public key certificates approach. It was suggested by Kohnfelder for the first time. In this approach, the public keys are exchanged by using certificates. This avoids the need to contact the public key authority for public key. Each certificate contains some other information in addition to public key. This other information includes time of the request and network address of the user who made the request. This time in the certificate helps to differentiate one request from other. Suppose user A's private key is captured by an attacker. Therefore, user A generates a new pair of public and private keys. Then he requested for the certificate by sending his public and private key pair. Meanwhile, the attacker uses the old certificate to communicate with user B. If user B have the old public key of A and encrypt the message using this old public key, the attacker can decrypt the message and read the message.

The information in the certificate is encrypted by the authority's private key. So, only the registered users who know the authority's public key can decrypt the message. When the user A sends the request for communication to user B with his certificate, user B first verifies the certificate of user A. The communication between the two users takes place as follows:

Step 1 User A requests the certificate authority for a certificate. For this he sends his name and his public key.

Step 2 The certificate authority responds to the request by sending encrypted certificate for user A. The certificate is encrypted using authority's private key. Each certificate contains some other information in addition to public key. This other information includes time of the request and network address of the user who made the request.

Step 3 User A uses this certificate for communication with user B. Then user B decrypts the certificate using authority's public key and verifies the user A authentication. Also, user B stores the public key of A for further communication.

Step 4 User B requests the certificate authority for a certificate. For this he sends his name and his public key.

Step 5 The certificate authority sends the certificate to user B.

Step 6 User B sends the certificate to user A. User A verifies the information by decrypting the certificates by authority's public key and stores the public key of user B.

After exchanging the certificates, user A and B start communication. They use each other's public key for communication. The detail process is as shown in Figure 8.4.

8.3 DIFFIE-HELLMAN KEY EXCHANGE

Whether we use symmetric encryption or public key encryption technique, transfer of key among the users is the main issue. Key exchange in public key cryptography is simple as compared to symmetric encryption. In Section 8.2 we discuss different approaches of key transfer for public key cryptography. But for symmetric encryption, the key may be handled personally using different media. Symmetric encryption techniques are faster than public key cryptography.

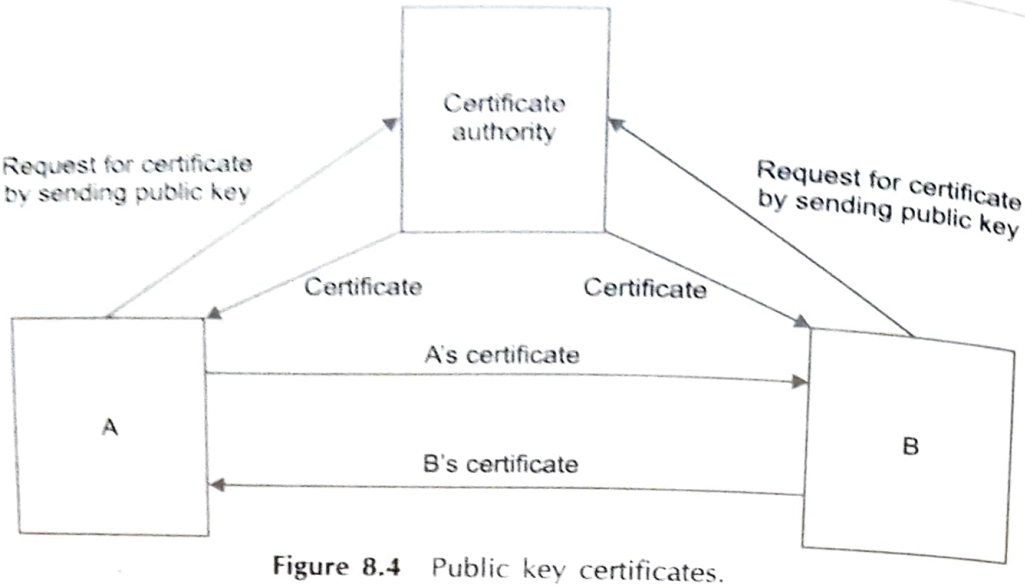

Figure 8.4 Public key certificates.

Diffie–Hellman designed an algorithm which helps for key transmission. This algorithm is known as *Diffie–Hellman key exchange algorithm*. The name is derived from the two scientists Whitefield Diffie and Martin Hellman who developed this algorithm. They developed a protocol called *key agreement protocol* in 1976. In this, the private key is exchangeable among the users using public key techniques. This algorithm can be used for key exchange and not for encryption or decryption. Using exchangeable keys, both the parties generate a key called *shared secret key* which can be used for encryption and decryption. So, actual shared secret key is never transmitted through the insecure channel. The performance of this algorithm is better than any other algorithm.

8.3.1 Description

Suppose A and B are the two users who wish to communicate with each other. They decided to use Diffie–Hellman key exchange algorithm for sharing their keys. The working of Diffie–Hellman algorithm is illustrated from the following steps.

Step 1 First, user A and user B both agree on two numbers, n and g. Where n is a large prime number and g is a primitive root of n.

Step 2 User A selects X_A as his private key randomly, such that $X_A < n$. X_A is a large positive integer number.

Step 3 Similarly, user B selects X_B as his private key randomly, such that $X_B < n$. X_B is a large positive integer number.

Step 4 User A computes his public key, Y_A, using $Y_A = (g^{X_A}) \bmod n$.

Step 5 Similarly, user B computes his public key, Y_B, using $Y_B = (g^{X_B}) \bmod n$.

Step 6 Both the users now exchange their public keys over the insecure channel.

Step 7 User A computes the key, k, called *shared secret key*, using the formula $k = (Y_B^{X_A}) \bmod n$.

Step 8 User B computes the key, k, called shared secret key, using the formula, $k = (Y_A^{X_B}) \bmod n$.

Step 9 Now, user A and user B communicate with each other using one of the symmetric encryption techniques. They use the shared secret key, k, as the encryption key for the selected algorithm. This key k was never transmitted over the insecure channel.

The Diffie–Hellman key exchange algorithm uses prime number n as modulo and primitive root of n. Table 8.1 illustrates the working of Diffie–Hellman algorithm.

Table 8.1 Working Of Diffice–Hallman algorithm

User A		User B	
Private key	**Calculation**	**Private key**	**Calculation**
X_A	n, g $Y_A = g^{X_A} \bmod n \rightarrow$ *A's public key* $(Y_B^{X_A}) \bmod n$ Shared key	X_B	n, g $\leftarrow Y_B = g^{X_B} \bmod n$ *B's public key* $(Y_A^{X_B}) \bmod n$ Shared key

User A and user B select n and g such that $n = 19$ and $g = 7$.

User A select his private key $X_A = 8$, then computes his public key $Y_A = g^{X_A} \bmod n$

$$Y_A = 7^8 \bmod 19 = 11$$

Therefore, A having private key $X_A = 8$ and public key $Y_A = 11$.

A sends his public key $Y_A = 11$ to B.

User B selects his private key $X_B = 10$, then computes his public key $Y_B = g^{X_B} \bmod n$

$$7^{10} \bmod 19 = 7$$

Therefore, B having private key $X_B = 10$ and public key $Y_B = 7$.

B sends his public key $Y_B = 7$ to A

A computes shared secret key using: $(Y_B\ X_A) \bmod n$

$$(7^8 \bmod 19) = 11$$

B computes shared secret key using: $(Y_A X_B) \bmod n$

$$11^{10} \bmod 19 = 11$$

Both users A and B have the same value of shared secret key. Note that only private keys, i.e., X_A, X_B and shared secret keys $Y_B^{X_A} = Y_A X_B$ should keep secret. All the other values, i.e., g, n, Y_A and Y_B need not be secret. The shared secret key can be used for encryption. This key is known only to A and B. Larger values of X_A, X_B, and n provide more security and make difficult for cryptanalysis. If these values are small, it is possible for the cryptanalyst to find out shared secret key easily. In the above example, we know that the values of $n = 19$, $g = 7$, $Y_A = 11$ and $Y_B = 7$ are publicaly available. Therefore, one can try $g^{X_A X_B} \bmod 19$. There are only 18 values and this equation is possible and easily determined the shared secret key. Therefore, the values of n and g should be large.

Considering above problem, Table 8.2 will illustrate how the cryptanalyst tries to find the shared secret key.

Table 8.2 Shared Secret Key

User A		User B		Cryptanalyst	
Knows	**Doesn't know**	**Knows**	**Doesn't know**	**Knows**	**Doesn't know**
$n = 19$	$X_B = 10$	$n = 19$	$X_A = 8$	$n = 19$	$X_A = 8$
$g = 7$		$g = 7$		$g = 7$	$X_B = 10$
$X_A = 8$		$X_B = 10$			$k_s = 11$
$7^8 \bmod 19 = 11$		$7^{10} \bmod 19 = 7$		$7^{X_A} \bmod 19 = 11$	
$7^{X_A} \bmod 19 = 7$		$7^{X_A} \bmod 19 = 11$		$7^{X_B} \bmod 19 = 7$	
$7^8 \bmod 19 = 11$		$11^{10} \bmod 19 = 11$		$k_s = 7^{X_A} \bmod 19$	
$11^{X_B} \bmod 19 = 11$		$7^{X_A} \bmod 19 = 11$		$k_s = 11^{X_B} \bmod 19$	
$7^8 \bmod 19 = 11^{X_B} \bmod 19$		$11^{10} \bmod 19 = 7^{X_A} \bmod 19$		$7^{X_A} \bmod 19 = 11^{X_B} \bmod 19$	
$K_s = 11$		$K_s = 11$			

It should not be possible for user A or user B to compute each other's private key using all available information. For this, the values of n and g should be very large. If the values of n and g are small then it is possible for A and B to find out each other's private key. In this case, it is also possible for the cryptanalyst to find out fake shared secret key by guessing his private and public keys. Then he tries for A's and B's public key.

8.3.2 Security

Diffie–Hellman algorithm is secure for large values of n and g. For small values, it is possible for the attackers to find out the X_A and X_B using discrete logarithm. So, they can derive the shared secret key once they know the private key of the users. But for large values of n and g it is very difficult to obtain the private keys of the user. To calculate a large prime n, a Sophie Germain prime number m is used. It is called a *safe prime* as in that case the factors of n is only 2 and m. g is selected as the order of m instead of n so that it is difficult to compute X_A from g^{X_A}.

After computing the secret key, there is no need of X_A and X_B. So, the private keys of the users are destroyed once the secret key is obtained. This provides forward security. But man-in-the-middle attack is possible against Diffie–Hellman algorithm. We will discuss it in the next section.

8.3.3 Man-in-the-Middle Attack

Man-in-the-middle attack is also known as *bucket brigade attack*. Suppose the two users A and B are in communication with each other. They want to exchange their

public keys. Let Y_A and Y_B are their public keys. User A receives the public key Y_B indirectly. Suppose the public key of user B is computed by the attacker and send it to user A. There is no way for him to know whether Y_B sends by B or somebody else. User A calculates the shared secret key k_s from Y_B. Now, encrypt the message B uses the k_s and start communication. The attacker captures the encrypted message in between and able to decrypt it as instead of user B, the attacker's key is used by user A to compute the shared secret key.

Let us take an example. Suppose Ajay wants to communicate with Seema. Assume n and g are publicly known keys.

A writes a letter and send it through a courier as "Dear Seema, I would like to meet you in the garden tomorrow, 456, A. (Here 456 is treated as public key). Suppose there is Henry (an attacker) who works for the courier. Henry picks his own private key Z_H, and computes $Y_H = gZ_H \bmod n$, slightly modifies the letter by replacing the number 456 by 1234, prints the new version of the letter, and sends it to B. Later, B replies, "I will meet you tomorrow. My magic number is 14035, B." Henry again modifies the letter replaces 14035 by his own number and sends it again to A. Now, A computes the key using 456 and uses it for communication with A, and computes the key using 14035 and uses it for communication with B.

In this example, man-in-the-middle attack is established by Henry. He uses two distinct keys, one with "Seema" and the other with "Ajay", and then tries to masquerade as "Seema" to "Ajay" and/or vice-versa, perhaps by decrypting and re-encrypting messages passed between them.

8.3.4 Authentication

Authentication of the two users has not taken place in Diffie–Hellman algorithm. So, it is suffered by man-in-the-middle attack. To avoid this attack some mechanism should be used to authenticate the users to each other. For this, any authentication algorithm can be used with Diffie–Hellman algorithm. One of the solutions is to use digital signature during transmitting the public key.

8.4 ELLIPTIC CURVE ARITHMETIC

Victor Millar first time proposed the elliptic curve cryptography (ECC) in 1985. Also Neal Koblitz proposed the Elliptic curve cryptography in the same year. ECC is standardised in 1990 and used for commercial purpose. It is more secure even for a small key. Due to small key size, the performance of Elliptic curve cryptography is better as compared to other algorithms. There is no sub-exponential algorithm due to which the cryptanalyst can break it. For selecting a small prime numbers or smaller finite fields, greater degree of security can be achieved. This saves hardware implementations for the algorithm. Elliptic curve cryptography uses discrete logarithms which provide more challenging task for finding keys of equivalent length. Today ECC is used in ad-hoc wireless networks and mobile networks.

Elliptic curve groups are formed by using elliptic curves. A group is a set of similar elements with some user-defined arithmetic operations on these elements. For elliptic