Symmetric Encryption Algorithms

Symmetric algorithms are broken down into two main types: stream and block ciphers. Block ciphers encrypt data in chunks (blocks), whereas stream ciphers encrypt data one bit at a time. So, what are some of the most commonly used or well-known symmetric algorithms?

- **Data Encryption Standard (DES):** DES is a type of block cipher that encrypts data in 64-bit blocks and using a single key that is one of three sizes (64-bit, 128-bit and 192-bit keys). However, one of every 8 bits is a parity bit, meaning that a single-length key that is 64 bits is really like using a 56-bit key. Although DES is one of the earliest symmetric encryption algorithms, it is viewed as insecure and has been deprecated.
- **Triple Data Encryption Standard (TDEA/3DES):** Unlike DES, triple DES can use two or three keys, which enables this algorithm to use multiple rounds of encryption (or more accurate, a round of encryption, round of decryption and another round of encryption). While 3DES is more secure than its DES predecessor, it is not as secure as its successor, AES.
- **Advanced Encryption Standard (AES):** This encryption algorithm is what you will most commonly find is use across the internet. The advanced encryption standard is more secure and efficient than DES and 3DES with key options that are 128 bits, 192 bits and 256 bits. However, while it is also a type of block cipher, it operates differently than DES and 3DES because it is based on a substitution-permutation network instead of the Feistel cipher.

How Secure are Symmetric Keys?

- The strength of any cryptographic key depends on a few specific considerations.
  - The length of the key,
  - The randomness (entropy) of how it was generated,
  - How long it takes to reverse it to figure out is individual components ?
- Now, you may be wondering whether a cybercriminal could just reverse the process to guess the variables to calculate the numbers that were used. It is possible, but the reality of that happening is so remote that it is not practical. I say that because even though cybercrimnals understand how these calculations work, it is incredibly difficult to reverse the process to find out the secret number you or your recipient used to generate your matching session keys.
- It goes back to the concept that was discussed in the video that shows the uses of mixing of specific colors to create a shared value. While it's easy to combine colors to create the shared value, it's virtually impossible to deconstruct those values to figure out exactly which shades of the colors were used to create them.
- Furthermore, since the numbers that are used in these calculations are massive, it would take way longer than any cybercriminal would have during a session to figure them out. Even with the most current supercomputers, an attacker would have to spend hundreds if not thousands of years trying to figure out the individual numbers you both used. I do not know about you, but we mortals do not have that much time to spend on such tasks.
- So, now that we know what symmetric encryption is and how it works, how is it used in the real world?

Examples of Where You are Already Using Symmetric Encryption

- Symmetric encryption is useful in many cases and has implementation opportunities across various industries. For example, symmetric encryption is useful for encrypting banking-related data as well as data storage.

Banking

- Ever heard of PCI DSS? The Payment Card Industry Data Security Standards is a set of 12 requirements that businesses or organizations that accept credit card payments must adhere to. Symmetric encryption is a key component of PCI compliance, as it directly correlates to requirement No. 3, which focuses on protecting at-rest cardholder data.

1.3 ASYMMETRIC KEY CRYPTOGRAPHY

- With censorship becoming increasingly prevalent in today's world, encryption has allowed everyone to have a firm hold of their private information. Asymmetric encryption, also known as asymmetric-key cryptography, has become a staple in digital security, thanks to its multiple applications and varied advantages.
- Look at the following story to understand the need for asymmetric key cryptography.
- Joe is a journalist who needs to communicate with Ryan over long-distance messaging. There is a single key for both encryption and decryption of data in such

a system. This key must be kept secret and should belong to the sender and receiver of the messages only. Because of the critical nature of the information, Joe uses symmetric encryption while sending his messages to prevent them from falling into the wrong hands.

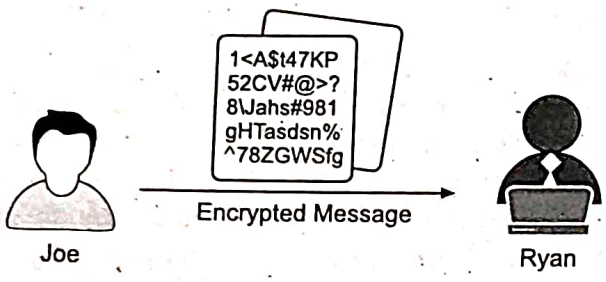


Fig. 1.10

- However, the tricky part comes when Ryan needs to receive the decryption key for the encrypted data.

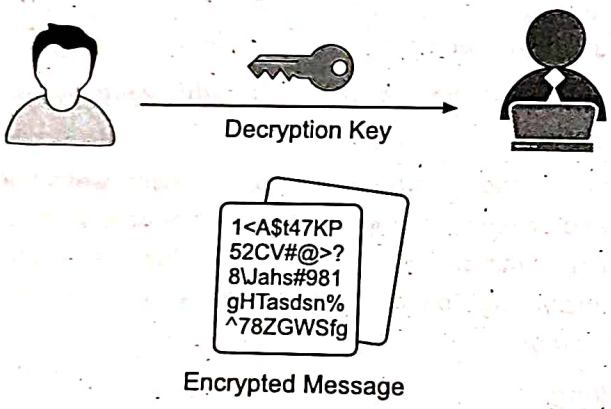


Fig. 1.11

- You face a critical problem here when it comes to the key exchange. The encrypted messages in transit are not a risk to either sender or receiver since unauthorized people can not read the contents. But the decryption key, on the other hand, if intercepted, exposes all the information Joe has to disclose to Ryan.

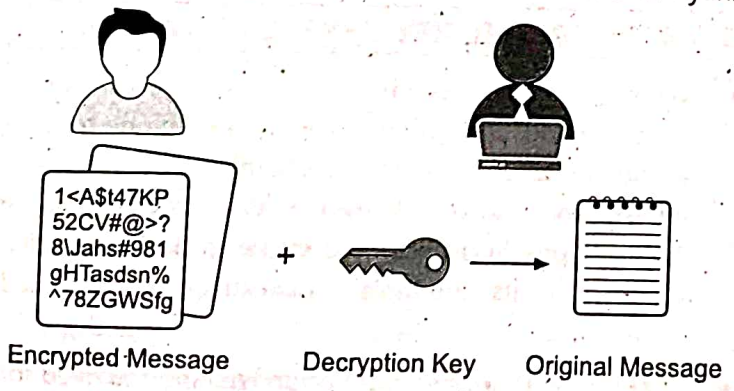


Fig. 1.12

- This is the dilemma asymmetric encryption has fixed because of its multi-key architecture. Now, this tutorial will take you through its entire workflow.

What is Asymmetric Encryption?

- Asymmetric encryption algorithms use two differe keys for encryption and decryption. The key used f encryption is the public key and the key used f decryption is the private key. Both the keys m belong to the receiver.

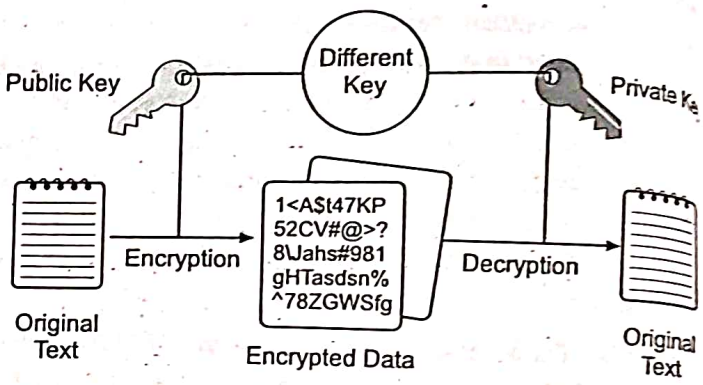


Fig. 1.13

- As you can see in the above image, using different keys for encryption and decryption has helped avoid the problem of key exchange, as seen in the case of symmetric encryption.

For example, if Alice needs to send a message to Bob, both the keys, private and public, must belong to Bob.

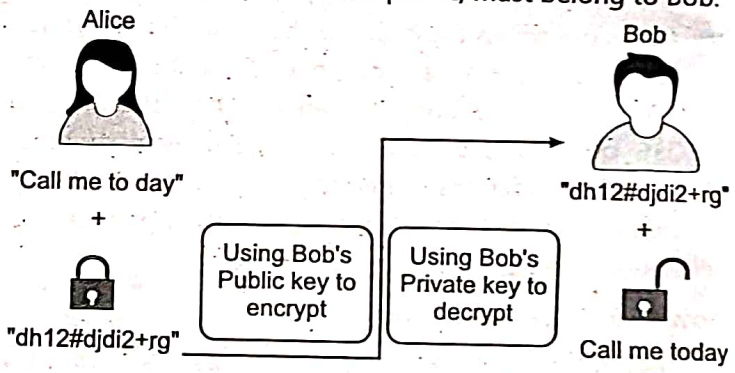


Fig. 1.14

- The process for the above image is as follows:

Step 1 : Alice uses Bob's public key to encrypt the message.

Step 2 : The encrypted message is sent to Bob.

Step 3 : Bob uses his private key to decrypt the message.

- To understand the asymmetric key cryptography architecture clearly, consider the process of sending and receiving letters via physical mailboxes.
- As shown below, anyone who has the postal address of the receiver (public key in our case) can send any message they want.

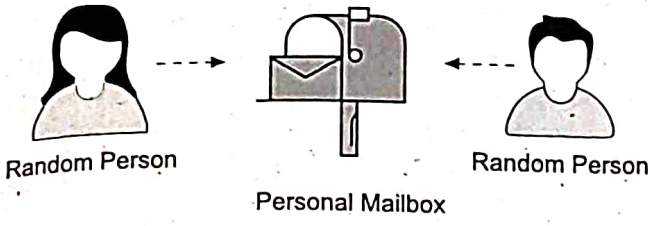


Fig. 1.15

- However, only the receiver can read all his/her messages thanks to the mailbox key that no other person can have.

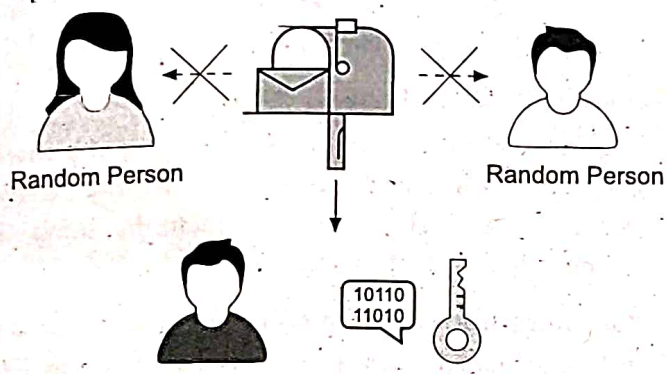


Fig. 1.16

- This eliminates the need to exchange any secret key between sender and receiver, thereby reducing the window of exploitation.
- Now that you understand the base terminology and process behind asymmetric key cryptography, this tutorial will take you through its applications.

Where is Asymmetric Key Cryptography used?

- Asymmetric key cryptography has found use in many authentication domains thanks to its thorough identity verification process. Some applications are as follows:

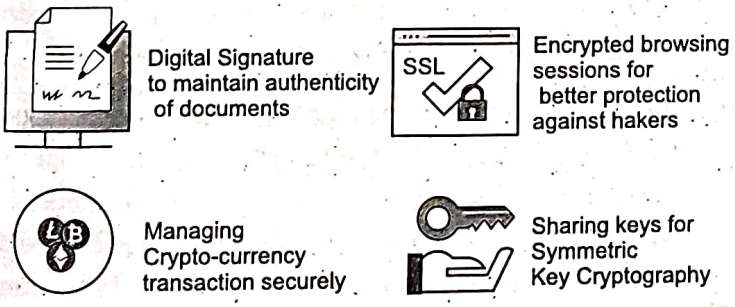


Fig. 1.17

- ➢ **Digital Signatures:** Verification of document origin and signature authenticity is possible today thanks to asymmetric key cryptography.
- ➢ **TLS/SSL Handshake:** Asymmetric key cryptography plays a significant role in verifying website server authenticity, exchanging the necessary encryption keys required and generating a session using those keys to ensure maximum security. Instead of the rather insecure HTTP website format.
- ➢ **Cryptocurrency:** Asymmetric key cryptography uses Blockchain Technology to authorize Cryptocurrency transactions and maintain the integrity of its decentralized architecture.
- ➢ **Key Sharing:** This cryptography category can also be used to exchange secret keys for symmetric encryption since keeping such keys private is of utmost importance in its system.

Why is Asymmetric Key Cryptography Called Public Key Cryptography?

- The most significant advantage of using asymmetric key cryptography over symmetric encryption is the non-reliance on a single point of failure key. Since the key used to encrypt is already public, the key used to decrypt the data is supposed to be private and need not be shared. Asymmetric key cryptography is also called public-key cryptography because of its open nature.
- This contrasts with symmetric encryption, where the single key used for both encryption and decryption is supposed to be kept secret, hence called private key cryptography.
- Now, have a look at the most widely used asymmetric key cryptography algorithm today.

What is RSA Encryption?

- RSA encryption is an asymmetric encryption algorithm named after its founders (Rivest, Shamir & Adleman) that uses block cipher methodology to encrypt data. It has been adopted worldwide in many industries like VPNs, chat applications, browser authorization and email encryption, to name a few.

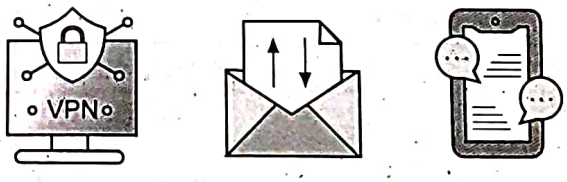


Fig. 1.18

- RSA algorithm is also used for the signing of documents, where the sender can sign a document using his own private key and the receiver verifies the document using the sender's public key. Since both keys are mathematically linked, it is impossible to replace either of the keys with a fraudulent piece.

What are the Advantages of Using Asymmetric Key Cryptography?

- Asymmetric encryption has a few advantages over symmetric encryption, which uses a single key for encryption and decryption of data. Some of these advantages are:

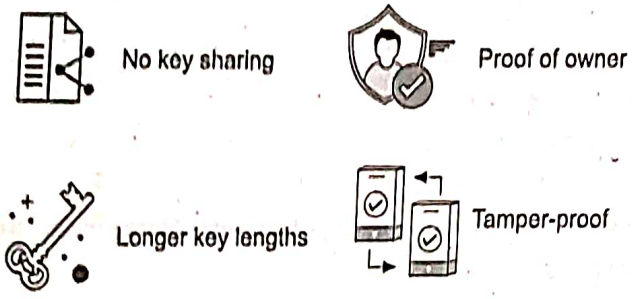


Fig. 1.19

- **No Key Sharing:** Asymmetric key cryptography overcomes the biggest flaw of symmetric key cryptography since it does not need to exchange any keys that can decrypt data.
- **Proof of Owner:** Since it links the private and public keys together, a message is decrypted using a private key. It stands as evidence that the message originated from the rightful owner who has the private key.
- **Longer Key Lengths:** Asymmetric encryption algorithms have key sizes up to 4096 bits that significantly increase the cipher and ciphertext security.
- **Tamper-Proof:** Hackers can not modify data during transmission since doing so will prevent the receiver's private key from decrypting the message, thus informing the receiver that the message has been meddled with.

1.4 ELLIPTIC CURVE CRYPTOGRAPHY (ECC)

- Elliptic Curve Cryptography (ECC) is a key-based technique for encrypting data. ECC focuses on pairs of public and private keys for decryption and encryption of web traffic.
- ECC is frequently discussed in the context of the Rivest–Shamir–Adleman (RSA) cryptographic algorithm. RSA achieves one-way encryption of things like emails, data and software using prime factorization.

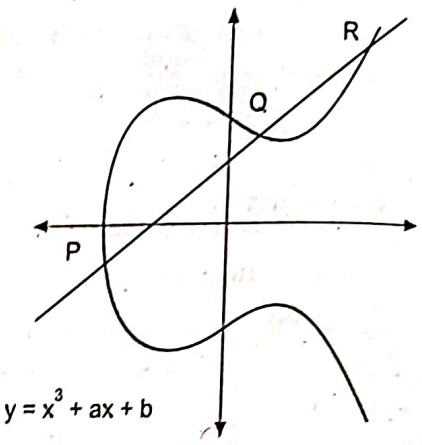


Fig. 1.20

What is Elliptic Curve Cryptography?

- ECC, an alternative technique to RSA, is a powerful cryptography approach. It generates security between key pairs for public key encryption by using the mathematics of elliptic curves.
- RSA does something similar with prime numbers instead of elliptic curves, but ECC has gradually been growing in popularity recently due to its smaller key size and ability to maintain security. This trend will probably continue as the demand on devices to remain secure increases due to the size of keys growing drawing on scarce mobile resources. This is why it is so important to understand elliptic curve cryptography in context.
- In contrast to RSA, ECC bases its approach to public key cryptographic systems on how elliptic curves are structured algebraically over finite fields. Therefore, ECC creates keys that are more difficult mathematically, to crack. For this reason, ECC is considered to be the next generation implementation of public key cryptography and more secure than RSA.
- It also makes sense to adopt ECC to maintain high levels of both performance and security. That is because ECC is increasingly in wider use as websites strive for greater online security in customer data and greater mobile optimization, simultaneously. More sites using ECC to secure data means a greater need for this kind of quick guide to elliptic curve cryptography.
- An elliptic curve for current ECC purposes is a plane curve over a finite field which is made up of the points satisfying the equation:

$$y^2 = x^3 + ax + b$$

- In this elliptic curve cryptography example, any point on the curve can be mirrored over the x-axis and the curve will stay the same. Any non-vertical line will intersect the curve in three places or fewer.

Elliptic Curve Cryptography vs RSA

- The difference in size to security yield between RSA and ECC encryption keys is notable. The Table 1.4 below shows the sizes of keys needed to provide the same level of security. In other words, an elliptic curve cryptography key of 384 bit achieves the same level of security as an RSA of 7680 bit.

Table 1.4

RSA Key Length (bit)	ECC Key Length (bit)
1024	160
2048	224
3072	256
7680	384
15360	521

- There is no linear relationship between the sizes of ECC keys and RSA keys. That is, an RSA key size that is twice as big does not translate into an ECC key size that is doubled. This compelling difference shows that ECC key generation and signing are substantially quicker than for RSA and also that ECC uses less memory than does RSA.
- Also, unlike in RSA, where both are integers, in ECC the private and public keys are not equally exchangeable. Instead, in ECC the public key is a point on the curve, while the private key is still an integer.
- A quick comparison of the advantages and disadvantages of ECC and RSA algorithms looks like this:
  - ➢ ECC features smaller ciphertexts, keys and signatures, and faster generation of keys and signatures. Its decryption and encryption speeds are moderately fast. ECC enables lower latency than inverse throughout by computing signatures in two stages. ECC features strong protocols for authenticated key exchange and support for the tech is strong.
  - ➢ The main disadvantage of ECC is that it is not easy to securely implement. Compared to RSA, which is much simpler on both the verification and encryption sides, ECC is a steeper learning curve and a bit slower for accumulating actionable results.
  - ➢ However, the disadvantages of RSA catch up with you soon. Key generation is slow with RSA, and so is decryption and signing, which are not always that easy to implement securely.

Advantages of Elliptic Curve Cryptography

- Public-key cryptography works using algorithms that are easy to process in one direction and difficult to process in the reverse direction. For example, RSA relies on the fact that multiplying prime numbers to get a larger number is easy, while factoring huge numbers back to the original primes is much more difficult.
- However, to remain secure, RSA needs keys that are 2048 bits or longer. This makes the process slow and it also means that key size is important.
- Size is a serious advantage of elliptic curve cryptography, because it translates into more power for smaller, mobile devices. It is far simpler and requires less energy to factor than it is to solve for an elliptic curve discrete logarithm, so for two keys of the same size, RSA's factoring encryption is more vulnerable.
- Using ECC, you can achieve the same security level using smaller keys. In a world where mobile devices must do more and more cryptography with less computational power, ECC offers high security with faster, shorter keys compared to RSA.

How Secure is Elliptic Curve Cryptography?

- There are several potential vulnerabilities to elliptic curve cryptography, including side-channel attacks and twist-security attacks. Both types aim to invalidate the ECC's security for private keys.
- Side-channel attacks including differential power attacks, fault analysis, simple power attacks and simple timing attacks, typically result in information leaks. Simple countermeasures exist for all types of side-channel attacks.
- An additional type of elliptic curve attack is the twist-security attack or fault attack. Such attacks may include invalid-curve attacks and small-subgroup attacks, and they may result in the private key of the victim leaking out. Twist-security attacks are typically simply mitigated with careful parameter validation and curve choices.
- Although there are certain ways to attack ECC, the advantages of elliptic curve cryptography for wireless security mean it remains a more secure option.

What is an Elliptic Curve Digital Signature?

- An Elliptic Curve Digital Signature Algorithm (ECDSA) uses ECC keys to ensure each user is unique and every transaction is secure. Although this kind of Digital Signing Algorithm (DSA) offers a functionally indistinguishable outcome as other DSAs, it uses the smaller keys you had expect from ECC and therefore is more efficient.

What is Elliptic Curve Cryptography Used For?

- ECC is among the most commonly used implementation techniques for digital signatures in cryptocurrencies. Both Bitcoin and Ethereum apply the Elliptic Curve Digital Signature Algorithm (ECDSA) specifically in signing transactions.
- However, ECC is not used only in cryptocurrencies. It is a standard for encryption that will be used by most web applications going forward due to its shorter key length and efficiency.

1.5 CRYPTOGRAPHIC 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 Fig. 1.21 illustrated hash function :

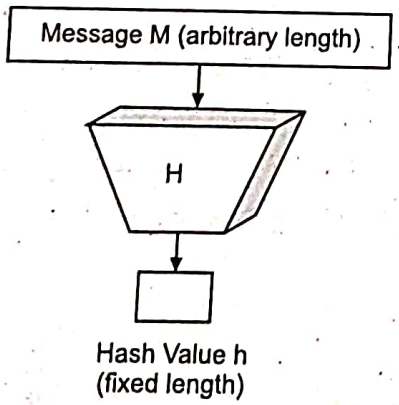


Fig. 1.21

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 :

- **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 Fig. 1.22 demonstrates hash function :

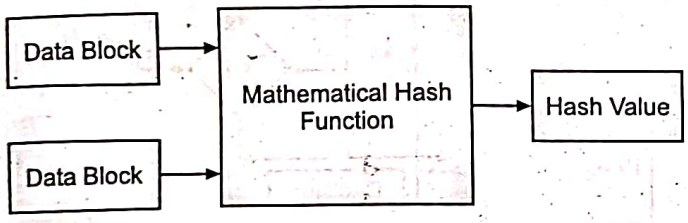


Fig. 1.22

- 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 Fig. 1.23.

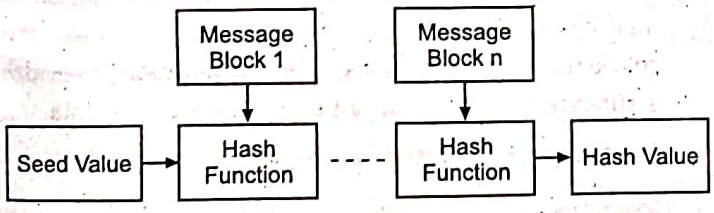


Fig. 1.23

- 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 RIPEMD 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 RIPEMD, 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.

(i) Password Storage

- Hash functions provide protection to passwo storage.
  - Instead of storing password in clear, mostly logon processes store the hash values passwords in the file.
  - The Password file consists of a table of pairs whic are in the form (user id, h(P)).
  - The process of logon is depicted in the followin Fig. 1.24.

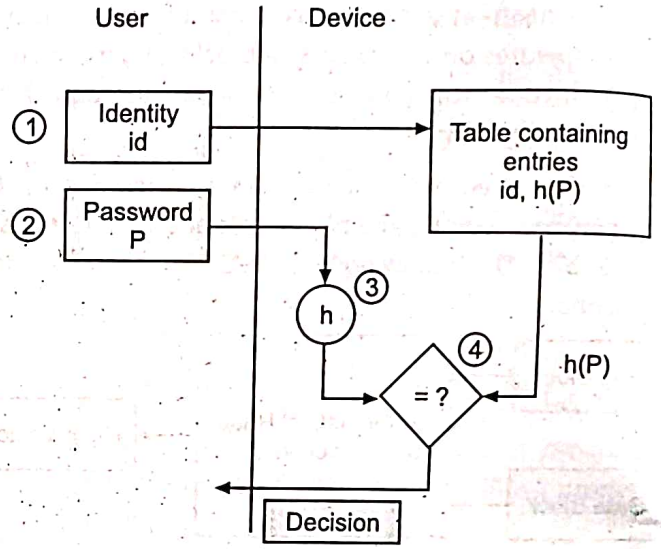


Fig. 1.24

  - 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.

(ii) Data Integrity Check

- 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 Fig. 1.25.

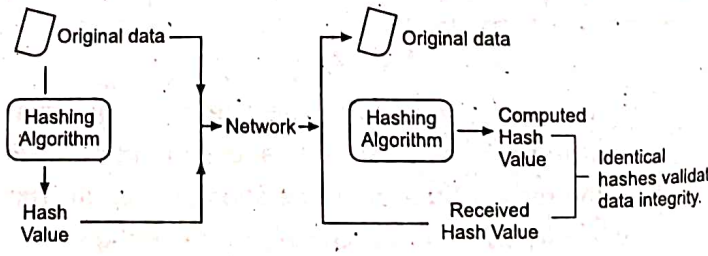


Fig. 1.25

- 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 check application is useful only if the user is sure about the originality of file.

SHA 256

- Among the many advancements seen in network security, encryption and hashing have been the core principles of additional security modules. The secure hash algorithm with a digest size of 256 bits or the SHA 256 algorithm, is one of the most widely used hash algorithms. While there are other variants, SHA 256 has been at the forefront of real-world applications.
- To understand the working of the SHA 256 algorithm, you need first to understand hashing and its functional characteristics.

What is Hashing?

- Hashing is the process of scrambling raw information to the extent that it cannot reproduce it back to its original form. It takes a piece of information and passes it through a function that performs mathematical operations on the plaintext. This function is called the hash function and the output is called the hash value/digest.

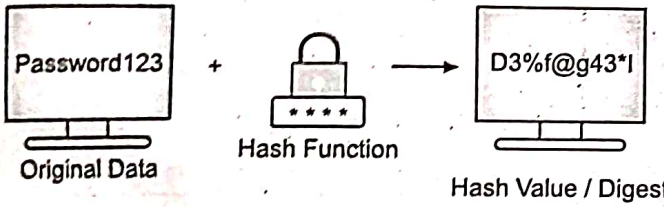


Fig. 1.26

- As seen from the above image, the hash function is responsible for converting the plaintext to its respective hash digest. They are designed to be irreversible, which means your digest should not provide you with the original plaintext by any means necessary. Hash functions also provide the same output value if the input remains unchanged, irrespective of the number of iterations.
- There are two primary applications of hashing:
  - (i) **Password Hashes:** In most website servers, it converts user passwords into a hash value before being stored on the server. It compares the hash value re-calculated during login to the one stored in the database for validation.

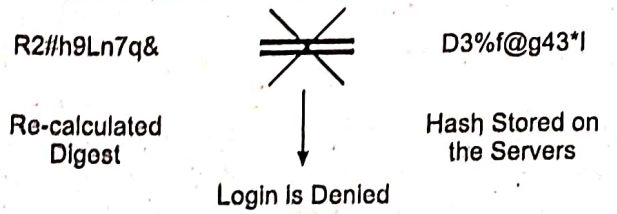


Fig. 1.27

  - (ii) **Integrity Verification:** When it uploads a file to a website, it also shared its hash as a bundle. When a user downloads it, it can recalculate the hash and compare it to establish data integrity.

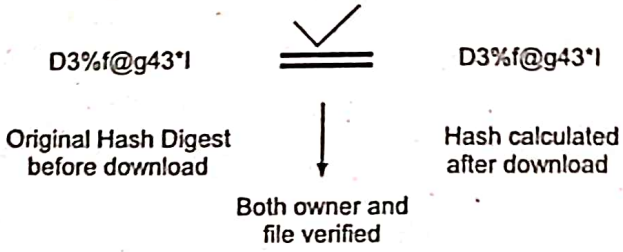


Fig. 1.28

- Now that you understand the working of hash functions, look at the key topic in hand - SHA 256 algorithm.

What is the SHA-256 Algorithm?

- SHA 256 is a part of the SHA 2 family of algorithms, where SHA stands for Secure Hash Algorithm. Published in 2001, it was a joint effort between the NSA and NIST to introduce a successor to the SHA 1 family, which was slowly losing strength against brute force attacks.
- The significance of the 256 in the name stands for the final hash digest value, i.e. irrespective of the size of plaintext/cleartext, the hash value will always be 256 bits.
- The other algorithms in the SHA family are more or less similar to SHA 256. Now, look into knowing a little more about their guidelines.

What are the Characteristics of the SHA-256 Algorithm?

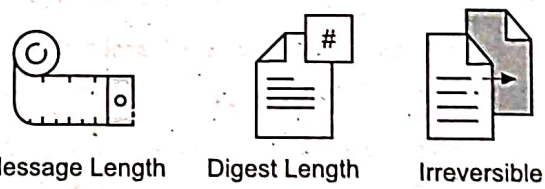


Fig. 1.29

Some of the standout features of the SHA algorithm are as follows:

- (i) **Message Length:** The length of the cleartext should be less than 264 bits. The size needs to be in the comparison area to keep the digest as random as possible.
- (ii) **Digest Length:** The length of the hash digest should be 256 bits in SHA 256 algorithm, 512 bits in SHA-512 and so on. Bigger digests usually suggest significantly more calculations at the cost of speed and space.

(iii) Irreversible: By design, all hash functions such as the SHA 256 are irreversible. You should neither get a plaintext when you have the digest beforehand nor should the digest provide its original value when you pass it through the hash function again.

Now that you got a fair idea about the technical requirements for SHA, you can get into its complete procedure, in the next section.

Steps in SHA-256 Algorithm

You can divide the complete process into five different segments, as mentioned below:

1. Padding Bits

It adds some extra bits to the message, such that the length is exactly 64 bits short of a multiple of 512. During the addition, the first bit should be one and the rest of it should be filled with zeroes.

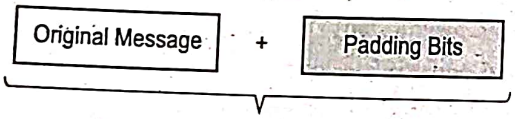


Fig. 1.30

2. Padding Length

- You can add 64 bits of data now to make the final plaintext a multiple of 512. You can calculate these 64 bits of characters by applying the modulus to your original cleartext without the padding.

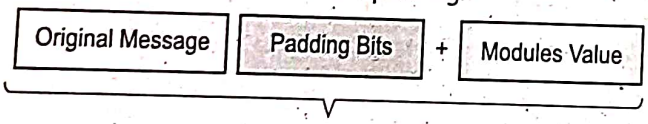


Fig. 1.31

3. Initializing the Buffers

- You need to initialize the default values for eight buffers to be used in the rounds as follows:

a = 0x6a09e667

b = 0xbb67ae85

c = 0x3c6ef372

d = 0xa54ff53a

e = 0x510e527f

f = 0z9b05688c

g = 0x1f83d9ab

h = 0x5be0cd19

You also need to store 64 different keys in an array ranging from K[0] to K[63]. They are initialized as follows:

k[0..63]:=

0x428a2f98,	0x71374491,	0xb5c0fbcf,	0xe9b5dba5,	0x3956c25b,	0x59f111f1,	0x923f82a4,	0xab1c5ed5,
0xd807aa98,	0x12835b01,	0x243185be,	0x550c7dc3,	0x72be5d74,	0x80deb1fe,	0x9bdc06a7,	0xc19bf174,
0xe49b69c1,	0xefbe4786,	0x0fc19dc6,	0x240ca1cc,	0x2de92c6f,	0x4a7484aa,	0x5cb0a9dc,	0x76f988da,
0x983e5152,	0xa831c66d,	0xb00327c8,	0xbf597fc7,	0xc6e00bf3,	0xd5a79147,	0x06ca6351,	0x14292967,
0x27b70a85,	0x2e1b2138,	0x4d2c6dfc,	0x53380d13,	0x650a7354,	0x766a0abb,	0x81c2c92e,	0x92722c85,
0xa2bfe8a1,	0xa81a664b,	0xc24b8b70,	0xc76c51a3,	0xd192e819,	0xd6990624,	0xf40e3585,	0x106aa070,
0x19a4c116,	0x1e376c08,	0x2748774c,	0x34b0bcb5,	0x391c0cb3,	0x4ed8aa4a,	0x5b9cca4f,	0x682e6ff3,
0x748f82ee,	0x78a5636f,	0x84c87814,	0x8cc70208,	0x90befffa,	0xa4506ceb,	0xbef9a3f7,	0xc67178f2,

4. Compression Functions

- The entire message gets broken down into multiple blocks of 512 bits each. It puts each block through 64 rounds of operation, with the output of each block serving as the input for the following Fig. 1.32. The entire process is as follows:

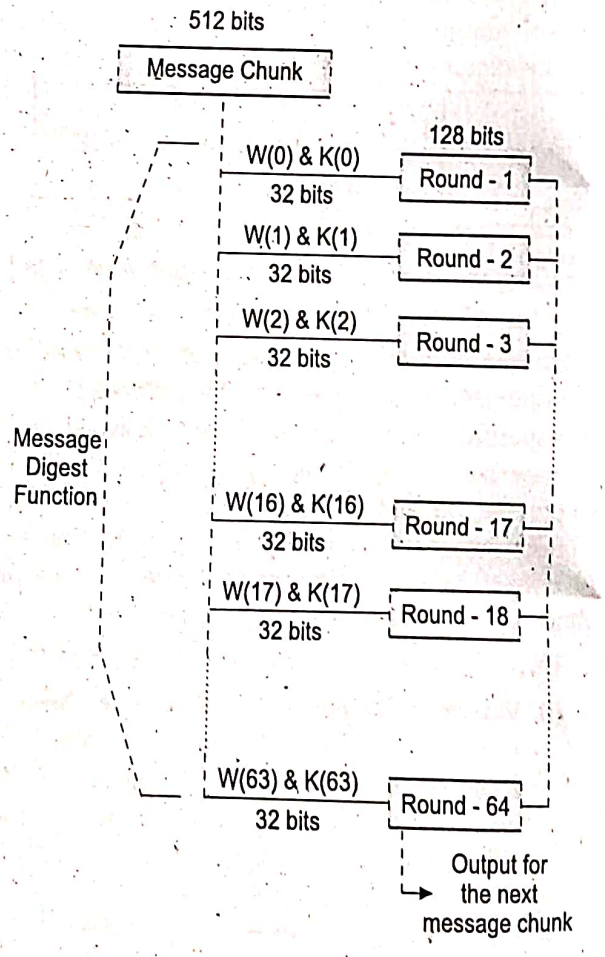


Fig. 1.32

- While the value of K[i] in all those rounds is pre-initialized, W[i] is another input that is calculated individually for each block, depending on the number of iterations being processed at the moment.

5. **Output**

- With each iteration, the final output of the block serves as the input for the next block. The entire cycle keeps repeating until you reach the last 512-bit block, and you then consider its output the final hash digest. This digest will be of the length 256-bit, as per the name of this algorithm.
- With the SHA 256 algorithm being implemented thoroughly since the early 90s, there are specific applications that you can look into. You will see them in the next section.

Applications of SHA Algorithm

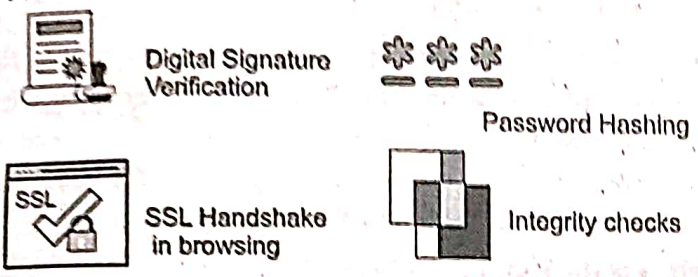


Fig. 1.33

- As seen in the image above, the SHA algorithm is being used in a lot of places, some of which are as follows:
  - **Digital Signature Verification:** Digital signatures follow asymmetric encryption methodology to verify the authenticity of a document/file. Hash algorithms like SHA 256 go a long way in ensuring the verification of the signature.
  - **Password Hashing:** As discussed above, websites store user passwords in a hashed format for two benefits. It helps foster a sense of privacy and it lessens the load on the central database since all the digests are of similar size.
  - **SSL Handshake:** The SSL handshake is a crucial segment of the web browsing sessions and it is done using SHA functions. It consists of your web browsers and the web servers agreeing on encryption keys and hashing authentication to prepare a secure connection.
  - **Integrity Checks:** As discussed above, verifying file integrity has been using variants like SHA 256 algorithm and the MD5 algorithm. It helps maintain the full value functionality of files and makes sure they were not altered in transit.

1.6 DIGITAL SIGNATURE ALGORITHM (DSA)

- The Covid-19 pandemic has given a new life to the work-from-home initiative, taking the corporate world into an untapped phase. Without a doubt, most of the users reading this have had to digitally sign some official documents over the past couple of years because of the lack of face-to-face interaction and standard distance constraints. To maintain the authenticity and integrity of such documents holding critical information, the DSA Algorithm was proposed and passed as a global standard for verifying digital signatures.
- You utilize two distinct keys in asymmetric encryption methods, one for encryption and the other for decryption. You use the public key for encryption; meanwhile, you use the private key for decryption. However, you must generate both keys from the receiver's end.

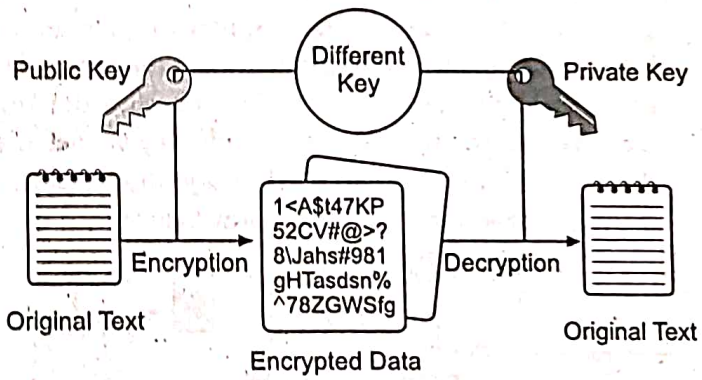


Fig. 1.34

- Using separate keys for encryption and decryption, as seen in the Fig. 1.34 has helped eliminate key exchange, as seen in the case of symmetric encryption.

  For example, if Alice needs to send a message to Bob, both the private and public keys must belong to Bob.

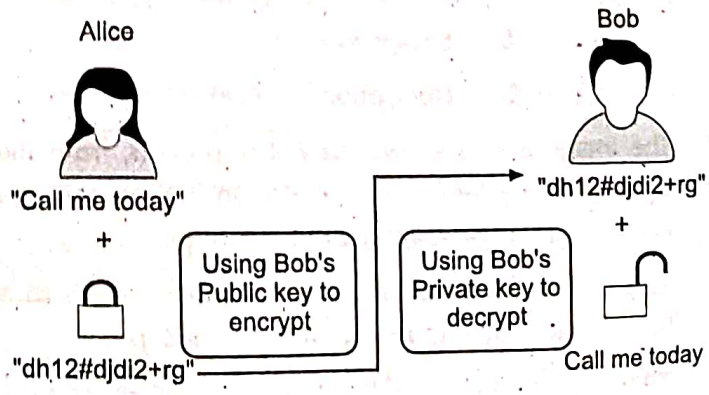


Fig. 1.35

The process for the above image is as follows:

Step 1 : Alice first uses Bob's public key to encrypt the message.

Step 2 : The encrypted message reaches Bob.

Step 3 : Bob decrypts the message with his secret key.

- This eliminates the requirement for the sender and recipient to exchange any secret keys, minimizing the window of opportunity for exploitation.
- Now that you learned how asymmetric encryption happens, you will look at how the digital signature architecture is set up.
- The objective of digital signatures is to authenticate and verify documents and data. This is necessary to avoid tampering and digital modification or forgery during the transmission of official documents.
- With one exception, they work on the public key cryptography architecture. Typically, an asymmetric key system encrypts using a public key and decrypts with a private key. For digital signatures, however, the reverse is true. The signature is encrypted using the private key and decrypted with the public key. Because the keys are linked, decoding it with the public key verifies that the proper private key was used to sign the document, thereby verifying the signature's provenance.

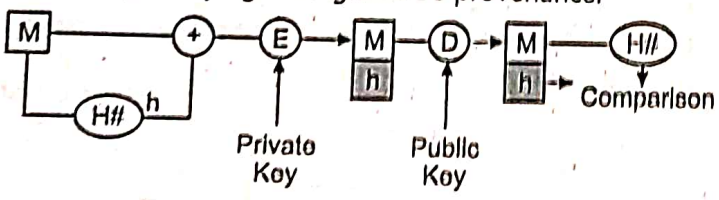


Fig. 1.36

M - Plaintext

H - Hash function

h - Hash digest

'+' - Bundle both plaintext and digest

E - Encryption

D - Decryption

- The image above shows the entire process, from the signing of the key to its verification. So, go through each step to understand the procedure thoroughly.

Step 1 : M, the original message is first passed to a hash function denoted by H# to create a digest.

Step 2 : Next, it bundles the message together with the hash digest h and encrypts it using the sender's private key.

Step 3 : It sends the encrypted bundle to the receiver who can decrypt it using the sender's public key.

Step 4 : Once it decrypts the message, it is passed through the same hash function (H#), to generate a similar digest.

Step 5 : It compares the newly generated hash with the bundled hash value received along with the message. If they match, it verifies data integrity.

- There are two industry-standard ways to implement the above methodology. They are:

  (I) RSA Algorithm

  (II) DSA Algorithm

- Both the algorithms serve the same purpose, but the encryption and decryption functions differ quite a bit. So, now that you understand how it is supposed to function while verifying the signature, let's deep dive into our focus for today, the DSA Algorithm.

What is the DSA Algorithm?

- Digital Signatures Algorithm is a FIPS (Federal Information Processing Standard) for digital signatures. It was proposed in 1991 and globally standardized in 1994 by the National Institute of Standards and Technology (NIST). It functions on the framework of modular exponentiation and discrete logarithmic problems, which are difficult to compute as a force-brute system.
- DSA Algorithm provides three benefits, which are as follows:

  (I) Message Authentication: You can verify the origin of the sender using the right key combination.

  (II) Integrity Verification: You cannot tamper with the message since it will prevent the bundle from being decrypted altogether.

  (III) Non-repudiation: The sender cannot claim they never sent the message if verifies the signature.

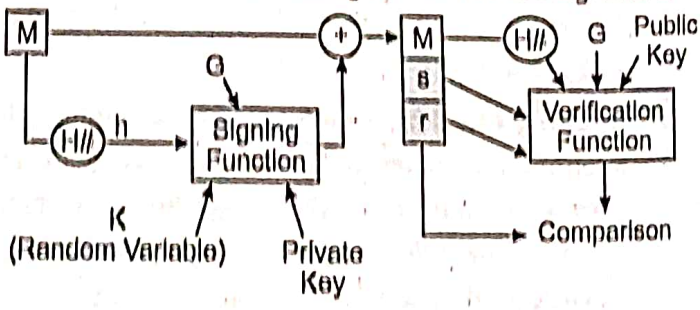


Fig. 1.37

- The image above shows the entire procedure of the DSA algorithm. You will use two different functions here, a signing function and a verification function. The difference between the image of a typical digital signature verification process and the one above is the encryption and decryption part. They have distinct parameters, which you will look into in the next section of this lesson on the DSA Algorithm.

Steps in DSA Algorithm

Keeping the image above in mind, go ahead and see how the entire process works, starting from creating the key pair to verifying the signature at the end.

1. Key Generation

- You first choose a prime number q which is known as the prime divisor.
- Another prime number, p is chosen such that p-1 mod q = 0.
- Choose an integer g (1<g<p), satisfying the two conditions, g**q mod p = 1 and g = h**((p–1)/q) mod p
- x is our private key and it is a random integer such that 0 < x < q.
- y is our public key and you can calculate it as y = gx mod p.
- Now the private key package is {p,q,g,x}.
- The public key package is {p,q,g,y}.

2. Signature Generation

- It passes the original message (M) through the hash function.(H#) to get our hash digest(h).
- It passes the digest as input to a signing function, whose purpose is to give two variables as output, s and r.
- Apart from the digest, you also use a random integer k such that 0 < k < q.
- To calculate the value of r, you use the formula r = (gk mod p) mod q.
- To calculate the value of s, you use the formula s = [K-1(h+x .R)mod q].
- It then packages the signature as {r,s}.
- The entire bundle of the message and signature (M,r,s) are sent to the receiver.

3. Signature Verification

- You use the same hash function (H#) to generate the digest h.
- You then pass this digest off to the verification function, which needs other variables as parameters too.
- Compute the value of w such that: s*w mod q = 1
- Calculate the value of u1 from the formula, u1 = h*w mod q
- Calculate the value of u2 from the formula, u2 = r*w mod q
- The final verification component v is calculated as v = [((gu1 , yu2) mod p) mod q].
- It compares the value of v to the value of r received in the bundle.
- If it matches, the signature verification is complete.
- Having understood the functionality of the DSA Algorithm, you must know the advantages this algorithm offers over alternative standards like the RSA algorithm.

Advantages of DSA

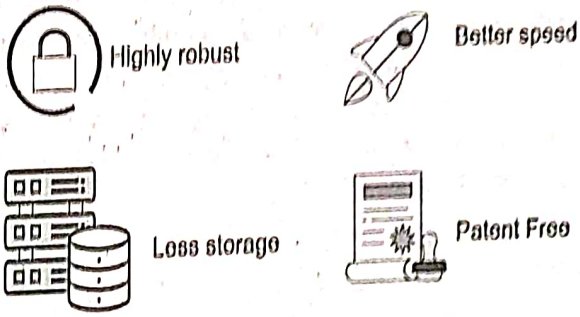


Fig. 1.38

- **Highly Robust:** DSA is highly robust in the security and stability aspect compared to alternative signature verification algorithms.
- **Better Speed:** The key generation is much faster compared to the RSA algorithm and such.
- **Less Storage:** DSA requires less storage space to work its entire cycle.
- **Patent Free:** When NIST released it, it was patent-free to enable its global use free of cost.

1.7 MERKEL TREE

- Merkle tree is a fundamental part of Blockchain Technology. It is a mathematical **data structure** composed of hashes of different blocks of data and which serves as a summary of all the transactions in a

block. It also allows for efficient and secure verification of content in a large body of data. It also helps to verify the consistency and content of the data. Both Bitcoin and Ethereum use Merkle Trees structure. Merkle Tree is also known as **Hash Tree**.

- The concept of Merkle Tree is named after **Ralph Merkle** who patented the idea in **1979**. Fundamentally, it is a data structure tree in which every **leaf node** labelled with the hash of a data block and the **non-leaf node** labelled with the cryptographic hash of the labels of its child nodes. The leaf nodes are the lowest node in the tree.

How do Merkle Trees Work?

- A Merkle tree stores all the transactions in a block by producing a digital fingerprint of the entire set of transactions. It allows the user to verify whether a transaction can be included in a block or not.
- Merkle trees are created by repeatedly calculating hashing pairs of nodes until there is only one hash left. This hash is called the Merkle Root or the Root Hash. The Merkle Trees are constructed in a bottom-up approach.
- Every leaf node is a hash of transactional data and the non-leaf node is a hash of its previous hashes. Merkle trees are in a binary tree, so it requires an even number of leaf nodes. If there is an odd number of transactions, the last hash will be duplicated once to create an even number of leaf nodes.

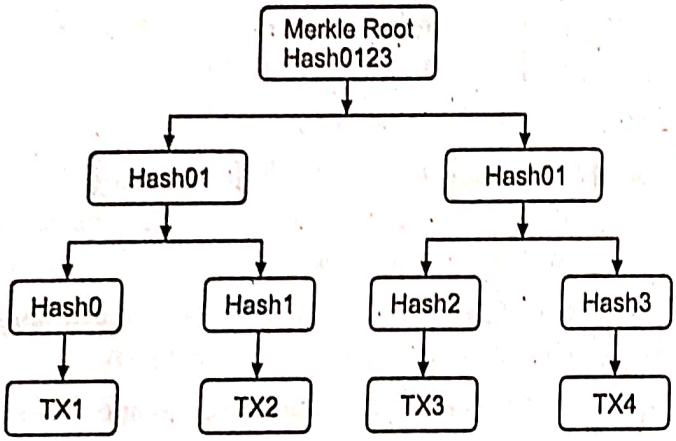


Fig. 1.39

- The above example is the most common and simple form of a Merkle tree, i.e., **Binary Merkle Tree**. There are four transactions in a block: **TX1**, **TX2**, **TX3** and **TX4**. Here you can see, there is a top hash which is the hash of the entire tree, known as the **Root Hash** or the **Merkle Root**. Each of these is repeatedly hashed, and stored in each leaf node, resulting in Hash 0, 1, 2 and 3. Consecutive pairs of leaf nodes are then summarized into a parent node by hashing **Hash0** and **Hash1** resulting in **Hash01**, and separately hashing **Hash2** and **Hash3** resulting in **Hash23**. The two hashes (**Hash01** and **Hash23**) are then hashed again to produce the Root Hash or the Merkle Root.
- Merkle Root is stored in the **block header**. The block header is the part of the Bitcoin block which gets hashed in the process of mining. It contains the hash of the last block, a Nonce and the Root Hash of all the transactions in the current block in a Merkle Tree. So having the Merkle root in block header makes the transaction **tamper-proof**. As this Root Hash includes the hashes of all the transactions within the block these transactions may result in saving the disk space.

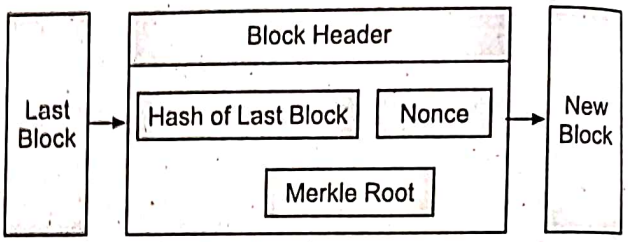


Fig. 1.40

- The Merkle Tree maintains the **integrity** of the data. If any single detail of transactions or order of the transaction's changes, then these changes reflected in the hash of that transaction. This change would cascade up the Merkle Tree to the Merkle Root changing the value of the Merkle root and thus invalidating the block. So everyone can see that Merkle tree allows for a quick and simple test of whether a specific transaction is included in the set or not.

Merkle Trees have Three Benefits

- It provides a means to maintain the integrity and validity of data.
- It helps in saving the memory or disk space as the proofs, computationally easy and fast.
- Their proofs and management require tiny amounts of information to be transmitted across networks.

CASE STUDY: COMPARE THE SYMMETRIC AND ASYMMETRIC CRYPTOGRAPHY ALGORITHMS

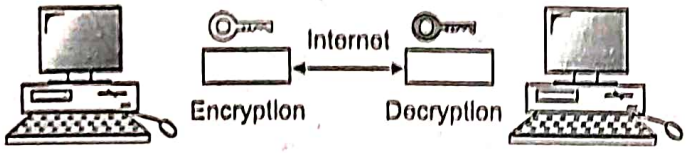
- Encryption means that the sender converts original information into another form and sends the unintelligible message over the network. It helps us to secure data that we send, receives and store. Data can be text messages saved on our cell phone, logs stored

on our fitness watch and details of banking sent by your online account.

- It is the procedure of taking ordinary text, such as a text or email and transforming it into an unreadable type of format known as "cipher text." The ciphertext is converted back to the real form when the recipient accesses the message, which is known as decryption. It helps to protect the digital information either saved on or spread through a network such as an Internet on computer systems.

Symmetric Encryption

Symmetric encryption encrypts and decrypts the information using a single password. In this encryption technique, the message is encrypted with a key, and the same key is used for decrypting the message. It is the simplest and commonly known encryption technique. It makes it easy to use but less secure.



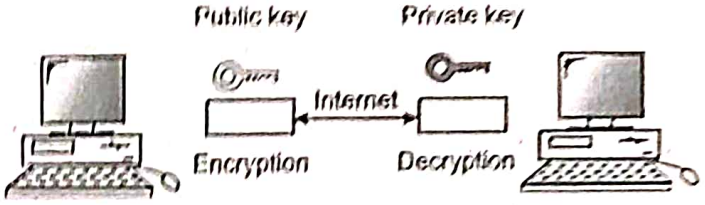
Symmetric Encryption

Fig. 1.41

- It is called symmetric encryption because the same key is responsible for encrypting or decrypting the data. The single key used in symmetric encryption is used to encrypt plain text into ciphertext and that same key is used to decrypt that ciphertext back into plain text.
- Symmetric encryption is also called secret key encryption. The algorithm behind the symmetric encryption executes faster and less complex, so it is the preferred technique to transmit the data in bulk.

Asymmetric Encryption

- Asymmetric encryption uses two keys for encryption and decryption. It is based on the technique of public and private keys. A public key, which is interchanged between more than one user. Data is decrypted by a private key, which is not exchanged. It is slower but more secure. The public key used in this encryption technique is available to everyone, but the private key used in it is not disclosed.



Asymmetric Encryption

Fig. 1.42

- The drawback of this encryption is that it takes more time than the symmetric encryption process. Asymmetric encryption is slower than secret-key encryption because, in secret key encryption, a single shared key is used to encrypt and decrypt the message, while in public-key encryption, two different keys are used, both related to each other by a complex mathematical process. Therefore, we can say that encryption and decryption take more time in public-key encryption.
- In asymmetric encryption, a message that is encrypted using a public key can be decrypted by a private key, while if the message is encrypted by a private key can be decrypted by using the public key. Asymmetric encryption is widely used in day-to-day communication channels, especially on the internet.
- That is about the description of both encryption techniques. Both encryption techniques have their own benefits and limitations, but from a security perspective, asymmetric encryption is a better choice. Now, let's see the comparison chart between both techniques. We are comparing asymmetric and symmetric encryption based on some characteristics.

Symmetric Encryption v/s Asymmetric Encryption

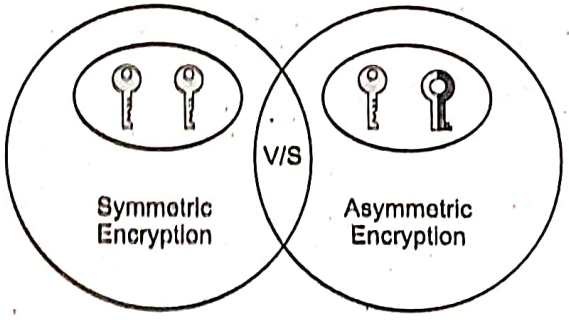


Fig. 1.43

Table 1.5

On the Basis of	Symmetric Encryption	Asymmetric Encryption
Keys used	It uses a single shared key (secret key) to encrypt and decrypt the message.	It uses two different keys for encryption and decryption.
Size	The size of ciphertext in symmetric encryption could be the same or smaller than the plain text.	The size of ciphertext in asymmetric encryption could be the same or larger than the plain text.
Efficiency	It is efficient as this technique is recommended for large amounts of text.	It is inefficient as this technique is used only for short messages.
Speed	The encryption process of symmetric encryption is faster as it uses a single key for encryption and decryption.	The encryption process in asymmetric encryption is slower as it uses two different keys; both keys are related to each other through the complicated mathematical process.
Purpose	Symmetric encryption is mainly used to transmit bulk data.	It is mainly used in smaller transactions. It is used for establishing a secure connection channel before transferring the actual data.
Security	It is less secured as there is a use of a single key for encryption.	It is safer as there are two keys used for encryption and decryption.
Algorithms	The algorithms used in symmetric encryption are 3DES, AES, DES and RC4.	RSA, DSA, Diffie-Hellman, ECC, Gamal and El.
Existence	It is an old technique.	It is a new technique.

EXERCISE

1. What is cryptography? Differentiate between cla cryptography and modern cryptography.
2. What are the various cryptography primitive Explain the role of these primitives in cryptography
3. Explain the various components of a ba cryptosystem.
4. What are the different types of cryptosystem?
5. What is symmetric key cryptography? Explain t working of symmetric key cryptography with suita diagram.
6. Enlist the various symmetric key algorithms.
7. Explain the various advantages and disadvantages symmetric key algorithms.
8. Why asymmetric key cryptography is called as pub key cryptography?
9. What is asymmetric key cryptography? Explain t working of asymmetric key cryptography wi suitable diagram.
10. What is elliptic curve cryptography? Explain th working of elliptic curve cryptography with suitab diagram.
11. What is hash function? Enlist the various features ar properties of Hash functions.
12. Explain the process of designing hashing algorithm?
13. Enlist the popular hash functions.
14. Enlist the applications of hash function.
15. What is SHA-256 algorithm? Describe th characteristics of SHA 256 algorithm.
16. Explain in brief various steps involved in SHA-25 algorithm.
17. What are the different applications of SHA 25 algorithm?
18. What is digital signature algorithm? Explain the working of digital signature algorithm with suitable diagram.
19. What are the different steps involved in DSA?
20. Enlist the advantages and disadvantages of DSA.
21. What is merkel tree? Explain the working of merkel tree with suitable diagram.
22. What is the significance of merkel tree in Blockchain?