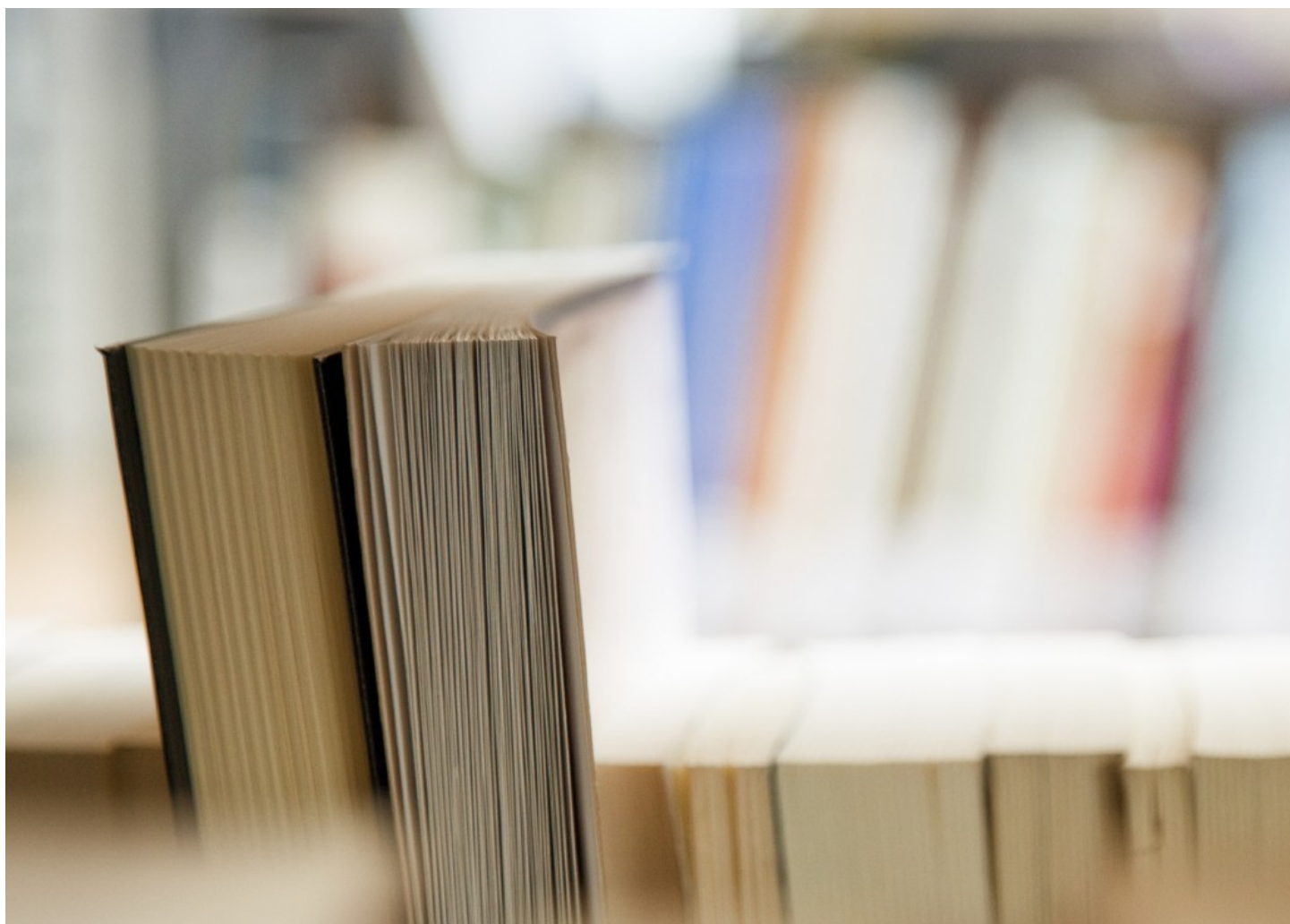




CRYPTUS CYBER
SECURITY PVT. LTD.



What is cryptography?

"Cryptography" comes from the Greek words **kryptos**, meaning "concealed, hidden, veiled, secret, or mysterious," and **graphia**, meaning "writing"; thus, cryptography is "the art of secret writing."

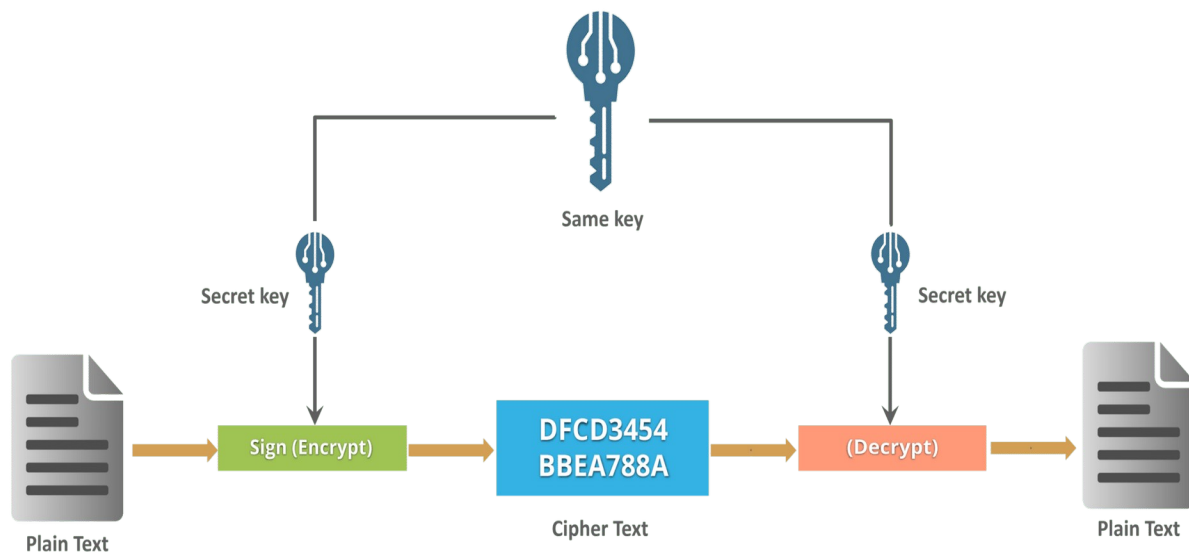
Cryptography is a method of protecting information and communications through the use of codes, so that only those for whom the information is intended can read and process it.

It's like putting a message in a locked box that only the intended recipient can open. It's used to make sure that only the right people can access and understand sensitive information, like passwords, messages, or financial data.

Objectives of Cryptography

- **Confidentiality:** Assurance that the information is accessible only to those authorized to access it.
- **Integrity:** Trustworthiness of data or resources in terms of preventing improper and unauthorized changes.
- **Authentication:** Assurance that the communication, document, or data is genuine.
- **Nonrepudiation:** Guarantee that the sender of a message cannot later deny having sent the message and that the recipient cannot deny having received the message.

How Cryptography Works:-



Step 1.

Encryption: The process starts with encryption, where the original information, known as plaintext, is transformed into a scrambled format called cipher text. This is done using mathematical algorithms and a secret key. The key is like a secret code that only the intended recipient knows.

Step 2.

Key: The key is a crucial part of the process. It determines how the encryption is done and how the cipher text will look. Without the correct key, it's nearly impossible to decipher the cipher text back into plaintext.

Step 3.

Transmission: The encrypted message or data can then be safely transmitted over insecure channels, like the internet or public networks, without worrying about eavesdroppers understanding the content.

Step 4.

Decryption: When the encrypted message reaches the intended recipient, they use the corresponding decryption key to reverse the encryption process. This transforms the cipher text back into its original plaintext form, making it readable and understandable again.

Basic Terms:-

1. **Plain Text :-** Is an unencrypted message
2. **Cipher Text :-** Is an Encrypted Message
3. **KEY:** - In cryptography, a key is a string of characters used within an encryption algorithm for altering data so that it appears random. Like a Physical Key, It locks (encrypts) data so that only someone with the right key can unlock (decrypt) it.

"Hello" +  = "KZ0KVey8l1c="

4. **Algorithm:** - set of mathematical and logical rules use in cryptography Function.

Difference between Encryption and Decryption:-

| Encryption | Decryption |
|---|---|
| 1. Encryption is the process of converting normal message into meaningless message. | While decryption is the process of converting meaningless message into its original form. |
| 2. Encryption is the process which take place at sender's end. | While decryption is the process which take place at receiver's end. |
| 3. Its major task is to convert the plain text into cipher text. | While its main task is to convert the cipher text into plain text. |
| 4. Any message can be encrypted with either secret key or public key. | Whereas the encrypted message can be decrypted with either secret key or private key. |
| 5. In encryption process, sender sends the data to receiver after encrypted it. | Whereas in decryption process, receiver receives the information (Cipher text) and convert into plain text. |

What is cryptanalysis?

Attackers may implement various cryptography attacks to evade the security of a cryptographic system by exploiting vulnerabilities in code, ciphers, cryptographic protocols, or key management schemes. This process is known as cryptanalysis.

Cryptanalysis is the study of ciphers, ciphertext, or cryptosystems with the ability to identify vulnerabilities in them and thus extract plaintext from ciphertext even if the cryptographic key or algorithm used to encrypt the plaintext is unknown.

This section deals with various cryptography attacks that an attacker performs to compromise cryptographic systems as well as various cryptanalysis techniques and tools that help in breaching cryptographic security.

Types of cryptanalysis:-

1. Known-Plaintext Analysis (KPA): In this type of attack, some plaintext-cipher text pairs are already known. Attacker maps them in order to find the encryption key. This attack is easier to use as a lot of information is already available.

2. Chosen-Plaintext Analysis (CPA): In this type of attack, the attacker chooses random plaintexts and obtains the corresponding cipher texts and tries to find the encryption key. It's very simple to implement like KPA but the success rate is quite low.

3. Cipher text-Only Analysis (COA): In this type of attack, only some cipher-text is known and the attacker tries to find the corresponding encryption key and plaintext. It's the hardest to implement but is the most probable attack as only cipher text is required

4. Man-In-The-Middle (MITM) attack: In this type of attack, attacker intercepts the message/key between two communicating parties through a secured channel.

5. Brute-force attack: This attack involves trying every possible key until the correct one is found. While this attack is simple to implement, it can be time-consuming and computationally expensive, especially for longer keys.

Cryptanalysis Methods

- **Linear Cryptanalysis**

Linear cryptanalysis is based on finding affine approximations to the action of a cipher. It is commonly used on block ciphers. This technique was invented by Mitsuru Matsui. It is a known plaintext attack and uses a linear approximation to describe the behavior of the block cipher. Given sufficient pairs of plaintext and corresponding ciphertext, bits of information about the key can be obtained. Obviously, the more pairs of plaintext and ciphertext one has, the greater are the chances of success.

Remember that cryptanalysis is an attempt to crack cryptography. For example, with the 56-bit data encryption standard (DES), **key brute-forcing could take up to 256 attempts**. Linear cryptanalysis requires 2^{43} known plaintexts. This is better than brute-forcing but is still impractical in most situations. The math may be a bit complex for novice cryptographers, but let us look at the basics of it.

With this method, a linear equation expresses the equality of two expressions, which consists of XORed binary variables. For example, the following equation XORs the sum of the first and third plaintext bits, and the first ciphertext bit is equal to the second bit of the key:

$$P_1 \oplus P_3 \oplus C_1 = K_2$$

You can use this method to slowly re-create the key that was used. After doing this for each bit, you will have an equation of the form

$$P_{i1} \oplus P_{i2} \oplus \dots \oplus C_{j1} \oplus C_{j2} \oplus \dots = K_{k1} \oplus K_{k2} \oplus \dots$$

You can then use **Matsui's Algorithm 2**, using known plaintext-ciphertext pairs, to guess the values of the key bits involved in the approximation. For each set of values of the key bits on the right-hand

side (referred to as a partial key), count how many times the approximation holds true over all the known plaintext-ciphertext pairs; call this count T. The partial key whose T has the greatest absolute difference from half the number of plaintext-ciphertext pairs is designated as the most likely set of values for those key bits.

- **Differential Cryptanalysis**

Differential cryptanalysis is a form of cryptanalysis applicable to symmetric-key algorithms. It was invented by Eli Biham and Adi Shamir. Essentially, it is the examination of differences in input and how that affects the resultant difference in the output. It originally worked only with chosen plaintext. It can also work with known **plaintext** and **ciphertext**.

- **Integral Cryptanalysis**

Integral cryptanalysis was first described by Lars Knudsen. This attack is particularly useful against block ciphers based on substitution-permutation networks as an extension of differential cryptanalysis. The differential analysis looks at pairs of inputs that differ in only one bit position, with all other bits being identical. Integral analysis for block size b holds $b-k$ bits constant and runs the other k bits through all 2^k possibilities. For $k = 1$, this is just differential cryptanalysis, but with $k > 1$, it is a new technique.

- **Quantum Cryptanalysis**

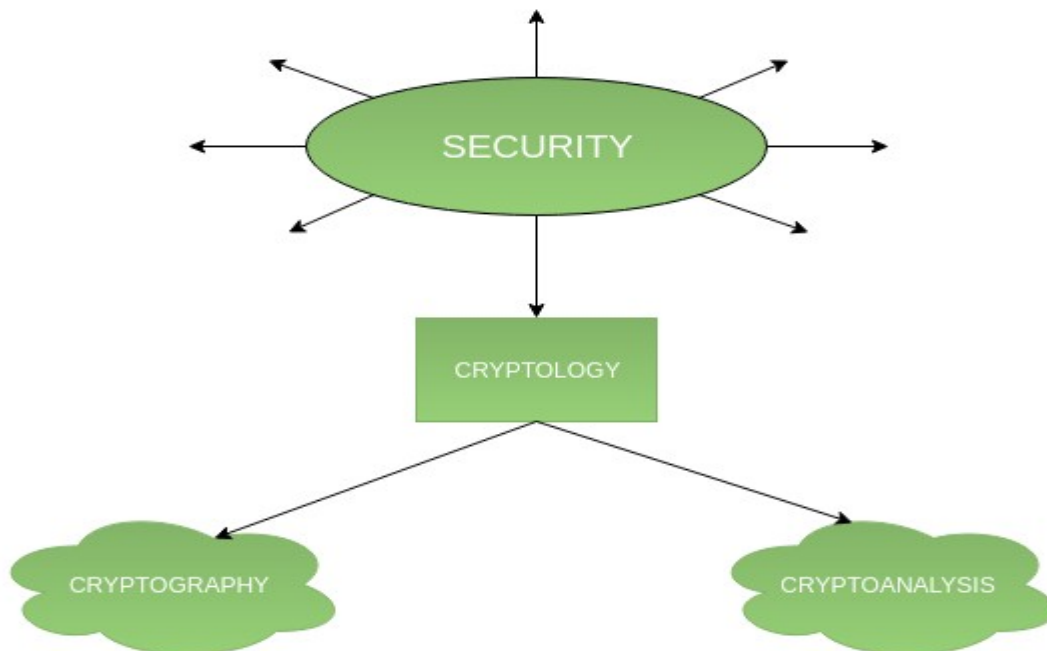
Quantum cryptanalysis is the process of cracking cryptographic algorithms using a quantum computer. Attackers can use Shor's quantum factoring algorithm on public-key cryptographic algorithms such as RSA and Elliptic Curve Diffie-Hellman (ECDH) to find the factors of large numbers in polynomial time and Grover's quantum search algorithm to make brute-force key search faster for block ciphers (AES) or hash functions (SHA).

To perform cryptanalysis, attackers must obtain the encrypted content, and the process requires significant time and quantum resources. Given below are the resources required to conduct cryptanalysis.

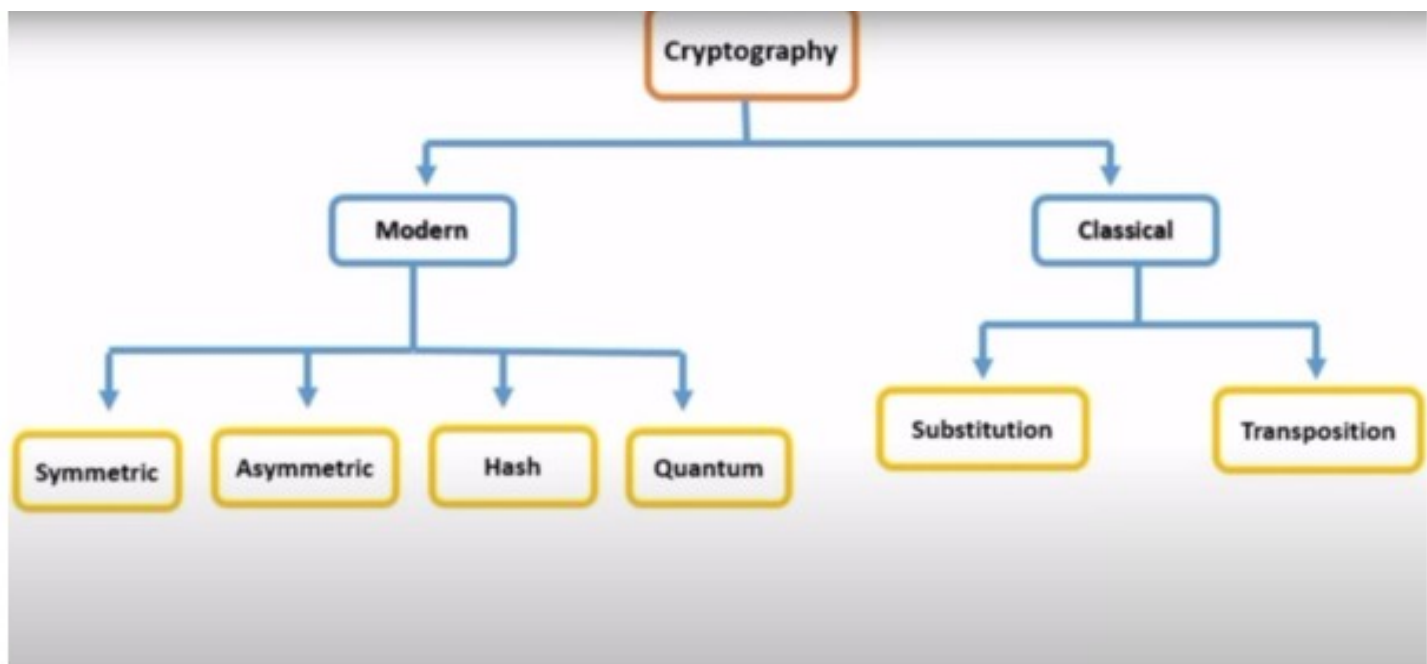
- **Circuit Width:** Specifies how many quantum bits or qubits are required in a time step
- **Circuit Depth:** Specifies the time steps required for a circuit
- **Number of Gates:** Specifies how many quantum gates are implemented in a circuit
- **Number of T-Gates:** Specifies how many T-gates are implemented in a circuit
- **T-Depth:** Specifies the time steps required for a T-gate
- **MAXDEPTH:** Specifies the maximum depth of a circuit (e.g., 240, 264, or 296)

What is cryptology?

Cryptology is the science of secure communications. Cryptology is a broader term that encompasses both cryptography and cryptanalysis. While cryptography focuses on the creation of secure communication through the use of codes and ciphers, cryptanalysis is the study of methods used to break those codes and ciphers, often by attempting to decipher encrypted messages without the proper key. In essence, cryptology involves the development of techniques for secure communication (cryptography) as well as the study of methods to break those techniques (cryptanalysis).



Types of Cryptography: -



Classical Cryptography:-

Substitution Technique in Cryptography

Substitution technique is a classical encryption technique where the characters present in the original message are replaced by the other characters or numbers or by symbols. If the plain text (original message) is considered as the string of bits, then the substitution technique would replace bit pattern of plain text with the bit pattern of cipher text.

Substitution Technique:-

Monoalphabetic Cipher:-

Monoalphabetic cipher is a substitution cipher, where the cipher alphabet for each plain text alphabet is fixed, for the entire encryption.

In simple words, if the alphabet 'p' in the plain text is replaced by the cipher alphabet 'd'. Then in the entire plain text wherever alphabet 'p' is used, it will be replaced by the alphabet 'd' to form the cipher text.

Example:-

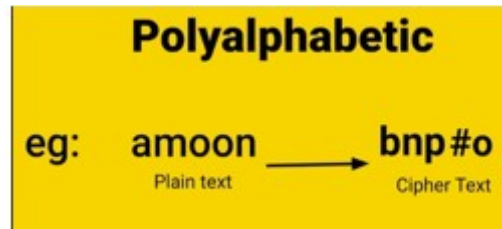


Polyalphabetic Cipher:-

Polyalphabetic cipher is far more secure than a monoalphabetic cipher. As monoalphabetic cipher maps a plain text symbol or alphabet to a cipher text symbol and uses the same cipher text symbol wherever that plain text occurs in the message.

But polyalphabetic cipher, each time replaces the plain text with the different cipher text.

Example:-



Polyalphabetic cipher is any cipher based on substitution, using multiple substitution alphabets. each alphabetic character of plain text is mapped on to a unique alphabetic character of a unique alphabetic character of a cipher text.

Difference between Polyalphabetic and Monoalphabetic:-

| Monoalphabetic | Polyalphabetic |
|---|---|
| 1) symbol in plain text is mapped to a fixed symbol in cipher text. O \Rightarrow P Boom \Rightarrow Cppn | 1) plain text and the characters in the cipher text is one-to-many O \Rightarrow P Boom \Rightarrow Cp *n |
| 2 Not Secure | 2 Secure |

Caesar Cipher

This is the simplest substitution cipher by Julius Caesar. In this substitution technique, to encrypt the plain text, each alphabet of the plain text is replaced by the alphabet three places further in it. And to decrypt the cipher text each alphabet of cipher text is replaced by the alphabet three places before it.

Let us take a simple example:

Plain Text: meet me tomorrow

Cipher Text: phhw ph wrpruurz

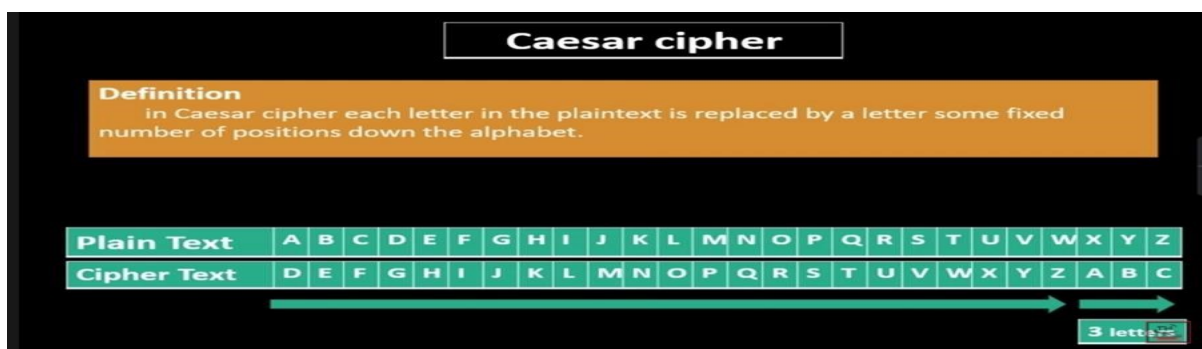
Look at the example above, we have replaced, 'm' with 'p' which occurs three places after, 'm'. Similarly, 'e' is replaced with 'h' which occurs in three places after 'e'.

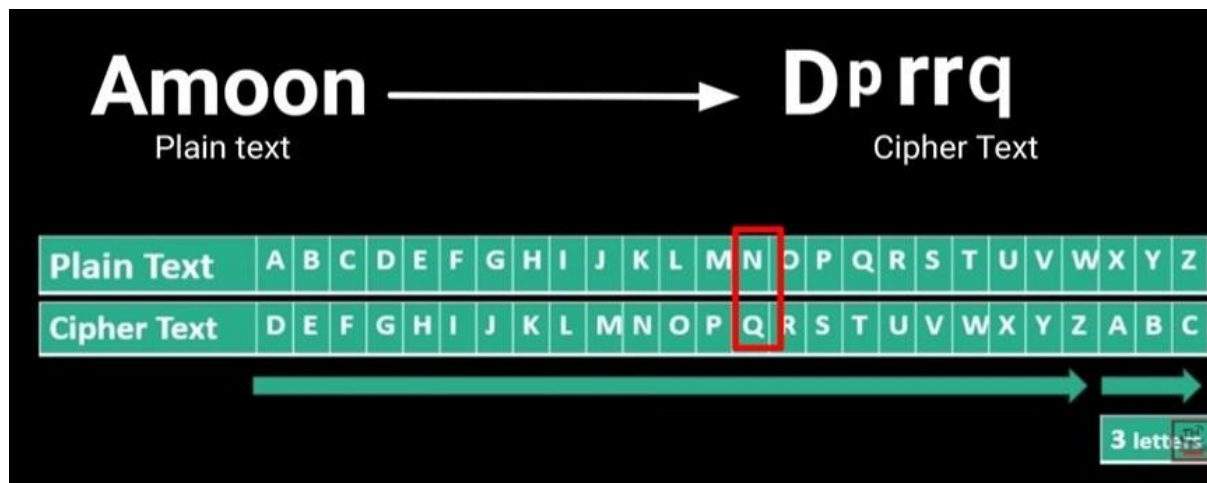
Note: If we have to replace the letter 'z' then the next three alphabets counted after 'z' will be 'a' 'b' 'c'. So, while counting further three alphabets if 'z' occurs it circularly follows 'a'.

There are also some drawbacks of this simple substitution technique. If the hacker knows that the Caesar cipher is used then to perform brute force cryptanalysis, he has only to try 25 possible keys to decrypt the plain text.

The hacker is also aware of the encryption and decryption algorithm.

Example:-





Encryption

$$C = E(3, P) (P+3) \bmod 26$$

Decryption

$$C = D(3, P) (P-3) \bmod 26$$

Vigenere Cipher

It is type of Polyalphabetic Cipher

It is Classical Cryptography

The encryption of the original text is done using the Vigenere Sequence or Vigenere table

Example:-

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| B | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A |
| C | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B |
| D | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C |
| E | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D |
| F | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E |
| G | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F |
| H | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G |
| I | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H |
| J | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I |
| K | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J |
| L | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K |
| M | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L |
| N | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M |
| O | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N |
| P | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
| Q | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P |
| R | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q |
| S | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R |
| T | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S |
| U | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T |
| V | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U |
| W | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V |
| X | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W |
| Y | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X |
| Z | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y |

| Plain Text | key | Cipher Text |
|------------|-----|-------------|
| F | S | X |
| L | U | F |
| O | N | B |
| W | S | O |
| E | U | Y |
| R | N | E |

Vigenere Table

Website: - <https://cryptii.com/pipes/vigenere-cipher>

Cryptii [Help us build the next cryptii](#)

VIEW
Plaintext ▾
flower

ENCODE DECODE
Vigenère cipher ▾
VARIANT
Standard Vigenère cipher ▾
KEY
sun
KEY MODE
Repeat ▾
ALPHABET
abcdefghijklmnopqrstuvwxyz
CASE STRATEGY
Maintain case ▾
FOREIGN CHARS
Include Ignore
→ Encoded 6 chars

VIEW
Ciphertext ▾
xfboye

Decrypt the message with the help of website:-

One time pad (OTP)

1. Difficult to crack encryption
2. Sender always use the new key for encryption



3. Plain text and key size will be same
4. Key will be use only one time. Next time key will be change
5. It is use in polyalphabetic cipher and its substitution cipher cryptography technique
6. It is also called vernam cipher

Example:-

Plain text: - Technical Haroon

| | 1) Technical | 2) Haroon |
|------------------|--------------|-----------|
| Alphabet | | |
| Number | | |
| Key | | |
| Number | | |
| Sum= | | |
| (-) 26 if > 25 | | |
| Cipher Text | | |

| | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |

| | | | | | | | | | | |
|------------------|----|----|----|----|----|----|----|----|----|--|
| Alphabet | T | E | C | H | N | I | C | A | L | |
| Number | 19 | 4 | 2 | 7 | 13 | 8 | 2 | 0 | 11 | |
| Key | R | S | Q | U | V | P | Q | X | M | |
| Number | 17 | 18 | 16 | 20 | 21 | 15 | 16 | 23 | 12 | |
| Sum= | 36 | 22 | 18 | 27 | 34 | 23 | 18 | 23 | 23 | |
| (-) 26 if > 25 | 10 | 22 | 18 | 1 | 8 | 23 | 18 | 23 | 23 | |
| Cipher Text | K | W | S | B | I | X | S | X | X | |

| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |

| | | | | | | |
|----------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
| Alphabet | H | A | R | O | O | N |
| Number | 7 | 0 | 17 | 14 | 14 | 13 |
| Key | u | x | y | c | c | v |
| Number | 20 | 23 | 24 | 2 | 2 | 21 |
| Sum= | 27 | 23 | 41 | 16 | 16 | 34 |
| (-) 26 if > 25 | 1 | 23 | 15 | 16 | 16 | 8 |
| Cipher Text | B | X | P | Q | Q | I |

| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |

| | | |
|--------------------|----------|------------------|
| Plain text | = | Technical |
| Cipher Text | = | Kwsbixsxx |
| Plain text | = | Haroon |
| Cipher Text | = | Bxpqqi |

(Manually method)

Tool:-

Website: -

<https://www.boxentriq.com/codebreaking/one-time-pad>

One-time pad

Encrypt Decrypt

technical

Clear Options

Result

KWSBIXSX

Encryption key

rsquvpqxm

One-time pad

Encrypt Decrypt

haroon

Clear Options

Result

BXPQQI

Encryption key

uxyccv

Playfair Cipher:-

1. Playfair Cipher was the first practical digraph substitution cipher
2. In playfair cipher unlike traditional cipher we encrypt a pair of alphabets(digraphs) instead of a single alphabet
3. It is a polygraphic cipher and using substitution technique.

Example:-

Playfair Cipher

$5 \times 5 = 25$

Plain Text : **Yellow**
Key : **Teacher**

| | | | | | |
|---|---|---|---|---|---|
| Y | E | L | L | O | W |
| Y | E | L | X | L | O |
| W | C | Q | A | I | Q |

| | | | | |
|-----|---|---|---|---|
| T | E | A | C | H |
| R | B | D | F | G |
| i/j | K | L | M | N |
| O | P | Q | S | U |
| V | W | X | Y | Z |

| | | | | | |
|---|---|---|---|---|---|
| Y | E | L | L | O | W |
| Y | E | L | X | L | O |
| W | C | Q | A | I | Q |

Playfair Cipher

Plain Text **Y E L L O W**

Key **T E A C H E R**

Cipher Text **W C Q A I Q X Y**

Website:- <https://www.boxentriq.com/code-breaking/playfair-cipher>

Playfair cipher

Encrypt Decrypt

yellow

Clear Options

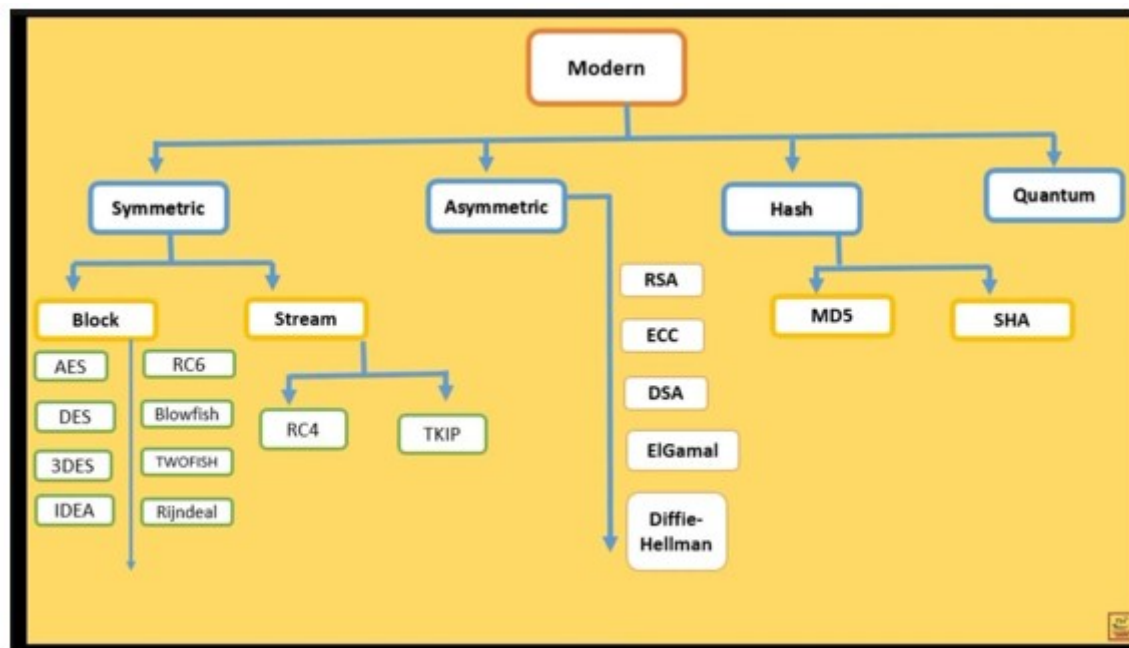
Result

WCQAIQXY

Encryption key

teacher

Modern Cryptography:-

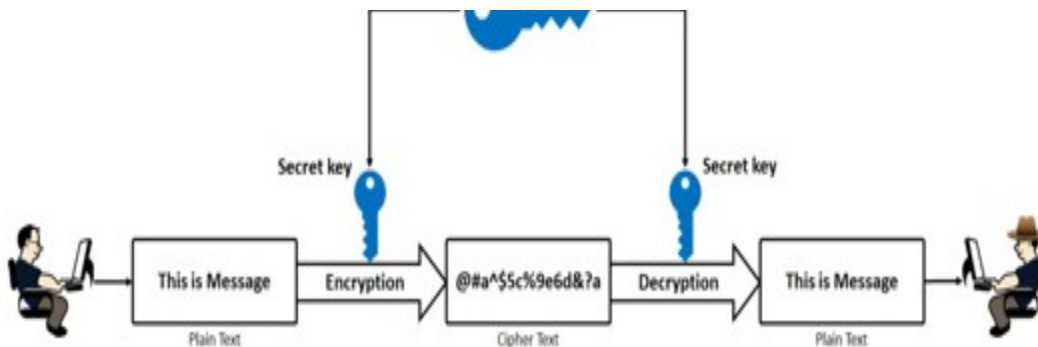


1. Symmetric Key Cryptography :-

Symmetric key cryptography is also called Private key Cryptography. In this approach, both the sender and receiver will use the same key for encrypting and decrypting the message. It's also known as secret-key cryptography or shared-key cryptography. In symmetric-key cryptography, both the sender and receiver share the same key, which they keep secret from everyone else.

How symmetric key cryptography typically works?

- **Key Generation:** A key is generated by the sender or agreed upon between the sender and receiver.
- **Encryption:** The sender uses the key to encrypt the plaintext data. This process scrambles the data into ciphertext, which can only be decrypted back to its original form using the same key.
- **Decryption:** The receiver uses the same key to decrypt the ciphertext and recover the original plaintext.



Asymmetric Key Cryptography:-

Asymmetric Key Cryptography, also known as public-key cryptography, is a cryptographic approach that uses a pair of keys for encryption and decryption. Unlike symmetric key cryptography, where the same key is used for both encryption and decryption, asymmetric key cryptography uses two different keys: a public key and a private key.

How asymmetric key cryptography typically works:-

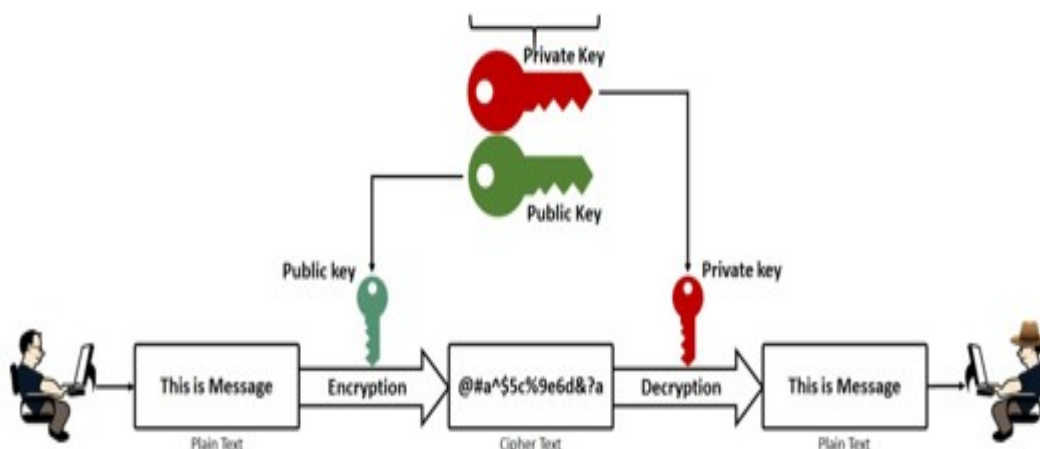
Key Generation: Each user generates a pair of keys - a public key and a private key. These keys are mathematically related in such a way that data encrypted with one key can only be decrypted with the other key in the pair.

Public Key Distribution: The public key is made freely available to anyone who wants to send encrypted messages to the owner of the key. It can be shared openly and doesn't need to be kept secret.

Private Key Protection: The private key is kept secret and known only to the owner. It should be securely stored and not shared with anyone else.

Encryption: When someone wants to send a secure message to the owner of a public key, they encrypt the message using the recipient's public key. This ensures that only the owner of the corresponding private key can decrypt and read the message.

Decryption: The recipient uses their private key to decrypt the encrypted message and recover the original plaintext.



Strengths and Weaknesses of Crypto Methods

| | Symmetric Encryption | Asymmetric Encryption |
|------------------|--|---|
| Strengths | <p>Faster and easier to implement, as the same key is used to encrypt and decrypt data</p> <p>Requires less processing power</p> <p>Can be implemented in application-specific integrated chip (ASIC).</p> | <p>Convenient to use, as the distribution of keys to encrypt messages is not required</p> |
| | <p>Prevents widespread message security compromise as different secret keys are used to communicate with different parties</p> | <p>Enhanced security, as one need not share or transmit private keys to anyone</p> |
| | <p>The key is not bound to the data being transferred on the link; therefore, even if the data are intercepted, it is not possible to decrypt it</p> | <p>Provides digital signatures that cannot be repudiated</p> |

| Weakness | Symmetric Encryption | Asymmetric Encryption |
|-----------------|---|---|
| | Lack of secure channel to exchange the secret key | Slow in processing and requires high processing power |
| | Difficult to manage and secure too many shared keys that are generated to communicate with different parties | Widespread message security compromise is possible (i.e., an attacker can read complete messages if the private key is compromised) |
| | Provides no assurance about the origin and authenticity of a message, as the same key is used by both the sender and the receiver | Messages received cannot be decrypted if the private key is lost |
| | Vulnerable to dictionary attacks and brute-force attacks | Vulnerable to man-in-the-middle and brute-force attacks |

Hashing algorithm:-

A hashing algorithm is a mathematical function that garbles data and makes it unreadable.

Hashing algorithms are one-way programs, so the text can't be unscrambled and decoded by anyone else. And that's the point.

Hashing protects data at rest, so even if someone gains access to your server, the items stored there remain unreadable.

Hashing can also help you prove that data isn't adjusted or altered after the author is finished with it. And some people use hashing to help them make sense of reams of data.

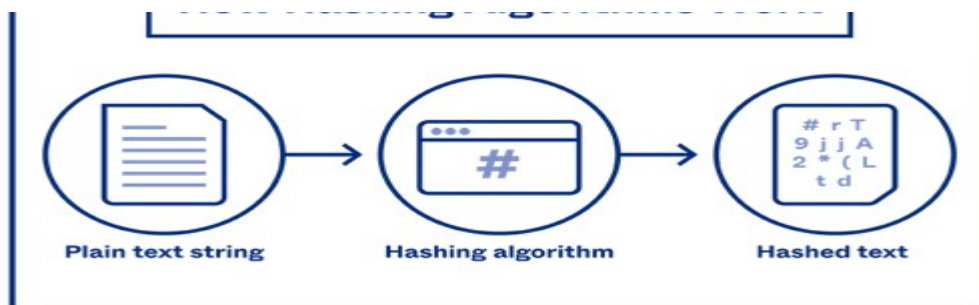
What Is a Hashing Algorithm?

Dozens of different hashing algorithms exist, and they all work a little differently. But in each one, people type in data, and the program alters it to a different form.

All hashing algorithms are:

- **Mathematical:** Strict rules underlie the work an algorithm does, and those rules can't be broken or adjusted.
- **Uniform:** Choose one type of hashing algorithm, and data of any character count put through the system will emerge at a length predetermined by the program.
- **Consistent:** The algorithm does just one thing (compress data) and nothing else.
- **One way:** Once transformed by the algorithm, it's nearly impossible to revert the data to its original state.

It's important to understand that hashing and encryption are different functions. You might use them in concert with one another. But don't use the terms interchangeably.



How Does a Hashing Algorithm Work?

It's possible to create an algorithm with nothing more than a chart, a calculator, and a basic understanding of math. But most people use computers to help.

Most hashing algorithms follow this process:

Create the message. A user determines what should be hashed.

- **Choose the type:** Dozens of hashing algorithms exist, and the user might decide which works best for this message.
- **Enter the message:** The user taps out the message into a computer running the algorithm.
- **Start the hash:** The system transforms the message, which might be of any length, to a predetermined bit size. Typically, programs break the message into a series of equal-sized blocks, and each one is compressed in sequence.
- **Store or share:** The user sends the hash (also called the "message digest") to the intended recipient, or the hashed data is saved in that form.

The process is complicated, but it works very quickly. In seconds, the hash is complete.

What Are Hashing Algorithms Used For?

The very first hashing algorithm, developed in 1958, was used for classifying and organizing data. Since then, developers have discovered dozens of uses for the technology.

Your company might use a hashing algorithm for:

- **Password storage:** You must keep records of all of the username/password combinations people use to access your resources. But if a hacker gains entry, stealing unprotected data is easy. Hashing ensures that the data is stored in a scrambled state, so it's harder to steal.
- **Digital signatures:** A tiny bit of data proves that a note wasn't modified from the time it leaves a user's outbox and reaches your inbox.

- **Document management:** Hashing algorithms can be used to authenticate data. The writer uses a hash to secure the document when it's complete. The hash works a bit like a seal of approval. A recipient can generate a hash and compare it to the original. If the two are equal, the data is considered genuine. If they don't match, the document has been changed.
- **File management:** Some companies also use hashes to index data, identify files, and delete duplicates. If a system has thousands of files, using hashes can save a significant amount of time.

Hashing Algorithm Examples

It may be hard to understand just what these specialized programs do without seeing them in action.

Imagine that we'd like to hash the answer to a security question. We've asked, "Where was your first home?" The answer we're given is, "At the top of an apartment building in Queens." Here's how the hashes look with:

- MD5: 72b003ba1a806c3f94026568ad5c5933
- SHA256:f6bf870a2a5bb6d26ddbda8e903f3867f729785a36f89bfae896776777d50af

Now, imagine that we've asked the same question of a different person, and her response is, "Chicago." Here's how hashes look with:

- MD-5: 9cfa1e69f507d007a516eb3e9f5074e2
- SHA256:0f5d983d203189bbffc5f686d01f6680bc6a83718a515fe42639347efc92478e

Notice that the original messages don't have the same number of characters. But the algorithms produce hashes of a consistent length each time.

And notice that the hashes are completely garbled. It's nearly impossible to understand what they say and how they work.

Popular Hashing Algorithms Explained

Many different types of programs can transform text into a hash, and they all work slightly differently.

Common hashing algorithms include:

MD-5: This is one of the first algorithms to gain widespread approval. It was designed in 1991, and at the time, it was considered remarkably secure.

Since then, hackers have discovered how to decode the algorithm, and they can do so in seconds. Most experts feel it's not safe for widespread use since it is so easy to tear apart.

RIPEMD-160: The RACE Integrity Primitives Evaluation Message Digest (or RIPEMD160) was developed in Belgium in the mid-1990s. It's considered remarkably secure, as hackers haven't quite figured out how to crack it.

SHA: Algorithms in the SHA family are considered slightly more secure. The first versions were developed by the United States government, but other programmers have built on the original frameworks and made later variations more stringent and harder to break. In general, the bigger the number after the letters "SHA," the more recent the release and the more complex the program.

For example, SHA-3 includes sources of randomness in the code, which makes it much more difficult to crack than those that came before. It became a standard hashing algorithm in 2015 for that reason.

Whirlpool: In 2000, designers created this algorithm based on the Advanced Encryption Standard. It's also considered very secure.

The government may no longer be involved in writing hashing algorithms. But the authorities do have a role to play in protecting data. The Cryptographic Module Validation Program, run in part by the National Institute of Standards and Technology, validates cryptographic modules. Companies can use this resource to ensure that they're using technologies that are both safe and effective.

The government may no longer be involved in writing hashing algorithms. But the authorities do have a role to play in protecting data. The Cryptographic Module Validation Program, run in part by the National Institute of Standards and Technology, validates cryptographic modules. Companies can use this resource to ensure that they're using technologies that are both safe and effective.

Data Encryption Standard (DES)

DES is a standard for data encryption that uses a secret key for both encryption and decryption (symmetric cryptosystem). DES uses a 64-bit secret key, of which 56 bits are generated randomly and the other 8 bits are used for error detection. It uses a data encryption algorithm (DEA), a secret key block cipher employing a 56-bit key operating on 64-bit blocks. DES is the archetypal block cipher—an algorithm that takes a fixed-length string of plaintext bits and transforms it into a ciphertext bit string of the same length. The design of DES allows users to implement it in hardware and use it for single-user encryption, such as to store files on a hard disk in encrypted form.

DES provides 72 quadrillion or more possible encryption keys and chooses a random key for the encryption of each message. Because of the inherent weakness of DES vis-à-vis today's technologies, some organizations use triple DES (3DES), in which

they repeat the process three times for added strength until they can afford to update their equipment to AES capabilities.

Triple Data Encryption Standard (3DES)

Eventually, it became obvious that DES would no longer be secure. The U.S. Federal Government began a contest seeking a replacement cryptography algorithm. However, in the meantime, 3DES was created as an interim solution. Essentially, it performs DES three times with three different keys. 3DES uses a "key bundle" that comprises three DES keys, K1, K2, and K3. Each key is a standard 56-bit DES key. It then performs the following process: DES encrypt with K1, DES decrypt with K2, DES encrypt with K3. There are three options for the keys. In the first option, all three keys are independent and different. In the second option, K1 and K3 are identical. In the third option, all three keys are the same; therefore, you are literally applying the same DES algorithm three times with the same key. The first option is the most secure, while the third is the least secure.

Advanced Encryption Standard (AES)

The Advanced Encryption Standard (AES) is a National Institute of Standards and Technology (NIST) specification for the encryption of electronic data. It also helps to encrypt digital information such as telecommunications, financial, and government data. US government agencies have been using it to secure sensitive but unclassified material.

AES consists of a symmetric-key algorithm: both encryption and decryption are performed using the same key. It is an iterated block cipher that works by repeating the defined steps multiple times. It has a 128-bit block size, with key sizes of 128, 192, and

256 bits for AES-128, AES-192, and AES-256, respectively. The design of AES makes its use efficient in both software and hardware. It works simultaneously at multiple network layers.

AES Pseudocode

Initially, the system copies the cipher input into the internal state and then adds an initial round key. The system transforms the state by iterating a round function in a number of cycles. The number of cycles may vary with the block size and key length. After completing rounding, the system copies the final state into the cipher output.

```
state = in
AddRoundKey (state, w)
  for round = 1 step 1 to Nr-1
    SubBytes (state)
    ShiftRows (state)
    MixColumns (state)
    AddRoundKey (state, w+round*Nb)
  end for
SubBytes (state)
ShiftRows (state)
AddRoundKey (state, w+Nr*Nb)
out = state
end
```

RC4, RC5, and RC6 Algorithms

Symmetric encryption algorithms developed by RSA Security are discussed below.

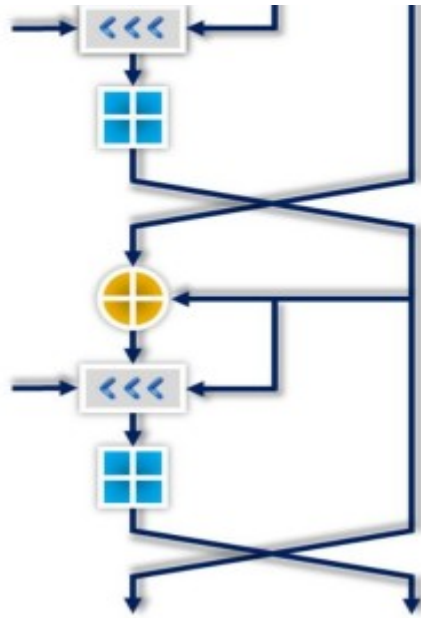
RC4

RC4 is a variable key-size symmetric-key stream cipher with byte-oriented operations, and it is based on the use of a random permutation. According to some analyses, the period of the cipher is likely to be greater than 10,100. Each output byte uses 8 to 16 system operations; thus, the cipher can run fast when used in software. RC4 enables safe communications such as for traffic encryption (which secures websites) and for websites that use the SSL protocol.

RC5

RC5 is a fast symmetric-key block cipher designed by Ronald Rivest for RSA Data Security (now RSA Security). The algorithm is a parameterized algorithm with a variable block size, a variable key size, and a variable number of rounds. The block sizes can be 32, 64, or 128 bits. The range of the rounds can vary from 0 to 255, and the size of the key can vary from 0 to 2,040 bits. This built-in variability can offer flexibility at all levels of security. The routines used in RC5 are key expansion, encryption, and decryption.

In the key expansion routine, the secret key that a user provides is expanded to fill the key table (the size of which depends on the number of rounds). RC5 uses a key table for both encryption and decryption. The encryption routine has three fundamental operations: integer addition, bitwise XOR, and variable rotation. The intensive use of data-dependent rotation and the combination of different operations make RC5 a secure encryption algorithm.



RC6

RC6 is a symmetric-key block cipher derived from RC5. It is a parameterized algorithm with a variable block size, key size, and number of rounds. Two features that differentiate RC6 from RC5 are integer multiplication (which is used to increase the diffusion, achieved in fewer rounds with increased speed of the cipher) and the use of four 4-bit working registers rather than two 2-bit registers. RC6 uses four 4-bit registers instead of two 2-bit registers because the block size of the AES is 128 bits.

Comparison of Cryptographic Algorithms

| Algorithm | Working Structure | Key/Block Size (bits) | Known Attacks |
|-----------|-------------------|-----------------------|--------------------|
| DES | Feistel | 56 (8 bits parity)/64 | Brute-force attack |

| | | | |
|-------------------|------------------------------------|-----------------|--|
| 3DES | Feistel | 112 or 168 bits | Block collision attack |
| AES | Substitution-permutation | Up to 256/128 | Side-channel attack |
| RC4 | Random-permutation | Up to 2048/2064 | NOMORE attack |
| RC5 | Feistel | Up to 2040/128 | Timing attack |
| RC6 | Feistel | Up to 256/128 | Brute-force attack |
| Blowfish | Feistel | 32-448 bits | Birthday attack and known-plaintext attack |
| Twofish | Feistel | Up to 256/128 | Power analysis attack |
| Threefish | Tweakable block cipher/Non-Feistel | Up to 1024/1024 | Boomerang attack |
| Serpent | Substitution-permutation | Up to 256/128 | XSLand Meet-in-the-Middle attack |
| TEA | Feistel | Up to 128/64 | Related-key attack |
| CAST-128 | Feistel | Up to 128/64 | Known-plaintext attack |
| GOST Block Cipher | Feistel | 256/64 | Chosen-key attack |
| RSA | Factorization | Variable | Brute force and |

| | | | |
|----------------|--|---------------|-------------------------------------|
| | | | timing attack |
| Diffie-Hellman | Elliptic Curves/Algebraic | Variable | Man-in-the-Middle attack |
| YAK | Nondeterministic Finite automata (NFA) | Variable | Keyshare and key replication attack |
| MD5 | Merkle-Damgard Construction | Variable | Collision attack |
| MD6 | Merkle-Damgard Construction | Variable | Brute-force attack/Birthday attack |
| SHA | Merkle-Damgard Construction | 160/512 | Collision attack |
| RIPEMD-160 | Merkle-Damgard Construction | Up to 320/512 | Collision attack |
| HMAC | Merkle-Damgard Construction | Variable | Brute-force attack |

Message Digest (One-way Hash) Functions

Hash functions calculate a unique fixed-size bit string representation, called a message digest, of any arbitrary block of information. Message digest functions distill the information contained in a file (small or large) into a single fixed-length number, typically between 128 and 256 bits. If any given bit of the function's input is changed, every output bit has a 50% chance of changing. Given an input file and its corresponding message digest, it should be nearly impossible to find another file with the same message digest value, as it is computationally infeasible to have two files with the same message digest value.

Message digest functions are also called one-way hash functions because they produce values that are nearly impossible to invert, resistant to attack, mostly unique, and widely distributed.

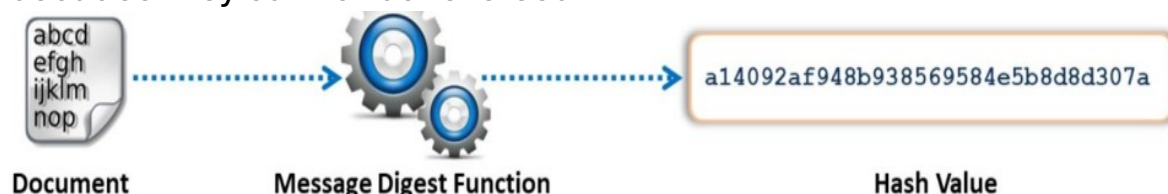
Message digest algorithms themselves do not participate in encryption and decryption operations. They allow the creation of digital signatures and message authentication codes (MACs) as well as the derivation of encryption keys from passphrases.

The main role of a cryptographic hash function is to provide integrity in document management. Cryptographic hash functions are an integral part of digital signatures. They are relatively faster than digital signature algorithms; hence, their characteristic feature is to calculate the signature of the document's hash value, which is smaller than the document. In addition, digests help to hide the contents or source of the document.

Widely used message digest functions include the following algorithms:

- MD5
- SHA

Note: Message digests are also called one-way hash functions because they cannot be reversed.



Message Digest Function: MD5 and MD6

MD2, MD4, MD5, and MD6 are message digest algorithms used in digital signature applications to compress a document securely before the system signs it with a private key. The algorithms can be of variable length, but the resulting message digest always has a size of 128 bits.

The structures of all three algorithms (MD2, MD4, and MD5) appear similar, although the design of MD2 is reasonably different from that of MD4 and MD5. MD2 supports 8-bit machines, while MD4 and MD5

support 32-bit machines. The algorithm pads the message with extra bits to ensure that the number of bits is divisible by 512. The extra bits may include a 64-bit binary message.

Attacks on versions of MD4 have become increasingly successful. Research has shown how an attacker launches collision attacks on the full version of MD4 within a minute on a typical PC. MD5 is slightly more secure but is slower than MD4. However, both the message digest size and the padding requirements remain the same.

MD5 is a widely used cryptographic hash function that takes a message of arbitrary length as input and outputs a 128-bit (16-byte) fingerprint or message digest of the input. MD5 can be used in a wide variety of cryptographic applications and is useful for digital signature applications, file integrity checking, and storing passwords. However, MD5 is not collision resistant; therefore, it is better to use the latest algorithms, such as MD6, SHA-2, and SHA-3.

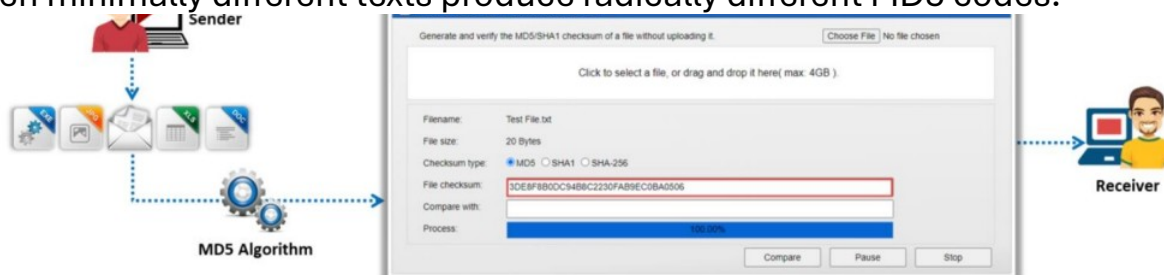
MD6 uses a Merkle-tree-like structure to allow for large-scale parallel computation of hashes for very long inputs. It is resistant to differential cryptanalysis attacks.

To calculate the effectiveness of hash functions, check the output produced when the algorithm randomizes an arbitrary input message.

The following are examples of minimally different message digests:

- echo "There is CHF1500 in the blue bo" | md5sum
e41a323bdf20eadafd3f0e4f72055d36
- echo "There is CHF1500 in the blue box" | md5sum
7a0da864a41fd0200ae0ae97afd3279d
- echo "There is CHF1500 in the blue box." | md5sum
2db1ff7a70245309e9f2165c6c34999d

Even minimally different texts produce radically different MD5 codes.



Secure Hashing Algorithm (SHA)

The NIST has developed the Secure Hash Algorithm (SHA), specified in the **Secure Hash Standard (SHS)** and published as a federal information-processing standard (FIPS PUB 180). It generates a cryptographically secure one-way hash. Rivest developed the SHA, which is similar to the message digest algorithm family of hash functions. It is slightly slower than MD5, but its larger message digest makes it more secure against brute-force collision and inversion attacks.

SHA encryption is a series of five different cryptographic functions, and it currently has three generations: SHA-1, SHA-2, and SHA-3.

- **SHA-0:** A retronym applied to the original version of the 160-bit hash function published in 1993 under the name SHA, which was withdrawn from trade due to an undisclosed "**significant flaw**" in it. It was replaced with a slightly revised version, namely SHA-1.
- **SHA-1:** It is a 160-bit hash function that resembles the former MD5 algorithm developed by Ron Rivest. It produces a 160-bit digest from a message with a maximum length of $(2^{64} - 1)$ bits. It was designed by the National Security Agency (NSA) to be part of the Digital Signature Algorithm (DSA). It is most commonly used in security protocols such as PGP, TLS, SSH, and SSL. As of 2010, SHA-1 is no longer approved for cryptographic use because of its cryptographic weaknesses.
- **SHA-2:** SHA2 is a family of two similar hash functions with different block sizes, namely SHA-256, which uses 32-bit words, and SHA-512, which uses 64-bit words. The truncated versions of each standard are SHA-224 and SHA-384.
- **SHA-3:** SHA-3 uses sponge construction in which message blocks are XORed into the initial bits of the state, which the algorithm then invertibly permutes. It supports the same hash

lengths as SHA-2 but differs in its internal structure considerably from the rest of the SHA family.

Comparison of SHA functions (SHA-0, SHA-1, SHA-2, and SHA-3)

| SHA (reference) | | add | (4*32) | sub | $2^{64}-1$ | shr | and, or, xor, rot | found) |
|-----------------|-------------|--------------|---------------|------|-------------|-----|--|---------------------------|
| SHA-0 | | 160 | 160 (5*32) | 512 | $2^{64}-1$ | 80 | Add mod 2^{32} , and, or, xor, rot | <34 (collisions found) |
| SHA-1 | | 160 | 160 (5*32) | 512 | $2^{64}-1$ | 80 | Add mod 2^{32} , and, or, xor, rot | <63 (collisions found) |
| SHA-2 | SHA-224 | 224 | 256 | 512 | $2^{64}-1$ | 64 | Add mod 2^{32} , and, or, xor, shr, rot | 112 |
| | SHA-256 | 256 | (8*32) | | | | | 128 |
| | SHA-384 | 384 | | | | | | 192 |
| | SHA-512 | 512 | 512 | 1024 | $2^{128}-1$ | 80 | Add mod 2^{64} , and, or, xor, shr, rot. | 256 |
| | SHA-512/224 | 224 | (8*64) | | | | | 112 |
| | SHA-512/256 | 256 | | | | | | 128 |
| SHA-3 | SHA3-224 | 224 | | 1152 | | | | 112 |
| | SHA3-256 | 256 | | 1088 | | | | 128 |
| | SHA3-384 | 384 | 1600 | 832 | | | | 192 |
| | SHA3-512 | 512 | (5*5*64) | 576 | ∞ | 24 | And, xor, not, rot | 256 |
| | SHAKE128 | d(arbitrary) | | 1344 | | | | Min(d/2,128) |
| | sHAKE256 | d(arbitrary) | | 1088 | | | | Min(d/2,256) |

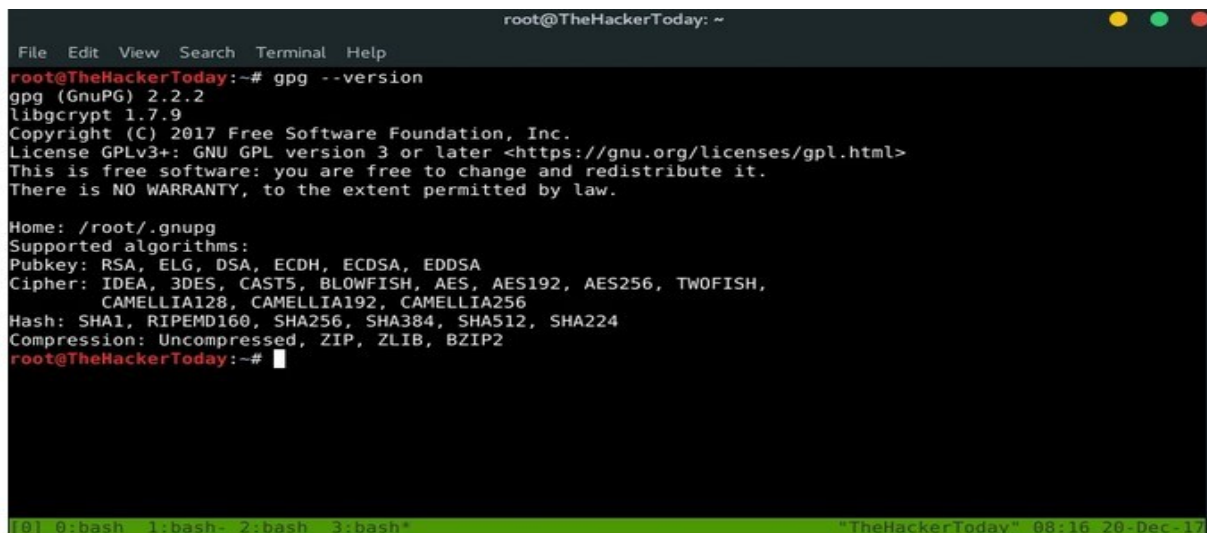
Practical Of GPG Tool

Today we're going to encrypt a file or directory using Gpg tool which can be installed in any Linux version. If you are really concerned about your privacy and you don't want your friends to sneak into your laptop or files you can use strong passwords, hide files somewhere in safe locations, or in some cases, you can encrypt files. You can do pretty much everything from encrypting a file to an entire hard drive.

Also Read: Creating an Encrypted Folder in Kali Linux/Ubuntu/Windows & Mac using TrueCrypt

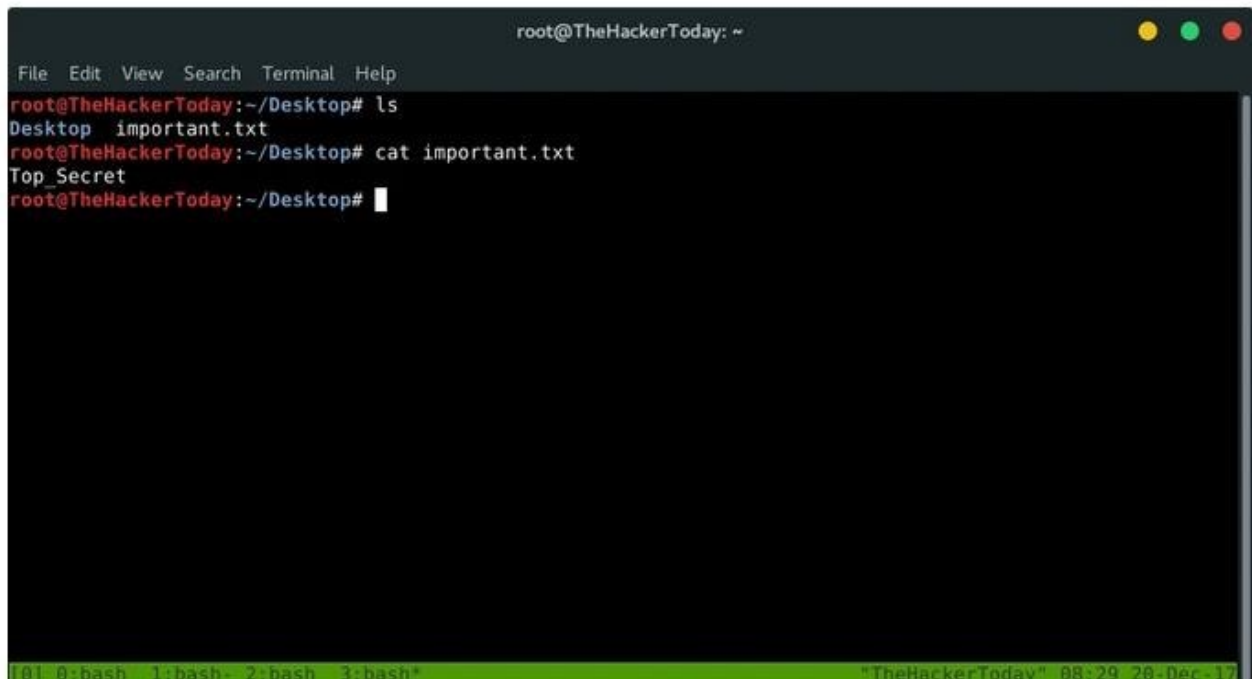
Gpg is a free tool that is used to encrypt a single file or folder with few commands, the only way to decrypt those files is with a password.

Let's get started! For this tutorial, I'm using Kali Linux and it has Gpg preinstalled not just Kali it comes pre-installed in every Linux version.

A screenshot of a terminal window titled 'root@TheHackerToday: ~'. The terminal shows the command 'gpg --version' being executed. The output displays the version of GnuPG (2.2.2) and libgcrypt (1.7.9), along with copyright information for the Free Software Foundation, Inc. It lists supported algorithms for public keys (RSA, ELG, DSA, ECDH, ECDSA, EDDSA), ciphers (IDEA, 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH, CAMELLIA128, CAMELLIA192, CAMELLIA256), hashes (SHA1, RIPEMD160, SHA256, SHA384, SHA512, SHA224), and compression (Uncompressed, ZIP, ZLIB, BZIP2). The terminal window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. At the bottom, there is a status bar showing '[0] 0: bash 1: bash- 2: bash 3: bash*' and the date 'TheHackerToday 08:16 20-Dec-17'.

How to Encrypt/Decrypt a File in Linux using gpg (Kali Linux)

Let's say you have file name important.txt and it contains some classified information or some secret stuff that you wanna hide. This 'important.txt' file contains the text "Top_Secret" or something totally depends on your work, let's say it's a password.

A terminal window titled 'root@TheHackerToday: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the following commands and output:

```
root@TheHackerToday:~/Desktop# ls
Desktop  important.txt
root@TheHackerToday:~/Desktop# cat important.txt
Top_Secret
root@TheHackerToday:~/Desktop#
```

The terminal has a green status bar at the bottom with the text '01 0: bash 1: bash 2: bash 3: bash*' and a timestamp 'TheHackerToday 08:29 28-Dec-17'.

Now, Before everything we have to generate a key first. You will be prompted to enter some security information. Use the defaults when available, otherwise enter your name and email address. You will also be prompted for a passphrase. Remember this passphrase.

```
root@TheHackerToday: ~  
File Edit View Search Terminal Help  
root@TheHackerToday:~/Desktop# gpg --gen-key  
gpg (GnuPG) 2.2.2; Copyright (C) 2017 Free Software Foundation, Inc.  
This is free software: you are free to change and redistribute it.  
There is NO WARRANTY, to the extent permitted by law.  
  
Note: Use "gpg --full-generate-key" for a full featured key generation dialog.  
  
GnuPG needs to construct a user ID to identify your key.  
  
Real name: fsociety  
Email address:  
You selected this USER-ID:  
    "fsociety"  
  
Change (N)ame, (E)mail, or (O)kay/(Q)uit? O  
We need to generate a lot of random bytes. It is a good idea to perform  
some other action (type on the keyboard, move the mouse, utilize the  
disks) during the prime generation; this gives the random number  
generator a better chance to gain enough entropy.  
We need to generate a lot of random bytes. It is a good idea to perform  
some other action (type on the keyboard, move the mouse, utilize the  
disks) during the prime generation; this gives the random number  
generator a better chance to gain enough entropy.  
gpg: key E31A77E678BF2232 marked as ultimately trusted  
gpg: directory '/root/.gnupg/openpgp-revocs.d' created  
gpg: revocation certificate stored as '/root/.gnupg/openpgp-revocs.d/28D50B0A3811B8B98B094945E31A77E678BF2232.rev'  
public and secret key created and signed.  
  
pub   rsa3072 2017-12-20 [SC] [expires: 2019-12-20]  
      28D50B0A3811B8B98B094945E31A77E678BF2232  
uid           fsociety  
sub   rsa3072 2017-12-20 [E] [expires: 2019-12-20]  
  
root@TheHackerToday:~/Desktop#  
[0] 0: bash 1: bash 2: bash 3: bash "The LAZY script" 08:49 20-Dec-17
```

gpg --gen-key

After generating the key. We have to encrypt our file.

Type

gpg -e -r fsociety important.txt

If you remember fsociety is our USER-ID. After typing that command your file will be encrypted and another file will be generated with a .gpg extension to delete your original non-encrypted file.

```
root@TheHackerToday: ~  
File Edit View Search Terminal Help  
root@TheHackerToday:~/Desktop# ls  
Desktop important.txt important.txt.gpg  
root@TheHackerToday:~/Desktop#  
[0] 0:bash 1:bash- 2:bash 3:bash* "TheHackerToday" 08:34 20-Dec-17
```

Now you'll see two files "important.txt" and "important.txt.gpg" let's cat to see the difference.

```
root@TheHackerToday: ~  
File Edit View Search Terminal Help  
root@TheHackerToday:~/Desktop# cat important.txt  
Top_Secret  
root@TheHackerToday:~/Desktop# cat important.txt.gpg  
5;wZP#aaL<uI*  
[0]^rcE]pb[01;31mroot@TheHackerToday:~/Desktop#  
[0] 0:bash 1:bash- 2:bash 3:bash* "TheHackerToday" 08:35 20-Dec-17
```

As you can see gpg has encoded our string or password inside "important.txt" file and now you can delete your previous text file.

```
root@TheHackerToday: ~  
File Edit View Search Terminal Help  
root@TheHackerToday:~/Desktop# ls  
Desktop  important.txt.gpg  
root@TheHackerToday:~/Desktop# cat important.txt.gpg  
5:wZP#aal<uI*  
8^rcEJpb[01;3lmroot@TheHackerToday:~/Desktop#
```

Now, It's time to decrypt our "important.txt.gpg" back to "important.txt" and readable text.

Type: `gpg -d -o decrypted.txt`

`important.txt.gpg`

You will be prompted to enter a password for the key and boom!

```
root@TheHackerToday: ~  
File Edit View Search Terminal Help  
root@TheHackerToday:~/Desktop# gpg -d -o decrypted.txt important.txt.gpg  
gpg: encrypted with 3072-bit RSA key, ID B7513DAE64F7AE6B, created 2017-12-20  
"fsociety"  
root@TheHackerToday:~/Desktop#  
root@TheHackerToday:~/Desktop# ls  
decrypted.txt  Desktop  important.txt.gpg  
root@TheHackerToday:~/Desktop# cat decrypted.txt  
Top_Secret  
root@TheHackerToday:~/Desktop#
```

MD5 and MD6 Hash Calculators

MD5 and MD6 hash calculators that use different hash algorithms to convert plaintext into its equivalent hash value are discussed below.

MD5 Calculator

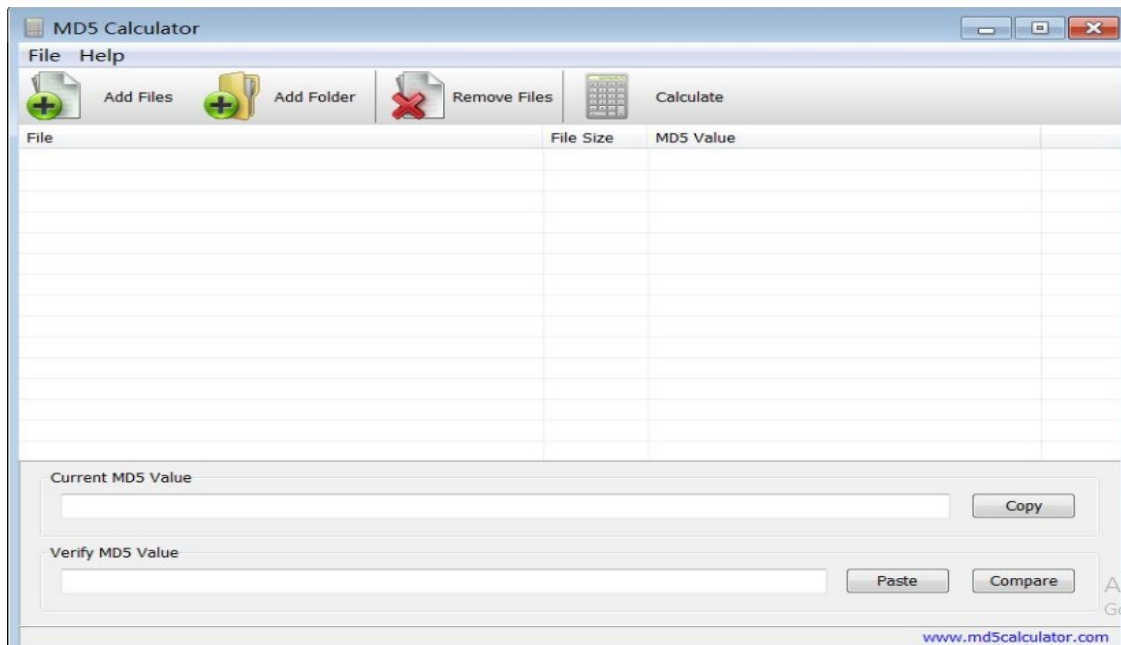
Source: <https://www.md5calculator.com/>

MD5 Calculator is a simple application that calculates the MD5 hash of a given file. It can be used with large files (e.g., several gigabytes in size). It features a progress counter and a text field from which the final MD5 hash can be easily copied to the clipboard. MD5

Calculator can be used to check the integrity of a file.

It allows you to calculate the MD5 hash value of the selected file.

Right-click the file and choose "MD5 Calculator;" the program will calculate the MD5 hash. The MD5 Digest field contains the calculated value. To compare this MD5 digest with another, one can paste the other value into the Compare To field. Obviously, an equal to sign ("=") appears between the two values if they are equal; otherwise, the less than ("<") or greater than (">") sign will tell you that the values are different.



Cryptography Attacks

Attackers conduct cryptography attacks by assuming that the cryptanalyst has access to the encrypted information. A cryptography attack or cryptanalysis involves the study of various principles and methods of decrypting the ciphertext back to the plaintext without knowledge of the key.

The various types of cryptography attacks are as follows:

- **Ciphertext-only Attack**

Ciphertext-only is less effective but much more likely for the attacker. The attacker only has access to a collection of ciphertexts. This is much more likely than known plaintext but is also the most difficult. The attack is completely successful if the corresponding plaintexts (or even better, the key) can be deduced. The ability to obtain any information at all about the underlying plaintext is still considered a success. So what does the attacker do with the ciphertexts he/she has accumulated? You can analyze them for patterns, trying to find something that would give you a hint as to the key that was used to crack them. Often, the result of this attack is just a partial break and not a complete break.

- **Adaptive Chosen-plaintext Attack**

In this type of attack, an attacker has complete access to the plaintext message including its encryption, and he/she can also modify the content of the message by making a series of interactive queries, choosing subsequent plaintext blocks based on the information from the previous encryption queries and functions. To perform this attack, an attacker needs to interact with the encryption device.

- **Chosen-plaintext Attack**

A chosen plaintext attack is a highly effective type of cryptanalysis attack. In this attack, the attacker obtains the ciphertexts corresponding to a set of plaintexts of his/her own choosing. This allows the attacker to attempt to derive the key used and thus decrypt other messages encrypted with that key. Basically, since the attacker knows the plaintext and the resultant ciphertext, he/she gains many insights into the key used. This technique can be difficult but is not impossible.

- **Related-Key Attack**

The related-key attack is similar to the chosen plaintext attack, except that the attacker can obtain ciphertexts encrypted under two different keys. This is actually a very useful attack if you can obtain the plaintext and matching ciphertext. The attack requires that the differing keys be closely related, e.g., in a wireless environment where subsequent keys might be derived from previous keys. Then, while the keys are different, they are close. Much like the ciphertext-only attack, this type of attack is most likely only going to yield a partial break.

- **Dictionary Attack**

In this attack, the attacker constructs a dictionary of plaintext along with its corresponding ciphertext that he/she has analyzed and obtained for a certain period of time. After building the dictionary, if the attacker obtains the ciphertext, he/she uses the already built dictionary to find the corresponding plaintext. Attackers use this technique to decrypt keys, passwords, passphrases, and ciphertext.

- **Known-plaintext Attack**

In this attack, the only information available to the attacker is some plaintext blocks along with the corresponding ciphertext and algorithm used to encrypt and decrypt the text. Using this information, the key used to generate the ciphertext is deduced so as to decipher other messages. This attack works on block ciphers and is an example of linear cryptanalysis. The known plaintext blocks are generated using a series of intelligent guesses and logic, and not by accessing the plaintext over a channel.

- **Chosen-ciphertext Attack**

The attacker obtains the plaintexts corresponding to an arbitrary set of ciphertexts of his own choosing. Using this information, the attacker tries to recover the key used to encrypt the plaintext. To perform this attack, the attacker must have access to the communication channel between the sender and the receiver.

There are two variants of this attack:

- **Lunchtime or Midnight Attack:** In this attack, the attacker can have access to the system for only a limited amount of time or can access only a few plaintext-ciphertext pairs.
- **Adaptive Chosen-ciphertext Attack:** In this attack, the attacker selects a series of ciphertexts and then observes the resulting plaintext blocks.

- **Rubber Hose Attack**

Attackers extract cryptographic secrets (e.g., the password to an encrypted file) from a person by coercion or torture. In general, people under pressure cannot maintain security, and they will reveal secrets or hidden information. Attackers torture

victims to reveal secret keys or passwords used to encrypt the information.

- **Chosen-key Attack**

In this type of attack, an attacker not only breaks a ciphertext but also breaks into a larger system, which is dependent of that ciphertext. The attacker usually breaks an n -bit key cipher into $2^{n/2}$ operations. Once an attacker breaks the cipher, he gets access to the system, and he can control the whole system, access confidential data, and perform further attacks.

- **Timing Attack**
















It is based on repeatedly measuring the exact execution times of modular exponentiation operations. The attacker tries to break the ciphertext by analyzing the time taken to execute the encryption and decryption algorithm for various inputs. In a computer, the time taken to execute a logical operation may vary based on the input given. An attacker tries to extract the plaintext by giving varying inputs.

- **Man-in-the-Middle Attack**

This attack is performed against a cryptographic protocol. Here, an attacker intercepts the communication between a client and a server and negotiates the cryptographic parameters. Using this attack, an attacker can decrypt the encrypted content and obtain confidential information such as system passwords. An attacker can also inject commands that can modify the data in transit. The attacker usually performs an

MITM attack on public-key cryptosystems where key exchange is required before communication takes place.

Online MD5 Decryption Tools

| | | |
|---|---|---|
|  MD5 Decoder https://www.dcode.fr |  MD5 Decryption https://www.md5online.org |  cmd5 https://www.cmd5.org |
|  CrackStation https://crackstation.net |  MD5Decrypter https://www.md5decrypter.com |  Decrypt MD5 Hash https://hashtoolkit.com |
|  Md5 Decrypt & Encrypt https://md5decrypt.net |  OnlineHashCrack https://www.onlinehashcrack.com |  Online MD5 Hashed Validator https://www.javainuse.com |
|  md5hashing https://md5hashing.net |  HashKiller.io https://hashkiller.io |  Decode MD5 Hash https://md5.web-max.ca |
|  MD5 Encrypt/Decrypt https://10015.io |  Md5.My-Addr.com http://md5.my-addr.com |  md5 decoder tool http://md5.my-addr.com |

How to Defend Against Cryptographic Attacks

The following countermeasures can be adopted to prevent cryptographic attacks:

- Access of cryptographic keys should be given directly to the application or user.
- IDS should be deployed to monitor exchanging and access of keys.
- Passphrases and passwords must be used to encrypt the key, if stored in the disk.
- Keys should not be present inside the source code or binaries.

- For certificate signing, the transfer of private keys should not be allowed.
- For symmetric algorithms, a key size of 168 bits or 256 bits should be preferred for a secure system, especially in the case of large transactions.
- Message authentication must be implemented for the encryption of symmetric-key protocols.
- For asymmetric algorithms, a key size of 1536 bits or 2048 bits should be considered for secure and highly protected applications.
- In the case of hash algorithms, a key size of 168 or 256 bits should be considered.
- Only recommended tools or products should be used rather than self-engineered crypto algorithms or functions.
- Impose a limit on the number of operations per key.
- The output of the hash function should have a larger bit length that makes it difficult to decrypt.
- Design applications and protocols that can avoid simple encryption key relationships, i.e., each encrypted key should be created from a key derivation function (KDF).
- Upgrade to the latest security standards.
- Use strong key schedules to mitigate the risks of related key attacks.
- Enforce hardware-backed security such as hardware security modules (HSMs) to enhance the cryptographic key security.
- Do not use a single cryptographic key for multiple purposes.
- Use redundant cryptosystems to encrypt data multiple times.