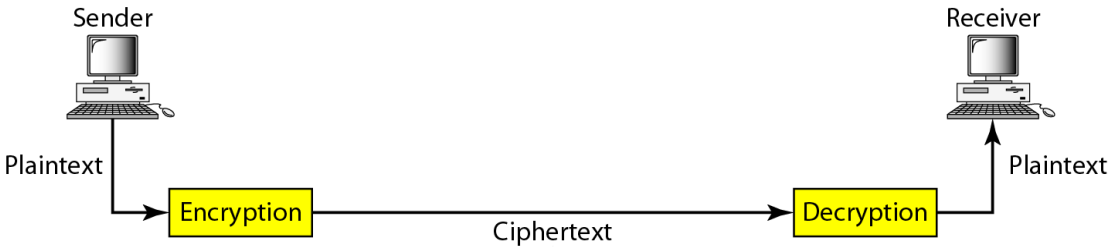
Cryptography

Chapter Index

5.1 Introduction to Basic Encryption and Decryption:

The term 'Cryptography' means the concept of encryption and decryption together. Cryptography is the technique in which the original 'plain text' message is 'encrypted' i.e. converted into a coded form called 'cipher text' at the sender's end, which is then transmitted to the receiver. The receiver then 'decrypts' i.e. converts the 'cipher text' back into the 'plain text' to get the original message back.



Cryptography is also called as an art or technique to achieve secure communication between the communicating parties by encoding the messages between them such that no third party can gain anything useful out of interception.

Various techniques are utilized for this purpose of cryptography. Broadly these techniques fall into two categories.

1) Symmetric key cryptography: - in which the 'key' element used, is the 'same' for both encryption as well as decryption and
2) Asymmetric key cryptography - in which the 'key' element used, is different for both encryption as well as decryption.

    - Symmetric key cryptography is also known as 'private or secret key cryptography' (*please refer to section 5.5 of these notes for details*) whereas

    - Asymmetric key cryptography is also known as 'public key cryptography', (*please refer to section 5.6 of these notes for details*)

The techniques used in symmetric key cryptography are as below.

Substitution technique - the very basic technique, which makes use of simple letter substitution to generate cipher text.

Specific methods used in this type include

1. Caesar cipher (used by Julius Caesar),
2. Modified Caesar Cipher,
3. Mono-alphabetic cipher,
4. Homophonic substitution cipher,
5. Polygram substitution cipher
6. Polyalphabetic cipher etc.

Now let us study them (Substitution Technique) one by one:

1. Caesar Cipher

A cryptographic scheme proposed by Julius Caesar is one special case of substitutional cipher where each alphabet is the message is replaced by an alphabet, three places down the line, in the alphabetical order.

Thus “A” becomes “D” and “B” becomes “E”

Plain text	A	B	C	D	E	F	G	H	I	J	K	L	M	N
Cipher Text	D	E	F	G	H	I	J	K	L	M	N	O	P	Q

Plain text	O	P	Q	R	S	T	U	V	W	X	Y	Z
Cipher Text	R	S	T	U	V	W	X	Y	Z	A	B	C

Caesar Cipher is very simple. But this simplicity comes with a cost. Obviously it is a very weak scheme.

Algorithm to break Caesar cipher

1. Read each alphabet in the cipher text message, and search for it in the second row of the figure above
2. When a match is found, replace that alphabet in the cipher text message with the corresponding alphabet in the same column but the first row of the table (e.g. if the alphabet in cipher text is J, replace it with G).
3. Repeat the process for all alphabets in the cipher text message.

The process shown above will reveal the original plain text. Thus, given a cipher text message L ORYH BRX, it is easy to work backwards and obtain the plain text I LOVE YOU as shown below.

Cipher text		L	O	R	Y	H	B	R	X
Plain text		I	L	O	V	E	Y	O	U

Caesar Cipher is good in theory, but not so good in practice.

Let Ke be the encryption key and Kd be the decryption key. Here we have assumed that the value of Ke = 3 and thus Kd would also be 3,

Let us now try and complicate the Caesar Cipher to make an attacker's life difficult.

2. Modified Version of Caesar Cipher

How can we generalize Caesar Cipher a bit more? Let us assume that the cipher text alphabets corresponding to the original plain text alphabets may not necessarily be three places down the order, but instead, can be any places down the order. This can complicate matters a bit.

Thus, we are now saying that an alphabet A in plain text would not necessarily be replaced by D. It can be replaced by any valid alphabet, i.e. by E or by F or by G, and so on. Once the replacement scheme is decided, it would be constant and will be used for all other alphabets in that message. As we know, the English language contains 26 alphabets. Thus, an alphabet A can be replaced by any other alphaber in the English alphabet set, (i.e. B through Z).

Of course, it does not make sense to replace an alphabet by itself (i.e. replacing A with A). Thus, for each alphabet, we have 25 possibilities of replacement. Hence, to break a message in the modified version of Caesar Cipher, our earlier algorithm would not work.

Let us write a new algorithm to break this version of Caesar Cipher, as shown:

1. Let k be a number equal to 1.
2. Read the complete cipher text message.
3. Replace each alphabet in the cipher text message with an alphabet that is k positions down the order.
4. Increment k by 1.
5. If k is less than 26, then go to step 2. Otherwise, stop the process.
   6. The original text message corresponding to the cipher text message is one of the 25 possibilities produced by the above steps.

We write down all the 25 possibilities and try to make sense. Whichever makes some sense we keep and the other 24 are rejected. Trying out all possibilities is called Brute-Force Attack.

3. Mono-alphabetic Cipher

The major weakness of the Caesar Cipher is its predictability. Once we decide to replace an alphabet in a plain text message with an alphabet that is k positions up or down the order, we replace all other alphabets in the plain text message with the same technique. Thus, the cryptanalyst has to tryout a maximum of 25 possible attacks, and she is assured of a success.

Now imagine that rather than using a uniform scheme for all the alphabets in a given plain text message, we decide to use random substitution. This means that in a given plain text message, each A can be replaced by any other alphabet (B through Z), each B can also be replaced by any other random alphabet (A or C through Z), and so on. The crucial difference being, there is no relation between the replacement of B and replacement of A. That is, if we have decided to replace each A with D, we need not necessarily replace each B with E-we can replace each B with any other character I

To put it mathematically, we can now have any permutation or combination of the 26 alphabets, which means (26 x 25 x 24 x 23 x ... 2) or 4 x 1026 possibilities I This is extremely hard to crack. It might actually take years to tryout these many combinations even with the most modern computers.

4. Homophonic Substitution Cipher:

The Homophonic Substitution Cipher is very similar to Mono Alphabetic Cipher. In a plain substitution cipher technique, we replace one alphabet with another, but in this scheme, the difference is that instead of having a fixed substitution, We can, choose the alphabet from a set. So in this technique, A can be replaced by D,H,P,R; B can be replaced by E,I,Q,S etc.

Homophonic Substitution Cipher also involved substitution of one plain text character with a Cipher Text character at a time. However the cipher text character can be any one of the chosen set.

5. Polygram Substitution Cipher.

In Polygram Substitution Cipher technique, rather than replacing one plain text alphabet with one cipher text alphabet at a time, a block of alphabets is replaced with another block. For instance, HELLO could be replaced with YUQQW, but HELL could be replaced by a totally different cipher text block TEUL

6. Poly-alphabetic Substitution Cipher.

This cipher uses multiple one character keys. Each of the keys encrypts one plain text character. The first key encrypts the first plain text character; the second key encrypts the second plain text character, and so on. After al the keys are used, they are recycled. Thus if we have 30 one letter keys, every 30th character in the plain text would be replaced with the same key. This number is called as the period of the cipher.

In some cases, the mono alphabetic cipher technique is used round after round over already converted plain text and its cipher text. The more number of rounds, the more complex the cipher becomes.

Transposition technique - Modified version of substitution technique because this not only substitutes letters but also makes some sort of permutation over the plain text in order to generate cipher text. Specific examples include

1. Rail fence technique
2. Simple columnar transposition
3. Simple columnar transposition with multiple rounds
4. Vemam cipher,
5. Book cipher etc.

Now let us study them (Transposition Technique) one by one:

1. Rail Fence Technique:

It uses a simple algorithm as:

1. Write down the plain text message as a sequence of diagonals.
2. Read the plain text written in step 1 as a sequence of rows.

Example: Original Plain text message: " Come home tomorrow"

1. After we arrange the plain text diagonally, it would like as follows:

C		M		H		M		T		M		R		O	
	O		E		O		E		O		O		R		W

2. Now read the text row by row, write it sequentially. Thus we have:

C-M-H-M-T-M-R-O-O-E-O-E-O-O-R-W

2. Simple Columnar Transposition Technique:

Basic Technique:

The idea is to:

a. Write the plain text message row by row in a rectangle of a pre-defined size.
b. Read the message column-by column, however, it need not be in the order of columns 1,2,3 etc. It can be any random order such as 2,1,3 etc.

c. The message thus obtained is the cipher text message.

Original Plain text Message: Secrets have to be kept.

1. Let us consider a rectangle with S columns. Therefore, when we write the message into the rectangle row by row it would look as follows:

Column 1	Column 2	Column 3	Column 4	Column 5	Column 6
C	O	M	E	H	O
M	E	T	O	M	O
R	R	O	W		

2. Now read the text in the order of the columns. 4,6,1,2,5,3
3. The cipher text thus obtained is:

E-O-W-O-O-C-M-R-O-E-R-H-M-M-T-O

3. Simple columnar transposition technique with multiple rounds:

Here, the basic Simple columnar technique is repeated for multiple rounds. The more number of rounds, the more complex the cipher becomes. Hence, it is more difficult to crack.

The Basic algorithm:

1. Write the plain text message row-by-row in a rectangle of a pre-determined size
2. Read the message column by column in a random sequence
3. The message thus obtained as the cipher text message of round 1
4. Use this output as a plain text for the next step

5. Vemam Cipher (One-Time Pad)

The Vemam Cipher, also called as One-Time Pad, is implemented using a random set of non-repeating characters as the input cipher text. The most significant point her is that once an input cipher text for transposition is used; it is never used again for any other message (hence the name one-time). The length of the cipher text is equal to the length of the original plain text.

Since, it is used as one-time pad and is discarded after a single use, this technique is highly secure and suitable for small plain text message, but is impractical for large messages.

6. Book Cipher / Running Block Key Cipher:

The idea used is quite simple and similar in principle to Vernam Cipher. For producing cipher text, some portion of text from a book is used, which serves the purpose of a one-time pad. This, the characters from a book are used as one time pad, and they are added to the input plain text messages.

Every process of encryption and decryption is necessarily associated with a 'key'- the combination used for encryption and/or decryption, and an algorithm i.e. the rules or steps used for both encryption and decryption. The requirement of 'same' key as in case of 'symmetric' key cryptography leads to a common problem called 'problem of key distribution', i.e. how the two parties should agree upon a 'common' key that has to be used for the process. This is as described below.

Problem of Key distribution in Symmetric Key cryptography:

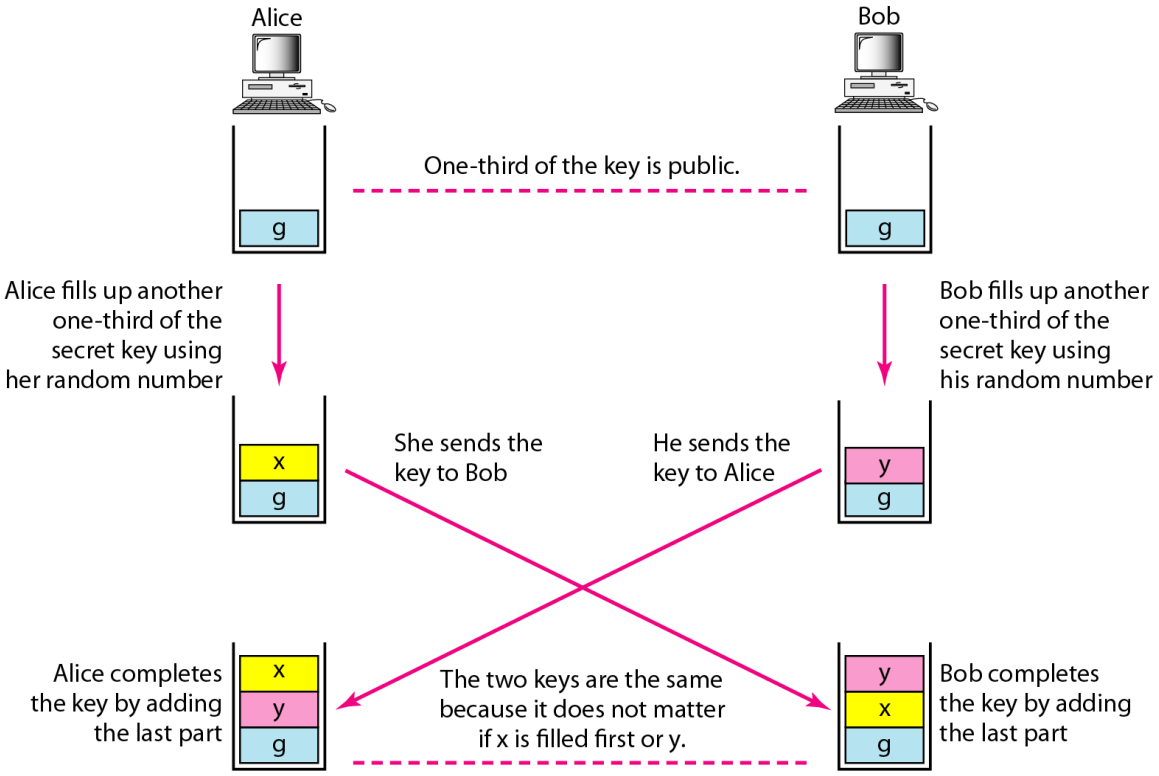
As in case of symmetric key cryptography, the key that has to be used for both encryption and decryption should be the 'same' this leads to a problem that how the two parties requiring secure communication can 'agree' or 'decide' upon a common key, without letting any third person know about it? There can be many ways in which the two parties will try to communicate assuming it is secure, but it may not be so. e.g. even if they exchange letters, seal envelopes into locked boxes, talk over open media for the common key, or send the key along with the locked boxes, whatever may be the means used, it turns out to be practically non-viable or difficult to implement.

That is to say, there are very much chances of intercepting the communication between two parties if any of these methods are used. This is called the 'problem of key distribution'.

In order to come out of this problem, one good solution was given by two scientists jointly known as 'Diffie-Hellman key exchange algorithm'.

5.2 The Diffie-Hellman Key exchange algorithm:

Whitefield Diffie and Martin Hellman, in 1976 have come out with a good solution to the problem of key distribution as mentioned above. The steps of this algorithm are as given below. (It must be noted, that this is NOT an encryption or decryption algorithm but is only used for agreeing upon a. symmetric key. Once it is done, some specific algorithm should be used for the purpose of encryption/decryption.)



Steps for algorithm:

Assume two parties viz. 'first' and 'second' want to communicate securely.

1. Let 'first' and 'second' agree upon two large prime nos., say n and g. These need not be kept secured. (i.e. everyone can know these values.)
2. 'First' chooses another large random no. say x to calculate another number A such that, A = g^xmod n. (Note, value of x is only known to 'first'!)
3. This no. A is then sent by 'first' to 'second'.
4. 'Second' also chooses another large random no. say y to calculate another number B such that,

5. B = g^y mod n. (Note, value of y is only known to 'second'!)
6. This no. B is then sent by 'second' to 'first'.
7. Now, independently, 'first' calculates the key KI as: KI = B^xmod n
8. Also, 'second' independently calculates the key K2 as: K2 = A^y mod n
9. As it should be required here in symmetric key cryptography, KI = K2.

Example:

Let us take an actual example, to illustrate above algorithm.
Assuming values such as n= 11, g=7, x=3 and y=6,
we have following equations:

1. Value of A=7^3mod 11 =343 mod 11 =2.
2. Value of B = 7^6mod 11 = 117649 mod 11 = 4.
3. Key KI = 4^3mod 11 = 64 mod 11 = 9.
4. And, Key K2 = 2^6mod 11 = 64 mod 11 = 9.
5. Thus, we find that KI = K2.
6. Hence the algorithm is proved.

Problems with the algorithm:

Although, it is seen that this algorithm turns out to be a good solution to the above mentioned key distribution problem, still it does not solve all the problems! This is because the algorithm can fail if a hacker makes what is called as the man-in-the-middle attack. This way, even though the two parties will feel that they are talking to each other, practically they are in-turn communicating with the hacker as he places himself in between them and switches back and forth the communication.

The second problem is regarding the no. of keys required. In our example, we have just seen the situation of only two communicating parties. What would be the situation if a third party say 'third' is added!

One must think of the situation when communication between first-second, second-third as well as third-first must be secure! This would obviously require three keys! Then assume how many keys would be required to securely communicate between 1000 people that to independently?

To find out this answer, one formula is used. It says, the total no. of keys required to securely communicate between 'n' individuals is = n (n-l) / 2. Hence in our example for 1000 people, 1000(999)/2 = 499500 keys would be needed. This certainly increases the complications further.

In order to recover from these problems, the second technique (mentioned in the beginning) comes into picture, i.e. the Asymmetric Key cryptography. This states that two types of keys would be required, one each for encryption and decryption.

5.3 The concept of Public key and Private key:

The Asymmetric key cryptography is also known as a 'public key cryptography', which uses a key-pair rather than a single key. The importance of this scheme is that only one key-pair is required to securely communicate between any number of other parties. (unlike the huge no. of keys that we've seen with earlier method.) Hence, one problem is overcome right away. One of these two keys is called public key (which can be announced to the world) and another is private key (obviously to be kept with oneself). This is to be followed by everyone who wants to communicate securely.

The working of public and private kevs:

Asymmetric key cryptography (using public and private keys) works as under:
Suppose, X wants to send a message to Y without having to worry about its security.

1. Then X and Y should each have a private key and a public key.
   - **X should keep its private key secret.**
   - **Y should keep its private key secret**.
   - X should inform Y about its public key.
   - Y should inform X about its public key

   (Both now have their own set of keys ready.)

2. When X wants to send message to Y, X encrypts with Y's public key (as it is known to everyone)
3. X then sends this message to Y.
4. Then, Y decrypts this message using his own private key (known only to Y)

   [This ensures in this case, that the message can be encrypted & sent by anyone, but can only be decrypted by Y. Hence, any interception will not result in knowing the sensitive information as key is only with Y.]
   Similarly, on the other side, if Y wants to send the message to X, reverse method is performed.

5. Y encrypts the message using X's public key and sends this to X
6. On receiving the message, X can further decrypt it using his own private key.

The basis of this working lies in the assumption of large prime number with only two factors. If one of the factors is used for encryption process, only the other factor shall be used for decryption. The **best example** of an asymmetric key cryptography algorithm is the famous **RSA algorithm** (developed by **Rivest, Shamir and Adleman** at MIT in 1978, based on the framework setup by Diffie & Hellman earlier).

What would happen if your private key were made public????

The answer is in just one word! –

Get Bankrupted!
– However rich you were! Now popper!!

The receiver of your private key can, not only withdraw all that you have but also can also avail credit for banks and enjoy and you keep paying throughout your life!

5.4 The concept of Hash (Message Digest):

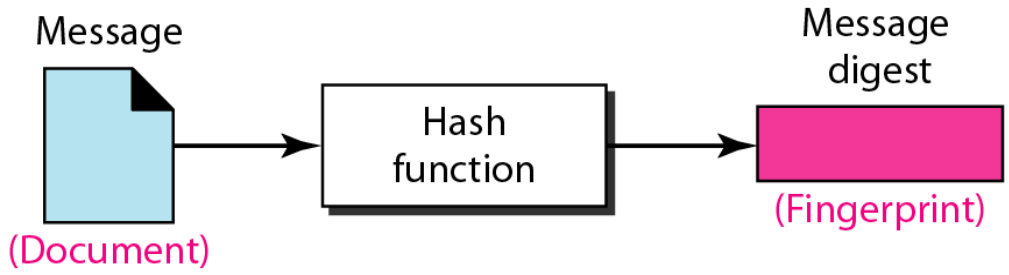
Signing the Digest

We said before that public-key encryption is efficient if the message is short.

Using a public key to sign the entire message is very inefficient if the message is very long.

The solution is to let the sender sign a digest of the document instead of the whole document. The sender creates a miniature version or digest of the document and signs it; the receiver then checks the signature on the miniature.

To create a digest of the message, we use a hash function. The hash function creates a fixed-size digest from a variable-length message, as shown in Figure



The two most common hash functions are called MD5 (Message Digest 5) and SHA-I (Secure Hash Algorithm I). The first one produces a 120-bit digest. The second produces a 160-bit digest.

Note that a hash function must have two properties to guarantee its success.

First, hashing is one-way; the digest can only be created from the message, not vice versa.

Second, hashing is a one-to-one function; there is little probability that two messages will create the same digest. We will see the reason for this condition shortly.

After the digest has been created, it is encrypted (signed) using the sender's private key. The encrypted digest is attached to the original message and sent to the receiver.

Idea of a Message Digest.

The concept of message digests is based on similar principles. However, it is slightly wider in scope. For instance, suppose that we have a number 4000 and we divide it by 4 to get 1000 Thus, 4 can become a fingerprint of the number 4000. Dividing 4000 by 4 will always yield 1000. If we change either 4000 or 4, the result will not be 1000.

Another important point is, if we are simply given the number 4, but are not given any further information, we would not be able to trace back the equation 4 x 1000 = 4000. Thus, we have one more important concept here. The fingerprint of a message (in this case, the number 4) does not tell anything about the original message (in this case, the number 4000). This is because there are infinite other possible equations, which can produce the result 4.

Another simple example of message digest is shown in fig. Let us assume that we want to calculate the message digest of a number 7391753. Then, we multiply each digit in the number with the next digit (excluding it if it is 0), and disregarding the first digits of the multiplication operation, if the result is a two-digit number.

Thus, we perform a hashing operation (or a message digest algorithm) over a block of data to produce its hash or message digest, which is smaller in size than the original message. This concept is shown in fig.

Actually, the message digests are not so small and straightforward to compute. Message digests usually consist of 128 or more bits. This means that the chance of any two-message digests being the same is anything between 0 and at least 2128. The message digest length is chosen to be so long with a purpose. This minimizes that the scope for two messages digests being the same.

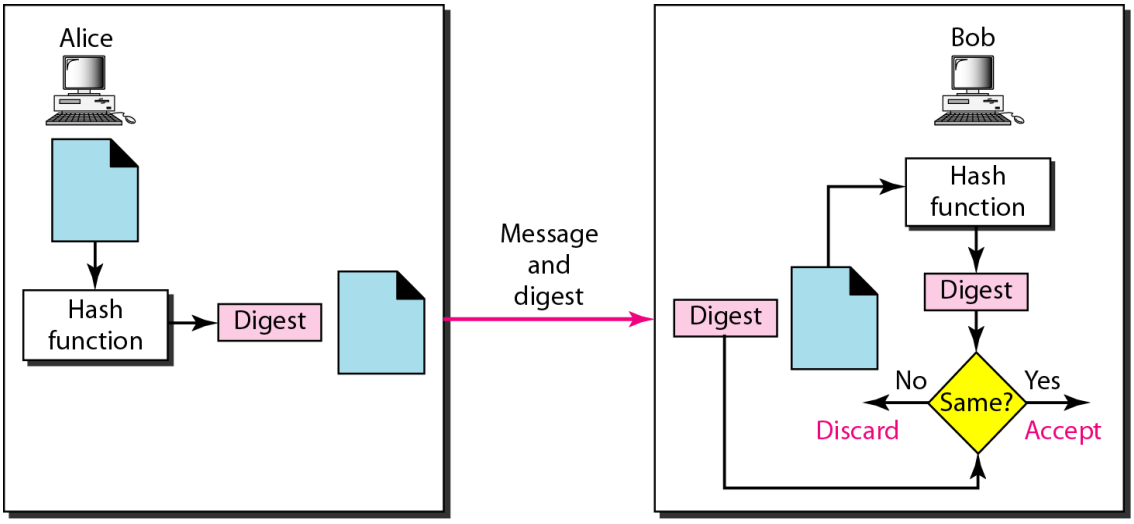
Requirement of a message digest

We can summarize the requirements of the message digest concept, as follows:

- Given a message, it should be very easy to find its corresponding message digest. Also for a given message, the message digest must always be the same.
- Given a message digest, it should be very difficult to find the original message for which the digest was created.
- Given any two messages, if we calculate their message digests, the two message digests must be different.

Another basis of message digest is that it should not give any clue or indication of the original message. i.e. it should not be possible to revert back to original message from the digest. Also, for a given message it's digest should be the same always.

Different algorithms are used to convert original message into its message digest. The popularly used ones are MD5 or Message Digest 5 (developed by Rivest) a modified version of earlier MD4, MD3 and MD2, while the first one was simply MD, and the SHA (Secure Hash Algorithm) developed by National Institute of Standards and Technology (NISI) in 1993. SHA-l is promoted & prominently used than the MD5 algorithm.



5.5 Digital Signatures:

In earlier discussion of Asymmetric key cryptography, we had considered the only situation, in which if X is sender & Y receiver, then X encrypts the message with Y's public key and on receiving, Y decrypts with his own private key. This method only ensures secure communication between the two. Now consider another situation. If X is sender and Y is receiver, X encrypts the message using his own private key! On receiving, Y decrypts it using X's public key. The purpose behind this move is 'authentication'. It is clear that, only X knows his private key.

So, when Y receives this message (encrypted with X's private key), it is an indication or proof that it has originated only from X and none else! Remember that in earlier scheme, the purpose was only 'confidentiality' and the origin of message was not the concern.

Now, one may say that if someone else wants to intercept this communication it should be easy. i.e. anyone can decrypt the message who knows X's public key. This is true, but then it will not be possible for anyone to again encrypt this message as only X knows his private key. Thus receiver here will not be fooled that message came from X This scheme confirms the origin of the message. So, in this case X cannot deny that he has sent the message to Y, because it was encrypted with X's private key, known only to X

The above discussion forms the basis for the concept called ‘ Digital Signature’’ In case of our normal operations, we make use of our (handwritten) signatures. These are used to confirm the 'origin' or the 'authentication' of the individual. In the Internet world, it would be difficult to use any such method in practice. Hence the concept of 'Digital signatures' was evolved.

This technique is vitally important in the E-commerce concept used in the Internet. It proves as a valid mechanism for 'authenticity' of individual. Most of the financial transactions done over Internet make use of this method.
Techniques of Digital signatures:

Actual working of Digital signatures involves the use of a concept called 'Message digest' or 'hash'. Message digest is something like the summary of original message. (works similar to the CRC checksum concept) This is basically used to verify the 'integrity' of data i.e. to ensure that the message has not been modified after it was sent by sender and before it reaches the receiver.

The Digital Signature Standard (DSS) was developed by NIST first in 1991. It suggests using the SHA-1 algorithm for calculating the message digest. This digest is further used for performing Digital signatures, by using the algorithm called Digital Signature Algorithm (DSA). In DSA, message digest is encrypted with the sender's private key to form the Digital Signature (DS). This signature is

transmitted further along with the original message. It is also possible to use the earlier RSA algorithm for performing digital signatures. RSA is prominently used over DSA as DSA turns out to be more complicated.

Steps for the process:

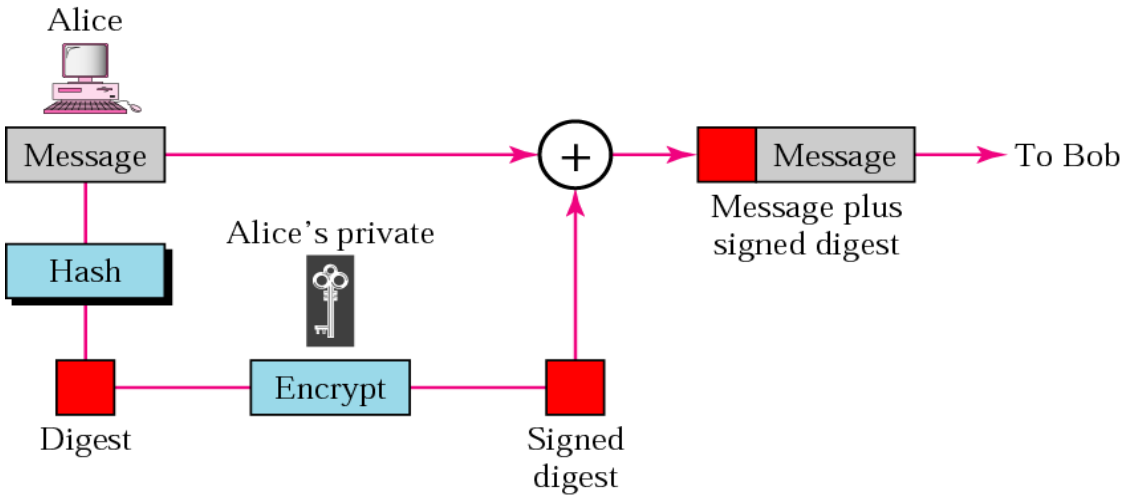
Sender's Side:

1. If X is the sender, the SHA-1 algorithm is used to first calculate the message digest (MD 1) of original message.
2. This MD1 is further encrypted using RSA with X's private key. This output is called the Digital Signature (DS) of X.
3. Further, the original message (M) along with the Digital signature (DS) is sent to receiver.

Receiver's Side:

4. Y thus receives the original message (M) and X's digital signature. Y uses the same message digest algorithm used by X to calculate the message digest (MD2) of received message (M).
5. Also, Y uses X's public key to decrypt the digital signature. The outcome of this decryption is nothing but original message digest (MD1) calculated by X.
6. Y, then compares this digest MD1 with the digest MD2 he has just calculated in step 4. If both of them are matching, i.e. MDl = MD2, Y can accept the original message (M) as correctly authenticated and assured to have originated from X. whereas, if they are different, the message shall be rejected.

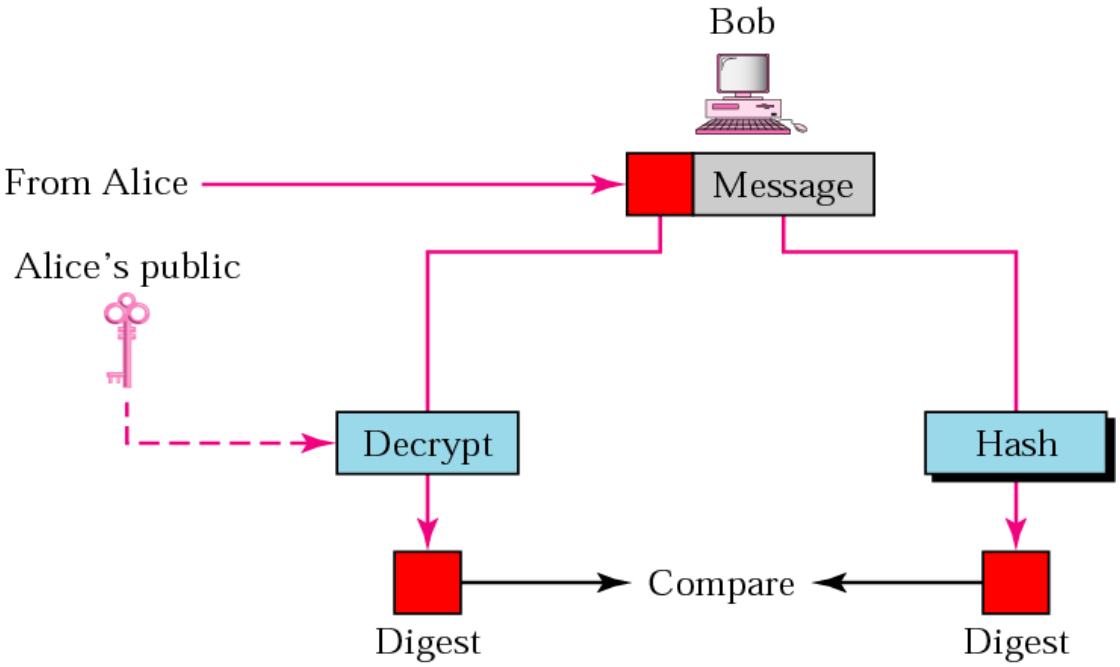
This method turns out to be foolproof. Even if an attacker intercepts anywhere in between, it is not likely for him to again sign the modified/read message, as only X in this case will know the private key! Hence, even if intercepted, this method remains very much secure and reliable!



The Sender's Side

Modus Operandi – Digital Signature:

After the digest has been created, it is encrypted (signed) using the sender's private key. The encrypted digest is attached to the original message and sent to the receiver. Figure (on previous page) shows the sender site.



The Receiver's Side

The receiver receives the original message and the encrypted digest. He separates the two. He applies the same hash function to the message to create a second digest. He also decrypts the received digest, using the public key of the sender. If the two digests are the same, all three security measures are preserved. Figure 30.7 shows the receiver site.

Properties of Digital Signature:

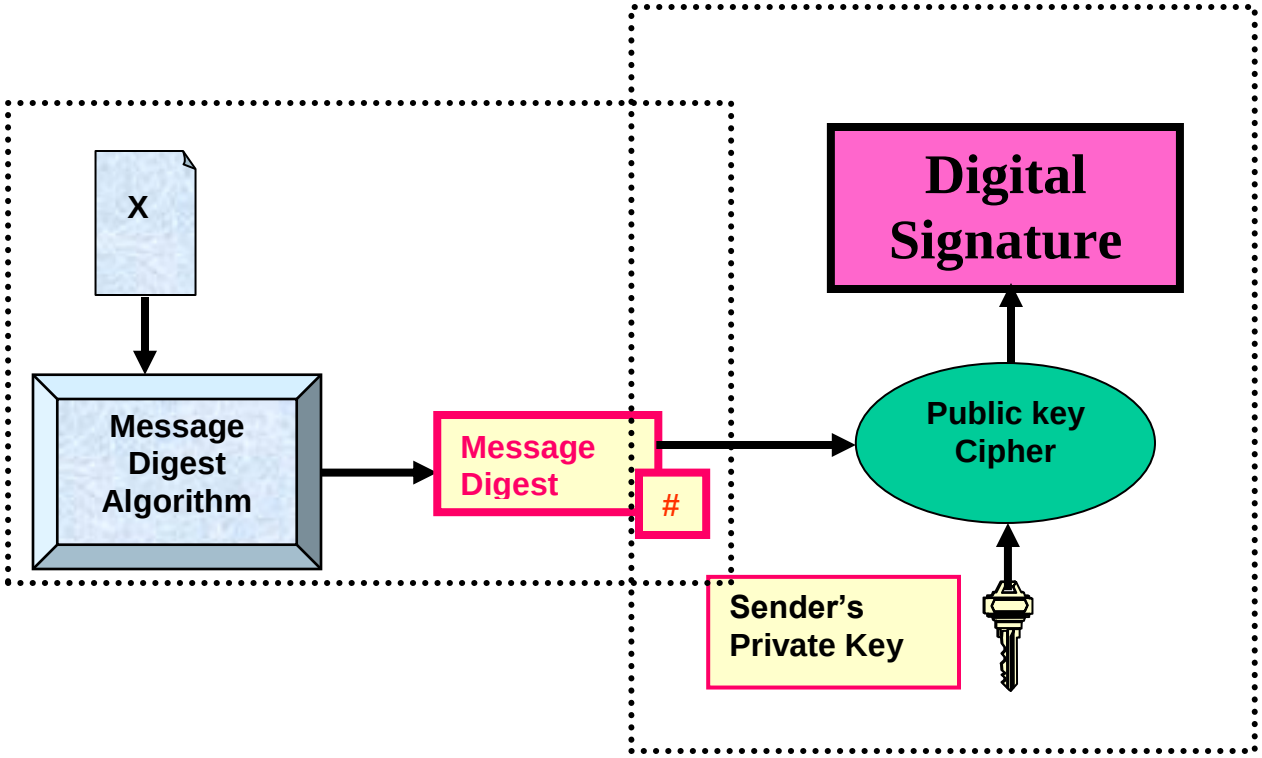
- Digital signature does not provide privacy. If there is a need for privacy, another layer of encryption/decryption must be applied.

- **Digital signatures can provide**
  1. Integrity,
  2. Authentication, and
  3. Nonrepudiation.

1. Integrity The integrity of a message is preserved because if Eve intercepted the message and partially or totally changed it, the decrypted message would be unreadable.

2. Authentication We can use the following reasoning to show how a message can be authenticated. If Eve sends a message while pretending that it is coming from Alice, she must use her own private key for encryption. The message is then decrypted with the public key of Alice and will therefore be nonreadable. Encryption with Eve's private key and decryption with Alice's public key result in garbage.

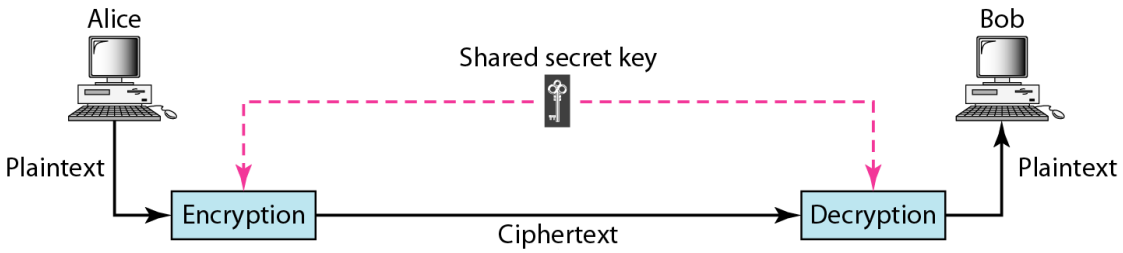
3. Nonrepudiation Digital signature also provides for nonrepudiation. Bob saves the message received from Alice. If Alice later denies sending the message, Bob can show that encrypting and decrypting the saved message with Alice's private and public key can create a duplicate of the saved message. Since only Alice knows her private key, she cannot deny sending the message.



5. 5 SYMMETRIC- KEY CRYPTOGRAPHY

We can divide all the cryptography algorithms in the world into two groups: symmetric-key (sometimes called secret-key) cryptography algorithms and public-key (sometimes called asymmetric) cryptography algorithms.

In symmetric-key cryptography, the same key is used by both parties. The sender uses this key and an encryption algorithm to encrypt data; the receiver uses the same key and the corresponding decryption algorithm to decrypt the data



In symmetric-key cryptography, the same key is used by the sender (for encryption) and the receiver (for decryption). The key is shared.

In symmetric-key cryptography, the algorithm used for decryption is the inverse of the algorithm used for encryption. This means that if the encryption algorithm uses a combination of addition and multiplication, the decryption algorithm uses a combination of division and subtraction.

Note that the symmetric-key cryptography algorithms are so named because the same key can be used in both directions.

In symmetric-key cryptography, the same key is used in both directions.

Symmetric-key algorithms are efficient; it takes less time to encrypt a message using a symmetric-key algorithm than it takes to encrypt using a public-key algorithm. The reason is that the key is usually smaller. For this reason, symmetric-key algorithms are used to encrypt and decrypt long messages.

Symmetric-key cryptography is often used for long messages.

Disadvantages of symmetric key:

A symmetric-key algorithm has two major disadvantages.

1. Each pair of users must have a unique symmetric key.
This means that if N people in the world want to use this method, there needs to be N(N - 1)/2 symmetric keys.

For example, for 1 thousand people to communicate, 1000 * 999 /2 = 4,99,500 (4 lakhs 99 thousand and five hundred symmetric keys are needed. The distribution of the keys between two parties can be difficult.

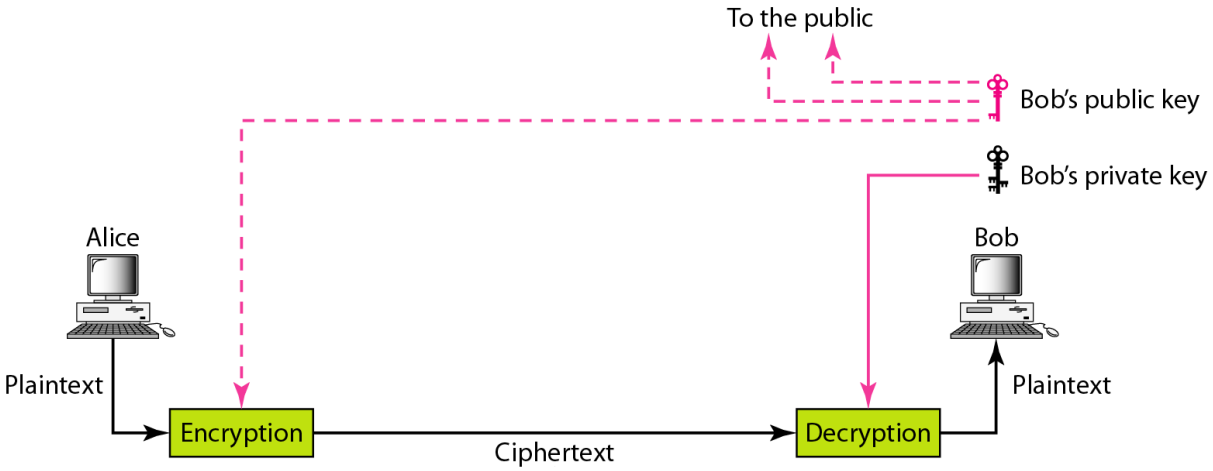
2. The sender needs to exchange the key to the receiver. It may be hijacked in between!

---

5. 6 Asymmetric Key Cryptography:

In public-key cryptography, there are two keys: a private key and a public key. The private key is kept by the receiver. The public key is announced to the public.

Imagine Alice, as shown in Figure 29.20, wants to send a message to Bob. Alice uses the public key to encrypt the message. When the message is received by Bob, the private key is used to decrypt the message.



In public-key encryption/decryption, the public key that is used for encryption is different from the private key that is used for decryption.

The public key is available to the public; the private key is available only to an individual.

Public-key encryption/decryption has two advantages.

First, it removes the restriction of a shared symmetric key between two entities (e.g., persons) who need to communicate with each other. A shared symmetric key is shared by the two parties and cannot be used when one of them wants to communicate with a third party. In public-key encryption! decryption, each entity creates a pair of keys; the private one is kept, and the public one is distributed. Each entity is independent, and the pair of keys created can be used to communicate with any other entity.

The second advantage is that the number of keys needed is reduced tremendously.

In this system, for I thousand users to communicate, only 1 thousand pairs of keys ie 2000 keys are needed, not 4,99,500, as was the case in symmetric-key cryptography.

Public-key cryptography also has two disadvantages.

The big disadvantage is the complexity of the algorithm. If we want the

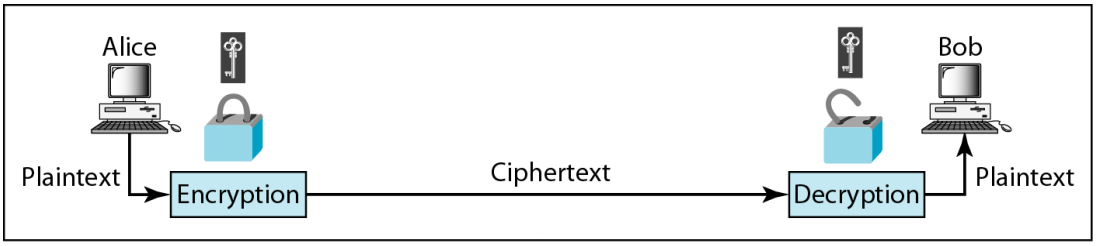
method to be effective, the algorithm needs large numbers. Calculating the ciphertext from plaintext using the long keys takes a lot of time. That is the main reason that public-key cryptography is not recommended for large amounts of text.

Public-key algorithms are more efficient for short messages.

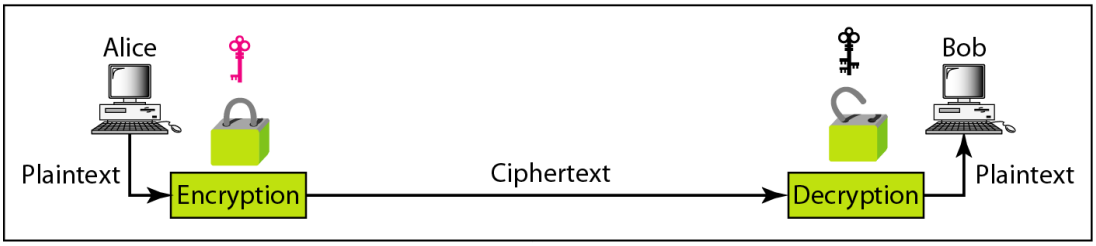
The second disadvantage of the public-key method is that the association between an entity and its public key must be verified. If Alice sends her public key via an email to Bob, then Bob must be sure that the public key really belongs to Alice and nobody else.

One point needs to re-mentioned that if your private key were made public you would Get Bankrupted in no time!

5.7 Compare and contrast between Symmetric Key Cryptography and Asymmetric Key Cryptography:



a. Symmetric-key cryptography



b. Asymmetric-key cryptography

S. No.	Characteristic	Symmetric Key Cryptography	Asymmetric Key Cryptography
1	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
2		Ke = Kd	Kd ≠ Kd
3	Speed of encryption/decryption	Very fast	Slower
4	Size of resulting encrypted text	Usually same as or less than the original clear text size	More than the original clear text size
5	Key agreement / exchange	A big problem	No problem at all
6	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

7	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)
8	Efficiency in usage	Symmetric key cryptography is often used for long messages	Public key algorithm are more efficient for short messages

The above table shows that both symmetric key cryptography and asymmetric key cryptography have nice features.

Also, both have some areas where better alternatives are generally desired. Asymmetric key cryptography solves the major problem of key agreement / key exchange as well as scalability.

However, it is far slower and produces huge chunks of cipher text as compared to symmetric key Cryptography (essentially because it uses large keys and complex algorithms as compared to symmetric key cryptography).

How nice 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.

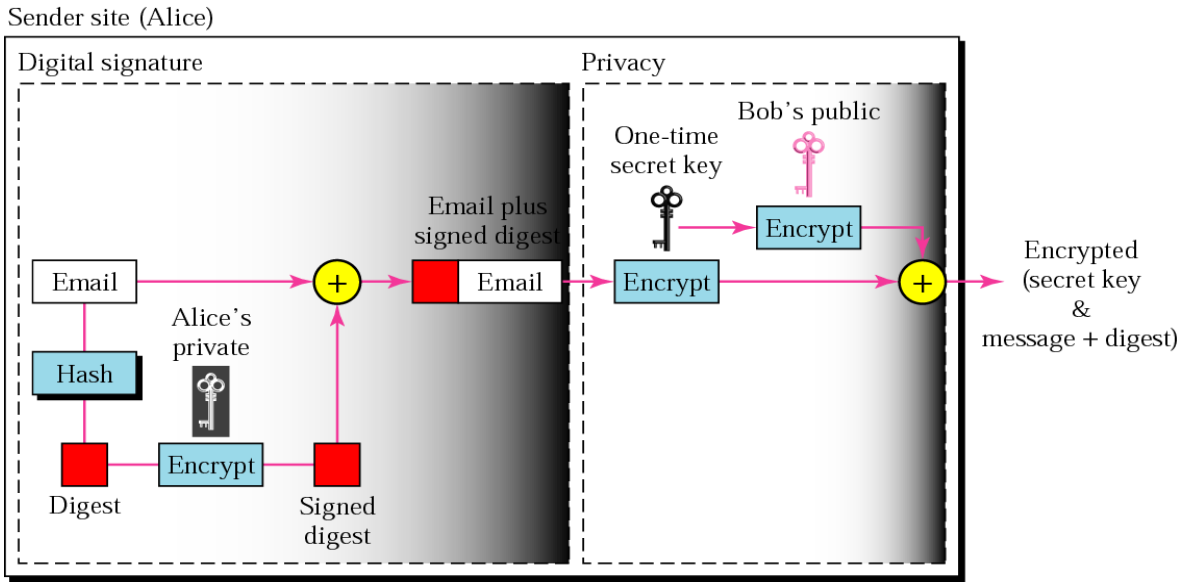
In practice symmetric key cryptography and asymmetric key cryptography are combined to have a very efficient security solutions.

Pretty Good Privacy:

The implementation of security at the application layer is more feasible and simpler, particularly when the Internet communication involves only two parties, as in the case of email and TELNET. The sender and the receiver can agree to use the same protocol and to use any type of security services they desire. In this section, we discuss one protocol used at the application layer to provide security: PGP.

Pretty Good Privacy (PGP) was invented by Phil Zimmermann to provide all four aspects of security (privacy, integrity, authentication, and nonrepudiation) in the sending of email.

PGP uses digital signature (a combination of hashing and public-key encryption) to provide integrity, authentication, and nonrepudiation. It uses a combination of secret-key and public-key encryption to provide privacy. Specifically, it uses one hash function, one secret key, and two private-public key pairs. See Figure below



The figure shows how PGP creates secure email at the sender site. The email message is hashed to create a digest. The digest is encrypted (signed) using Alice's private key. The message and the digest are encrypted using the one-time secret key created by Alice. The secret key is encrypted using Bob's public key and is sent together with the encrypted combination of message and digest.

Figure below shows how PGP uses hashing and a combination of three keys to extract the original message at the receiver site. The combination of encrypted secret key and message plus digest is received. The encrypted secret key first is decrypted (using Bob's private key) to get the one-time secret key created by Alice. The secret key then is used to decrypt the combination of the message plus digest.