

116 CSC Symmetric / Asymmetric Encryption

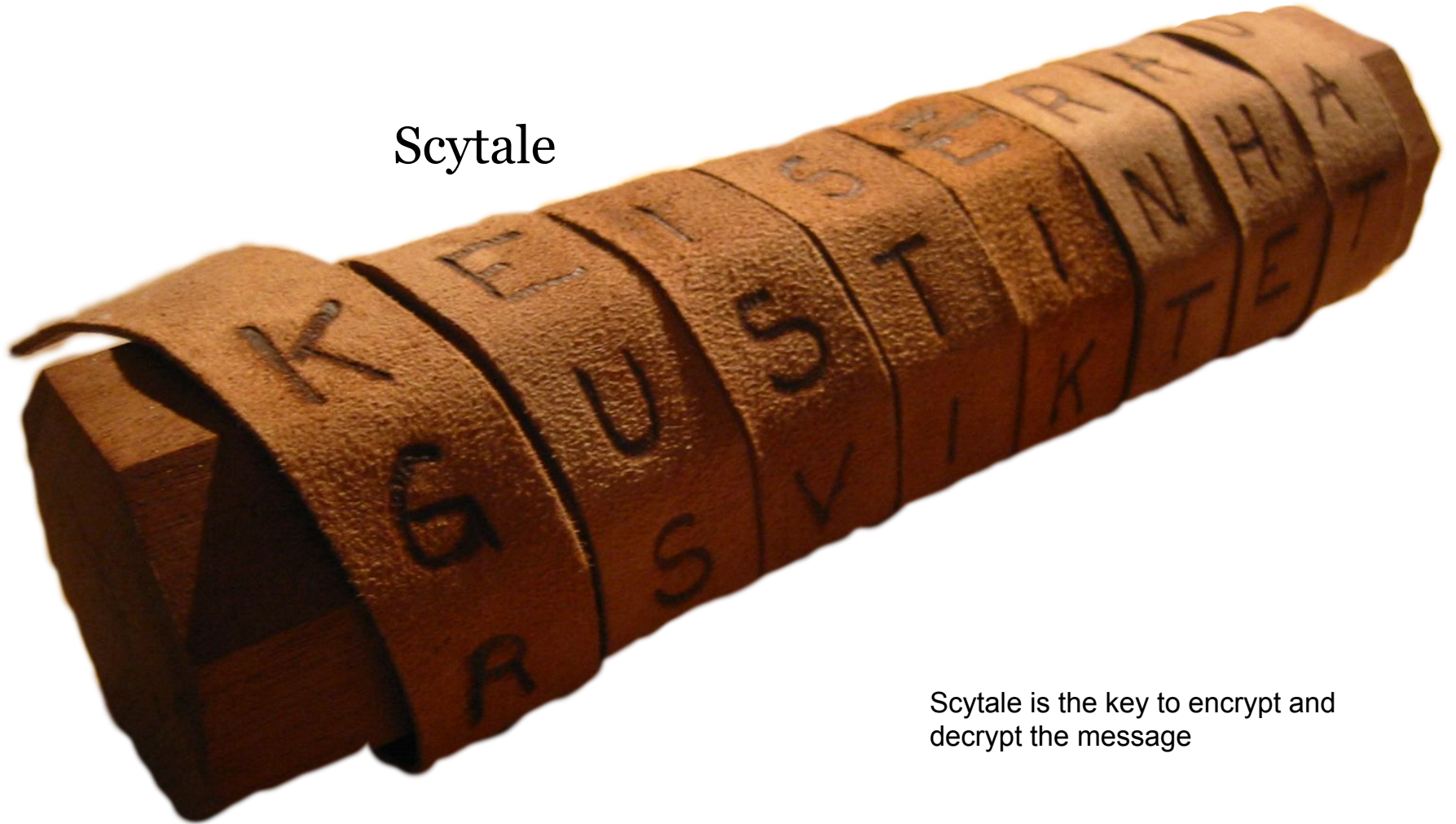
Instructor: Yusen Wu

Department of Neurology,
Department of Computer Science,
Frost Institute for Data Science and Computing,
University of Miami

**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	y
f	i	r	s	t	e	n	c
r	y	p	t	e	d	m	e
e	s	a	g	e			

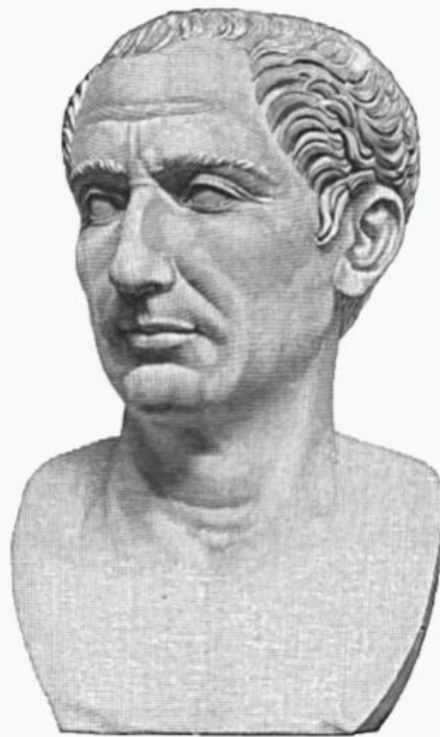
Ciphertext:


Tfrehiysirpasstgiteesedmnmyce

Question?

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.

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

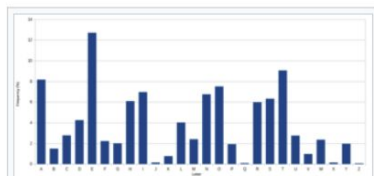
Random **T Y Z B C F H J**

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	fyyfhpfytshj

language distributions.^[23]

World War I

Cipher Disk



**1470 AD –
1940 AD**

600-200 years ago

World War II

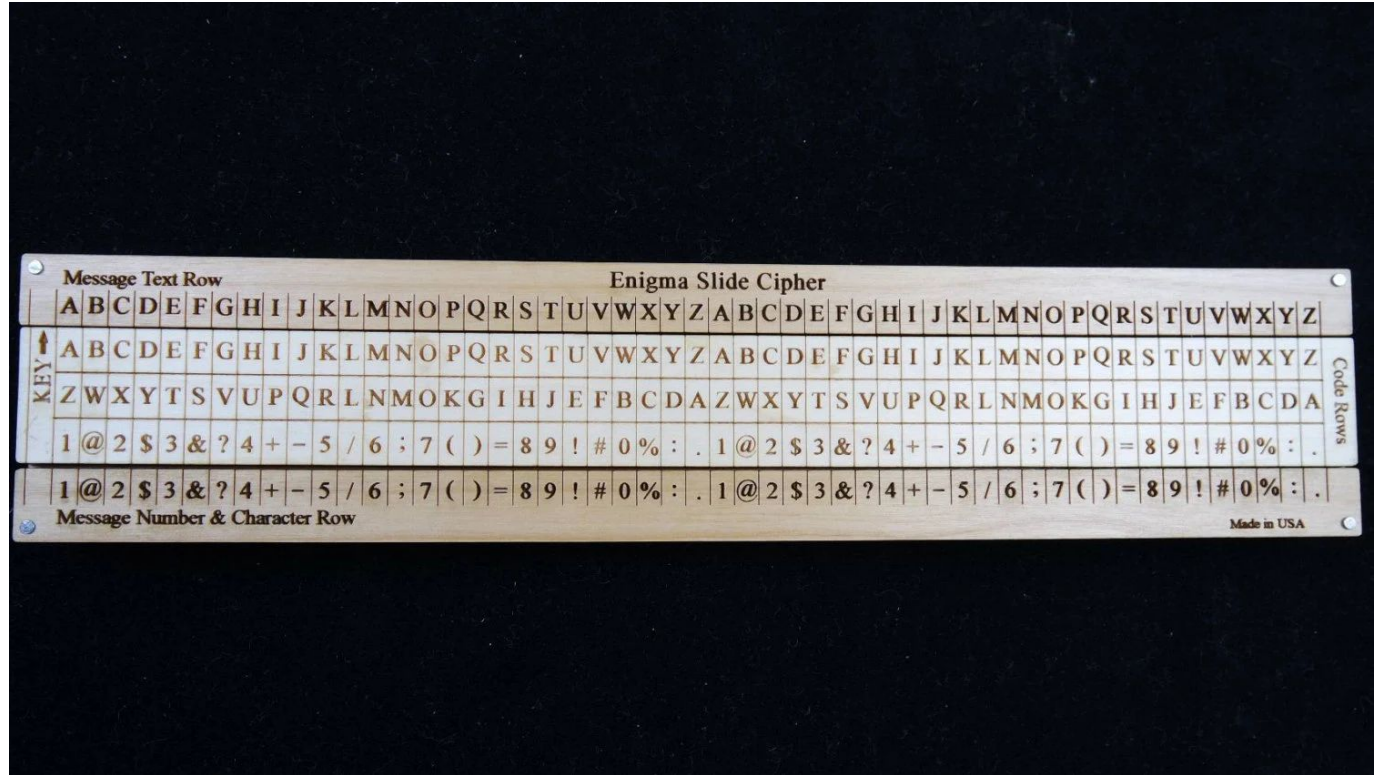


<https://www.cryptomuseum.com/crypto/enigma/i/index.htm>

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

Fig. 4. Standard military ENIGMA

Enigma



Change Keys every day in the wars

Plaintext:

HELLO

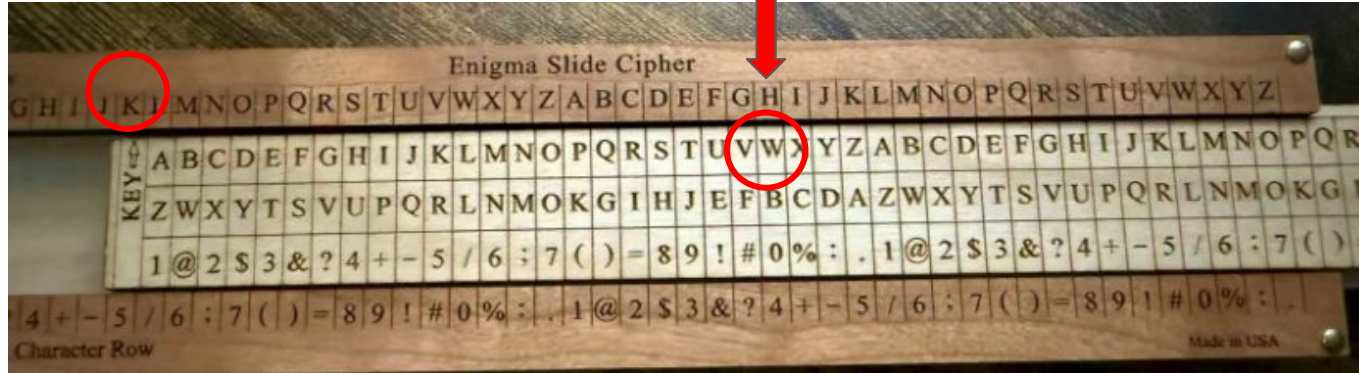
Keys:

KEYYKE

H

K

Ciphertext:



Key space

Char: 123

Encrypted message: **CGZ**

111 112 113 121 123 131 132 133

$$3 * 3 * 3 = \mathbf{27}$$

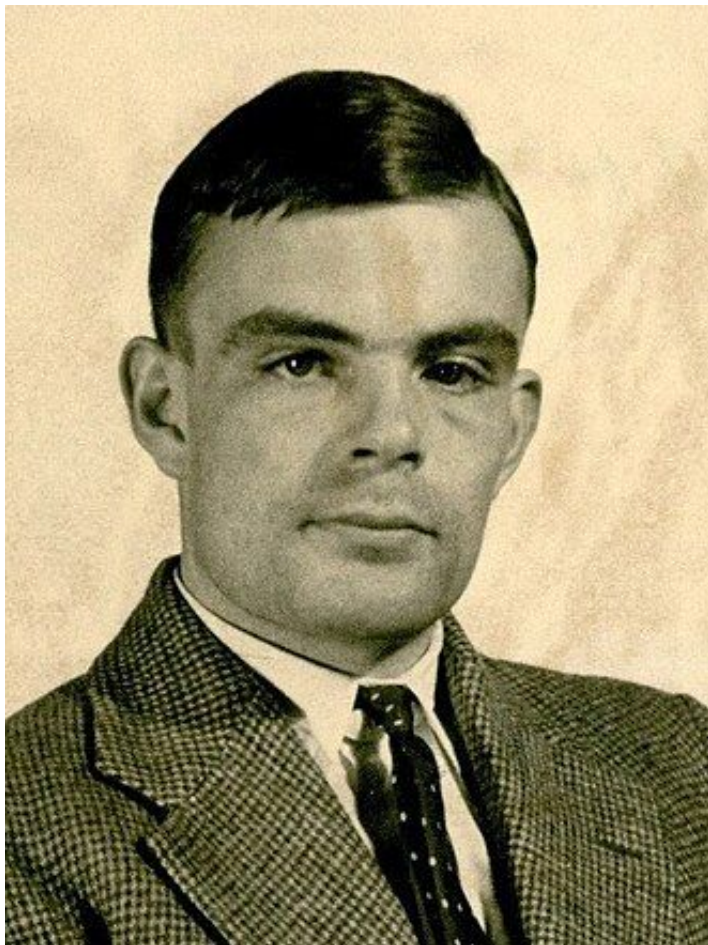
Key Space



52 unique characters

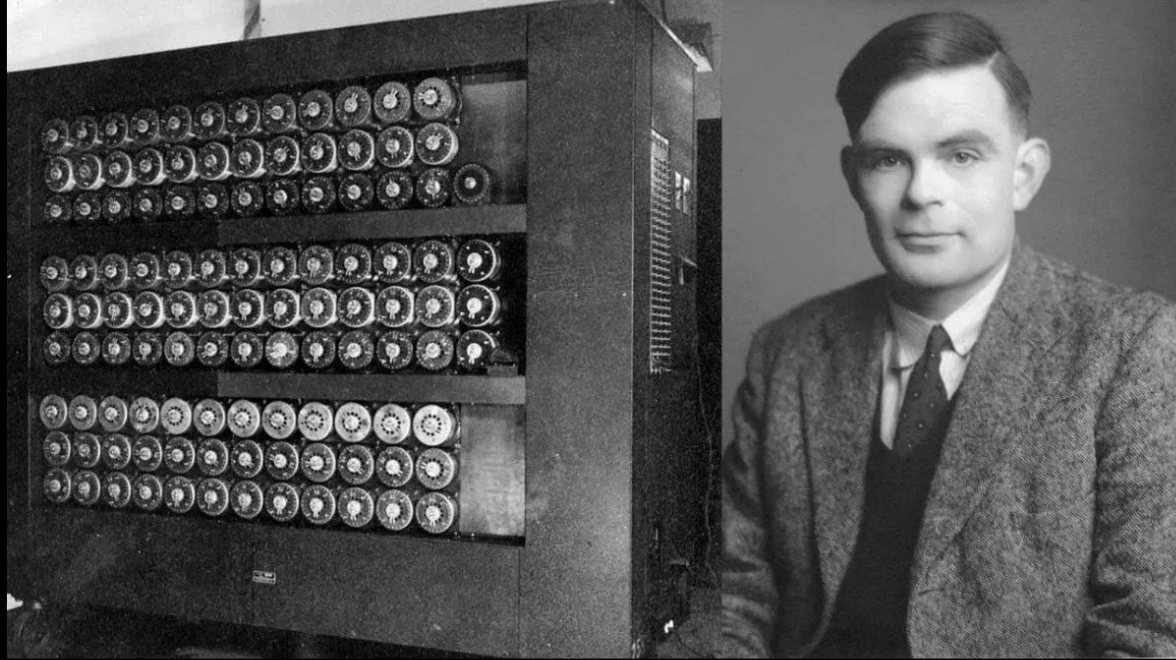
Question ?

**If I send you 100 encrypted letters,
how to calculate the key space?**



Alan Turing

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)

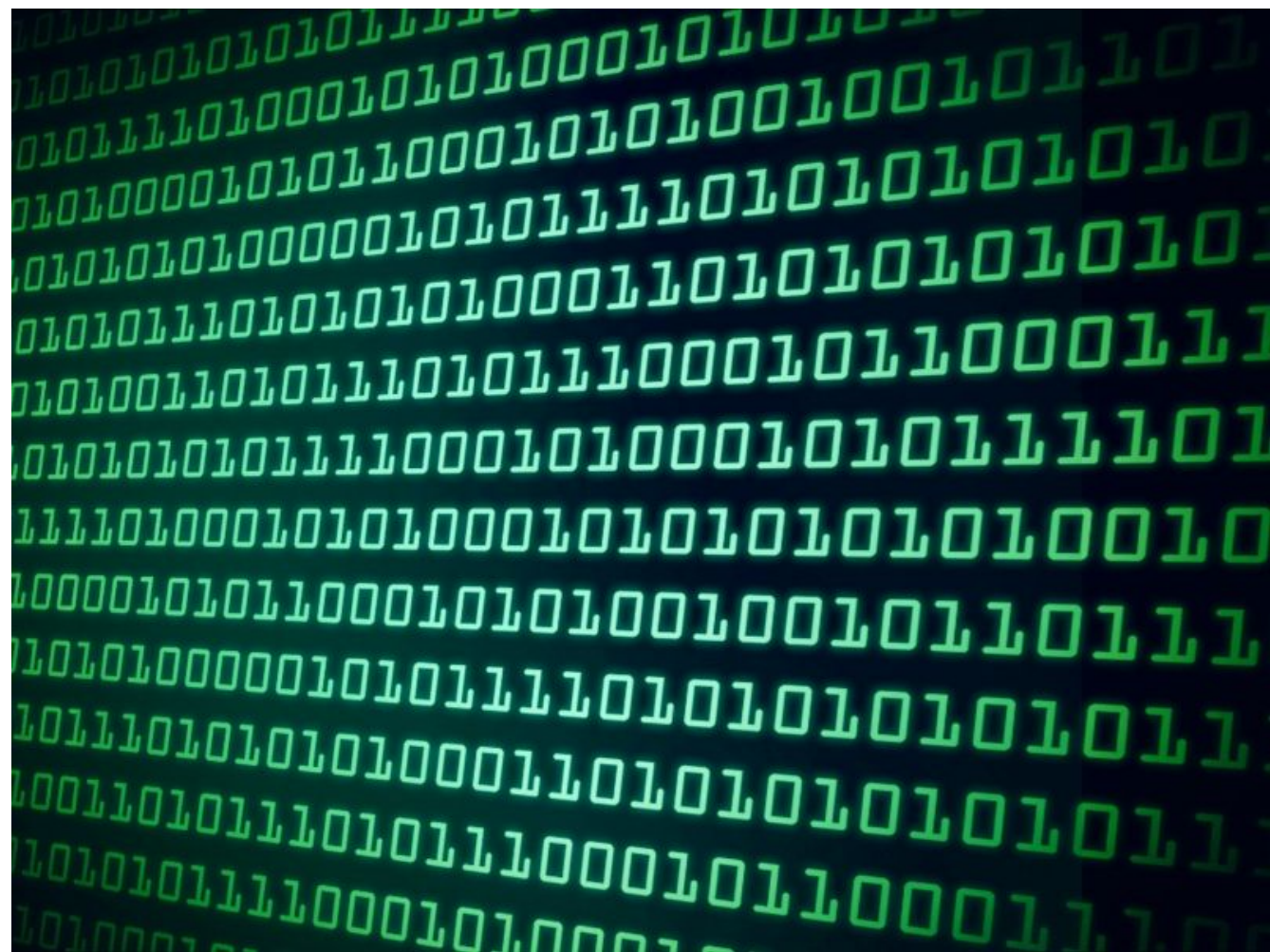
- Also known as Rijndael, designed by Joan Daemen and Vincent Rijmen.
- One of the **most widely used symmetric encryption** algorithms today.
- Supports key sizes of 128, 192, or 256 bits.
- Offers strong security and high performance, making it the top choice in most modern applications.

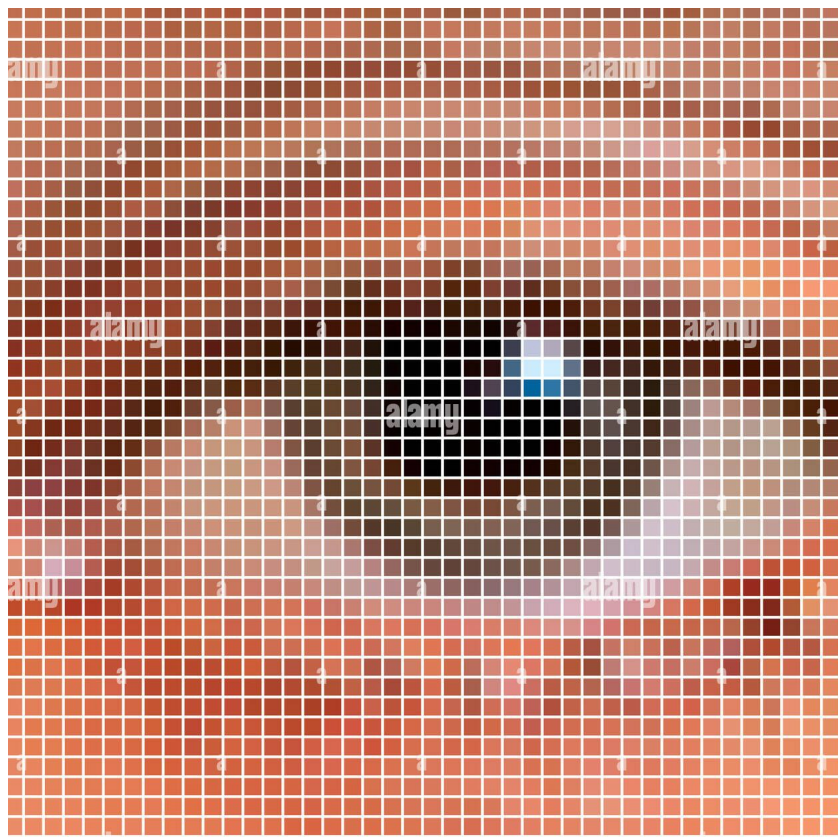
Blowfish

- 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.

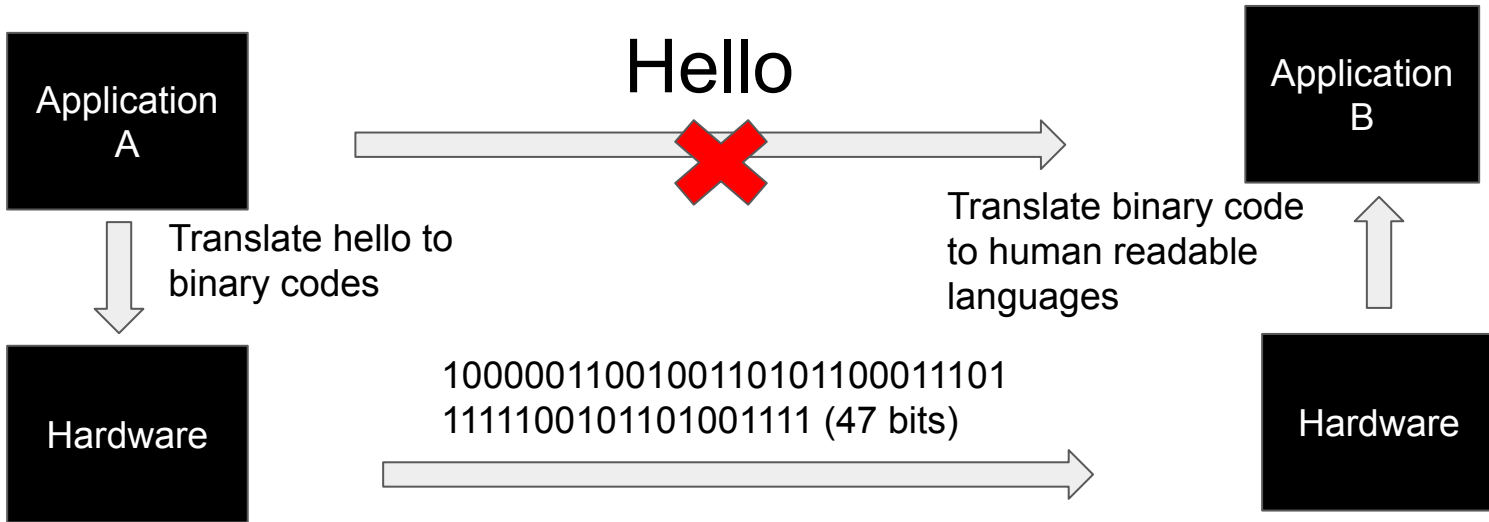




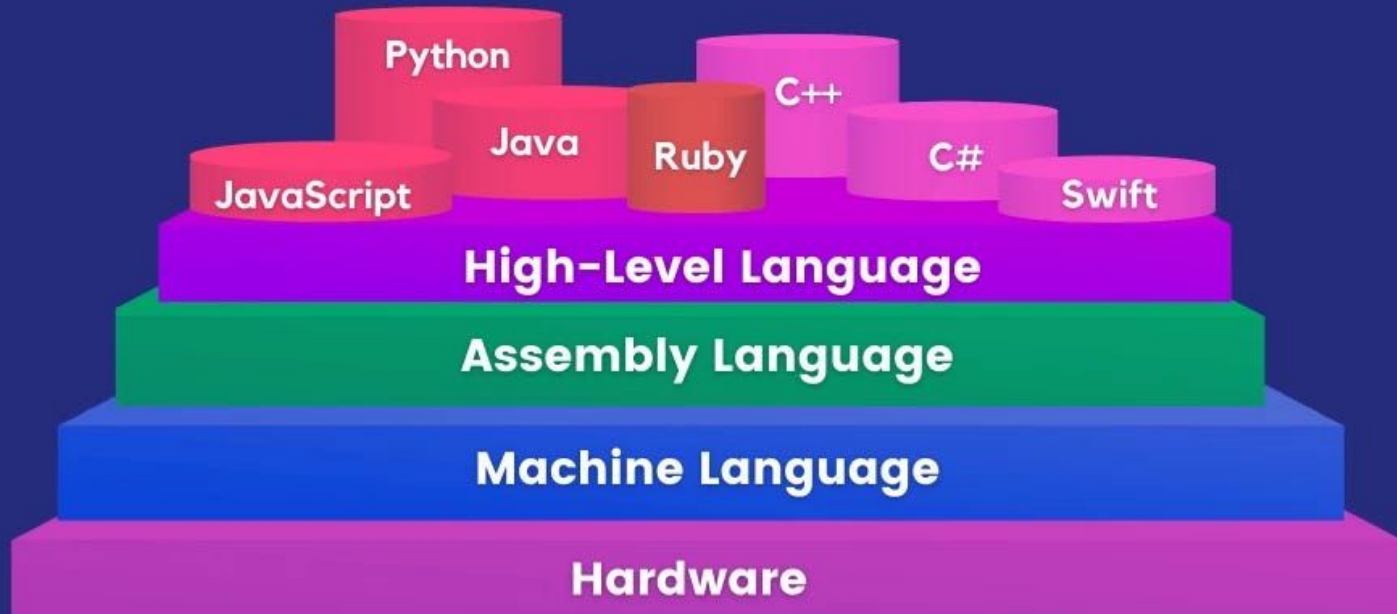
FFF FFF	CCC CCC	999 999	666 666	333 333	000 000	FPC C00	FF9 900	FF6 600	FP3 300										
99C C00					CC9 900	FPC C33	FPC C66	FF9 966	FF6 633	CC3 300								CC0 033	
CCF F00	CCF F33	333 300	666 600	999 900	CCC C00	FFF F00	CC9 933	CC6 633	330 300	660 000	990 000	CC0 000	FF0 000	FF3 366	FF0 033				
99F F00	CCF F66	99C C33	666 633	999 933	CCC C33	FFF F33	996 600	993 300	663 333	993 333	CC3 333	FF3 333	CC3 366	FF6 699	FF0 066				
66F F00	99F F66	66C C33	669 900	999 966	CCC C66	FFF F66	996 633	663 300	996 666	CC6 666	FF6 666	990 033	CC3 399	FF6 6CC	FF0 099				
33F F00	66F F33	339 900	66C C00	99F F33	CCC C99	FFF F99	CC9 966	CC6 600	CC9 999	FF9 999	FF3 399	CC0 066	990 066	FF3 3CC	FF0 0CC				
00C C00	33C C00	336 600	669 933	99C C66	CCF F99	FFF FCC	FPC C99	FF9 933	FPC CCC	FF9 9CC	CC6 699	993 366	660 033	CC0 099	330 033				
33C C33	66C C66	00F F00	33F F33	66F F66	99F F99	CCF FCC					CC9 9CC	996 699	993 399	990 399	663 366	660 066			
006 600	336 633	009 900	339 933	669 966	99C C99					FPC CFE	FF9 9FF	FF6 6FF	FF3 3FF	FF0 OFF	CC6 6CC	CC3 3CC			
003 300	00C C33	006 633	339 966	66C C99	99F FCC	CCF FFF	339 006	99C 669	CCC 999	CC9 999	996 600	663 660	330 660	990 0CC	CC0 0CC				
00F F33	33F F66	009 933	00C C66	33F F99	99F FFF	99C CCC	006 6CC	669 9CC	999 9FF	999 9CC	993 3FF	660 0CC	660 099	CC3 3FF	CC0 OFF				
00F F66	66F F99	33C C66	009 966	66F FFF	66C CCC	669 999	003 366	336 699	666 6FF	666 6CC	666 699	330 099	993 3CC	CC6 6FF	990 OFF				
00F F99	66F FCC	33C C99	33F FFF	33C CCC	339 999	336 666	006 699	003 399	333 3FF	333 3CC	333 399	333 366	663 3CC	996 6FF	660 OFF				
00F FCC	33F FCC	00F FFF	00C CCC	009 999	006 666	003 333	339 9CC	336 6CC	000 OFF	000 0CC	000 099	000 066	000 033	663 3FF	330 OFF				
00C C99	© 2006 VisiBone					009 9CC	33C CFE	66C CFE	669 9FF	336 6FF	003 3CC						330 0CC		
						00C CFE	009 9FF	006 6FF	003 3FF										

COMPUTER A

COMPUTER B



> hackr.io



ASCII Table

Dec = Decimal Value
Char = Character

128 chars

'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'

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?

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

96 Bits == 12 Bytes

8 Bits == 1 Byte

Binary Data



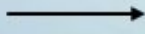
1 0 0 1 0 1 0 1

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

+



10101110

01100101

+



01110101

01101100

+



01001001

01011100

+



10110010

01101111

+



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

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).

Brute Force Attack is **Negligible** to
attack AES-256

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.

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.

RSA (Rivest–Shamir–Adleman)

1. Asymmetric encryption is NOT used for encrypting large messages.
2. Instead, it is mainly used to encrypt small secrets (like an AES key)

How Large Can RSA, ECC, or ElGamal Encrypt?

Asymmetric Algorithm	Max Message Size (for 2048-bit key)
RSA (2048-bit key)	~245 bytes (with padding)
ECC (256-bit key)	Very small, used for key exchange
ElGamal	Similar to RSA, also slow for large data

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???**

One time pad

The **One-Time Pad (OTP)** is a cryptographic method that provides **perfect secrecy**, meaning it is theoretically **unbreakable**

Why Isn't OTP Used in Practice?

✗ Key Distribution Problem

- The sender and receiver must share the same key in advance, which must be as long as the message.
- If you can securely transmit the key, why not send the message directly?

✗ Key Must Be Truly Random

- If the key is predictable, encryption is not secure.

✗ Key Can Only Be Used Once

- If a key is reused, attackers can detect patterns and break the encryption.

♦ **One-Time Pad is the only encryption method proven to be unbreakable.**

♦ **However, it is impractical for everyday use due to key management issues.**

♦ **Modern encryption (AES, RSA) is used instead, balancing security and usability.**

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.

1024 Bits

KEY length == 1024
Bits

Future **Quantum Computing** is
trying to destroy all the existed solutions



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

Post-quantum cryptography (PQC) is a new generation of cryptographic algorithms that are designed to protect against attacks from quantum computers.

Updates

Symmetric
Asymmetric

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.

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.

A — B

A use B's public key to encrypt the symmetric key
Send to B

B uses his/her private key to decrypt the ciphertext

A can send the messages to B by using symmetric
encryption

Question1:

In Asymmetric encryption,

can you use your **private key** to encrypt a secret,
and share this ciphertext to someone?

Discussion:

Application of these 2 Encryptions

File Encryption:

Protecting sensitive documents or compressed files.

Database Protection:

Securing stored sensitive data such as passwords or medical records.

HTTPS:

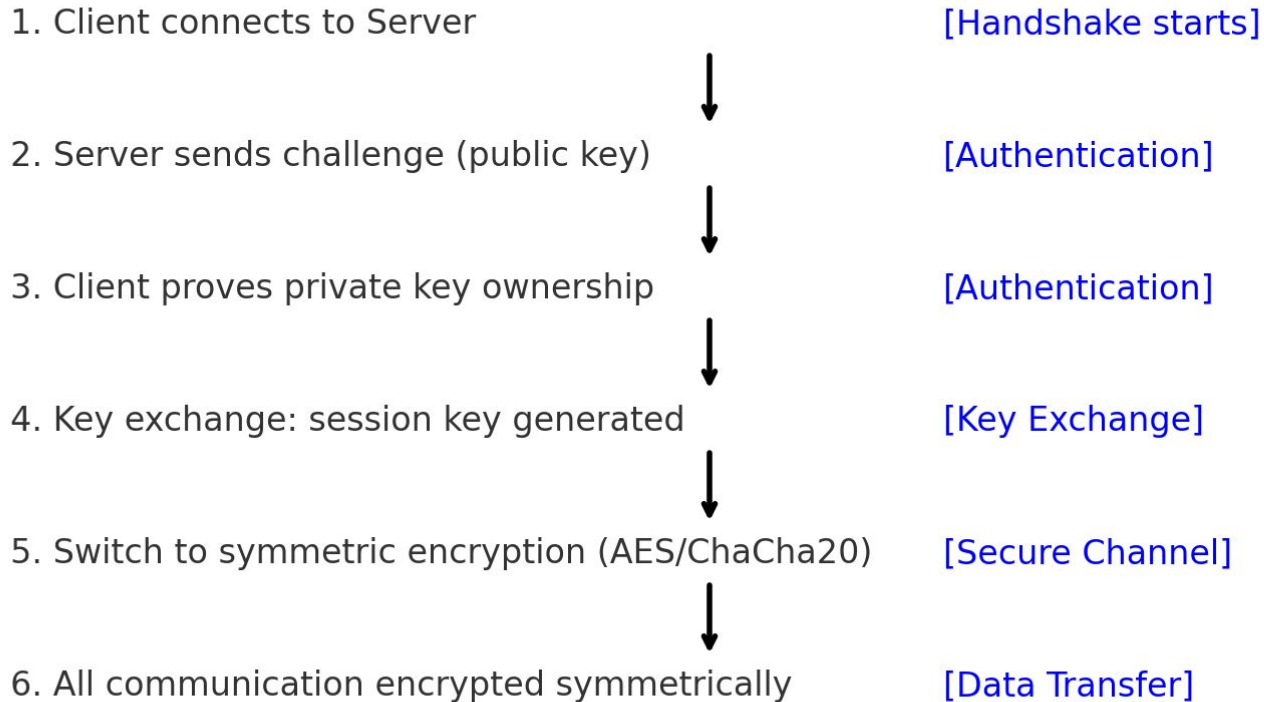
?

Browsers use asymmetric encryption to exchange keys securely, then switch to symmetric encryption for speed.

SSH/Github (for CS)

How to build a secure
connection with someone

□ SSH Connection Timeline



Additional HW for CSer(system design):

“If you were to design a **GROUP CHAT** messaging application, what kind of encryption strategy would you choose? How would you balance the requirements of security, scalability, maintainability, practice, and future upgradability?”

Answer with detailed design, workflow, technologies you use (the secure authentications, etc., ECSDA) and send to me before Sep 15th