What is Cryptography?

*Cryptography is the art and science of making a cryptosystem that is capable of providing information security.*

Cryptography deals with the actual securing of digital data. It refers to the design of mechanisms based on mathematical algorithms that provide fundamental information security services. You can think of cryptography as the establishment of a large toolkit containing different techniques in security applications.

What is Cryptanalysis?

*The art and science of breaking the cipher text is known as cryptanalysis.*

Cryptanalysis is the sister branch of cryptography and they both co-exist. The cryptographic process results in the cipher text for transmission or storage. It involves the study of cryptographic mechanism with the intention to break them. Cryptanalysis is also used during the design of the new cryptographic techniques to test their security strengths.

Threats to Data Integrity

When sensitive information is exchanged, the receiver must have the assurance that the message has come intact from the intended sender and is not modified inadvertently or otherwise. There are two different types of data integrity threats, namely passive and active.

Passive Threats

This type of threats exists due to accidental changes in data.

These data errors are likely to occur due to noise in a communication channel. Also, the data may get corrupted while the file is stored on a disk.

Error-correcting codes and simple checksums like Cyclic Redundancy Checks (CRCs) are used to detect the loss of data integrity. In these techniques, a digest of data is computed mathematically and appended to the data.

Active Threats

In this type of threats, an attacker can manipulate the data with malicious intent.

At simplest level, if data is without digest, it can be modified without detection. The system can use techniques of appending CRC to data for detecting any active modification.

At higher level of threat, attacker may modify data and try to derive new digest for modified data from exiting digest. This is possible if the digest is computed using simple mechanisms such as CRC.

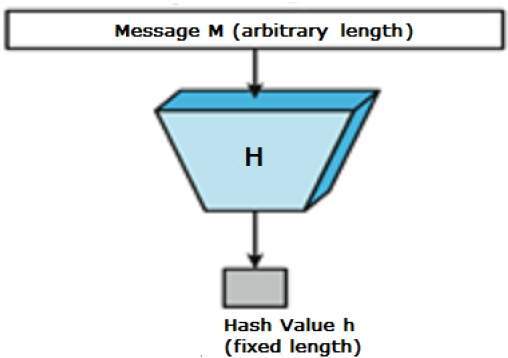
Security mechanism such as Hash functions are used to tackle the active modification threats.

Hash functions

Hash functions are extremely useful and appear in almost all information security applications.

A hash function is a mathematical function that converts a numerical input value into another compressed numerical value. The input to the hash function is of arbitrary length but output is always of fixed length.

Values returned by a hash function are called message digest or simply hash values. The following picture illustrated hash function −



Features of Hash Functions

The typical features of hash functions are −

Fixed Length Output (Hash Value)

Hash function coverts data of arbitrary length to a fixed length. This process is often referred to as hashing the data.

In general, the hash is much smaller than the input data, hence hash functions are sometimes called compression functions.

Since a hash is a smaller representation of a larger data, it is also referred to as a digest.

Hash function with n bit output is referred to as an n-bit hash function. Popular hash functions generate values between 160 and 512 bits.

**Efficiency of Operation**

Generally for any hash function h with input x, computation of h(x) is a fast operation.

Computationally hash functions are much faster than a symmetric encryption.

**Properties of Hash Functions**

In order to be an effective cryptographic tool, the hash function is desired to possess following properties −

1. Pre-Image Resistance

This property means that it should be computationally hard to reverse a hash function.

In other words, if a hash function h produced a hash value z, then it should be a difficult process to find any input value x that hashes to z.

This property protects against an attacker who only has a hash value and is trying to find the input.

Second Pre-Image Resistance

This property means given an input and its hash, it should be hard to find a different input with the same hash.

In other words, if a hash function h for an input x produces hash value h(x), then it should be difficult to find any other input value y such that h(y) = h(x).

This property of hash function protects against an attacker who has an input value and its hash, and wants to substitute different value as legitimate value in place of original input value.

Collision Resistance

This property means it should be hard to find two different inputs of any length that result in the same hash. This property is also referred to as collision free hash function.

In other words, for a hash function h, it is hard to find any two different inputs x and y such that h(x) = h(y).

Since, hash function is compressing function with fixed hash length, it is impossible for a hash function not to have collisions. This property of collision free only confirms that these collisions should be hard to find.

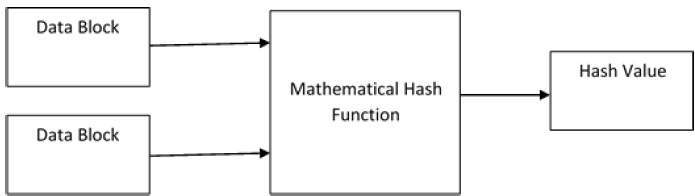
This property makes it very difficult for an attacker to find two input values with the same hash.

Also, if a hash function is collision-resistant then it is second pre-image resistant.

**Design of Hashing Algorithms**

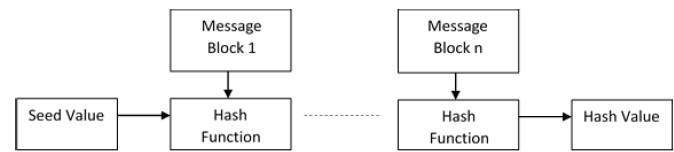
At the heart of a hashing is a mathematical function that operates on two fixed-size blocks of data to create a hash code. This hash function forms the part of the hashing algorithm.

The size of each data block varies depending on the algorithm. Typically the block sizes are from 128 bits to 512 bits. The following illustration demonstrates hash function −



Hashing algorithm involves rounds of above hash function like a block cipher. Each round takes an input of a fixed size, typically a combination of the most recent message block and the output of the last round.

This process is repeated for as many rounds as are required to hash the entire message. Schematic of hashing algorithm is depicted in the following illustration −



Since, the hash value of first message block becomes an input to the second hash operation, output of which alters the result of the third operation, and so on. This effect, known as an avalanche effect of hashing.

Avalanche effect results in substantially different hash values for two messages that differ by even a single bit of data.

Understand the difference between hash function and algorithm correctly. The hash function generates a hash code by operating on two blocks of fixed-length binary data.

Hashing algorithm is a process for using the hash function, specifying how the message will be broken up and how the results from previous message blocks are chained together.

**Popular Hash Functions**

**Let us briefly see some popular hash functions −**

**Message Digest (MD)**

MD5 was most popular and widely used hash function for quite some years.

The MD family comprises of hash functions MD2, MD4, MD5 and MD6. It was adopted as Internet Standard RFC 1321. It is a 128-bit hash function.

MD5 digests have been widely used in the software world to provide assurance about integrity of transferred file. For example, file servers often provide a pre-computed MD5 checksum for the files, so that a user can compare the checksum of the downloaded file to it.

In 2004, collisions were found in MD5. An analytical attack was reported to be successful only in an hour by using computer cluster. This collision attack resulted in compromised MD5 and hence it is no longer recommended for use.

Secure Hash Function (SHA)

Family of SHA comprise of four SHA algorithms; SHA-0, SHA-1, SHA-2, and SHA-3. Though from same family, there are structurally different.

The original version is SHA-0, a 160-bit hash function, was published by the National Institute of Standards and Technology (NIST) in 1993. It had few weaknesses and did not become very popular. Later in 1995, SHA-1 was designed to correct alleged weaknesses of SHA-0.

SHA-1 is the most widely used of the existing SHA hash functions. It is employed in several widely used applications and protocols including Secure Socket Layer (SSL) security.

In 2005, a method was found for uncovering collisions for SHA-1 within practical time frame making long-term employability of SHA-1 doubtful.

SHA-2 family has four further SHA variants, SHA-224, SHA-256, SHA-384, and SHA-512 depending up on number of bits in their hash value. No successful attacks have yet been reported on SHA-2 hash function.

Though SHA-2 is a strong hash function. Though significantly different, its basic design is still follows design of SHA-1. Hence, NIST called for new competitive hash function designs.

In October 2012, the NIST chose the Keccak algorithm as the new SHA-3 standard. Keccak offers many benefits, such as efficient performance and good resistance for attacks.

RIPEMD

The RIPEND is an acronym for RACE Integrity Primitives Evaluation Message Digest. This set of hash functions was designed by open research community and generally known as a family of European hash functions.

The set includes RIPEND, RIPEMD-128, and RIPEMD-160. There also exist 256, and 320-bit versions of this algorithm.

Original RIPEMD (128 bit) is based upon the design principles used in MD4 and found to provide questionable security. RIPEMD 128-bit version came as a quick fix replacement to overcome vulnerabilities on the original RIPEMD.

RIPEMD-160 is an improved version and the most widely used version in the family. The 256 and 320-bit versions reduce the chance of accidental collision, but do not have higher levels of security as compared to RIPEMD-128 and RIPEMD-160 respectively.

Whirlpool

This is a 512-bit hash function.

It is derived from the modified version of Advanced Encryption Standard (AES). One of the designer was Vincent Rijmen, a co-creator of the AES.

Three versions of Whirlpool have been released; namely WHIRLPOOL-0, WHIRLPOOL-T, and WHIRLPOOL.

Applications of Hash Functions

There are two direct applications of hash function based on its cryptographic properties.

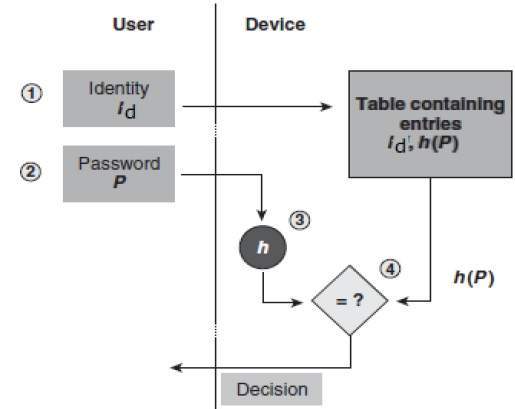
**Password Storage**

Hash functions provide protection to password storage.

Instead of storing password in clear, mostly all logon processes store the hash values of passwords in the file.

The Password file consists of a table of pairs which are in the form (user id, h(P)).

**The process of logon is depicted in the following illustration** −



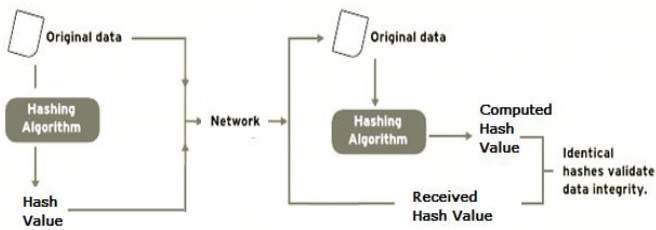
An intruder can only see the hashes of passwords, even if he accessed the password. He can neither logon using hash nor can he derive the password from hash value since hash function possesses the property of pre-image resistance.

**Data Integrity Check**

**Data integrity** refers to the overall completeness, accuracy and consistency of **data**. This can be indicated by the absence of alteration between two instances or between two updates of a **data** record, **meaning data** is intact and unchanged.

Data integrity check is a most common application of the hash functions. It is used to generate the checksums on data files. This application provides assurance to the user about correctness of the data.

The process is depicted in the following illustration −



The integrity check helps the user to detect any changes made to original file. It however, does not provide any assurance about originality. The attacker, instead of modifying file data, can change the entire file and compute all together new hash and send to the receiver. This integrity 1check application is useful only if the user is sure about the originality of file.

Cryptography Digital signatures

Digital signatures are the public-key primitives of message authentication. In the physical world, it is common to use handwritten signatures on handwritten or typed messages. They are used to bind signatory to the message.

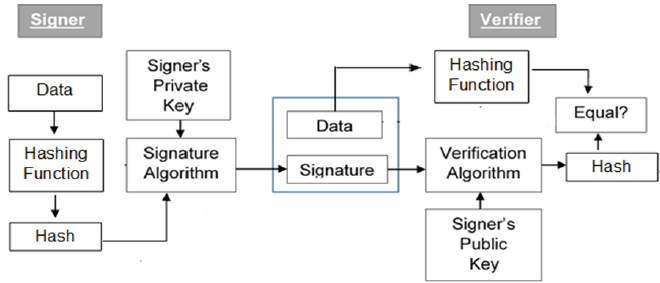
Similarly, a digital signature is a technique that binds a person/entity to the digital data. This binding can be independently verified by receiver as well as any third party.

Digital signature is a cryptographic value that is calculated from the data and a secret key known only by the signer.

In real world, the receiver of message needs assurance that the message belongs to the sender and he should not be able to repudiate the origination of that message. This requirement is very crucial in business applications, since likelihood of a dispute over exchanged data is very high.

Model of Digital Signature

As mentioned earlier, the digital signature scheme is based on public key cryptography. The model of digital signature scheme is depicted in the following illustration −



The following points explain the entire process in detail −

Each person adopting this scheme has a public-private key pair.

Generally, the key pairs used for encryption/decryption and signing/verifying are different. The private key used for signing is referred to as the signature key and the public key as the verification key.

Signer feeds data to the hash function and generates hash of data.

Hash value and signature key are then fed to the signature algorithm which produces the digital signature on given hash. Signature is appended to the data and then both are sent to the verifier.

Verifier feeds the digital signature and the verification key into the verification algorithm. The verification algorithm gives some value as output.

Verifier also runs same hash function on received data to generate hash value.

For verification, this hash value and output of verification algorithm are compared. Based on the comparison result, verifier decides whether the digital signature is valid.

Since digital signature is created by ‘private’ key of signer and no one else can have this key; the signer cannot repudiate signing the data in future.

It should be noticed that instead of signing data directly by signing algorithm, usually a hash of data is created. Since the hash of data is a unique representation of data, it is sufficient to sign the hash in place of data. The most important reason of using hash instead of data directly for signing is efficiency of the scheme.

Let us assume RSA is used as the signing algorithm. As discussed in public key encryption chapter, the encryption/signing process using RSA involves modular exponentiation.

Signing large data through modular exponentiation is computationally expensive and time consuming. The hash of the data is a relatively small digest of the data, hence signing a hash is more efficient than signing the entire data.

Importance of Digital Signature

Out of all cryptographic primitives, the digital signature using public key cryptography is considered as very important and useful tool to achieve information security.

Apart from ability to provide non-repudiation of message, the digital signature also provides message authentication and data integrity. Let us briefly see how this is achieved by the digital signature −

Message authentication − When the verifier validates the digital signature using public key of a sender, he is assured that signature has been created only by sender who possess the corresponding secret private key and no one else.

Data Integrity − In case an attacker has access to the data and modifies it, the digital signature verification at receiver end fails. The hash of modified data and the output provided by the verification algorithm will not match. Hence, receiver can safely deny the message assuming that data integrity has been breached.

Non-repudiation − Since it is assumed that only the signer has the knowledge of the signature key, he can only create unique signature on a given data. Thus the receiver can present data and the digital signature to a third party as evidence if any dispute arises in the future.

By adding public-key encryption to digital signature scheme, we can create a cryptosystem that can provide the four essential elements of security namely − Privacy, Authentication, Integrity, and Non-repudiation.

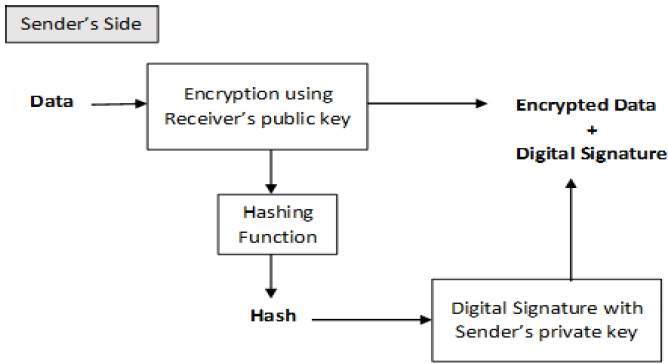
Encryption with Digital Signature

In many digital communications, it is desirable to exchange an encrypted messages than plaintext to achieve confidentiality. In public key encryption scheme, a public (encryption) key of sender is available in open domain, and hence anyone can spoof his identity and send any encrypted message to the receiver.

This makes it essential for users employing PKC for encryption to seek digital signatures along with encrypted data to be assured of message authentication and non-repudiation.

This can archived by combining digital signatures with encryption scheme. Let us briefly discuss how to achieve this requirement. There are two possibilities, sign-then-encrypt and encrypt-then-sign.

However, the crypto system based on sign-then-encrypt can be exploited by receiver to spoof identity of sender and sent that data to third party. Hence, this method is not preferred. The process of encrypt-then-sign is more reliable and widely adopted. This is depicted in the following illustration −



The receiver after receiving the encrypted data and signature on it, first verifies the signature using sender’s public key. After ensuring the validity of the signature, he then retrieves the data through decryption using his private key.