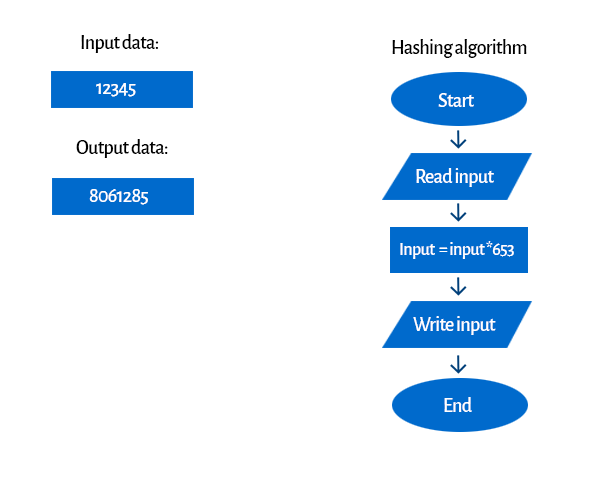
## **Hash**

A hashing algorithm’s goal is to generate a safe hash; but what is a hash?

*A hash is a value computed from a base input number using a hashing function.*

Shortly, the hash value is a summary of the original data. For instance, think of a paper document that you squeeze and squeeze so that, in the end, you aren’t even able to read the content. It’s almost (in theory) impossible to restore the original input without knowing what was the starting data.

Let’s take an example of a hashing algorithm:



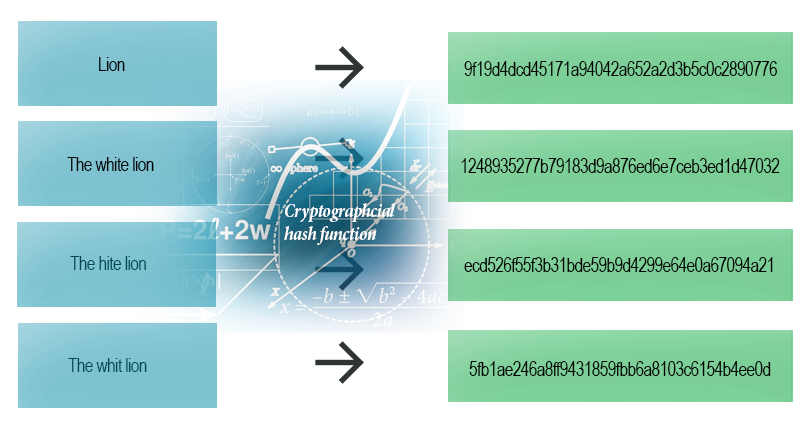
We could discuss if it’s a secure algorithm. We can help you — it isn’t. Of course, every input number is individual (we’ll talk more about this in the further sections), but it’s easy to guess how it works. But that just shows the idea.

**Hashing Algorithm**

A hashing algorithm is a cryptographic hash function. It is a mathematical algorithm that maps data of arbitrary size to a hash of a fixed size. It’s designed to be a one-way function, infeasible to invert. However, nowadays several hashing algorithms are being compromised. This happened to MD5, for example — a widely known hash function designed to be a cryptographic hash function, which is now so easy to reverse — that we could only use for verifying data against unintentional corruption.

It’s easy to figure out what the ideal cryptographic hash function should be like:

1. It should be fast to compute the hash value for any kind of data
2. It should be impossible to regenerate a message from its hash value (brute force attack as the only option)
3. It should avoid hash collisions, each message has its own hash.
4. Every change to a message, even the smallest one, should change the hash value. It should be completely different. It’s called the [avalanche effect](https://en.wikipedia.org/wiki/Avalanche_effect" \t "https://blog.jscrambler.com/hashing-algorithms/_blank)



Even the smallest change (one letter) makes the whole hash different (SHA-1 example)

## **What we use it for?**

Cryptographic hash functions are used notably in IT. We can use them for digital signatures, message authentication codes (MACs), and other forms of authentication. We can also use them for indexing data in hash tables, for fingerprinting, identifying files, detecting duplicates or as checksums (we can detect if a sent file didn’t suffer accidental or intentional data corruption). We’ll show you an example of the last feature.

## ***Message Digest***

A message digest is a cryptographic hash function containing a string of digits created by a one-way hashing formula.

Message digests are designed to protect the integrity of a piece of data or media to detect changes and alterations to any part of a message. They are a type of cryptography utilizing hash values that can warn the copyright owner of any modifications applied to their work.

Message digest hash numbers represent specific files containing the protected works. One message digest is assigned to particular data content. It can reference a change made deliberately or accidentally, but it prompts the owner to identify the modification as well as the individual(s) making the change. Message digests are algorithmic numbers.

This term is also known as a hash value and sometimes as a checksum.

# **MD4 Hash**

MD4 is a message digest algorithm (the fourth in a series) designed by Professor Ronald Rivest of MIT in 1990. It implements a cryptographic hash function for use in message integrity checks. The digest length is 128 bits. The algorithm has influenced later designs, such as the MD5, SHA and RIPEMD algorithms. MD4 is also used to compute NT-hash password digests on Microsoft Windows NT, XP and Vista.

MD4 was designed to be fast, which meant taking a few risks regarding security. By 1992 weaknesses had been found which led Rivest to produce a strengthened, but slower, version known as MD5. In 1998, Dobbertin [[1, 2]](http://practicalcryptography.com/hashes/md4-hash/" \l "references) found the first MD4 collisions, and he gave an algorithm for generating such collisions, with a work factor that is approximately equal to the computation of 2^20 MD4 hashes.

The MD4 hash should not be used for any cryptographic purposes.

## **The Algorithm**

The MD4 algorithm is described by Rivest in [RFC 1320](http://www.ietf.org/rfc/rfc1320.txt), along with an efficient implementation (in C).

MD4 operates on 32-bit words. Let M be the message to be hashed. The message M is padded so that its length (in bits) is equal to 448 modulo 512, that is, the padded message is 64 bits less than a multiple of 512. The padding consists of a single 1 bit, followed by enough zeros to pad the message to the required length. Padding is always used, even if the length of M happens to equal 448 mod 512. As a result, there is at least one bit of padding, and at most 512 bits of padding. Then the length (in bits) of the message (before padding) is appended as a 64-bit block.

The padded message is a multiple of 512 bits and, therefore, it is also a multiple of 32 bits. Let M be the message and N the number of 32-bit words in the (padded) message. Due to the padding, N is a multiple of 16.

A four-word buffer (A,B,C,D) is used to compute the message digest. Here each of A, B, C, D is a 32-bit register. These registers are initialized to the following values in hexadecimal:

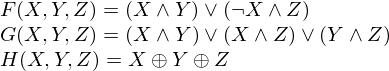
word A: 01 23 45 67

word B: 89 ab cd ef

word C: fe dc ba 98

word D: 76 54 32 10

We first define three auxiliary functions that each take as input three 32-bit words and produce as output one 32-bit word.



where IMG_257 is logical and, IMG_258 is logical or and IMG_259 is logical xor. Do the following:

/\* Process each 16-word block. \*/

For i = 0 to N/16-1 do

/\* Copy block i into X. \*/

For j = 0 to 15 do

Set X[j] to M[i\*16+j].

end /\* of loop on j \*/

/\* Save A as AA, B as BB, C as CC, and D as DD. \*/

AA = A

BB = B

CC = C

DD = D

/\* Round 1. \*/

/\* Let [abcd k s] denote the operation:

a = (a + F(b,c,d) + X[k]) <<< s. \*/

/\* Do the following 16 operations. \*/

[ABCD 0 3] [DABC 1 7] [CDAB 2 11] [BCDA 3 19]

[ABCD 4 3] [DABC 5 7] [CDAB 6 11] [BCDA 7 19]

[ABCD 8 3] [DABC 9 7] [CDAB 10 11] [BCDA 11 19]

[ABCD 12 3] [DABC 13 7] [CDAB 14 11] [BCDA 15 19]

/\* Round 2. \*/

/\* Let [abcd k s] denote the operation:

a = (a + G(b,c,d) + X[k] + 5A827999) <<< s. \*/

/\* Do the following 16 operations. \*/

[ABCD 0 3] [DABC 4 5] [CDAB 8 9] [BCDA 12 13]

[ABCD 1 3] [DABC 5 5] [CDAB 9 9] [BCDA 13 13]

[ABCD 2 3] [DABC 6 5] [CDAB 10 9] [BCDA 14 13]

[ABCD 3 3] [DABC 7 5] [CDAB 11 9] [BCDA 15 13]

/\* Round 3. \*/

/\* Let [abcd k s] denote the operation:

a = (a + H(b,c,d) + X[k] + 6ED9EBA1) <<< s. \*/

/\* Do the following 16 operations. \*/

[ABCD 0 3] [DABC 8 9] [CDAB 4 11] [BCDA 12 15]

[ABCD 2 3] [DABC 10 9] [CDAB 6 11] [BCDA 14 15]

[ABCD 1 3] [DABC 9 9] [CDAB 5 11] [BCDA 13 15]

[ABCD 3 3] [DABC 11 9] [CDAB 7 11] [BCDA 15 15]

/\* Then perform the following additions. (That is, increment each

of the four registers by the value it had before this block

was started.) \*/

A = A + AA

B = B + BB

C = C + CC

D = D + DD

end /\* of loop on i \*/

# **MD5 Hash**

In cryptography, MD5 (Message-Digest algorithm 5) is a widely used cryptographic hash function with a 128-bit hash value. As an Internet standard ([RFC 1321](http://www.ietf.org/rfc/rfc1321.txt)), MD5 has been employed in a wide variety of security applications, and is also commonly used to check the integrity of files. An MD5 hash is typically expressed as a 32 digit hexadecimal number.

MD5 is a strengthened version of [MD4](http://practicalcryptography.com/hashes/Cryptographic-category/md4/). Like MD4, the MD5 hash was invented by Professor Ronald Rivest of MIT. Also, MD5 was obviously used as the model for [SHA-1](http://practicalcryptography.com/hashes/Cryptographic-category/sha-1/), since they share many common features. MD5 and SHA-1 are the two most widely used hash algorithms today, but use of MD5 will certainly decline over time, since it is now considered broken [[2,3,4]](http://practicalcryptography.com/hashes/md5-hash/" \l "references).

The MD5 hash should not be used for cryptographic purposes.

## **The Algorithm**

The MD5 hash is described in [RFC 1321](http://www.ietf.org/rfc/rfc1321.txt) along with a C implementation. MD5 is similar to the [MD4](http://practicalcryptography.com/hashes/Cryptographic-category/md4/) hash. The padding and initialisation is identical.

MD5 operates on 32-bit words. Let M be the message to be hashed. The message M is padded so that its length (in bits) is equal to 448 modulo 512, that is, the padded message is 64 bits less than a multiple of 512. The padding consists of a single 1 bit, followed by enough zeros to pad the message to the required length. Padding is always used, even if the length of M happens to equal 448 mod 512. As a result, there is at least one bit of padding, and at most 512 bits of padding. Then the length (in bits) of the message (before padding) is appended as a 64-bit block.

The padded message is a multiple of 512 bits and, therefore, it is also a multiple of 32 bits. Let M be the message and N the number of 32-bit words in the (padded) message. Due to the padding, N is a multiple of 16.

A four-word buffer (A,B,C,D) is used to compute the message digest. Here each of A, B, C, D is a 32-bit register. These registers are initialized to the following values in hexadecimal:

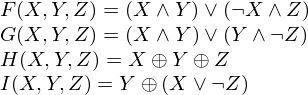
word A: 01 23 45 67

word B: 89 ab cd ef

word C: fe dc ba 98

word D: 76 54 32 10

We first define four auxiliary functions that each take as input three 32-bit words and produce as output one 32-bit word.



where IMG_257 is logical and, IMG_258 is logical or and IMG_259 is logical xor. Do the following:

/\* Process each 16-word block. \*/

For i = 0 to N/16-1 do

/\* Copy block i into X. \*/

For j = 0 to 15 do

Set X[j] to M[i\*16+j].

end /\* of loop on j \*/

/\* Save A as AA, B as BB, C as CC, and D as DD. \*/

AA = A

BB = B

CC = C

DD = D

/\* Round 1. \*/

/\* Let [abcd k s i] denote the operation

a = b + ((a + F(b,c,d) + X[k] + T[i]) <<< s). \*/

/\* Do the following 16 operations. \*/

[ABCD 0 7 1] [DABC 1 12 2] [CDAB 2 17 3] [BCDA 3 22 4]

[ABCD 4 7 5] [DABC 5 12 6] [CDAB 6 17 7] [BCDA 7 22 8]

[ABCD 8 7 9] [DABC 9 12 10] [CDAB 10 17 11] [BCDA 11 22 12]

[ABCD 12 7 13] [DABC 13 12 14] [CDAB 14 17 15] [BCDA 15 22 16]

/\* Round 2. \*/

/\* Let [abcd k s i] denote the operation

a = b + ((a + G(b,c,d) + X[k] + T[i]) <<< s). \*/

/\* Do the following 16 operations. \*/

[ABCD 1 5 17] [DABC 6 9 18] [CDAB 11 14 19] [BCDA 0 20 20]

[ABCD 5 5 21] [DABC 10 9 22] [CDAB 15 14 23] [BCDA 4 20 24]

[ABCD 9 5 25] [DABC 14 9 26] [CDAB 3 14 27] [BCDA 8 20 28]

[ABCD 13 5 29] [DABC 2 9 30] [CDAB 7 14 31] [BCDA 12 20 32]

/\* Round 3. \*/

/\* Let [abcd k s t] denote the operation

a = b + ((a + H(b,c,d) + X[k] + T[i]) <<< s). \*/

/\* Do the following 16 operations. \*/

[ABCD 5 4 33] [DABC 8 11 34] [CDAB 11 16 35] [BCDA 14 23 36]

[ABCD 1 4 37] [DABC 4 11 38] [CDAB 7 16 39] [BCDA 10 23 40]

[ABCD 13 4 41] [DABC 0 11 42] [CDAB 3 16 43] [BCDA 6 23 44]

[ABCD 9 4 45] [DABC 12 11 46] [CDAB 15 16 47] [BCDA 2 23 48]

/\* Round 4. \*/

/\* Let [abcd k s t] denote the operation

a = b + ((a + I(b,c,d) + X[k] + T[i]) <<< s). \*/

/\* Do the following 16 operations. \*/

[ABCD 0 6 49] [DABC 7 10 50] [CDAB 14 15 51] [BCDA 5 21 52]

[ABCD 12 6 53] [DABC 3 10 54] [CDAB 10 15 55] [BCDA 1 21 56]

[ABCD 8 6 57] [DABC 15 10 58] [CDAB 6 15 59] [BCDA 13 21 60]

[ABCD 4 6 61] [DABC 11 10 62] [CDAB 2 15 63] [BCDA 9 21 64]

/\* Then perform the following additions. (That is increment each

of the four registers by the value it had before this block

was started.) \*/

A = A + AA

B = B + BB

C = C + CC

D = D + DD

end /\* of loop on i \*/

The algorithm above uses a set o 64 constants T[i] for i = 1 to 64. Let T[i] denote the i-th element of the table, which is equal to the integer part of 4294967296 times abs(sin(i)), where i is in radians. The elements of the table are given in the appendix of [RFC 1321](http://www.ietf.org/rfc/rfc1321.txt).

## **Differences between MD5 and MD4**

The significant differences between MD4 and MD5 are the following:

1. MD5 has four rounds, whereas MD4 has only three. Consequently, the MD5 compression function includes 64 steps, whereas the MD4 compression function has 48 steps.
2. Each step of MD5 has a unique additive constant, T[i], whereas each round of MD4 uses a fixed constant
3. The function G in the second round of MD5 is less symmetric than the G function in MD4
4. Each step of MD5 adds the result of the previous step, which is not the case with MD4. The stated purpose of this modification is to produce a faster avalanche effect.
5. In MD5, the order in which input words are accessed in the second and third rounds is less similar to each other than is the case in MD4.
6. [RFC 1321](http://www.ietf.org/rfc/rfc1321.txt) claims that "the shift amounts in each round have been approximately optimized, to yield a faster ‘avalanche effect’." Also, the shifts employed in each round of MD5 are distinct, which is not the case in MD4.

MD5 Hash:-

The MD5 function is a cryptographic algorithm that takes an input of arbitrary length and produces a message digest that is 128 bits long. The digest is sometimes also called the "hash" or "fingerprint" of the input. MD5 is used in many situations where a potentially long message needs to be processed and/or compared quickly. The most common application is the creation and verification of [digital signatures](https://www.iusmentis.com/technology/digitalsignatures/" \o "Signatures have been used for centuries to prove the authenticity of a  document. In the digital domain, tampering with a document is very easy and leaves no traces. This makes it essential to have a...).

MD5 was designed by well-known cryptographer Ronald Rivest in 1991. In 2004, some serious flaws were found in MD5. The complete implications of these flaws has yet to be determined.

## How MD5 works

## Preparing the input

## The MD5 algorithm first divides the input in ****blocks**** of 512 bits each. 64 Bits are inserted at the end of the last block. These 64 bits are used to record the length of the original input. If the last block is less than 512 bits, some extra bits are 'padded' to the end.

Next, each **block** is divided into 16 **words** of 32 bits each. These are denoted as M0 ... M15.

### MD5 helper functions

#### The buffer

MD5 uses a buffer that is made up of four **words** that are each 32 bits long. These words are called A, B, C and D. They are initialized as

word A: 01 23 45 67

word B: 89 ab cd ef

word C: fe dc ba 98

word D: 76 54 32 10

#### The table

MD5 further uses a table K that has 64 elements. Element number i is indicated as Ki. The table is computed beforehand to speed up the computations. The elements are computed using the mathematical sin function:

Ki = abs(sin(i + 1)) \* 232

#### Four auxiliary functions

In addition MD5 uses four auxiliary functions that each take as input three 32-bit words and produce as output one 32-bit word. They apply the logical operators and, or, not and xor to the input bits.

F(X,Y,Z) = (X and Y) or (not(X) and Z)

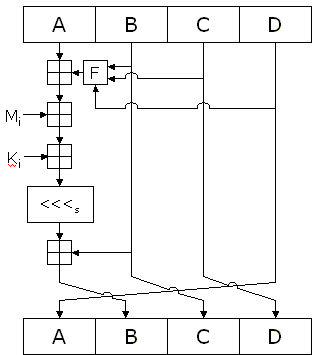
G(X,Y,Z) = (X and Z) or (Y and not(Z))

H(X,Y,Z) = X xor Y xor Z

I(X,Y,Z) = Y xor (X or not(Z))

### Processing the blocks

The contents of the four buffers (A, B, C and D) are now mixed with the words of the input, using the four auxiliary functions (F, G, H and I). There are four rounds, each involves 16 basic operations. One operation is illustrated in the figure below.



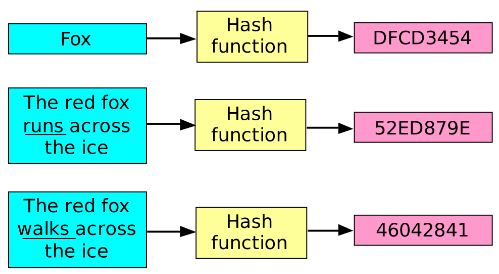
The figure shows how the auxiliary function F is applied to the four buffers (A, B, C and D), using message word Mi and constant Ki. The item "<<<s" denotes a binary left shift by s bits.

### The output

After all rounds have been performed, the buffers A, B, C and D contain the MD5 digest of the original input.

Sha:-****Secure Hash Algorithms****, also known as SHA, are a family of [cryptographic](https://brilliant.org/wiki/cryptography/" \o "cryptographic" \t "https://brilliant.org/wiki/secure-hashing-algorithms/_blank) functions designed to keep data secured. It works by transforming the data using a [hash function](https://brilliant.org/wiki/hash-function/" \o "hash function" \t "https://brilliant.org/wiki/secure-hashing-algorithms/_blank): an algorithm that consists of [bitwise operations](https://brilliant.org/wiki/bitwise-operations/" \o "bitwise operations" \t "https://brilliant.org/wiki/secure-hashing-algorithms/_blank), [modular additions](https://brilliant.org/wiki/modular-additions/?wiki_title=modular additions" \o "page not yet created" \t "https://brilliant.org/wiki/secure-hashing-algorithms/_blank), and [compression functions](https://brilliant.org/wiki/compression-functions/" \o "compression functions" \t "https://brilliant.org/wiki/secure-hashing-algorithms/_blank). The hash function then produces a fixed-size string that looks nothing like the original. These algorithms are designed to be [one-way functions](https://brilliant.org/wiki/one-way-functions/?wiki_title=one-way functions" \o "page not yet created" \t "https://brilliant.org/wiki/secure-hashing-algorithms/_blank), meaning that once they’re transformed into their respective hash values, it’s virtually impossible to transform them back into the original data. A few algorithms of interest are SHA-1, SHA-2, and SHA-3, each of which was successively designed with increasingly stronger encryption in response to hacker attacks. SHA-0, for instance, is now obsolete due to the widely exposed vulnerabilities.

A common application of SHA is to encrypting passwords, as the server side only needs to keep track of a specific user’s hash value, rather than the actual password. This is helpful in case an attacker hacks the database, as they will only find the hashed functions and not the actual passwords, so if they were to input the hashed value as a password, the hash function will convert it into another string and subsequently deny access. Additionally, SHAs exhibit the avalanche effect, where the modification of very few letters being encrypted causes a big change in output; or conversely, drastically different strings produce similar hash values. This effect causes hash values to not give any information regarding the input string, such as its original length. In addition, SHAs are also used to detect the tampering of data by attackers, where if a text file is slightly changed and barely noticeable, the modified file’s hash value will be different than the original file’s hash value, and the tampering will be rather noticeable.

*A small tweak in the original data produces a drastically different encrypted output. This is called the [avalanche effect](https://brilliant.org/wiki/avalanche-effect/" \o " avalanche effect" \t "https://brilliant.org/wiki/secure-hashing-algorithms/_blank) [[1]](https://brilliant.org/wiki/secure-hashing-algorithms/" \l "citation-1) .*

## SHA Characteristics

Cryptographic hash functions are utilized in order to keep data secured by providing three fundamental safety characteristics: pre-image resistance, second pre-image resistance, and collision resistance.

The cornerstone of cryptographic security lies in the provision of ****pre-image resistance****, which makes it hard and time-consuming for an attacker to find an original message, m,*m*, given the respective hash value, h\_m*hm*​. This security is provided by the nature of [one-way functions](https://brilliant.org/wiki/one-way-function/" \o "one-way functions" \t "https://brilliant.org/wiki/secure-hashing-algorithms/_blank), which is a key component of SHA. Pre-image resistance is necessary to ward off brute force attacks from powerful machines.

****One-way Function****

Alice and Bob are pen pals who share their thoughts via mail. When Alice visited Bob, she gave him a phone book of her city. In order to keep their messages safe from intruders, Alice tells Bob that she will encrypt the message. She tells Bob that he will find a bunch of numbers on every letter, and each sequence of numbers represents a phone number. Bob’s job is to find the phone number in the book and write down the first letter of the person’s last name. With this function, Bob is to decrypt the entire message.

To decrypt the message, Bob has to read the entire phone book to find all the numbers on the letter, whereas Alice can quickly find the letters and their respective phone numbers in order to encrypt her message. For this reason, before Bob is able to decrypt the message by hand, Alice can re-hash the message and keep the data secure. This makes Alice’s algorithm a one-way function[[2]](https://brilliant.org/wiki/secure-hashing-algorithms/" \l "citation-2).

The second safety characteristic is called ****second pre-image resistance****, granted by SHA when a message is known, m\_1*m*1​, yet it’s hard to find another message, m\_2*m*2​, that hashes to the same value: H\_{m\_1} = H\_{m\_2}*Hm*1​​=*Hm*2​​. Without this characteristic, two different passwords would yield the same hash value, deeming the original password unnecessary in order to access secured data.

The last safety characteristic is **[collision resistance](https://brilliant.org/wiki/collision-resistance/?wiki_title=collision resistance" \o "page not yet created" \t "https://brilliant.org/wiki/secure-hashing-algorithms/_blank)**, which is provided by algorithms that make it extremely hard for an attacker to find two completely different messages that hash to the same hash value: H\_{m\_1} = H\_{m\_2}*Hm*1​​=*Hm*2​​. In order to provide this characteristic, there must be a similar number of possible inputs to possible outputs, as more inputs than outputs, by the [pigeonhole principle](https://brilliant.org/wiki/pigeonhole-principle/" \o "pigeonhole principle" \t "https://brilliant.org/wiki/secure-hashing-algorithms/_blank), will definitively incur potential collisions. For this reason, collision resistance is necessary, as it implies that finding two inputs that hash to the same hash value is extremely difficult. Without collision resistance, digital signatures can be compromised as finding two messages that produce the same hash value may make users believe two documents were signed by two different people when one person was able to produce a different document with the same hash value.

Recent cryptographic functions have stronger security characteristics to block off recently developed techniques such as [length extension attacks](https://brilliant.org/wiki/length-extension-attacks/?wiki_title=length extension attacks" \o "page not yet created" \t "https://brilliant.org/wiki/secure-hashing-algorithms/_blank), where given a hash value, hash(m)*hash*(*m*), and the length of the original message, m*m*, an attacker can find a message, m’*m*’, and calculate the hash value of the concatenation of the original message and the new message: hash\ (m||m’)*hash* (*m*∣∣*m*’).

As a general guideline, a hash function should be as seemingly random as possible while still being [deterministic](https://brilliant.org/wiki/deterministic-functions/?wiki_title=deterministic" \o "page not yet created" \t "https://brilliant.org/wiki/secure-hashing-algorithms/_blank) and fast to compute.

## SHA-1

****Secure Hash Algorithm 1****, or SHA-1, was developed in 1993 by the U.S. government's standards agency National Institute of Standards and Technology (NIST). It is widely used in security applications and protocols, including [TLS](https://brilliant.org/wiki/tls/?wiki_title=TLS" \o "page not yet created" \t "https://brilliant.org/wiki/secure-hashing-algorithms/_blank), [SSL](https://brilliant.org/wiki/ssl/" \o "SSL" \t "https://brilliant.org/wiki/secure-hashing-algorithms/_blank), [PGP](https://brilliant.org/wiki/pgp/?wiki_title=PGP" \o "page not yet created" \t "https://brilliant.org/wiki/secure-hashing-algorithms/_blank), [SSH](https://brilliant.org/wiki/ssh/?wiki_title=SSH" \o "page not yet created" \t "https://brilliant.org/wiki/secure-hashing-algorithms/_blank), [IPsec](https://brilliant.org/wiki/ipsec/?wiki_title=IPsec" \o "page not yet created" \t "https://brilliant.org/wiki/secure-hashing-algorithms/_blank), and [S/MIME](https://brilliant.org/wiki/smime/?wiki_title=S/MIME" \o "page not yet created" \t "https://brilliant.org/wiki/secure-hashing-algorithms/_blank).

SHA-1 works by feeding a message as a [bit string](https://brilliant.org/wiki/binary-numbers/" \o "bit string" \t "https://brilliant.org/wiki/secure-hashing-algorithms/_blank) of length less than 2^{64}264 bits, and producing a 160-bit hash value known as a *message digest*. Note that the message below is represented in [hexadecimal](https://brilliant.org/wiki/hexadecimal-numbers/" \o "hexadecimal" \t "https://brilliant.org/wiki/secure-hashing-algorithms/_blank) notation for compactness.

There are two methods to encrypt messages using SHA-1. Although one of the methods saves the processing of sixty-four 32-bit words, it is more complex and time-consuming to execute, so the simple method is shown in the example below. At the end of the execution, the algorithm outputs blocks of 16 words, where each word is made up of 16 bits, for a total of 256 bits.

****Pseudocode****

Suppose the message ‘abc’ were to be encoded using SHA-1, with the message ‘abc’ in binary being

01100001\ 01100010\ 0110001101100001 01100010 01100011

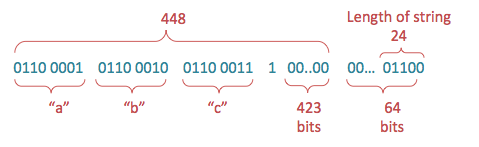
and that in hex being

616263.616263.

****1)**** The first step is to initialize five random strings of hex characters that will serve as part of the hash function (shown in hex):

\begin{aligned} H\_0 &= 67DE2A01\\ H\_1 &= BB03E28C\\ H\_2 &= 011EF1DC \\ H\_3 &= 9293E9E2\\ H\_4 &= CDEF23A9. \end{aligned}*H*0​*H*1​*H*2​*H*3​*H*4​​=67*DE*2*A*01=*BB*03*E*28*C*=011*EF*1*DC*=9293*E*9*E*2=*CDEF*23*A*9.​

****2)**** The message is then [padded](https://brilliant.org/wiki/padding/" \o "padded" \t "https://brilliant.org/wiki/secure-hashing-algorithms/_blank) by appending a 1, followed by enough 0s until the message is 448 bits. The length of the message represented by 64 bits is then added to the end, producing a message that is 512 bits long:

*Padding of string "abc" in bits, finalized by the length of the string, which is 24 bits.*

****3)**** The padded input obtained above, M*M*, is then divided into 512-bit chunks, and each chunk is further divided into sixteen 32-bit words, W\_0 … W\_{15}*W*0​…*W*15​. In the case of ‘abc’, there’s only one chunk, as the message is less than 512-bits total.

****4)**** For each chunk, begin the 80 iterations, i*i*, necessary for hashing (80 is the determined number for SHA-1), and execute the following steps on each chunk, M\_n:*Mn*​:

* For iterations 16 through 79, where 16 \leq i \leq 7916≤*i*≤79, perform the following operation:W(i) = S^1\big(W(i-3) \oplus W(i-8) \oplus W(i-14) \oplus W(i-16)\big),*W*(*i*)=*S*1(*W*(*i*−3)⊕*W*(*i*−8)⊕*W*(*i*−14)⊕*W*(*i*−16)),where XOR, or \oplus⊕, is represented by the following comparison of inputs x*x* and y:*y*:

|  |  |  |
| --- | --- | --- |
| x*x* | y*y* | Output |
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 0 |

* For example, when i*i* is 16, the words chosen are W(13), W(8), W(2), W(0)*W*(13),*W*(8),*W*(2),*W*(0), and the output is a new word, W(16)*W*(16), so performing the XOR, or \oplus⊕, operation on those words will give this:

|  |  |
| --- | --- |
| W(0)*W*(0) | 01100001\ 01100010\ 01100011\ 1000000001100001 01100010 01100011 10000000 |
| W(2)*W*(2) | 00000000\ 00000000\ 00000000\ 0000000000000000 00000000 00000000 00000000 |
| W(8)*W*(8) | 00000000\ 00000000\ 00000000\ 0000000000000000 00000000 00000000 00000000 |
| W(13)*W*(13) | 00000000\ 00000000\ 00000000\ 0000000000000000 00000000 00000000 00000000 |
|  | \oplus⊕ |
| W(16)*W*(16) | 01100001\ 01100010\ 01100011\ 1000000001100001 01100010 01100011 10000000 |

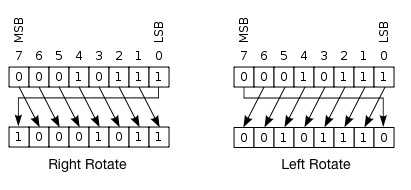
****Circular Shift Operation****

Now, the circular shift operation S^n(X)*Sn*(*X*) on the word X*X* by n*n* bits, n*n* being an integer between 00 and 3232, is defined by

S^n(X) = (X << n)\quad \textbf{OR}\quad (X >> 32-n),*Sn*(*X*)=(*X*<<*n*)**OR**(*X*>>32−*n*),

where X<<n*X*<<*n* is the ****left-shift**** operation, obtained by discarding the leftmost n*n* bits of X*X* and padding the result with n*n* zeroes on the right.

X>> 32-n*X*>>32−*n* is the ****right-shift**** operation obtained by discarding the rightmost n*n* bits of X*X* and padding the result with n*n* zeroes on the left. Thus S^n(X)*Sn*(*X*) is equivalent to a circular shift of X*X* by n*n* positions, and in this case the circular left-shift is used. [[3]](https://brilliant.org/wiki/secure-hashing-algorithms/" \l "citation-3)



So, a left shift S^n\big(W(i)\big),*Sn*(*W*(*i*)), where W(i)*W*(*i*) is 10010,10010, would produce 0100101001, as the rightmost bit 00 is shifted to the left side of the string. Therefore, W(16)*W*(16) would end up being

11000010\ 11000100\ 11000111\ 000000000.11000010 11000100 11000111 000000000.

****5)**** Now, store the hash values defined in step 1 in the following variables:

\begin{aligned} A &= H\_0\\ B &= H\_1\\ C &=H\_2\\ D &=H\_3\\ E &=H\_4. \end{aligned}*ABCDE*​=*H*0​=*H*1​=*H*2​=*H*3​=*H*4​.​

****6)**** For 8080 iterations, where 0 \leq i \leq 790≤*i*≤79, computeTEMP = S^5 \*(A) + f(i; B, C, D) + E + W(i) + K(i).*TEMP*=*S*5∗(*A*)+*f*(*i*;*B*,*C*,*D*)+*E*+*W*(*i*)+*K*(*i*).*See below for details on the logical function, ff, and on the values of K(i).K(i).* Reassign the following variables:

\begin{aligned} E&=D\\ D&=C\\ C&=S^{30}(B)\\ B&=A\\ A&=TEMP. \end{aligned}*EDCBA*​=*D*=*C*=*S*30(*B*)=*A*=*TEMP*.​

****7)**** Store the result of the chunk’s hash to the overall hash value of all chunks, as shown below, and proceed to execute the next chunk:

\begin{aligned} H\_0 &= H\_0 + A\\ H\_1 &= H\_1+B\\ H\_2 &= H\_2+C\\ H\_3 &= H\_3+D\\ H\_4 &= H\_4 + E. \end{aligned}*H*0​*H*1​*H*2​*H*3​*H*4​​=*H*0​+*A*=*H*1​+*B*=*H*2​+*C*=*H*3​+*D*=*H*4​+*E*.​

****8)**** As a final step, when all the chunks have been processed, the message digest is represented as the 160-bit string comprised of the ****OR**** logical operator, \lor∨, of the 5 hashed values:

HH = S^{128}(H\_0)\ \lor\ S^{96}(H\_1)\ \lor\ S^{64}(H\_2)\ \lor\ S^{32}(H\_3)\ \lor\ H\_4.*HH*=*S*128(*H*0​) ∨ *S*96(*H*1​) ∨ *S*64(*H*2​) ∨ *S*32(*H*3​) ∨ *H*4​.

So, the string ‘abc’ becomes represented by a hash value akin to ****a9993e364706816aba3e25717850c26c9cd0d89d****.

If the string changed to ‘abcd’, for instance, the hashed value would be drastically different so attackers cannot tell that it is similar to the original message. The hash value for 'abcd' is ****81fe8bfe87576c3ecb22426f8e57847382917acf****.

****Functions used in the algorithm****

A sequence of logical functions are used in SHA-1, depending on the value of i*i*, where 0 \leq i \leq 790≤*i*≤79, and on three 32-bit words B, C, and D, in order to produce a 32-bit output. The following equations describe the [logical functions](https://brilliant.org/wiki/truth-tables/" \o "logical functions" \t "https://brilliant.org/wiki/secure-hashing-algorithms/_blank), where \neg¬ is the logical *NOT*, \lor∨ is the logical *OR*, \land∧ is the logical *AND*, and \oplus⊕ is the logical XOR:

\begin{aligned} f(i;B,C,D)\ &=\ (B\ \land\ C) \lor \big((\neg B) \land D\big)&&\text{for }\ 0 \geq i \geq 19\\\\ f(i;B,C,D)\ &=\ B\ \oplus\ C\ \oplus\ D&&\text{for } \ 20 \geq i \geq 39\\\\ f(i;B,C,D)\ &=\ (B\ \land\ C)\ \lor\ (B\ \land\ D) \lor\ (C\ \land\ D)&&\text{for } \ 40 \geq i \geq 59\\\\ f(i;B,C,D)\ &=\ B \oplus\ C\ \oplus\ D&&\text{for } \ 60 \geq i\geq 79. \end{aligned}*f*(*i*;*B*,*C*,*D*) *f*(*i*;*B*,*C*,*D*) *f*(*i*;*B*,*C*,*D*) *f*(*i*;*B*,*C*,*D*) ​= (*B* ∧ *C*)∨((¬*B*)∧*D*)= *B* ⊕ *C* ⊕ *D*= (*B* ∧ *C*) ∨ (*B* ∧ *D*)∨ (*C* ∧ *D*)= *B*⊕ *C* ⊕ *D*​​for  0≥*i*≥19for  20≥*i*≥39for  40≥*i*≥59for  60≥*i*≥79.​

Additionally, a sequence of constant words, shown in hex below, is used in the formulas:

\begin{aligned} K(i) &= 5A827999, &&\text{where }\ 0 \leq i \leq 19\\\\ K(i) &= 6ED9EBA1, &&\text{where }\ 20 \leq i \leq 39\\\\ K(i) &= 8F1BBCDC, &&\text{where }\ 40 \leq i \leq 59\\\\ K(i) &= CA62C1D6, &&\text{where }\ 60 \leq i \leq 79. \end{aligned}*K*(*i*)*K*(*i*)*K*(*i*)*K*(*i*)​=5*A*827999,=6*ED*9*EBA*1,=8*F*1*BBCDC*,=*CA*62*C*1*D*6,​​where  0≤*i*≤19where  20≤*i*≤39where  40≤*i*≤59where  60≤*i*≤79.​

Albeit SHA-1 is still widely used, cryptanalysts in 2005 were able to find vulnerabilities on this algorithm that detrimentally compromised its security. These vulnerabilities came in the form of an algorithm that speedily finds [collisions](https://brilliant.org/wiki/collisions/" \o "collisions" \t "https://brilliant.org/wiki/secure-hashing-algorithms/_blank) with different inputs, meaning that two distinct inputs map to the same digest [[4]](https://brilliant.org/wiki/secure-hashing-algorithms/" \l "citation-4).

As of 2010, many organizations have recommended its replacement by SHA-2 or SHA-3. Companies like Microsoft, Google, or Mozilla have announced that their browsers will stop accepting SHA-1 encryption certificates by 2017 [[5]](https://brilliant.org/wiki/secure-hashing-algorithms/" \l "citation-5).

## SHA-2

Due to the exposed vulnerabilities of SHA-1, cryptographers modified the algorithm to produce SHA-2, which consists of not one but two hash functions known as SHA-256 and SHA-512, using 32- and 64-bit words, respectively. There are additional truncated versions of these hash functions, known as SHA-224, SHA-384, SHA-512/224, and SHA-512/256, which can be used for either part of the algorithm.

SHA-1 and SHA-2 differ in several ways; mainly, SHA-2 produces 224- or 256-sized digests, whereas SHA-1 produces a 160-bit digest; SHA-2 can also have block sizes that contain 1024 bits, or 512 bits, like SHA-1.

Brute force attacks on SHA-2 are not as effective as they are against SHA-1. A brute force search for finding a message that corresponds to a given digest of length L*L* using brute force would require 2^L2*L* evaluations, which makes SHA-2 a lot safer against these kinds of attacks.

## Common Attacks

Cryptography wouldn’t be as quickly developed if it weren’t for the attacks that compromise their effectiveness. One of the most common attacks is known as the [primeage attack](https://brilliant.org/wiki/primeage-attack/?wiki_title=primeage attack" \o "page not yet created" \t "https://brilliant.org/wiki/secure-hashing-algorithms/_blank), where pre-computed tables of solutions are used in a brute-force manner in order to crack passwords. The solution against these kinds of attacks is to compose a hash function that would take an attacker an exorbitant amount of resources, such as millions of dollars or decades of work, to find a message corresponding to a given hash value.

Most attacks penetrating SHA-1 are collision attacks, where a non-sensical message produces the same hash value as the original message. Generally, this takes time proportional to 2^{n/2}2*n*/2 to complete, where n*n* is the length of the message. This is the reason the message digests have increased in length from 160-bit digests in SHA-1 to 224- or 256-bit digests in SHA-2 [[6]](https://brilliant.org/wiki/secure-hashing-algorithms/" \l "citation-6).

Other attacks exist that attempt to exploit mathematical properties in order to crack hash functions. Amongst these is the [birthday attack](https://brilliant.org/wiki/birthday-attack/" \o "birthday attack" \t "https://brilliant.org/wiki/secure-hashing-algorithms/_blank), where higher likelihood of collisions are found when using random attacks with a fixed number of letter combinations (see the [pigeonhole principle](https://brilliant.org/wiki/pigeon-hole-principle/?wiki_title=pigeonhole principle" \o "page not yet created" \t "https://brilliant.org/wiki/secure-hashing-algorithms/_blank)), or the [rainbow table](https://brilliant.org/wiki/rainbow-table/?wiki_title=rainbow table" \o "page not yet created" \t "https://brilliant.org/wiki/secure-hashing-algorithms/_blank) attack, where a pre-computed hash table is used to reverse a hash function in order to crack passwords.

## Network Security issues:-

## Problem #1: Unknown Assets on the Network

There are many businesses that don’t have a complete inventory of all of the IT assets that they have tied into their network. This is a massive problem. If you don’t know what all of the assets are on your network, how can you be sure your network is secure?

The easiest fix for this is to conduct a review of all the devices on your network and identify all of the various platforms they run. By doing this, you can know what all of the different access points are on your network and which ones are most in need of security updates.

## Problem #2: Abuse of User Account Privileges

According to data cited by the [Harvard Business Review](https://hbr.org/2016/09/the-biggest-cybersecurity-threats-are-inside-your-company" \t "https://www.compuquip.com/blog/_blank), for the year of 2016, “60% of all attacks were carried out by insiders.” Whether it’s because of honest mistakes (accidentally sending info to the wrong email address or losing a work device), intentional leaks and misuse of account privileges, or identity theft arising from a phishing campaign or other social engineering attack that compromises their user account data, the people inside your business represent one of the biggest security problems you’ll ever face.

Because these threats come from trusted users and systems, they’re also among the hardest to identify and stop.

However, there are ways to minimize your risk in case of an insider attack. For example, if your company uses a policy of least privilege (POLP) when it comes to user access, you can limit the damage that a misused user account can do. In a POLP, every user’s access to the various systems and databases on your network is restricted to just those things that they need to do their jobs.

## Problem #3: Unpatched Security Vulnerabilities

Many businesses are concerned with “zero day” exploits. These exploits are those unknown issues with security in programs and systems that have yet to be used against anyone. However, zero day vulnerabilities aren’t the problem—unpatched known vulnerabilities are the problem.

As noted in one [CSO online article](https://www.csoonline.com/article/3075830/data-protection/zero-days-arent-the-problem-patches-are.html" \t "https://www.compuquip.com/blog/_blank), “around 6,300 unique vulnerabilities appeared in 2015. Symantec says that only 54 of them were classified as zero-days.”

This is because when a “zero day” exploit is used it can be discovered—becoming a known issue that the software vendor can begin working on. The more often the exploit is used, the more likely it is to get discovered and patched. Also, it takes a lot of effort to independently discover a completely unknown vulnerability in a system.

So, attackers generally prefer to stick to known exploits. In fact, as noted in the CSO article, “The Verizon Data Breach Report 2016 revealed that out of all detected exploits, most came from vulnerabilities dating to 2007. Next was 2011.”

In other words, vulnerabilities that were almost a decade old accounted for most of the breaches in 2016. Let that sink in.

The easiest fix for this problem is to maintain a strict schedule for keeping up with security patches. Also, gradually changing the programs and operating systems on your network to make them the same can simplify this process. For example, if every system is Windows-based or Mac-based (rather than a hodgepodge of Mac, Windows, Linux, etc.), then you only have to keep track of Mac OS or Windows OS security patch schedules and alerts.

## Problem #4: A Lack of Defense in Depth

Eventually, despite all of your best efforts, there will be a day where an attacker succeeds in breaching your network security. However, just how much damage this attacker will be capable of depends on how the network is structured.

The problem is that some businesses have an open network structure where once an attacker is in a trusted system, they have unfettered access to all systems on the network.

If the network is structured with strong segmentation to keep all of its discrete parts separate, then it’s possible to slow down the attacker enough to keep them out of vital systems while your security team works to identify, contain, and eliminate the breach.

## Problem #5: Not Enough IT Security Management

Another common issue for many companies is that even when they have all of the best cybersecurity solutions in place, they might not have enough people in place to properly manage those solutions.

When this happens, critical cybersecurity alerts may get missed, and successful attacks may not be eliminated in time to minimize damage.

However, finding a large enough internal IT security team to manage all of your needs can be an expensive and time-consuming process. Qualified professionals are in demand, and they know it.

To build up IT security staff quickly, many businesses use the services of a dedicated partner such as Compuquip Cybersecurity. This allows these businesses to access a full team of experienced cybersecurity professionals for a fraction of the cost of hiring them full-time internally.

Some businesses use these cybersecurity solutions partners to shore up their IT security departments in the short-term while they’re preparing their own internal cybersecurity teams.

**IMPERSONATION**

Impersonation is when a person plays the role of someone you are likely to trust or obey convincingly enough to fool you into allowing access to your office, to information, or to your information systems. This type of social engineering plays on our natural tendencies to believe that people are who they say they are, and to follow instructions when asked by an authority figure. It involves the conscious manipulation of a victim to obtain information without the individual realizing that a security breach is occurring.

Most common impersonation roles fall under the category of someone with authority, which leads us to ingratiation. Most people want to help, so they will go to great lengths to provide the required information (or access) to anyone with authority.

These tricks work because we all regularly interact with people we don’t know. Still, it’s human nature to trust these credentials like badges and IDs that we most likely do not know how to truly verify.

**MESSAGE CONFIDENTIALITY**

## Confidentiality

Confidentiality refers to protecting information from being accessed by unauthorized parties. In other words, only the people who are authorized to do so can gain access to sensitive data. Imagine your bank records. You should be able to access them, of course, and employees at the bank who are helping you with a transaction should be able to access them, but no one else should. A failure to maintain confidentiality means that someone who shouldn't have access has managed to get it, through intentional behavior or by accident. Such a failure of confidentiality, commonly known as a breach, typically cannot be remedied. Once the secret has been revealed, there's no way to un-reveal it. If your bank records are posted on a public website, everyone can know your bank account number, balance, etc., and that information can't be erased from their minds, papers, computers, and other places. Nearly all the major security incidents reported in the media today involve major losses of confidentiality.

**Message Confidentiality Threats**

An attacker can easily violate message confidentiality (and perhaps integrity) because of the public nature of networks. Eavesdropping and impersonation attacks can lead to a confidentiality or integrity failure. Here we consider several other vulnerabilities that can affect confidentiality.

**Misdelivery**

Sometimes messages are misdelivered because of some flaw in the network hardware or software. Most frequently, messages are lost entirely, which is an integrity or availability issue. Occasionally, however, a destination address is modified or some handler malfunctions, causing a message to be delivered to someone other than the intended recipient. All of these "random" events are quite uncommon.

More frequent than network flaws are human errors. It is far too easy to mistype an address such as 100064,30652 as 10064,30652 or 100065,30642, or to type "idw" or "iw" instead of "diw" for David Ian Walker, who is called Ian by his friends. There is simply no justification for a computer network administrator to identify people by meaningless long numbers or cryptic initials when "iwalker" would be far less prone to human error.

**Exposure**

To protect the confidentiality of a message, we must track it all the way from its creation to its disposal. Along the way, the content of a message may be exposed in temporary buffers; at switches, routers, gateways, and intermediate hosts throughout the network; and in the workspaces of processes that build, format, and present the message. In earlier chapters, we considered confidentiality exposures in programs and operating systems. All of these exposures apply to networked environments as well. Furthermore, a malicious attacker can use any of these exposures as part of a general or focused attack on message confidentiality.

Passive wiretapping is one source of message exposure. So also is subversion of the structure by which a communication is routed to its destination. Finally, intercepting the message at its source, destination, or at any intermediate node can lead to its exposure.

**Traffic Flow Analysis**

Sometimes not only is the message itself sensitive but the fact that a message exists is also sensitive. For example, if the enemy during wartime sees a large amount of network traffic between headquarters and a particular unit, the enemy may be able to infer that significant action is being planned involving that unit. In a commercial setting, messages sent from the president of one company to the president of a competitor could lead to speculation about a takeover or conspiracy to fix prices. Or communications from the prime minister of one country to another with whom diplomatic relations were suspended could lead to inferences about a rapprochement between the countries. In these cases, we need to protect both the content of messages and the header information that identifies sender and receiver.

**Message integrity:-**

In the world of secured communications, Message **Integrity** describes the concept of ensuring that **data has not been modified in transit**. This is typically accomplished with the use of a Hashing algorithm.  We learned earlier [what a Hashing Algorithm does.](https://www.practicalnetworking.net/series/cryptography/hashing-algorithm/" \t "https://www.practicalnetworking.net/series/cryptography/message-integrity/_blank) Now we can take a look at how they are actually used to provide Message Integrity.

The basic premise is a sender wishes to send a message to a receiver, and wishes for the integrity of their message to be guaranteed.  The sender will calculate a hash on the message, and include the digest with the message.

On the other side, the receiver will independently calculate the hash on just the message, and compare  the resulting digest with the digest which was sent with the message.  If they are the same, then the message must have been the same as when it was originally sent.

**Code integrity:-**

**Code integrity** is a measurement used in [software testing](https://en.wikipedia.org/wiki/Software_testing" \o "Software testing). It measures the how high the [source code](https://en.wikipedia.org/wiki/Source_code" \o "Source code)'s quality is when it is passed on to the QA, and is affected by how extensively the code was [unit tested](https://en.wikipedia.org/wiki/Unit_tests" \o "Unit tests) and [integration tested](https://en.wikipedia.org/wiki/Integration_test" \o "Integration test). Code integrity is a combination of code coverage and software quality, and is usually achieved by unit testing your code to reach high code coverage.

With code integrity, the developer can be sure that his code is written correctly when passed on to QA. This is, in fact, the expected quality level of the code. Code integrity helps companies release better products, with fewer bugs, in a shorter time.

Companies who practice code integrity avoid the classic scenario where the development stage is delayed, delaying the QA stage, delaying the release stage. The product is released with more bugs (due to time pressure), users report tons of bugs back to the development team, and they start working on version 1.1 shortly after releasing version 1.0, just to fix bugs that could have been avoided.

The QA department can’t measure the code’s integrity even after all their tests are run. The only way to measure code integrity, and be sure of your code, is by unit testing your code, and reaching high code coverage.

**Improve code integrity by:**

* Unit testing the code
* Integration testing
* Assigning a code integrity manager

**Advantages of working with code**

* Shorter development time - bugs that are found during the development stage are fixed faster and easier than bugs found in later stages.
* Lower development costs – It's cheaper to fix bugs that are found during the development stage than in later stages.
* Confidence in your code's quality – Releasing products with high code integrity means more positive feedback from your customers.
* Makes the QA's work much more efficient – The QA concentrates on testing the system, without worrying about bugs that could have been easily found through proper unit testing.

**Measuring code integrity:**

To measuring code integrity, use the following formula: 1 − (Non-covered bugs) / (Total bugs)

In words:, the 100% code integrity minus the number of bugs that weren’t covered by unit testing, divided by the total bugs found during the entire product cycle., including development is the code not in integrity.

**Denial of service:-**A **Denial-of-Service (DoS) attack** is an attack meant to shut down a machine or network, making it inaccessible to its intended users. DoS attacks accomplish this by flooding the target with traffic, or sending it information that triggers a crash. In both instances, the DoS attack deprives legitimate users (i.e. employees, members, or account holders) of the service or resource they expected.

Victims of DoS attacks often target web servers of high-profile organizations such as banking, commerce, and media companies, or government and trade organizations. Though DoS attacks do not typically result in the theft or loss of significant information or other assets, they can cost the victim a great deal of time and money to handle.

There are two general methods of DoS attacks: flooding services or crashing services. Flood attacks occur when the system receives too much traffic for the server to buffer, causing them to slow down and eventually stop. Popular flood attacks include:

* **Buffer overflow attacks** – the most common DoS attack. The concept is to send more traffic to a network address than the programmers have built the system to handle. It includes the attacks listed below, in addition to others that are designed to exploit bugs specific to certain applications or networks
* **ICMP flood** – leverages misconfigured network devices by sending spoofed packets that ping every computer on the targeted network, instead of just one specific machine. The network is then triggered to amplify the traffic. This attack is also known as the smurf attack or ping of death.
* **SYN flood** – sends a request to connect to a server, but never completes the [handshake](https://www.paloaltonetworks.com/documentation/glossary/what-is-a-port-scan). Continues until all open ports are saturated with requests and none are available for legitimate users to connect to.

Other DoS attacks simply exploit vulnerabilities that cause the target system or service to crash. In these attacks, input is sent that takes advantage of bugs in the target that subsequently crash or severely destabilize the system, so that it can’t be accessed or used.

An additional type of DoS attack is the [Distributed Denial of Service (DDoS) attack](https://www.paloaltonetworks.com/cyberpedia/what-is-a-ddos-attack). A DDoS attack occurs when multiple systems orchestrate a synchronized DoS attack to a single target. The essential difference is that instead of being attacked from one location, the target is attacked from many locations at once. The distribution of hosts that defines a DDoS provide the attacker multiple advantages:

* He can leverage the greater volume of machine to execute a seriously disruptive attack
* The location of the attack is difficult to detect due to the random distribution of attacking systems (often worldwide)
* It is more difficult to shut down multiple machines than one
* The true attacking party is very difficult to identify, as they are disguised behind many (mostly compromised) systems

Modern security technologies have developed mechanisms to defend against most forms of DoS attacks, but due to the unique characteristics of DDoS, it is still regarded as an elevated threat and is of higher concern to organizations that fear being targeted by such an attack.

**Firewall:**

The Internet is a dangerous place filled with all kinds of security threats. Therefore the moment your network – be it small or large – connects to the internet, it gets exposed to all kinds of security issues, unless it is protected by what is known as a ****firewall****, which is responsible for filtering traffic which flows into (and goes out of) your network, thereby greatly reducing the impact of malicious traffic traveling over public internet can have on your network.

****What Exactly Is A Firewall, In Simple Terms?****

Firewall, a term which is supposedly coined by AT&T’s Steven M.Bellovin, is a metaphor (wall on fire) indicating that it prevents intruders (like virus, trojans, ransomware, other types of malware and other such security threats) from breaking into networks and infecting them. Simply put, *****network firewall*****(s) protect your networks against security threats.

****Types Of Firewall****

The concept ‘network firewall’ has evolved over the years, and roughly speaking, you now have about 5 different types of them. Let’s take a look at them briefly.

* ****Packet Filtering Firewall:**** this is the first firewall created for network security which was basically responsible for filtering (inspecting) data packets coming into the network based on an established rule-set (or criteria) – like allowing data from only certain IP addresses, packet types, port numbers etc., – and ignoring those which don’t match them.
* ****Circuit Level Firewall Or Gateway:**** these firewalls operate at the session level – monitoring TCP handshakes – instead of at the data packets level. Simply put, they ensure the external sessions your network encounters (or engages with) are legitimate or not.
* ****Stateful Inspection Firewall:**** these network firewalls accomplish both data packet filtering and session (TCP handshake) filtering. In other words, they do the job of both packet filtering and circuit level filtering firewalls. This means they monitor all active sessions or connections and thus determine which network packets should be allowed or disallowed.
* ****Application Level Firewall:**** these firewalls operate at the application level. In other words, they filter the traffic only with regards to the application (or service) for which they are intended. For example, a firewall for monitoring traffic to all the web applications your network uses.

****Next Gen Firewall Is The Future****

The firewalls which are most popular at the moment are next-gen firewalls. The obvious reason being the modern-day malware which is quite complex in nature gives rise to the need for a more sophisticated firewall for monitoring (and safeguarding) the network efficiently. This firewall is the least-well delineated firewall of all and is usually a combination of stateful inspection and deep packet inspection firewalls.

****Use Comodo Internet Security (CIS)****

Comodo Internet Security, which comes equipped with impressive security features, contains the best firewall the IT security industry has to offer. Operating using the Comodo’s patented Default Deny Approach (which is implemented via Containment technology), it offers effective resistance not just against malware threats but against zero-day attacks too.

Apart from offering an award-winning best firewall, CIS also offers features like Secure Shopping, Spyware Scanning, technologies like HIPS (Host Intrusion Prevention System) and Viruscope, and other important features like unlimited product support, [virus removal](https://antivirus.comodo.com/free-antivirus.php?af=7639" \t "https://antivirus.comodo.com/blog/comodo-news/types-of-firewall/_blank), online backup, wifi security etc.

No one can deny the fact that the dynamic rise of the Internet has brought the world closer. But at the same time, it has left us with different kinds of security threats. To ensure the confidentiality and integrity of valuable information of a corporate network from the outside attacks, we must have some robust mechanism. This is where the **Firewall** comes into picture.

It can be compared with a security guard standing at the entrance of a minister’s home. He keeps an eye on everyone and physically checks every person who wishes to enter the house. It won’t allow a person to enter if he/she is carrying a harmful object like a knife, gun etc. Similarly, even if the person doesn’t possess any banned object but appears suspicious, the guard can still prevent that person’s entry.

The firewall acts as a guard. It guards a corporate network acting as a shield between the inside network and the outside world. All the traffic in either direction must pass through the firewall. It then decides whether the traffic is allowed to flow or not. The firewall can be implemented as hardware and software, or a combination of both.

IMG_256

**Types of Firewalls:**

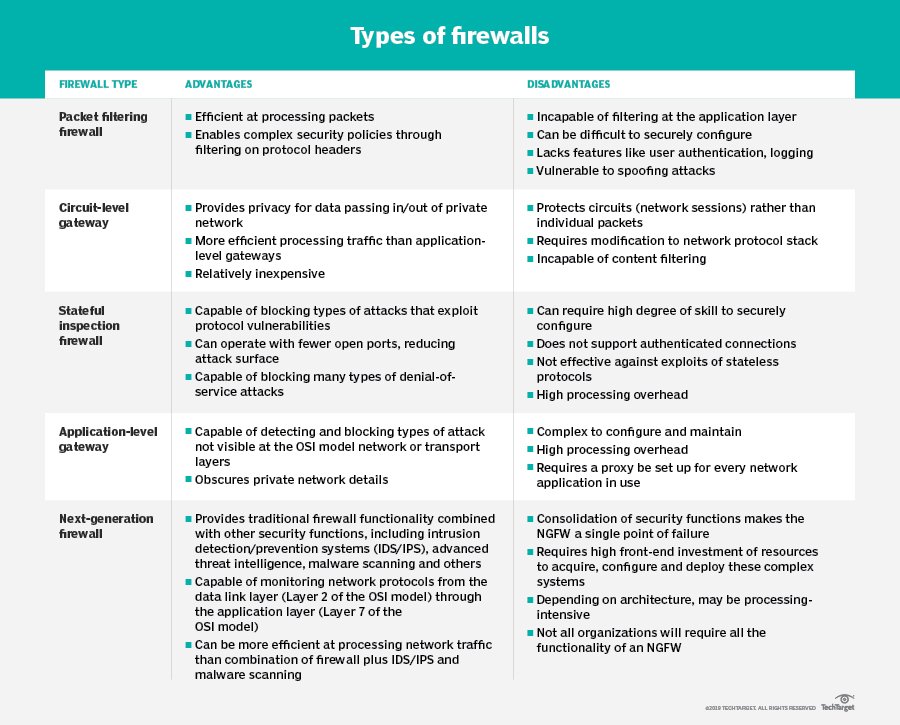
1. **Packet Filters –**  
   It works in the **network layer** of the OSI Model. It applies a set of rules (based on the contents of IP and transport header fields) on each packet and based on the outcome, decides to either forward or discard the packet.

For example, a rule could specify to block all incoming traffic from a certain IP address or disallow all traffic that uses UDP protocol. If there is no match with any predefined rules, it will take default action. The default action can be to ‘discard all packets’ or to ‘accept all packets’.

**Security threats** to Packet Filters:

* 1. **IP address Spoofing:**  
     In this kind of attack, an intruder from the outside tries to send a packet towards the internal corporate network with the source IP address set equal to one of the IP address of internal users.  
     **Prevention:**  
     Firewall can defeat this attack if it discards all the packets that arrive at the incoming side of the firewall, with source IP equal to one of the internal IPs.
  2. **Source Routing Attacks:**  
     In this kind of attack, the attacker specifies the route to be taken by the packet with a hope to fool the firewall.  
     **Prevention:**  
     Firewall can defeat this attack if it discards all the packets that use the option of source routing aka path addressing.
  3. **Tiny Fragment Attacks:**  
     Many times, the size of the IP packet is greater than the maximum size allowed by the underlying network such as Ethernet, Token Ring etc. In such cases, the packet needs to be [fragmented](https://www.geeksforgeeks.org/fragmentation-network-layer/" \t "https://www.geeksforgeeks.org/types-of-firewall-and-possible-attacks/_blank), so that it can be carried further. The attacker uses this characteristic of TCP/IP protocol. In this kind of attack, the attacker intentionally creates fragments of the original packet and send it to fool the firewall.  
     **Prevention:**  
     Firewall can defeat this attack if it discards all the packets which use the TCP protocol and is fragmented. *Dynamic Packet Filters* allow incoming TCP packets only if they are responses to the outgoing TCP packets.

1. **Application Gateways –**  
   It is also known as **Proxy server**. It works as follows:
   1. **Step-1:** User contacts the application gateway using a TCP/IP application such as HTTP.
   2. **Step-2:** The application gateway asks about the remote host with which the user wants to establish a connection. It also asks for the user id and password that is required to access the services of the application gateway.
   3. **Step-3:** After verifying the authenticity of the user, the application gateway accesses the remote host on behalf of the user to deliver the packets.
2. **Stateful Inspection Firewalls –**  
   It is also known as ‘Dynamic Packet Filters’. It keeps track of the state of active connections and uses this information to decide which packets to allow through it, i.e., it adapts itself to the current exchange of information, unlike the normal packet filters/stateless packet filters, which have hardcoded routing rules.
3. **Circuit-Level Gateways –**  
   It works at the **session layer** of the OSI Model. It is the advanced variation of *Application Gateway*. It acts as a virtual connection between the remote host and the internal users by creating a new connection between itself and the remote host. It also changes the source IP address in the packet and puts its own address at the place of source IP address of the packet from end users. This way, the IP addresses of the internal users are hidden and secured from the outside world.



## What is a DMZ Network?

In computer security, a DMZ Network (sometimes referred to as a “demilitarized zone”) functions as a subnetwork containing an organization's exposed, outward-facing services. It acts as the exposed point to an untrusted networks, commonly the Internet.

The goal of a DMZ is to add an extra layer of security to an organization's local area network. A protected and monitored network node that faces outside the internal network can access what is exposed in the DMZ, while the rest of the organization's network is safe behind a firewall.

When implemented properly, a DMZ Network gives organizations extra protection in detecting and mitigating security breaches before they reach the internal network, where valuable assets are stored.

### Purpose of a DMZ

The DMZ Network exists to protect the hosts most vulnerable to attack. These hosts usually involve services that extend to users outside of the local area network, the most common examples being email, web servers, and DNS servers. Because of the increased potential for attack, they are placed into the monitored subnetwork to help protect the rest of the network if they become compromised.

Hosts in the DMZ have tightly controlled access permissions to other services within the internal network, because the data passed through the DMZ is not as secure. On top of that, communications between hosts in the DMZ and the external network are also restricted to help increase the protected border zone. This allows hosts in the protected network to interact with the internal and external network, while the firewall separates and manages all traffic shared between the DMZ and the internal network. Typically, an additional firewall will be responsible for protecting the DMZ from exposure to everything on the external network.

All services accessible to users on communicating from an external network can and should be placed in the DMZ, if one is used. The most common services are:

* Web servers: Web servers responsible for maintaining communication with an internal database server may need to be placed into a DMZ. This helps ensure the safety of the internal database, which is often storing sensitive information. The web servers can then interact with internal database server through an application firewall or directly, while still falling under the umbrella of the DMZ protections.
* Mail servers: individual email messages, as well as the user database built to store login credentials and personal messages, are usually stored on servers without direct access to the internet. Therefore, an email server will be built or placed inside the DMZ in order to interact with and access the email database without directly exposing it to potentially harmful traffic.
* FTP servers: These can host critical content on an organization's site, and allow direct interaction with files. Therefore, an FTP server should always be partially isolated from critical internal systems.

A DMZ configuration provides additional security from external attacks, but it typically has no bearing on internal attacks such as sniffing communication via a packet analyzer or spoofing via email or other means.

### DMZ Designs

There are numerous ways to construct a network with a DMZ. The two major methods are a single firewall (sometimes called a three-legged model), or dual firewalls. Each of these system can be expanded to create complex architectures built to satisfy network requirements:

* Single firewall: A modest approach to network architecture involves using a single firewall, with a minimum of 3 network interfaces. The DMZ will be placed Inside of this firewall. The tier of operations is as follows: the external network device makes the connection from the ISP, the internal network is connected by the second device, and connections within the DMZ is handled by the third network device.
* Dual firewall: The more secure approach is to use two firewalls to create a DMZ. The first firewall (referred to as the “frontend” firewall) is configured to only allow traffic destined for the DMZ. The second firewall (referred to as the “backend” firewall) is only responsible for the traffic that travels from the DMZ to the internal network. An effective way of further increasing protection is to use firewalls built by separate vendors, because they are less likely to have the same security vulnerabilities. While more effective, this scheme can be more costly to implement across a large network.

## Why DMZ Networks are Important

On many home networks, internet enabled devices are built around a local area network which accesses the internet from a broadband router. However, the router serves as both a connection point and a firewall, automating traffic filtering to ensure only safe messages enter the local area network. So, on a home network, a DMZ can built by adding a dedicated firewall, between the local area network and the router. While more expensive, this structure can help to protect internal devices from sophisticated attacks better protects the inside devices from possible attacks by the outside.

DMZ’s are an essential part of network security for both individual users and large organizations. They provides an extra layer of security to the computer network by restricting remote access to internal servers and information, which can be very damaging if breached.

**VPN:-**

VPN stands for [Virtual Private Network (VPN)](https://www.geeksforgeeks.org/virtual-private-network-vpn-introduction/" \t "https://www.geeksforgeeks.org/types-of-virtual-private-network-vpn-and-its-protocols/_blank), that allows a user to connect to a private network over the Internet securely and privately. VPN creates an encrypted connection that is called VPN tunnel, and all Internet traffic and communication is passed through this secure tunnel.

Virtual Private Network (VPN) is basically of 2 types:

1. **Remote Access VPN:**  
   Remote Access VPN permits a user to connect to a private network and access all its services and resources remotely. The connection between the user and the private network occurs through the Internet and the connection is secure and private. Remote Access VPN is useful for home users and business users both.

An employee of a company, while he/she is out of station, uses a VPN to connect to his/her company’s private network and remotely access files and resources on the private network. Private users or home users of VPN, primarily use VPN services to bypass regional restrictions on the Internet and access blocked websites. Users aware of Internet security also use VPN services to enhance their Internet security and privacy.

1. **Site to Site VPN:**  
   A Site-to-Site VPN is also called as Router-to-Router VPN and is commonly used in the large companies. Companies or organizations, with branch offices in different locations, use Site-to-site VPN to connect the network of one office location to the network at another office location.
   * **Intranet based VPN:** When several offices of the same company are connected using Site-to-Site VPN type, it is called as Intranet based VPN.
   * **Extranet based VPN:** When companies use Site-to-site VPN type to connect to the office of another company, it is called as Extranet based VPN.

Basically, Site-to-site VPN create a imaginary bridge between the networks at geographically distant offices and connect them through the Internet and sustain a secure and private communication between the networks. In Site-to-site VPN one router acts as a VPN Client and another router as a VPN Server as it is based on Router-to-Router communication. When the authentication is validated between the two routers only then the communication starts.

**Types of Virtual Private Network (VPN) Protocols:**

1. **Internet Protocol Security (IPSec):**  
   Internet Protocol Security, known as IPSec, is used to secure Internet communication across an IP network. IPSec secures Internet Protocol communication by verifying the session and encrypts each data packet during the connection.

IPSec runs in 2 modes:

* + (i) Transport mode
  + (ii) Tunneling mode

The work of transport mode is to encrypt the message in the data packet and the tunneling mode encrypts the whole data packet. IPSec can also be used with other security protocols to improve the security system.

1. **Layer 2 Tunneling Protocol (L2TP):**  
   L2TP or Layer 2 Tunneling Protocol is a tunneling protocol that is often combined with another VPN security protocol like IPSec to establish a highly secure VPN connection. L2TP generates a tunnel between two L2TP connection points and IPSec protocol encrypts the data and maintains secure communication between the tunnel.
2. **Point–to–Point Tunneling Protocol (PPTP):**  
   PPTP or Point-to-Point Tunneling Protocol generates a tunnel and confines the data packet. Point-to-Point Protocol (PPP) is used to encrypt the data between the connection. PPTP is one of the most widely used VPN protocol and has been in use since the early release of Windows. PPTP is also used on Mac and Linux apart from Windows.
3. **SSL and TLS:**  
   SSL (Secure Sockets Layer) and TLS (Transport Layer Security) generate a VPN connection where the web browser acts as the client and user access is prohibited to specific applications instead of entire network. Online shopping websites commonly uses SSL and TLS protocol. It is easy to switch to SSL by web browsers and with almost no action required from the user as web browsers come integrated with SSL and TLS. SSL connections have “https” in the initial of the URL instead of “http”.
4. **OpenVPN:**  
   OpenVPN is an open source VPN that is commonly used for creating Point-to-Point and Site-to-Site connections. It uses a traditional security protocol based on SSL and TLS protocol.
5. **Secure Shell (SSH):**  
   Secure Shell or SSH generates the VPN tunnel through which the data transfer occurs and also ensures that the tunnel is encrypted. SSH connections are generated by a SSH client and data is transferred from a local port on to the remote server through the encrypted tunnel.

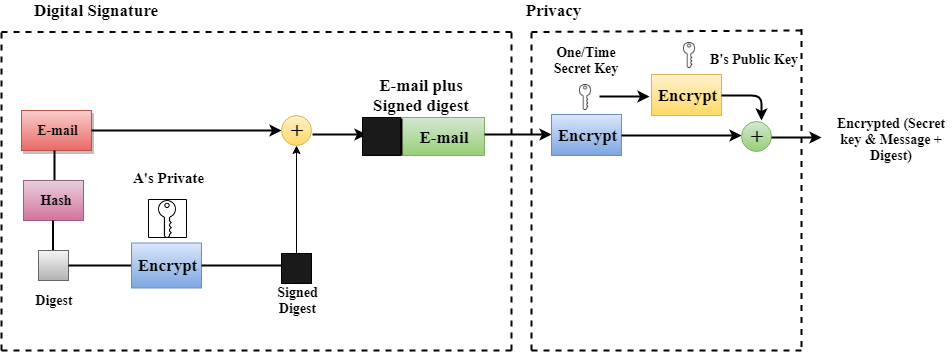
# PGP

* PGP stands for Pretty Good Privacy (PGP) which is invented by Phil Zimmermann.
* PGP was designed to provide all four aspects of security, i.e., privacy, integrity, authentication, and non-repudiation in the sending of email.
* PGP uses a digital signature (a combination of hashing and public key encryption) to provide integrity, authentication, and non-repudiation. PGP uses a combination of secret key encryption and public key encryption to provide privacy. Therefore, we can say that the digital signature uses one hash function, one secret key, and two private-public key pairs.
* PGP is an open source and freely available software package for email security.
* PGP provides authentication through the use of Digital Signature.
* It provides confidentiality through the use of symmetric block encryption.
* It provides compression by using the ZIP algorithm, and EMAIL compatibility using the radix-64 encoding scheme.

### Following are the steps taken by PGP to create secure e-mail at the sender site:

* The e-mail message is hashed by using a hashing function to create a digest.
* The digest is then encrypted to form a signed digest by using the sender's private key, and then signed digest is added to the original email message.
* The original message and signed digest are encrypted by using a one-time secret key created by the sender.
* The secret key is encrypted by using a receiver's public key.
* Both the encrypted secret key and the encrypted combination of message and digest are sent together.

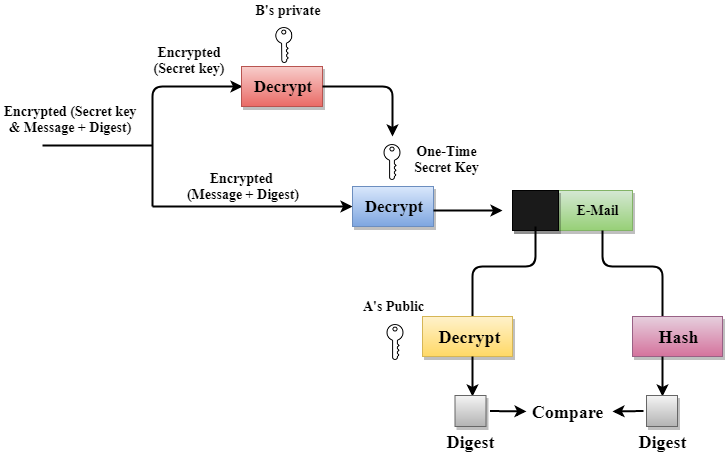
### PGP at the Sender site (A)



### Following are the steps taken to show how PGP uses hashing and a combination of three keys to generate the original message:

* The receiver receives the combination of encrypted secret key and message digest is received.
* The encrypted secret key is decrypted by using the sender's private key to get the one-time secret key.
* The secret key is then used to decrypt the combination of message and digest.
* The digest is decrypted by using the sender's public key, and the original message is hashed by using a hash function to create a digest.
* Both the digests are compared if both of them are equal means that all the aspects of security are preserved.

### PGP at the Receiver site (B)



### Disadvantages of PGP Encryption

* **The Administration is difficult:** The different versions of PGP complicate the administration.
* **Compatibility issues:** Both the sender and the receiver must have compatible versions of PGP. For example, if you encrypt an email by using PGP with one of the encryption technique, the receiver has a different version of PGP which cannot read the data.
* **Complexity:** PGP is a complex technique. Other security schemes use symmetric encryption that uses one key or asymmetric encryption that uses two different keys. PGP uses a hybrid approach that implements symmetric encryption with two keys. PGP is more complex, and it is less familiar than the traditional symmetric or asymmetric methods.
* **No Recovery:** Computer administrators face the problems of losing their passwords. In such situations, an administrator should use a special program to retrieve passwords. For example, a technician has physical access to a PC which can be used to retrieve a password. However, PGP does not offer such a special program for recovery; encryption methods are very strong so, it does not retrieve the forgotten passwords results in lost messages or lost files.