concept in the history of cryptography. Diffie and Hellman can, therefore, be regarded as the fathers of asymmetric-key cryptography

However, there is a lot of debate regarding who should get the credit for developing asymmetric-key cryptography. It is believed that James Ellis of the British Communications Electronic Security Group (CSEG) proposed the idea of asymmetric-key cryptography in the 1960s. His ideas were based on an anonymous paper written at the Bell Labs during the Second World War. However, Ellis could not devise a practical algorithm based on his ideas.

Simultaneously, the US National Security Agency (NSA) was also working on asymmetric-key cryptography. It is believed that the NSA system based on the asymmetric-key cryptography was operational in the mid-1970s.

Based on the theoretical framework of Diffie and Hellman, in 1977, Ron Rivest, Adi Shamir and Len Adleman at MIT developed the first major asymmetric-key cryptography system, and published their results in 1978. This method is called **RSA algorithm**.

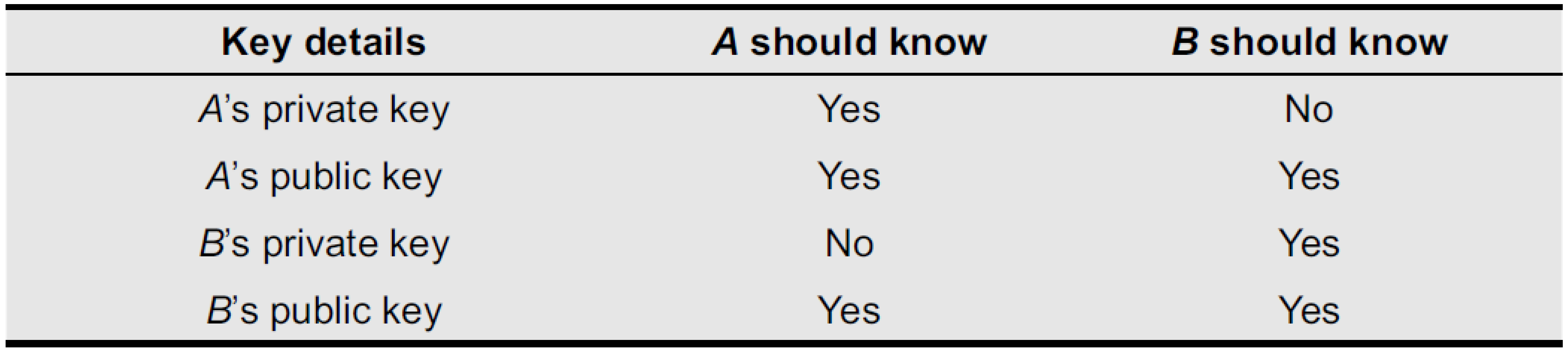
Even today, RSA is the most widely accepted public-key solution. It solves the problem of key agreements and distribution.

To communicate securely over any network, all one needs to do is to publish one's public key. All these public keys can then be stored in a database that anyone can consult. However, the private key only remains with the respective individuals.

AN OVERVIEW OF ASYMMETRIC-KEY CRYPTOGRAPHY

- In asymmetric-key cryptography, also called public key cryptography, two different keys (which form a key pair) are used. One key is used for encryption and only the other corresponding key must be used for decryption. No other key can decrypt the message—not even the original (i.e. the first) key used for encryption! The beauty of this scheme is that every communicating party needs just a key pair for communicating with any number of other communicating parties.
- One of the two keys is called the public key and the other is the private key.
- The private key remains with you as a secret. You must not disclose your private key to anybody. However, the public key is for the general public. It is disclosed to all parties that you want to communicate with.

- Suppose A wants to send a message to B without having to worry about its security.
- Then, A and B should each have a private key and a public key.
  - ● A should keep her private key secret.
  - ● B should keep her private key secret.
  - ● A should inform B about her public key.
  - ● B should inform A about her public key.
- Thus, we have a matrix as shown in Fig
- Key details A should know B should know

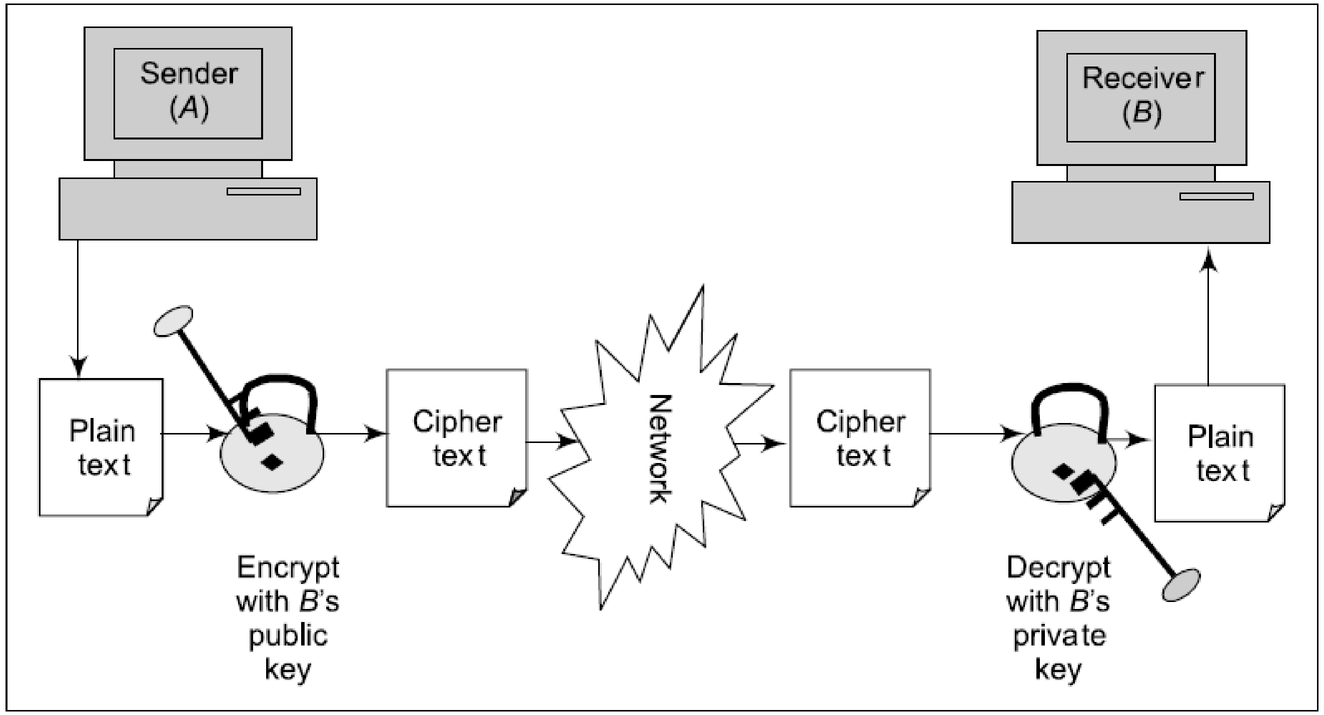


Key details	***A* should know**	***B* should know**
A's private key	Yes	No
A's public key	Yes	Yes
B's private key	No	Yes
B's public key	Yes	Yes

Armed with this knowledge, asymmetric-key cryptography works as follows:

1. When A wants to send a message to B, A encrypts the message using B's public key. This is possible because A knows B's public key.
2. A sends this message (which was encrypted with B's public key) to B.
3. B decrypts A's message using B's private key.

Thus, no one else can make any sense out of the message even if one can manage to intercept the message. This is because the intruder (ideally) does not know about B's private key. It is only B's private key that can decrypt the message.

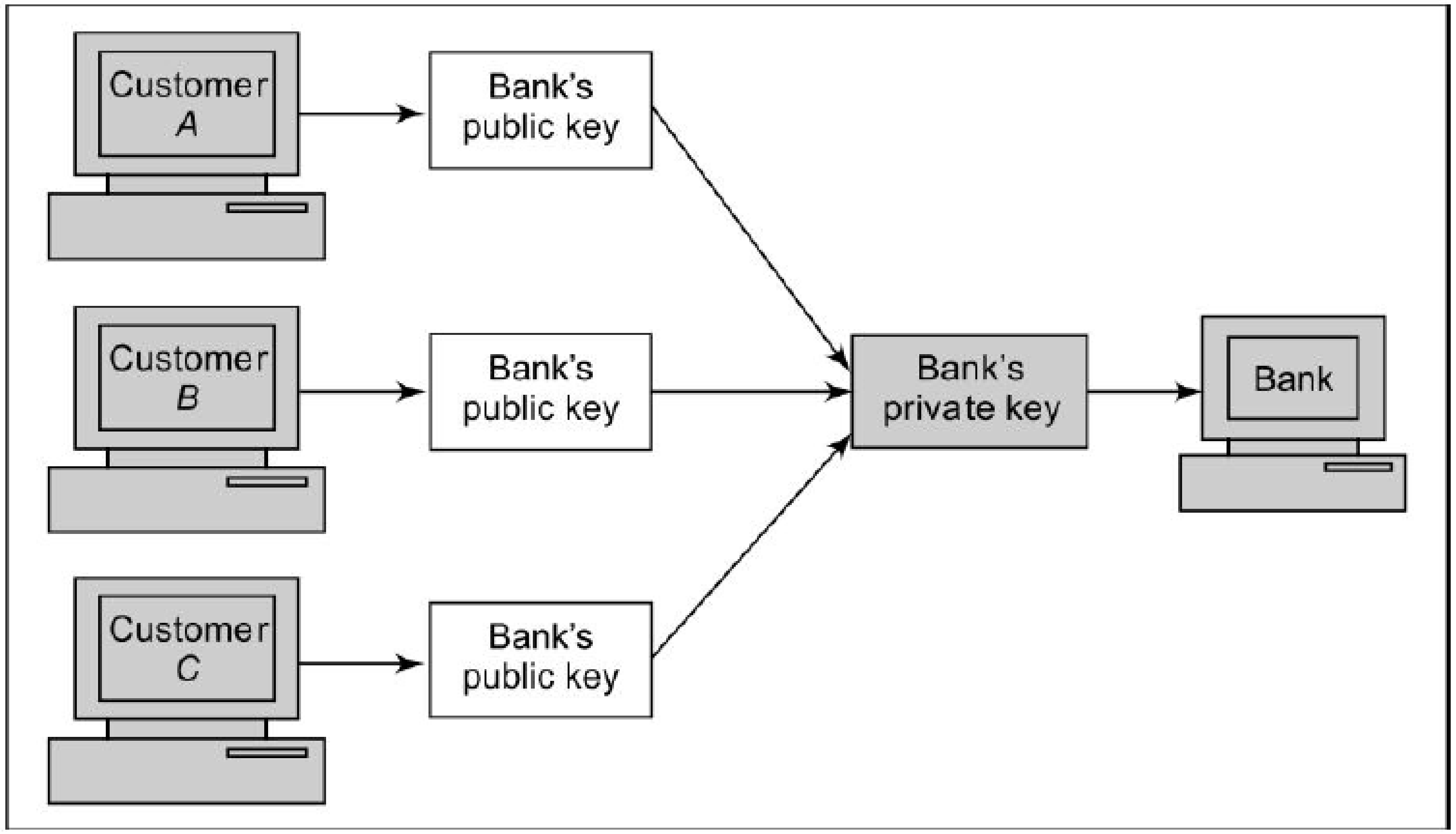


Example:

We can consider a practical situation that describes asymmetric-key cryptography as used in real life.

Suppose a bank accepts many requests for transactions from its customers over an insecure network.

The bank can have a private-key–public key pair.



Use of a public-key–private-key pair by a bank

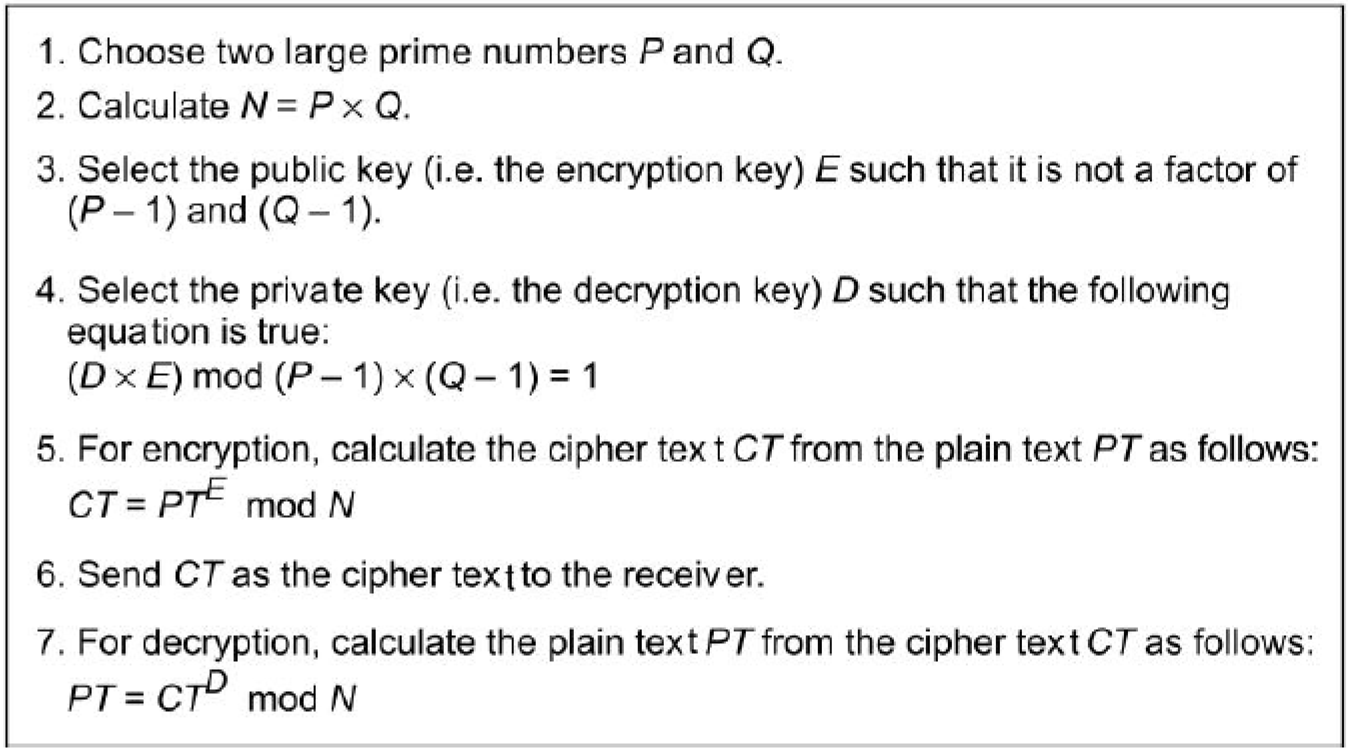
MKK

The RSA Algorithm

RSA algorithm is an asymmetric cryptography algorithm. Asymmetric actually means that it works on two different keys i.e. **Public Key** and **Private Key.** As the name describes that the Public Key is given to everyone and the Private key is kept private.

- The RSA algorithm is based on the mathematical fact that it is easy to find and multiply large prime numbers together, but it is extremely difficult to factor their product. The private and public keys in RSA are based on very large (made up of 100 or more digits) prime numbers.
- However, the real challenge in the case of RSA is the selection and generation of the public and private keys.
- Let us now understand how the public and private keys are generated, and using them, how we can perform encryption and decryption in RSA.

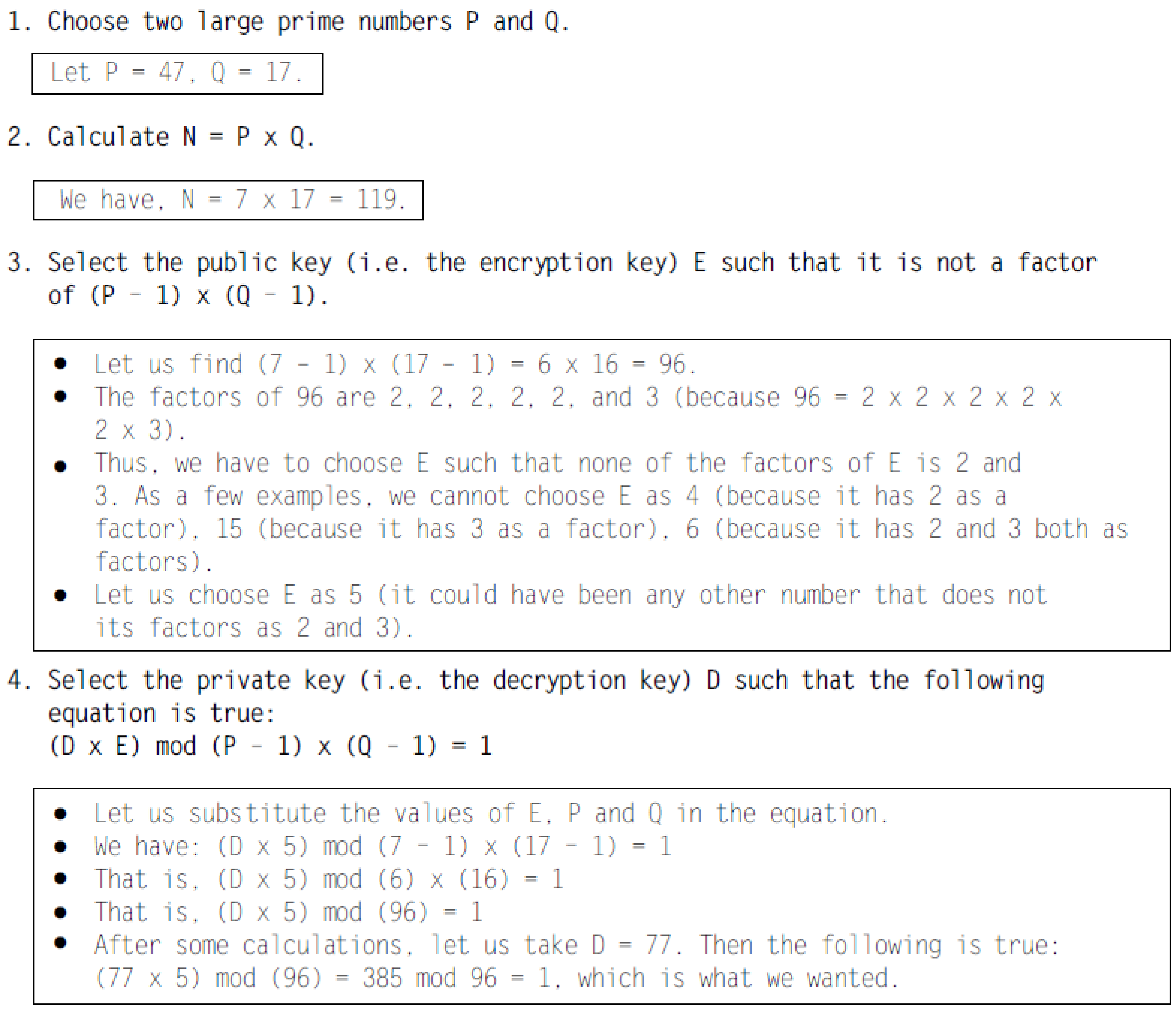
The RSA algorithm



1. Choose two large prime numbers P and Q.
2. Calculate $N = P \times Q$.
3. Select the public key (i.e. the encryption key) E such that it is not a factor of $(P - 1)$ and $(Q - 1)$.
4. Select the private key (i.e. the decryption key) D such that the following equation is true:
   $(D \times E) \bmod (P - 1) \times (Q - 1) = 1$
5. For encryption, calculate the cipher text CT from the plain text PT as follows:
   $CT = PT^E \bmod N$
6. Send CT as the cipher text to the receiver.
7. For decryption, calculate the plain text PT from the cipher text CT as follows:
   $PT = CT^D \bmod N$

MKK

Examples of RSA



1. Choose two large prime numbers P and Q.

   Let P = 47, Q = 17.

2. Calculate N = P x Q.

   We have, N = 7 x 17 = 119.

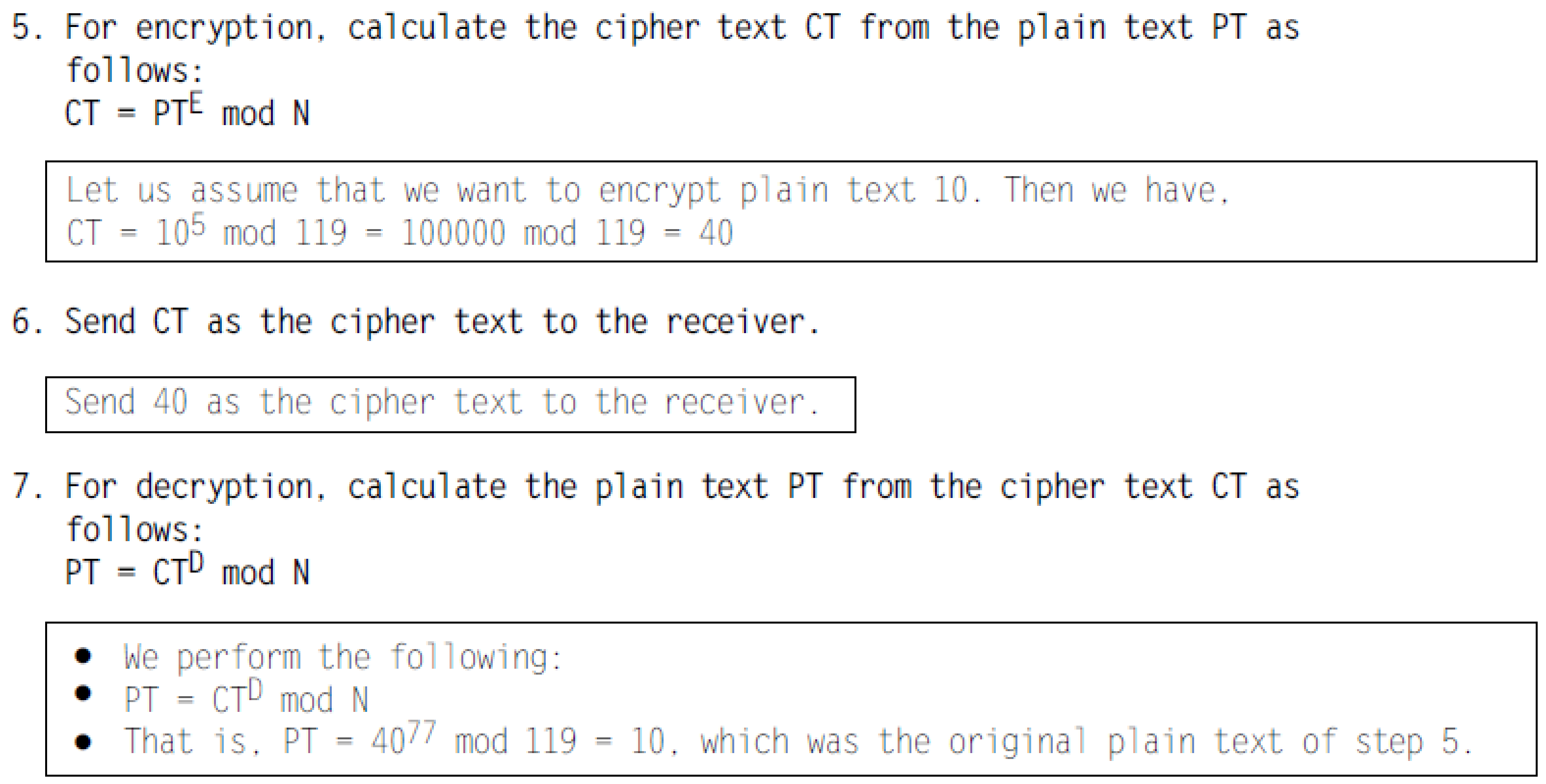
3. Select the public key (i.e. the encryption key) E such that it is not a factor of (P – 1) x (Q – 1).

   - Let us find (7 – 1) x (17 – 1) = 6 x 16 = 96.
   - The factors of 96 are 2, 2, 2, 2, 2, and 3 (because 96 = 2 x 2 x 2 x 2 x 2 x 3).
   - Thus, we have to choose E such that none of the factors of E is 2 and 3. As a few examples, we cannot choose E as 4 (because it has 2 as a factor), 15 (because it has 3 as a factor), 6 (because it has 2 and 3 both as factors).
   - Let us choose E as 5 (it could have been any other number that does not its factors as 2 and 3).

4. Select the private key (i.e. the decryption key) D such that the following equation is true:
   (D x E) mod (P – 1) x (Q – 1) = 1

   - Let us substitute the values of E, P and Q in the equation.
   - We have: (D x 5) mod (7 – 1) x (17 – 1) = 1
   - That is, (D x 5) mod (6) x (16) = 1
   - That is, (D x 5) mod (96) = 1
   - After some calculations, let us take D = 77. Then the following is true: (77 x 5) mod (96) = 385 mod 96 = 1, which is what we wanted.

Let us choose D as 77, because we can see that (5 * 77) mod 96 = 385 mod 96 = 1, which satisfies our condition.


5. For encryption, calculate the cipher text CT from the plain text PT as follows:
   $CT = PT^E \bmod N$

> Let us assume that we want to encrypt plain text 10. Then we have,
> $CT = 10^5 \bmod 119 = 100000 \bmod 119 = 40$

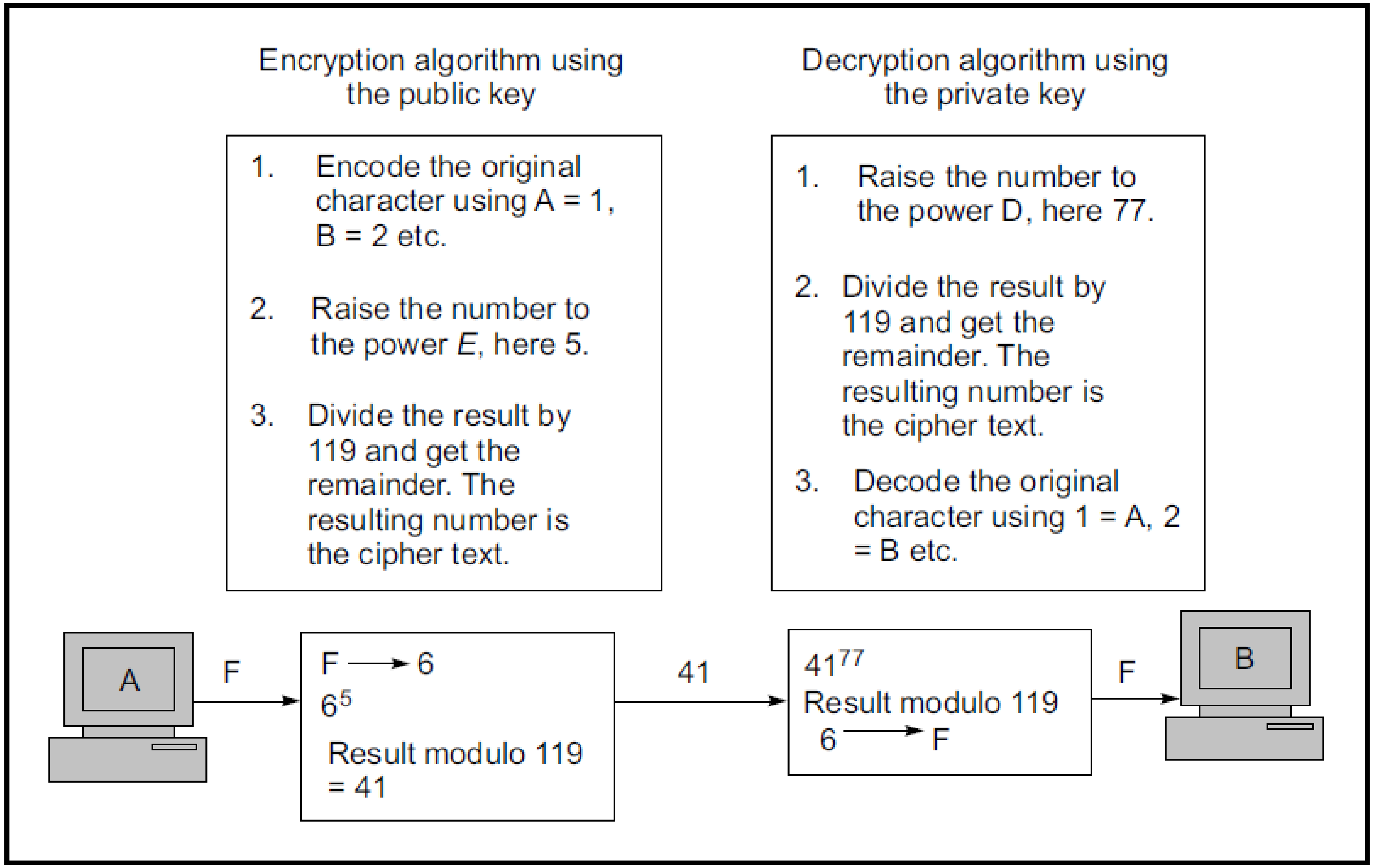
6. Send CT as the cipher text to the receiver.

> Send 40 as the cipher text to the receiver.

7. For decryption, calculate the plain text PT from the cipher text CT as follows:
   $PT = CT^D \bmod N$

> - We perform the following:
> - $PT = CT^D \bmod N$
> - That is, $PT = 40^{77} \bmod 119 = 10$, which was the original plain text of step 5.

- Now, based on these values, let us consider an encryption and decryption process.
- Here, A is the sender and B is the receiver. As we can see, here we use an encoding scheme of encoding alphabets as A = 1, B = 2, …, Z = 26.
- Let us assume that we want to encrypt a single alphabet F using this scheme, and with B's public key as 77 (known to A and B) and B's private key (known only to B) as 5.
- Then the description is as follows



Encryption algorithm using the public key

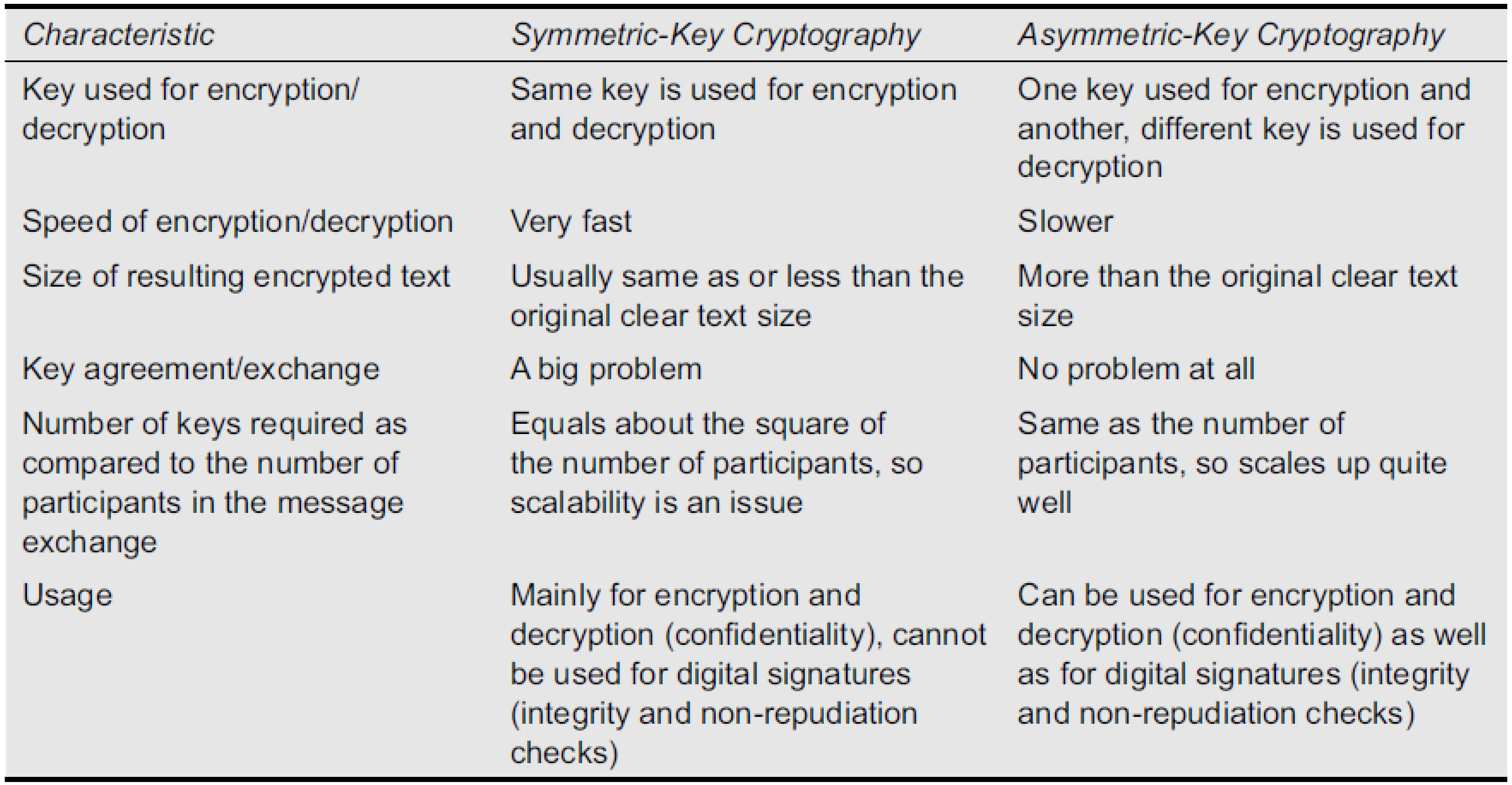
1. Encode the original character using A = 1, B = 2 etc.
2. Raise the number to the power *E*, here 5.
3. Divide the result by 119 and get the remainder. The resulting number is the cipher text.

Decryption algorithm using the private key

1. Raise the number to the power D, here 77.
2. Divide the result by 119 and get the remainder. The resulting number is the cipher text.
3. Decode the original character using 1 = A, 2 = B etc.

SYMMETRIC- AND ASYMMETRIC-KEY CRYPTOGRAPHY

- Comparison Between Symmetric- and Asymmetric-Key Cryptography



Characteristic	*Symmetric-Key Cryptography*	*Asymmetric-Key Cryptography*
Key used for encryption/ decryption	Same key is used for encryption and decryption	One key used for encryption and another, different key is used for decryption
Speed of encryption/decryption	Very fast	Slower
Size of resulting encrypted text	Usually same as or less than the original clear text size	More than the original clear text size
Key agreement/exchange	A big problem	No problem at all
Number of keys required as compared to the number of participants in the message exchange	Equals about the square of the number of participants, so scalability is an issue	Same as the number of participants, so scales up quite well
Usage	Mainly for encryption and decryption (confidentiality), cannot be used for digital signatures (integrity and non-repudiation checks)	Can be used for encryption and decryption (confidentiality) as well as for digital signatures (integrity and non-repudiation checks)

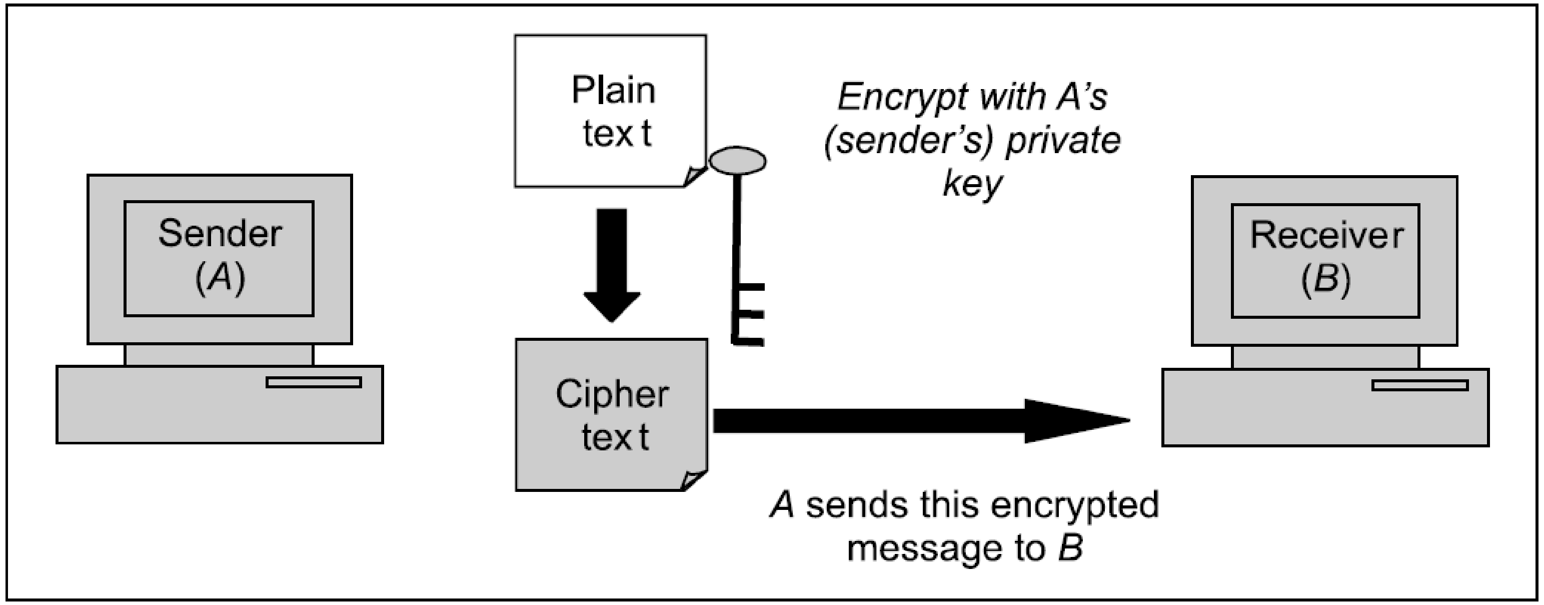
The Best of Both Worlds

How economic it would be, if we can combine the two cryptography mechanisms, so as to achieve the better of the two, and yet do not compromise on any of the features! More specifically, we need to ensure that the following objectives are met:

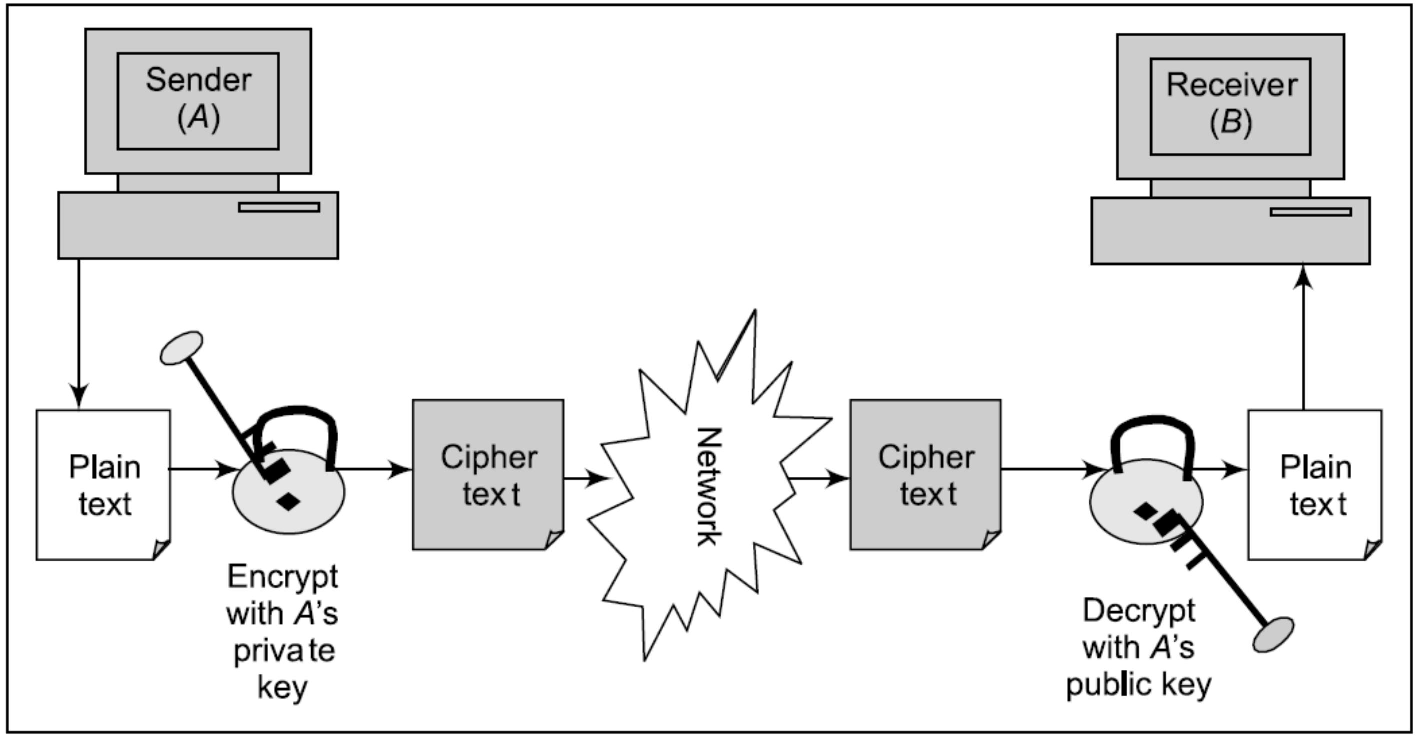
1. The solution should be completely secure.
2. The encryption and decryption processes must not take a long time.
3. The generated cipher text should be compact in size.
4. The solution should scale to a large number of users easily, without introducing any additional complications.
5. The key-distribution problem must be solved by the solution.

Digital Signatures

- Digital signatures are the public-key primitives of message authentication. Digital signature is a technique that binds a person/entity to the digital data.
- As we know in asymmetric-key cryptography:
  - "If A is the sender of a message and B is the receiver, A encrypts the message with B's public key and sends the encrypted message to B."
- Let us now consider another scheme, as follows:
  - If A is the sender of a message and B is the receiver, A encrypts the message with A's private key and sends the encrypted message to B.

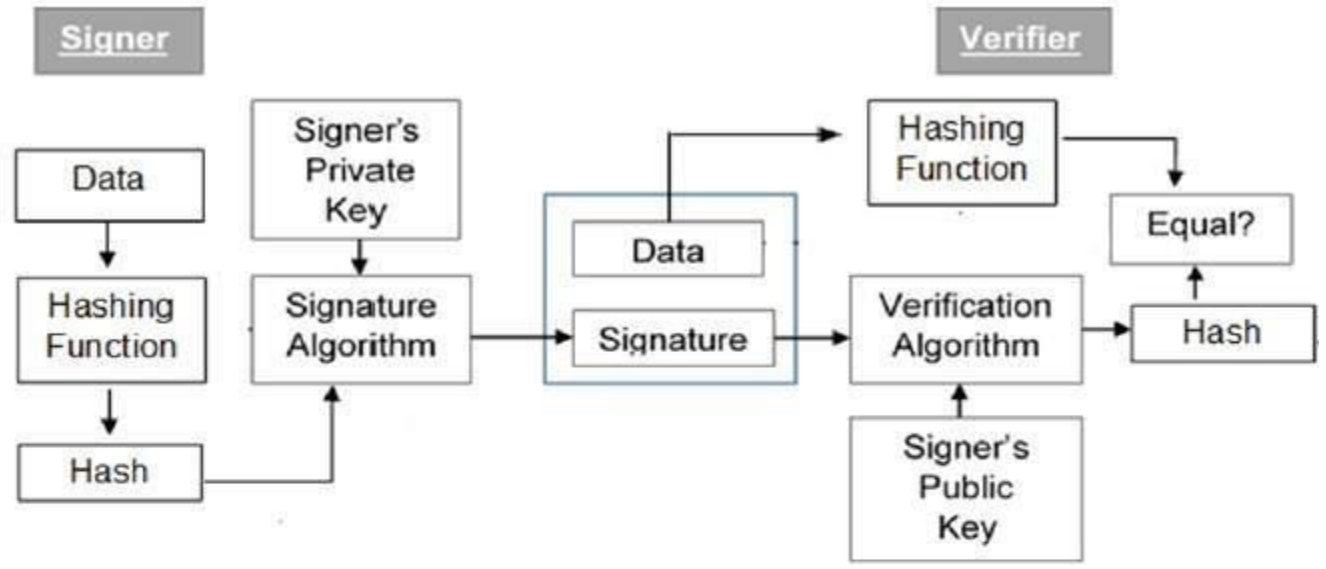

Plain text
Encrypt with A's (sender's) private key
Sender (A)
Receiver (B)
Cipher text
A sends this encrypted message to B

- Here B use A’s public key to decrypt it, and therefore, access the plain text.
- If the decryption is successful, it assures B that this message was indeed sent by A. This is because if B can decrypt a message with A’s public key, it means that the message must have been initially encrypted with A’s private key
- Therefore, someone posing as A (say C) could not have sent a message encrypted with A’s private key to B.
- Moreover, in the case of a dispute of non-repudiation



MKK

- Digital signatures have assumed great significance in the modern world of Web commerce.
- Most countries have already made provisions for recognizing a digital signature as a valid authorization mechanism, just like paper-based signatures.
- Digital signatures have legal status now.
  - For example, suppose you send a message to your bank over the Internet, to transfer some amount from your account to your friend's account, and digitally sign the message, this transaction has the same status as the one wherein you fill in and sign the bank's paper-based money-transfer slip.



MKK

KNAPSACK ALGORITHM

- Actually, Ralph Merkle and Martin Hellman developed the first algorithm for public-key encryption, called the Knapsack algorithm. It is based on the Knapsack problem.
- This is actually a simple problem.
- Given a pile of items, each with different weights, is it possible to put some of them in a bag (i.e. knapsack) in such a way that the knapsack has a certain weight?

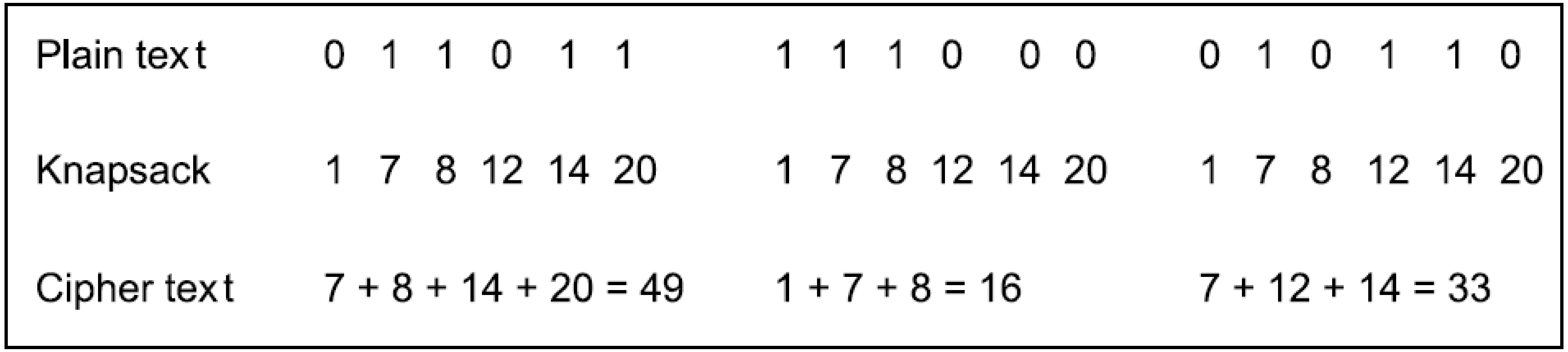
That is, if M1, M2, …, Mn are the given values and S is the sum,

find out bi so that:

S = b1M1 + b2M2 + … + bnMn

- Each *bi can be 0 or 1*. A *1* indicates that the item is in the knapsack, and a *0* indicates that it is not.
- A block of plain text equal in length to the number of items in the pile would select the items in the knapsack. The cipher text is the resulting sum.
- For example, if the knapsack is 1, 7, 8, 12, 14, 20 then the plain text and the resulting cipher text is as shown below

Knapsack example



Plain text	0 1 1 0 1 1	1 1 1 0 0 0	0 1 0 1 1 0
Knapsack	1 7 8 12 14 20	1 7 8 12 14 20	1 7 8 12 14 20
Cipher text	7 + 8 + 14 + 20 = 49	1 + 7 + 8 = 16	7 + 12 + 14 = 33

SOME OTHER ALGORITHMS

Elgamal digital signature

- The ElGamal digital-signature scheme uses the same keys, but a different algorithm.
- The algorithm creates two digital signatures.
- In the verification step, these two signatures are tallied.
- The key-generation process here is the same as earlier and hence we would not repeat the discussion. The public key remains (E1, E2, P) and the private key continues to be D.

Signature

- The signature process works as follows:

  1. The sender selects a random number R.
  2. The sender computes the first signature S1 using the equation S1 = E1R mod P.
  3. The sender computes the second signature S2 using the equation S2 = (M – D * S1) * R–1 mod (P – 1), where M is the original message that needs to be signed.
  4. The sender sends M, S1, and S2 to the receiver.

For example,

let E1 = 10, E2 = 4, P = 19, M = 14, D = 16, and R = 5.

Then we have:

$$S1 = E1^R \bmod P = 10^5 \bmod 19 = 3$$

$$S2 = (M - D \times S1) \times R^{-1} \bmod (P - 1) = (14 - 16 \times 3) \times 5^{-1} \bmod 18 = 4$$

Hence, the signature is (S1, S2) i.e. (3, 4). This is sent to the receiver.

Verification

- The verification process works as follows:
- 1. The receiver performs the first part of verification called V1 using the equation V1 = $E1^M$ mod P.
- 2. The receiver performs the second part of verification called as V2 using the equation V2 = $E2^{S1}$ * $S1^{S2}$ mod P.

In our example:

$$V1 = E1^M \bmod P = 10^{14} \bmod 19 = 16$$

$$V2 = E2^{S1} \times S1^{S2} \bmod P = 4^3 \times 3^4 \bmod 19 = 5184 \bmod 19 = 16$$

- Since V1 = V2, the signature is considered valid.

Attacks on digital signatures

In general, three types of attacks are attempted against digital signatures, as outlined below:

1. Chosen-message Attack

In the chosen-message attack, the attacker tricks a genuine user into digitally signing messages that the user does not normally intend to sign.

2. Known-message Attack

In the known-message attack, the attacker obtains the previous few messages and the corresponding digital signatures from a genuine user.

3. Key-only Attack

In the key-only attack, the assumption is that some information was made public by a genuine user. This attacker now tries to misuse this public information.

That's all about

UNIT 3

MKK