

116 CSC: Symmetric Encryption

Greece, 3rd century BC
~2,300 years ago.



Scytale



Scytale is the key to encrypt and decrypt the message

**This is my first
encrypted message**

G	H	S	T	I	N	H	A
T	h	i	s	i	s	m	f
i	r	s	t	e	n	c	r
y	p	t	e	d	m	e	s
s	a	g	e				

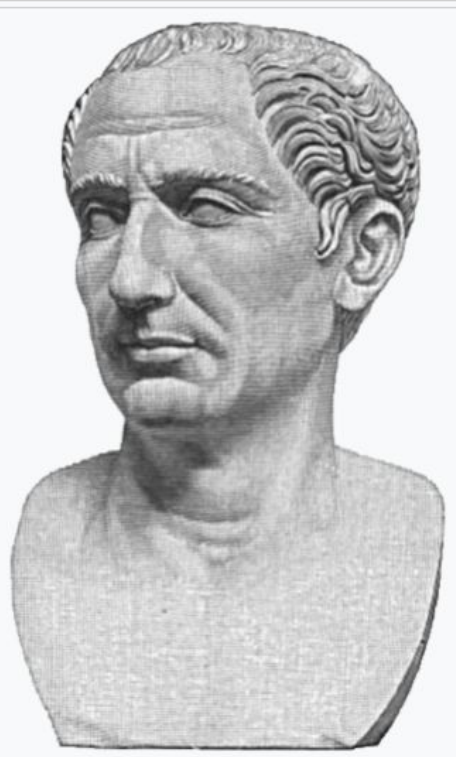
Ciphertext: Tiyshrpai stgste eiedsnmmcefrs


Limitations

1. All the texts are readable which are easy to be attacked.
2. T is the first number

Last century BC

~2000 year ago



The Caesar cipher is named  for [Julius Caesar](#), who used an alphabet where decrypting would shift three letters to the left.

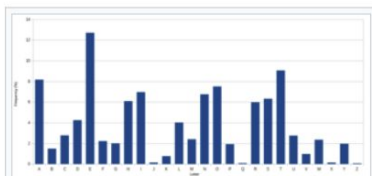
Plain	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Cipher	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Random	T	Y	Z	B	C	F	H	J																	

Plaintext: THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG

Ciphertext: QEB NRFZH YOLTK CLU GRJMP LSBO QEB IXWV ALD

Breaking the cipher [\[edit \]](#)

The Caesar cipher can be easily broken even in a [ciphertext-only scenario](#). Since there are only a limited number of possible shifts (25 in English), an attacker can mount a [brute force attack](#) by deciphering the message, or part of it, using each possible shift. The correct description will be the one which makes sense as English text.^[18] An example is shown on the right for the ciphertext "exxegoexsrgi"; the candidate plaintext for shift four "attackatonce" is the only one which makes sense as English text. Another type of brute force attack is to write out the alphabet beneath each letter of the ciphertext, starting at that letter. Again the correct decryption is the one which makes sense as English text. This technique is sometimes known as "completing the plain component".^{[19][20]}



The distribution of letters in a typical sample of English language text has a distinctive and predictable shape. A Caesar shift "rotates" this distribution, and it is possible to determine the shift by examining the resultant frequency graph.

Another approach is to match up the frequency distribution of the letters. By graphing the frequencies of letters in the ciphertext, and by knowing the expected distribution of those letters in the original language of the plaintext, a human can easily spot the value of the shift by looking at the displacement of particular features of the graph. This is known as [frequency analysis](#). For example, in the English language the plaintext frequencies of the letters E, T,

(usually most frequent), and Q, Z (typically least frequent) are particularly distinctive.^[21] Computers can automate this process by assessing the similarity between the observed frequency distribution and the expected distribution. This can be achieved, for instance, through the utilization of the [chi-squared statistic](#)^[22] or by minimizing the sum of squared errors between the observed and known

Decryption shift	Candidate plaintext
0	exxegoexsrgi
1	dwwdfndwrqfh
2	cvvcemcvqpeg
3	buubdlbupodf
4	attackatonce
5	zsszbjzsnmbd
6	yrryaiyrmlac
...	
23	haahjrhavujl
24	gzzgiqgzutik
25	fyyfhpftytshj

language distributions.^[23]

World War I

Cipher Disk



**1470 AD –
1940 AD**

600-200 years ago



Step back in time
with this
state-of-the-art
encryption machine
used during the
conflict between
Mexico and the
USA before World
War I

Mexican–American War

🌐 73 languages

Article Talk

Read Edit View history Tools

From Wikipedia, the free encyclopedia

The **Mexican–American War**,^[a] also known in the United States as the **Mexican War**, and in Mexico as the **United States intervention in Mexico**,^[b] was an invasion of [Mexico](#) by the [United States Army](#) from 1846 to 1848. It followed the 1845 American [annexation of Texas](#), which Mexico still considered its territory because it refused to recognize the [Treaties of Velasco](#), signed by President [Antonio López de Santa Anna](#) after he was captured by the [Texian Army](#) during the 1836 [Texas Revolution](#). The [Republic of Texas](#) was *de facto* an independent country, but most of its Anglo-American citizens who had moved from the United States to Texas after 1822 wanted to be annexed by the United States.^{[5][6]}

Sectional politics over [slavery in the United States](#) had previously prevented annexation because Texas would have been admitted as a slave state, upsetting the balance of power between Northern free states and Southern slave states.^[7] In the [1844 United States presidential election](#), Democrat [James K. Polk](#) was elected on a platform of expanding U.S. territory to [Oregon](#), [California](#) (also a Mexican territory), and Texas by any means, with the 1845 annexation of Texas furthering that goal.^[8] However, the boundary between Texas and Mexico was disputed, with the Republic of Texas and the U.S. asserting it to be the [Rio Grande](#) and Mexico claiming it to be the more-northern [Nueces River](#). Polk sent a diplomatic mission to Mexico in an attempt to buy the disputed territory, together with California and everything in between for \$25 million (equivalent to \$778 million in 2023), an offer the Mexican government refused.^{[9][10]} Polk then sent a group of 80 soldiers across the disputed territory to the Rio Grande, ignoring Mexican demands to withdraw.^{[11][12]} Mexican forces interpreted this as an attack and [repelled the U.S. forces](#) on April 25, 1846,^[13] a move which Polk used to convince the Congress of the United States to declare war.^[11]

Mexican–American War



Clockwise from top: [Winfield Scott](#) entering [Plaza de la Constitución](#) after the [Fall of Mexico City](#), U.S. soldiers engaging the retreating Mexican force during the [Battle of Resaca de la Palma](#), U.S. victory at [Churubusco](#) outside of Mexico City, Marines storming [Chapultepec castle](#) under a large U.S. flag, [Battle of Cerro Gordo](#)

Date	April 25, 1846 – February 2, 1848 (1 year, 9 months, 1 week and 1 day)
Location	Texas , New Mexico , California ; Northern, Central, and Eastern Mexico; Mexico City
Result	American victory ^[1]

World War II

Enigma == Puzzle



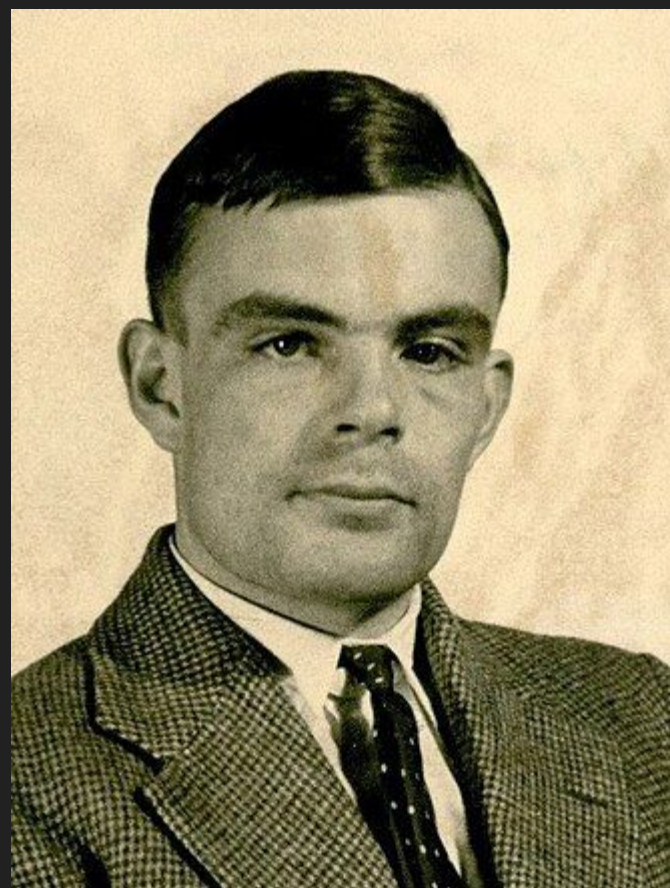
Change Keys every day in the wars



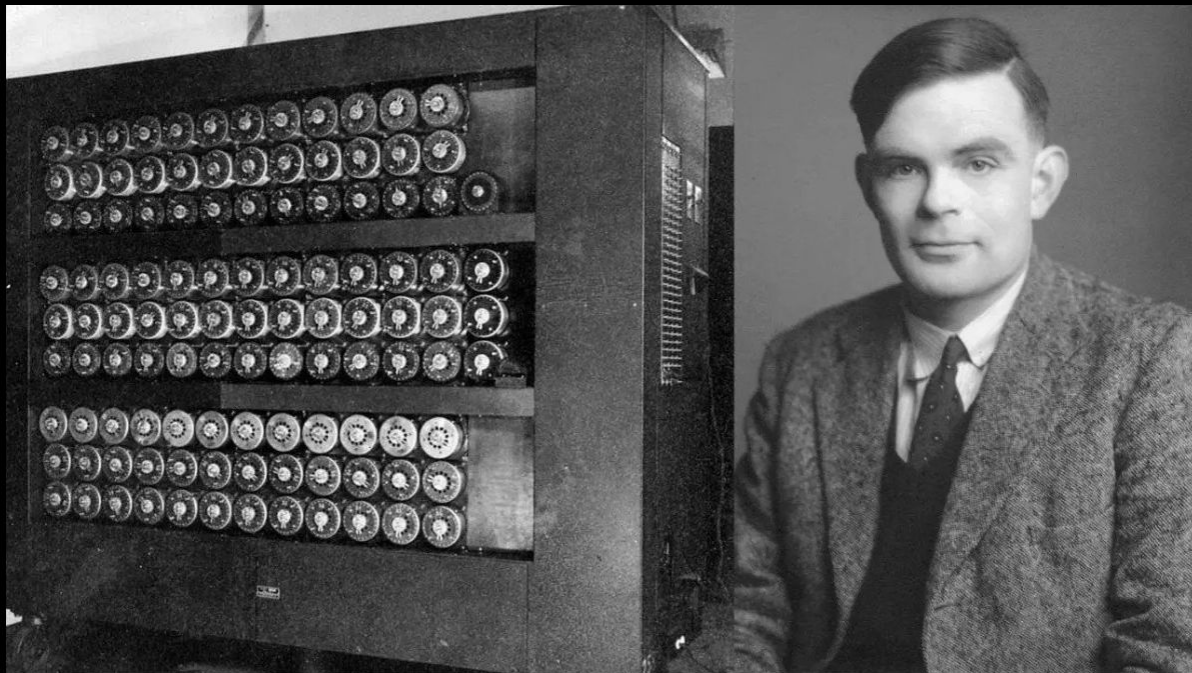
Fig. 4. Standard military ENIGMA

<https://www.cryptomuseum.com/crypto/enigma/i/index.htm>

The "Enigma" was a cipher machine extensively used by **Nazi Germany** during World War II to encode secret messages, considered so secure at the time that it was believed to be unbreakable



1936: Alan Turing & The Turing Machine - Pivotal Moments



Welcome to our fourth Pivotal Moments blog. If you follow us on [social media](#), you'll know what this is all about. To paraphrase Confucius, it's only by knowing where we've been that we can understand where we're going.



Modern Cryptography

100 year ago ~ present

Symmetric Encryption

Symmetric encryption is a type of encryption key management solution where only one key (a secret key) is used to both encrypt and decrypt.

DES (Data Encryption Standard)

- Originally designed **by IBM** and adopted by the U.S. government in 1977 as a Federal Data Processing Standard.
- Uses a 56-bit key and supports multiple operation modes (e.g., ECB, CBC).
- Now considered insecure primarily due to its relatively short key length, making it vulnerable to **brute-force attacks**.

AES (Advanced Encryption Standard)

- Also known as Rijndael, designed by Joan Daemen and Vincent Rijmen.
- One of the **most widely used symmetric encryption** algorithms today.
- Supports key sizes of 128, 192, or 256 bits.
- Offers strong security and high performance, making it the top choice in most modern applications.

Blowfish

- Designed by Bruce Schneier; supports a variable key length ranging from 32 to 448 bits.
- Known for its fast encryption speed and high security.
- Particularly suitable for environments with limited resources (e.g., embedded systems).

ChaCha20

- Designed by Daniel J. Bernstein; typically classified as a **stream cipher** but also considered a form of symmetric encryption.
- Renowned for its efficiency, speed, and strong security properties.
- Widely used in modern protocols, including certain cipher suites in TLS 1.3.

ASCII Table

Dec = Decimal Value
Char = Character

'5' has the int value 53

if we write '5'-'0' it evaluates to 53-48, or the int 5

if we write char c = 'B'+32; then c stores 'b'

128 chars

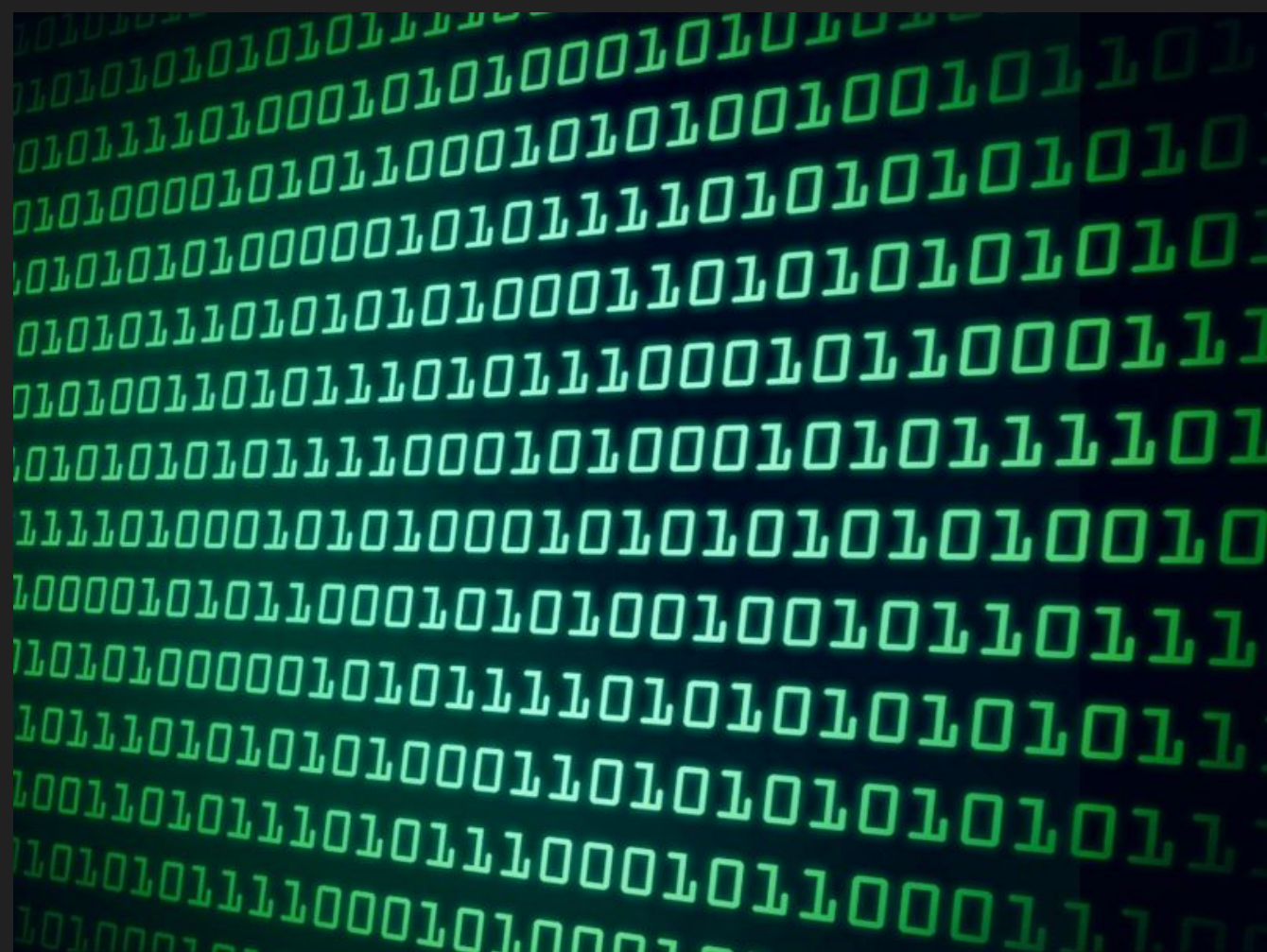
Dec	Char	Dec	Char	Dec	Char	Dec	Char
0	NUL (null)	32	SPACE	64	@	96	`
1	SOH (start of heading)	33	!	65	A	97	a
2	STX (start of text)	34	"	66	B	98	b
3	ETX (end of text)	35	#	67	C	99	c
4	EOT (end of transmission)	36	\$	68	D	100	d
5	ENQ (enquiry)	37	%	69	E	101	e
6	ACK (acknowledge)	38	&	70	F	102	f
7	BEL (bell)	39	'	71	G	103	g
8	BS (backspace)	40	(72	H	104	h
9	TAB (horizontal tab)	41)	73	I	105	i
10	LF (NL line feed, new line)	42	*	74	J	106	j
11	VT (vertical tab)	43	+	75	K	107	k
12	FF (NP form feed, new page)	44	,	76	L	108	l
13	CR (carriage return)	45	-	77	M	109	m
14	SO (shift out)	46	.	78	N	110	n
15	SI (shift in)	47	/	79	O	111	o
16	DLE (data link escape)	48	0	80	P	112	p
17	DC1 (device control 1)	49	1	81	Q	113	q
18	DC2 (device control 2)	50	2	82	R	114	r
19	DC3 (device control 3)	51	3	83	S	115	s
20	DC4 (device control 4)	52	4	84	T	116	t
21	NAK (negative acknowledge)	53	5	85	U	117	u
22	SYN (synchronous idle)	54	6	86	V	118	v
23	ETB (end of trans. block)	55	7	87	W	119	w
24	CAN (cancel)	56	8	88	X	120	x
25	EM (end of medium)	57	9	89	Y	121	y
26	SUB (substitute)	58	:	90	Z	122	z
27	ESC (escape)	59	;	91	[123	{
28	FS (file separator)	60	<	92	\	124	
29	GS (group separator)	61	=	93]	125	}
30	RS (record separator)	62	>	94	^	126	~
31	US (unit separator)	63	?	95	_	127	DEL

<https://www.rapidtables.com/convert/number/decimal-to-binary.html?x=97>

Hello == 72 101 108 108 111

1000 0011 0010 0110 1011 0001
1101 1111 1001 0110 1001 111 ==
47 bits

hello



Hello World!

Question?

8 Bits == 1 Byte

01001000 01100101 01101100
01101100 01101111 00100000
01010111 01101111 01110010
01101100 01100100 00100001

96 Bits == 12 Bytes

Binary Data



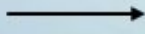
1 0 0 1 0 1 0 1

Algorithm Function



$f(x)$

Random
Encryption Key



0 1 1 1 0 0 1 0

Stream Cipher

Stream ciphers are the algorithms that encrypt basic information, one byte/bit at a time.

1 0 1 1 1 0 0 1



W³KO#

Ciphertext

Each Block

Encryption Key

Block Cipher

Encrypted Block

01001000

+



01100101

+



01101100

+



01011100

+



01101111

+



10101110

01110101

01001001

10110010

11101001



block ciphers separate the raw information into chunks of data of a fixed size.

10101110 + 01110101 + 01001001 + 10110010 + 11101001



10101110 01110101 01001001 10110010 11101001



W\$2M&

Ciphertext

Encryption is the process of transforming a message, usually called cyphered message, in such a way that only the intended parties can reverse it. The process of reversing encryption is called decryption. This implies that, as opposed to hashes, encryption is reversible.

There are two main types of encryption, symmetric and asymmetric, this chapter will cover symmetric encryption.

In symmetric encryption, the message to be sent is encrypted using a single secret password, also called key. Anyone with that secret key and decrypt the message and see the original content.

1000 Bytes

Key = 32 Bytes

AES (32 Bytes 256 bits)

Why secure ?

Let's put it this way: **brute-forcing** an AES-256 key—even with massive GPU power—is effectively impossible with current (and near-future) technology. The size of the AES-256 key space is 2^{256} , which is around 1.16×10^{77} . Even if you had a hypothetical machine that could test 10^{18} (one quintillion) keys per second—which is vastly more powerful than any general-purpose GPU cluster today—you would still need on the order of:

$$\frac{2^{256}}{10^{18}} \approx 10^{59} \text{ seconds}$$

To give that some perspective:

- There are about 3.154×10^7 seconds in a year.
- So that's about 10^{52} years of brute-forcing, many orders of magnitude greater than the age of the universe (roughly 1.38×10^{10} years).

In other words:

1. **It's not feasible** to brute-force AES-256 using any practical amount of GPUs.
2. **It's effectively negligible** (i.e., you can treat the probability of a successful brute force as zero for all realistic purposes).

Keys can include more than just numbers and characters.

They can be very long—up to 256 bits.

Even with GPU acceleration, guessing these keys remains extremely difficult.

Brute Force Attack is
Negligible to attack AES-256

Asymmetric Encryption

Asymmetric encryption, also known as public-key cryptography, is a method of encryption that uses **two different** keys: **a public key** that can be shared freely and **a private key that must be kept secret**, where data encrypted with the public key can only be decrypted using the corresponding private key, allowing secure communication without the need to pre-share a secret key with the recipient.

Game time!

RSA (Rivest–Shamir–Adleman)
ElGamal “**el-guh-MAHL**” Encryption
ECC (Elliptic Curve Cryptography)
Paillier (**‘pie-yay’**) Cryptosystem

Symmetric Encryption (Symmetric Encryption)

- **Advantages:** **Fast speed**, suitable for quickly encrypting and decrypting large-scale data.
- **Disadvantages:** Key management is difficult; the key must be securely shared and protected.

Asymmetric Encryption (Public-Key Encryption)

- **Advantages:** Public keys can be distributed openly, enabling secure key exchange and supporting digital signatures and identity authentication.
- **Disadvantages:** **Slower speed**, more complex algorithms, generally used only to encrypt a small amount of sensitive information or to exchange symmetric keys.

We usually use a combined
solution. How?

Demo



```
from cryptography.fernet import Fernet

# we will be encrypting the below string.
message = "hello geeks"

# generate a key for encryption and decryption
# You can use fernet to generate
# the key or use random key generator
# here I'm using fernet to generate key

key = Fernet.generate_key()

# Instance the Fernet class with the key

fernet = Fernet(key)

# then use the Fernet class instance
# to encrypt the string string must
# be encoded to byte string before encryption
encMessage = fernet.encrypt(message.encode())

print("original string: ", message)
print("encrypted string: ", encMessage)

# decrypt the encrypted string with the
# Fernet instance of the key,
# that was used for encrypting the string
# encoded byte string is returned by decrypt method,
# so decode it to string with decode methods
decMessage = fernet.decrypt(encMessage).decode()

print("decrypted string: ", decMessage)
```

Output:

```
(venv) C:\Users\user>python3 program.py
original string:  hello geeks
encrypted string:  b'gAAAAABgE4gyG_0ceYqYzE8_qRFbiQ6EO_6ms-uSXiCK9af2PTp4a8e_ONxc2Xy071rzaKxPHvG-jh0iq0CDEky3F_Qmjv8Cdw=='
decrypted string:  hello geeks
```



```
import rsa

# generate public and private keys with
# rsa.newkeys method, this method accepts
# key length as its parameter
# key length should be atleast 16
publicKey, privateKey = rsa.newkeys(512)

# this is the string that we will be encrypting
message = "hello geeks"

# rsa.encrypt method is used to encrypt
# string with public key string should be
# encode to byte string before encryption
# with encode method
encMessage = rsa.encrypt(message.encode(),
                          publicKey)

print("original string: ", message)
print("encrypted string: ", encMessage)

# the encrypted message can be decrypted
# with ras.decrypt method and private key
# decrypt method returns encoded byte string,
# use decode method to convert it to string
# public key cannot be used for decryption
decMessage = rsa.decrypt(encMessage, privateKey).decode()

print("decrypted string: ", decMessage)
```


Output:

```
original string: hello geeks
encrypted string: b'\x8f\xd0>\xce\xdf>\xec\x14\xb4R\x93\xab+\xcd\x18\xaci\x949\xfcCr\x8e\xe9\xfb
decrypted string: hello geeks
```


Future:

Quantum Computing is trying to
destory all the existed solutions



Peter Shor (pictured here in ) showed in 1994 that a scalable quantum computer would be able to break RSA encryption.