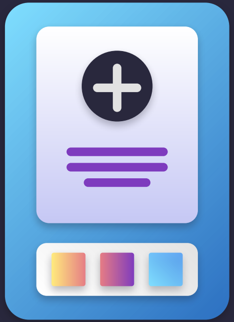
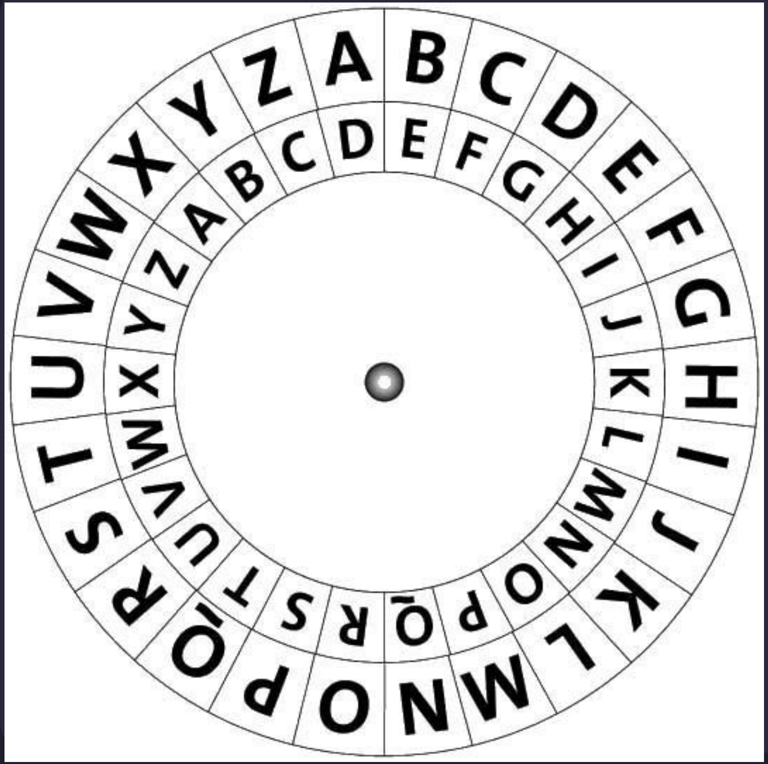
Public Key Encryption

By Benson Leung,
Marina Moshchenko, and
Mamudu Wally

/TABLE OF CONTENTS

Introduction

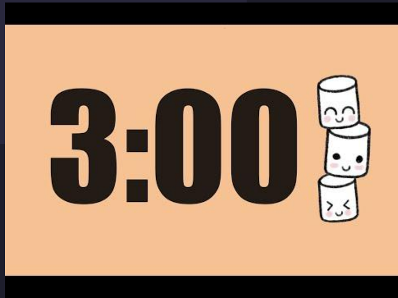


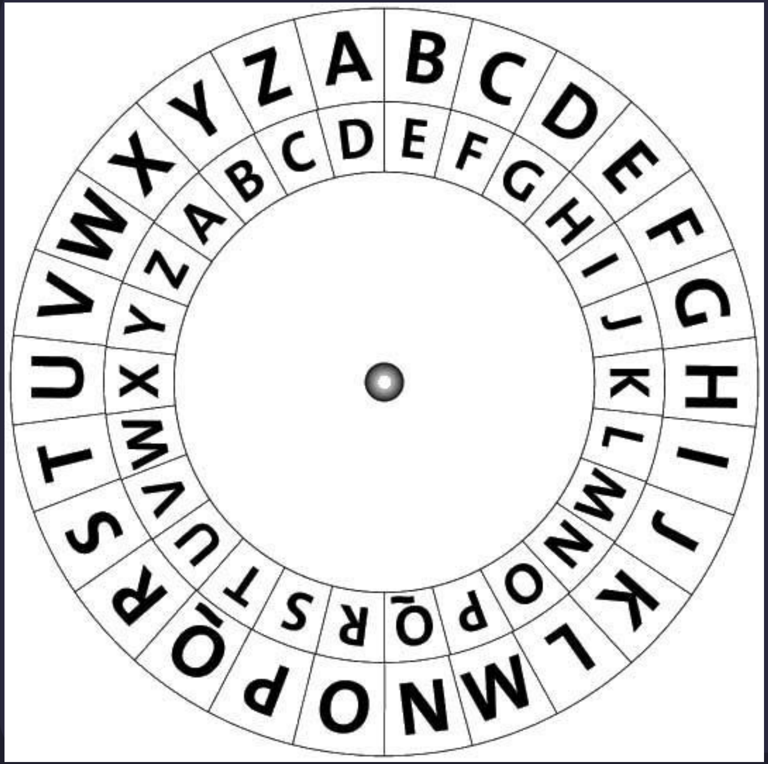


/Do Now - Part 1

1) Use a simple substitution cipher to encipher word “hello”.
2) For that: shift each letter of a word to a certain characters (your choice) in the same direction).
3) Post your result in to the slack.

WATERFALL STYLE





/Do Now - Part 2

4) Use online tool <u>dcode.fr/caesar-cipher</u> to DECRYPT any message from class

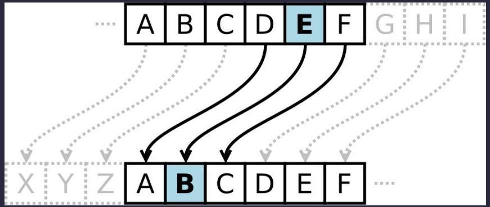
5) Post it to slack with the ciphered text.

Julius Caesar

YOUR LOGO HERE

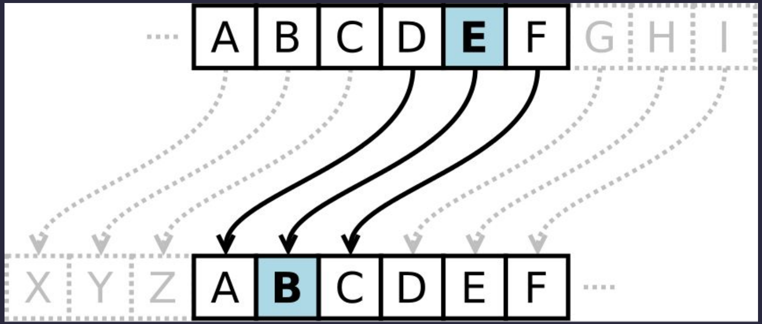
Caesar Cipher

A method of encryption where the letters of the original message are replaced with the corresponding letters in a shifted alphabet.

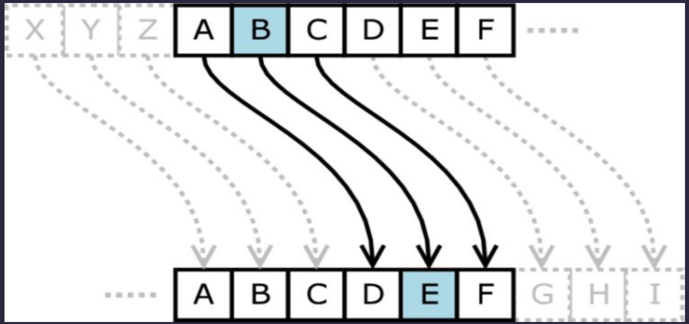


Compare and Contrast

Shift: 3 letters
Direction: LEFT



Shift: 3 letters
Direction: RIGHT



DIRECTION MATTERS!

Vigenère Cipher

Mathematical Approach (by adding letters' values)

Caesar Cipher: each letter of the alphabet is shifted a number (key = 0-25) of places (monoalphabetic)
Vigenère cipher has several Caesar ciphers in sequence with different shift values (polyalphabetic)

WORD: HELLO
KEY: TECH

A	B	C	D	E	F	G	H	I	J	K	L	M
↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕
0	1	2	3	4	5	6	7	8	9	10	11	12

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕
13	14	15	16	17	18	19	20	21	22	23	24	25

H (7) T(19) → 7+19=26; 26-26=0 (A)
E (4) E(4) → 4+4=8 (I)
L (11) C(2) → 11+2=13 (N)
L (11) H(7) → 11+7=18 (S)
O (14) T (19) → 14+19=33; 33-26=7 (H)

Cipher encoded word: **AINSH**

YOUR LOGO HERE

Activity

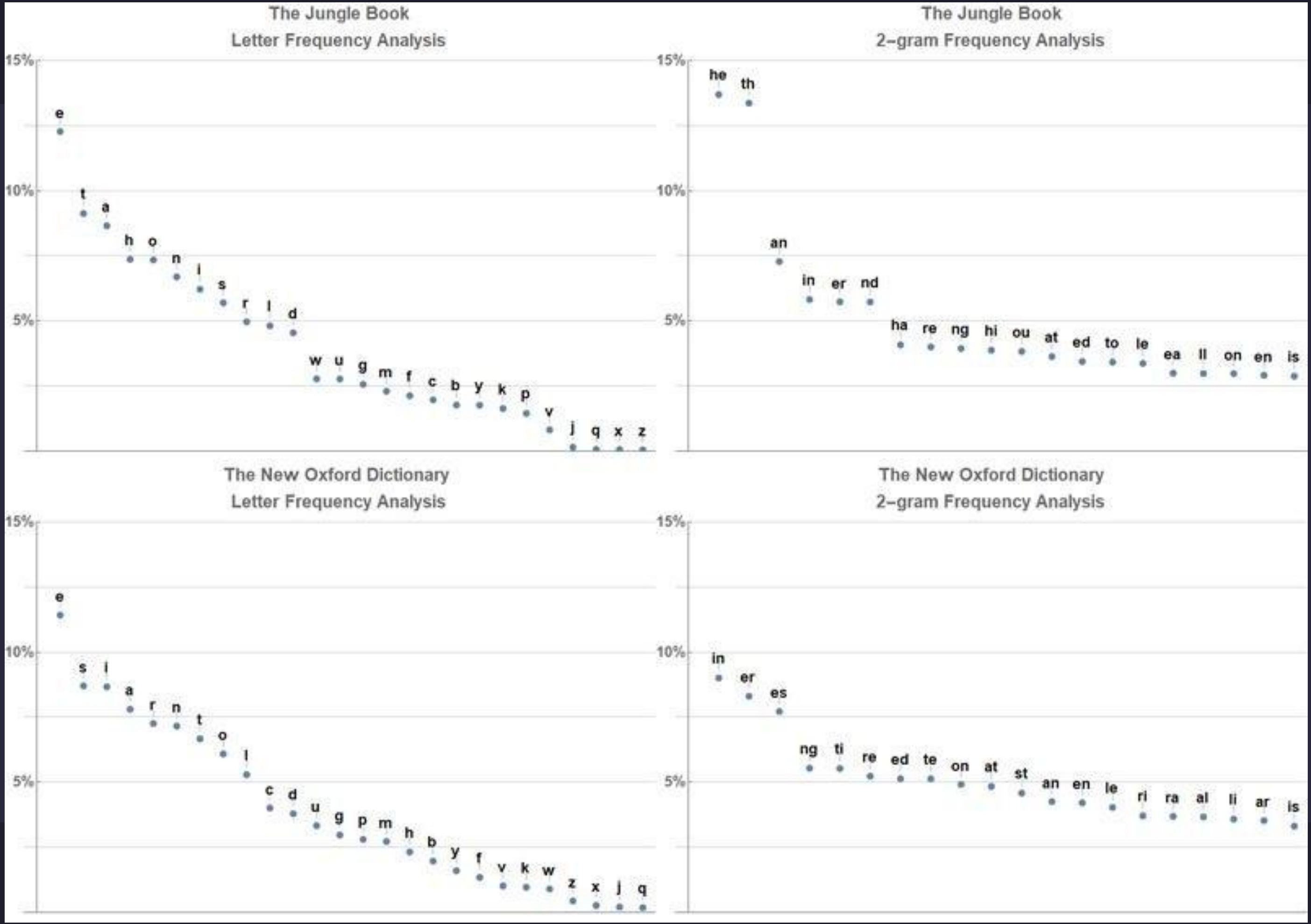
WORD: HELLO
KEY: TECH

H → H(T) → A
E → E(E) → I
L → L(C) → N
L → L(H) → S
O → O(T) → H

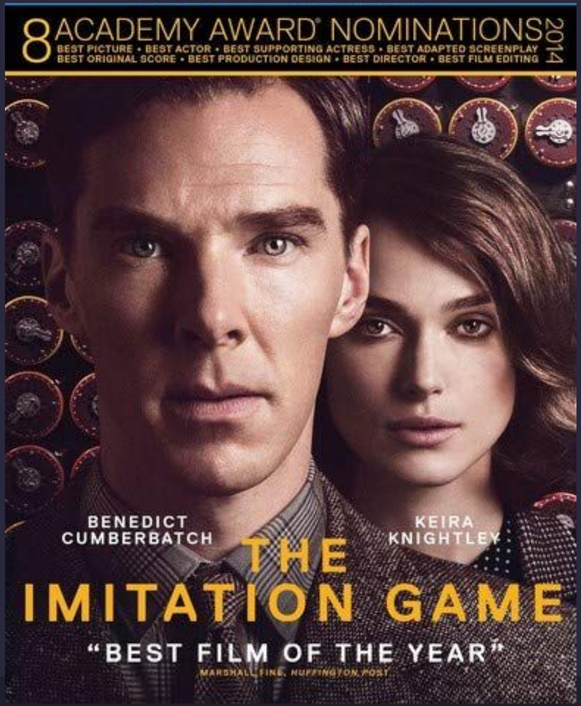
Cipher encoded word: **AINSH**

Decode AINSH with
https://www.dcode.fr/vigenere-cipher

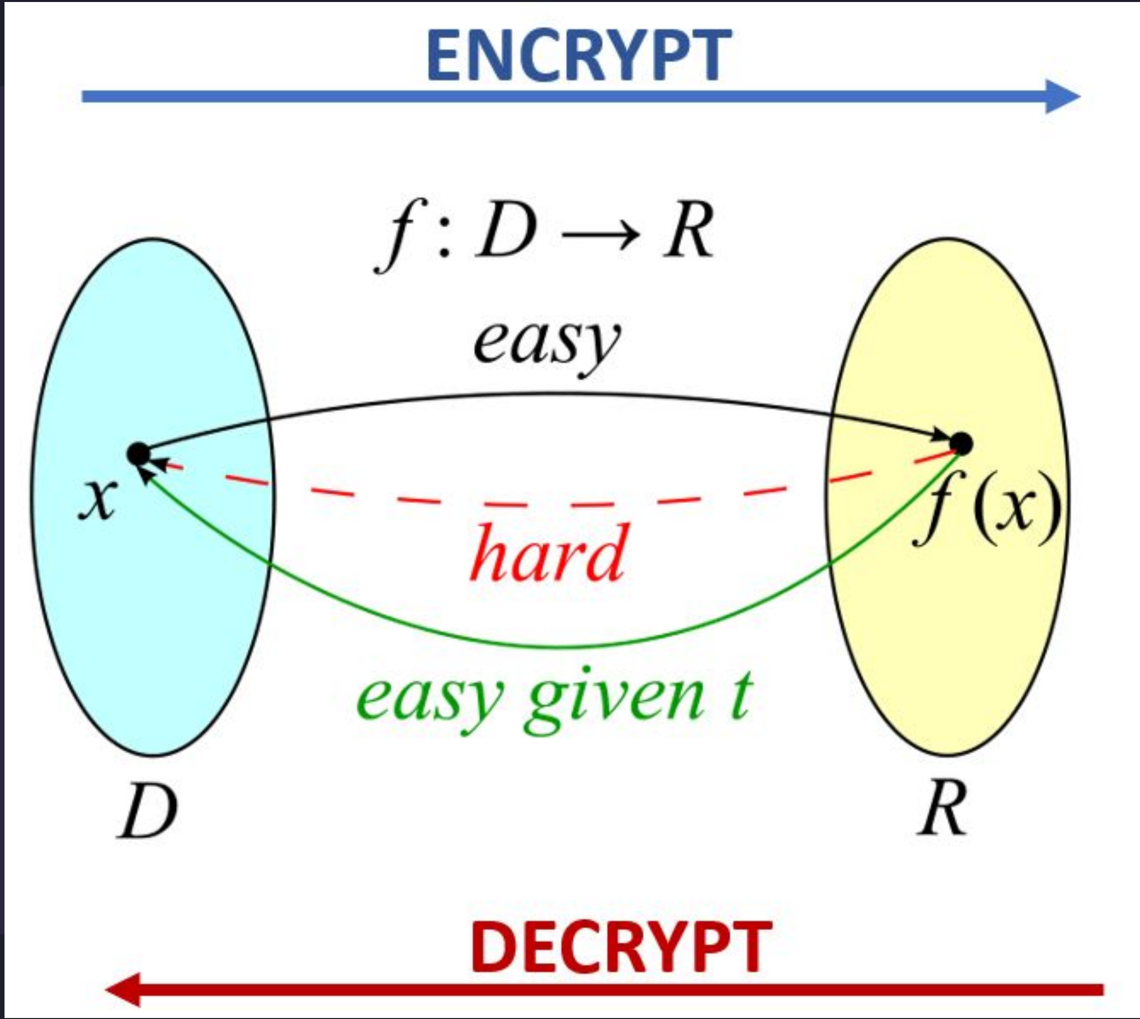
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T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


The Jungle Book
Letter Frequency Analysis
15%
10%
5%
e t a h o n i s r l d w u g m f c b y k p v j q x z
The Jungle Book
2–gram Frequency Analysis
15%
10%
5%
he th an in er nd ha re ng hi ou at ed to le ea ll on en is
The New Oxford Dictionary
Letter Frequency Analysis
15%
10%
5%
e s i a r n t o l c d u g p m h b y f v k w z x j q
The New Oxford Dictionary
2–gram Frequency Analysis
15%
10%
5%
in er es ng ti re ed te on at st an en le ri ra al li ar is

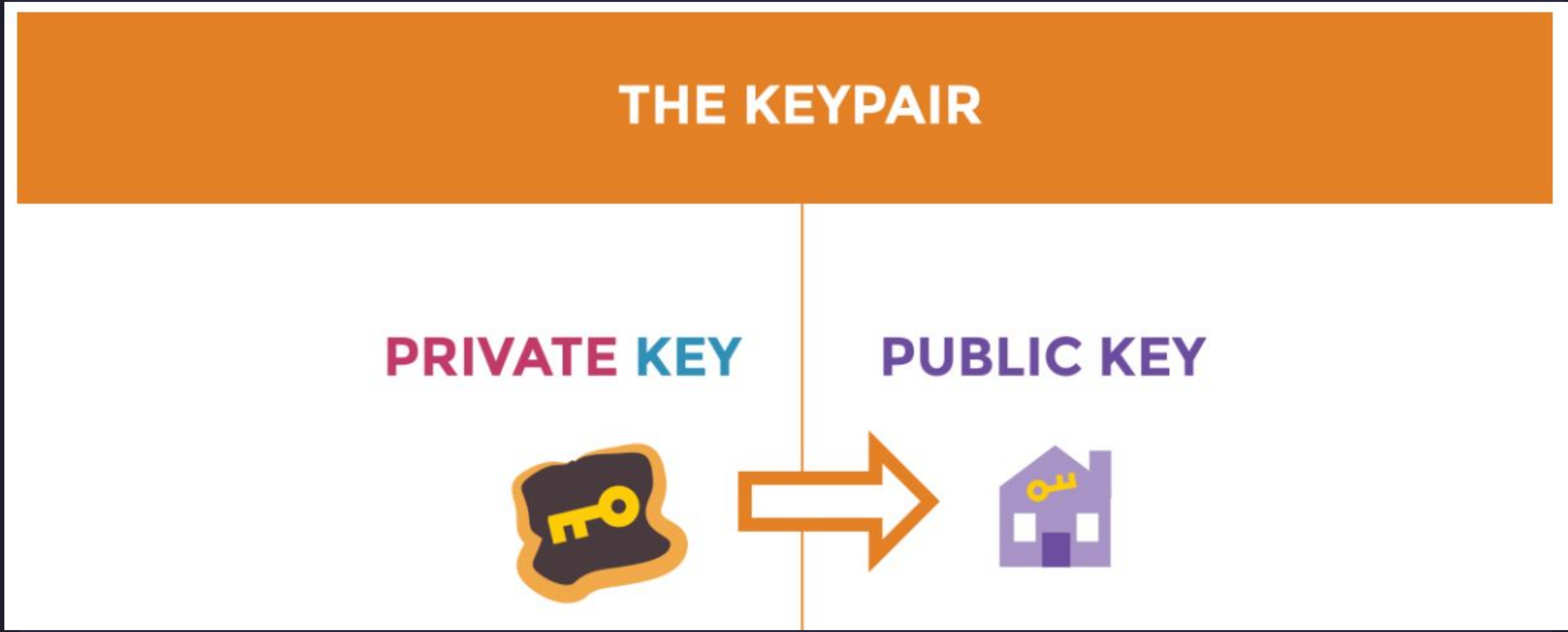
Enigma Code


8 ACADEMY AWARD® NOMINATIONS 2014
BEST PICTURE • BEST ACTOR • BEST SUPPORTING ACTRESS • BEST ADAPTED SCREENPLAY
BEST ORIGINAL SCORE • BEST PRODUCTION DESIGN • BEST DIRECTOR • BEST FILM EDITING
BENEDICT CUMBERBATCH
KEIRA KNIGHTLEY
THE IMITATION GAME
"BEST FILM OF THE YEAR"
MARSHALL FINE, HUFFINGTON POST

Trapdoor Function



Public Key to the Rescue



Think of a mailbox...

TRY IT! - ENCRYPT AND DECRYPT TEXT

GENERATE A PAIR OF PRIVATE AND PUBLIC KEYS:
https://www.codeusingjava.com/tools/rsa

1) The main person will send public key to team members.
2) They will use your **PUBLIC** key to encrypt the messages and send the messages back to you.
3) The main person will now use their **PRIVATE** key to decrypt all messages.
4) If we have time, switch main person and repeat until everyone has gone.

Timeline – History of Asymmetric Cryptography

1975

- Diffie imagines asymmetric cryptography
- Paper “New Direction in Cryptography” by Whitfield Diffie and Martie E. Hellman [Link]

1976

- Diffie-Hellman key exchange
- *Computational Diffie-Hellman* (CDH) assumption (allows two principals to set up a shared key given a public-key system)

1977

- RSA algorithm (public-key cryptosystem) - Rivest, Shamir, and Adelman shared the 2002 Turing Award for this development

1985

- ElGamal encryption system (based on DHKE algorithm)
- Elliptic Curve Cryptography (ECC), but in use since 2004-2005

ANY MODERN CRYPTO SYSTEM IS BASED ON A HARD MATHEMATICAL PROBLEM

RSA Cryptosystem

/RSA Cryptosystem

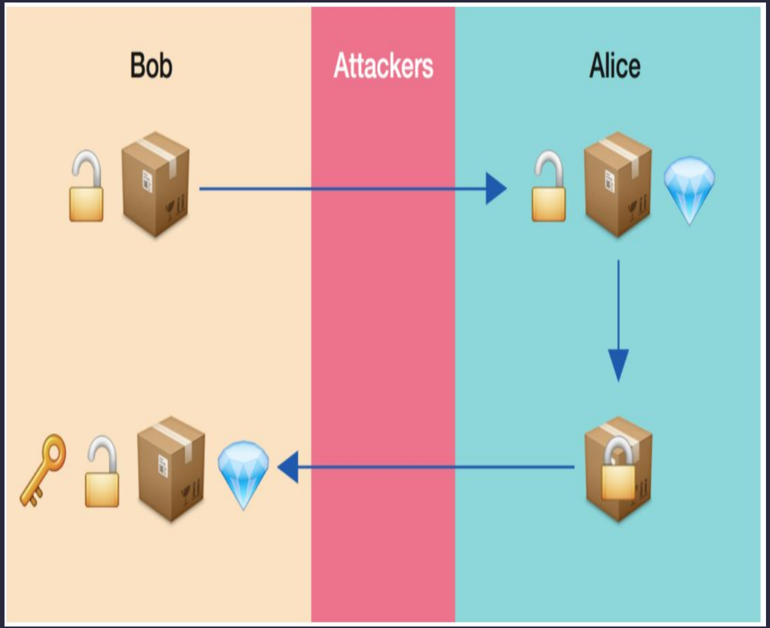
Invented by three scholars: Ron Rivest, Adi Shamir, and Len Adleman (RSA)

RSA is an encryption algorithm, used to securely transmit messages over the internet. It is based on the principle that it is easy to multiply large numbers, but factoring large numbers is very difficult. For example, it is easy to check that 31 and 37 multiply to 1147, but trying to find the factors of 1147 is a much longer process.

RSA is an example of public-key cryptography, which is illustrated by the following example: Suppose Alice wishes to send Bob a valuable diamond, but the jewel will be stolen if sent unsecured. Both Alice and Bob have a variety of padlocks, but they don't own the same ones, meaning that their keys cannot open the other's locks.

[1] How did Alice send the diamond to Bob?

The Process



1) Bob first sends Alice an unlocked padlock. Note that Bob would give anyone an unlocked padlock, as the only use for one is to send Bob something.
2) Alice adds Bob's lock and sends the package to Bob, and
3) Bob removes his lock and opens the package.

This example demonstrates the ideas behind public-key cryptography, though the concept is actually slightly different. In public-key cryptography, Alice encrypts her message using Bob's public key, which can only be decoded by Bob's private key.

/RSA Generation

In RSA, the public key is generated by multiplying two large prime numbers p and q together, and the private key is generated through a different process involving p and q. A user can then distribute his public key pq, and anyone wishing to send the user a message would encrypt their message using the public key. For all practical purposes, even computers cannot factor large numbers into the product of two primes, in the same way that factoring a number like 414863 by hand is virtually impossible.

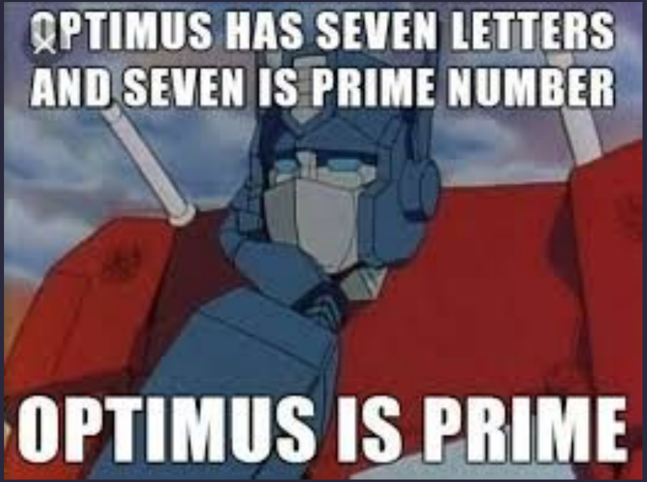
However, multiplying two numbers is much less difficult, so a potential factorization can be verified quickly, as the following multiple-choice problem shows:

Quick Check

In the Slack, choose your response:

[2] Which of the following is the prime factorization of 414863?

1) 577 x 709
2) 577 x 719
3) 587 x 709
4) 587 x 719



/The Algorithm

The implementation of RSA makes heavy use of modular arithmetic, Euler's theorem, and Euler's totient function. Notice that each step of the algorithm only involves multiplication, so it is easy for a computer to perform.

1) First, the receiver chooses two large prime numbers p and q. Their product, n=pq, will be half of the public key.
2) The receiver calculates $\phi(pq)=(p-1)(q-1)$ and chooses a number e relatively prime to $\phi(pq)$. In practice, e is often chosen to be $2^{16}+1=65537$, though it can be as small as 3 in some cases. e will be the other half of the public key.
3) The receiver calculates the modular inverse d of e modulo $\phi(n)$. In other words, $de \equiv 1 \pmod{\phi(n)}$. d is the private key.
4) The receiver distributes both parts of the public key: n and e. d is kept secret.

/Example

For example, suppose the receiver selected the primes $p=11$ and $q=17$, along with $e=3$.

1) The receiver calculates $n = pq = 11 \cdot 17 = 187$, which is half of the public key.
2) The receiver also calculates $\phi(n)=(p-1)(q-1) = 10 \cdot 16 = 160$.
   $e=3$ was also chosen.
3) The receiver calculates $d=107$, since then $de = 321 \equiv 1 (\bmod \phi(n))$ (since $\phi(n)=160$).
   a) d is the modulo inverse of e, which means that
      $d \cdot e = 1 \bmod 160$
4) The receiver distributes his public key: $n=187$ and $e=3$.

Encryption & Decryption Process

1) First, the sender converts his message into a number m. One common conversion process uses the ASCII alphabet:

A	B	C	D	E	F	G	H	I	J	K	L	M
65	66	67	68	69	70	71	72	73	74	75	76	77
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
78	79	80	81	82	83	84	85	86	87	88	89	90

Private	Public
p = 11	n = 187
q = 17	e = 3
d = 107	

2) For example, the message "HELLO" would be encoded as 7269767679. It is important that m<n, as otherwise the message will be lost when taken modulo n, so if n is smaller than the message, it will be sent in pieces.
3) The sender then calculates $c \equiv m^e \pmod{n}$. c is the ciphertext, or the encrypted message. Besides the public key, this is the only information an attacker will be able to steal.
4) The receiver computes $c^d \equiv m \pmod{n}$, thus retrieving the original number m.
5) The receiver translates m back into letters, retrieving the original message.

/Another Example

For example, suppose the receiver selected the primes p=11 and q=17, along with e=3.

Now suppose the sender wanted to send the message "HELLO". Since n is so small, the sender will have to send his message character by character.

1) 'H' is 72 in ASCII, so the message text is m=72.
2) The sender calculates $m^e = 72^3 \bmod 187 \equiv 183$, making the ciphertext c=183. Again, this is the only information an attacker can get, since the attacker does not have the private key.
3) The receiver calculates $c^d = 183^{107} \bmod 187 \equiv 72$, thus getting the message of m=72.
4) The receiver translates 72 into 'H'.
5) The rest of the letters are sent in the same way.

/Applications & Vulnerabilities

1. Because of the great difficulty in breaking RSA, it is almost universally used anywhere encryption is required: password exchange, banking, online shopping, and even cable television.
2. The strength of RSA is measured in key size, which is the number of bits in n=pq. As of 2016, 1024-bit (309 digits) keys are considered risky, and most newly generated keys are 4096-bit (1234 digits).

1. One common attack on RSA bypasses the algorithm altogether. A computer can quickly compute the greatest common divisor of two numbers using the Euclidean algorithm.
2. So an attacker can run this algorithm on two public keys.
3. If their greatest common divisor is not 1, then the attacker has found a prime number dividing both or multiple keys, therefore breaking two or more keys at the same time.

/Code for Finding GCD & Vulnerability on RSA

You have looked up several people's public keys, some of which are below. They are vulnerable because they are divisible by the same prime. Which prime is that?

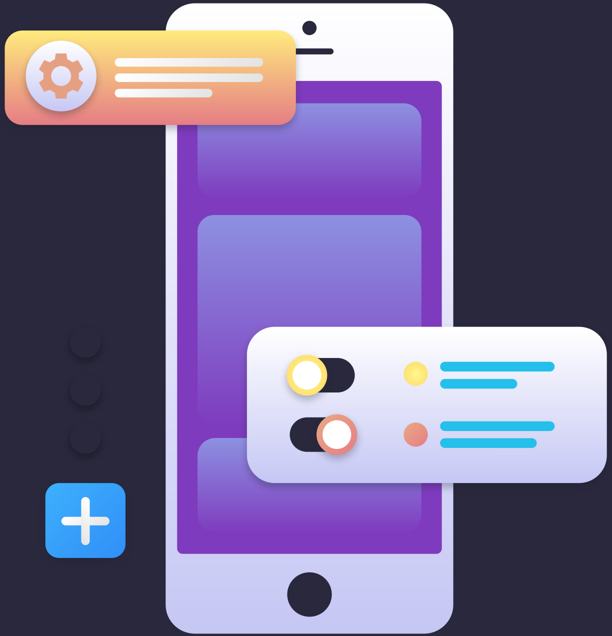
Key #1: 1196311
Key #2: 1250559
Key #3: 1362733
Key #4: 1462991
Key #5: 1509349

a. 983
b. 1217
c. 1361
d. 1439

RSA Code for finding gcd

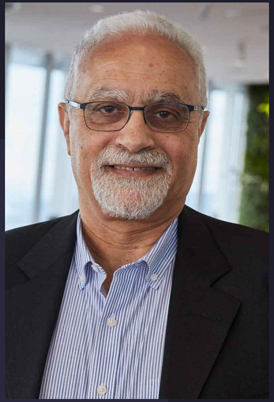
YOUR LOGO HERE

Elgamal Encryption



INDEX.HTML

What is Elgamal Encryption?



Elgamal encryption system is an asymmetric public key encryption system.

It was created by Taher Elgamal in 1985.

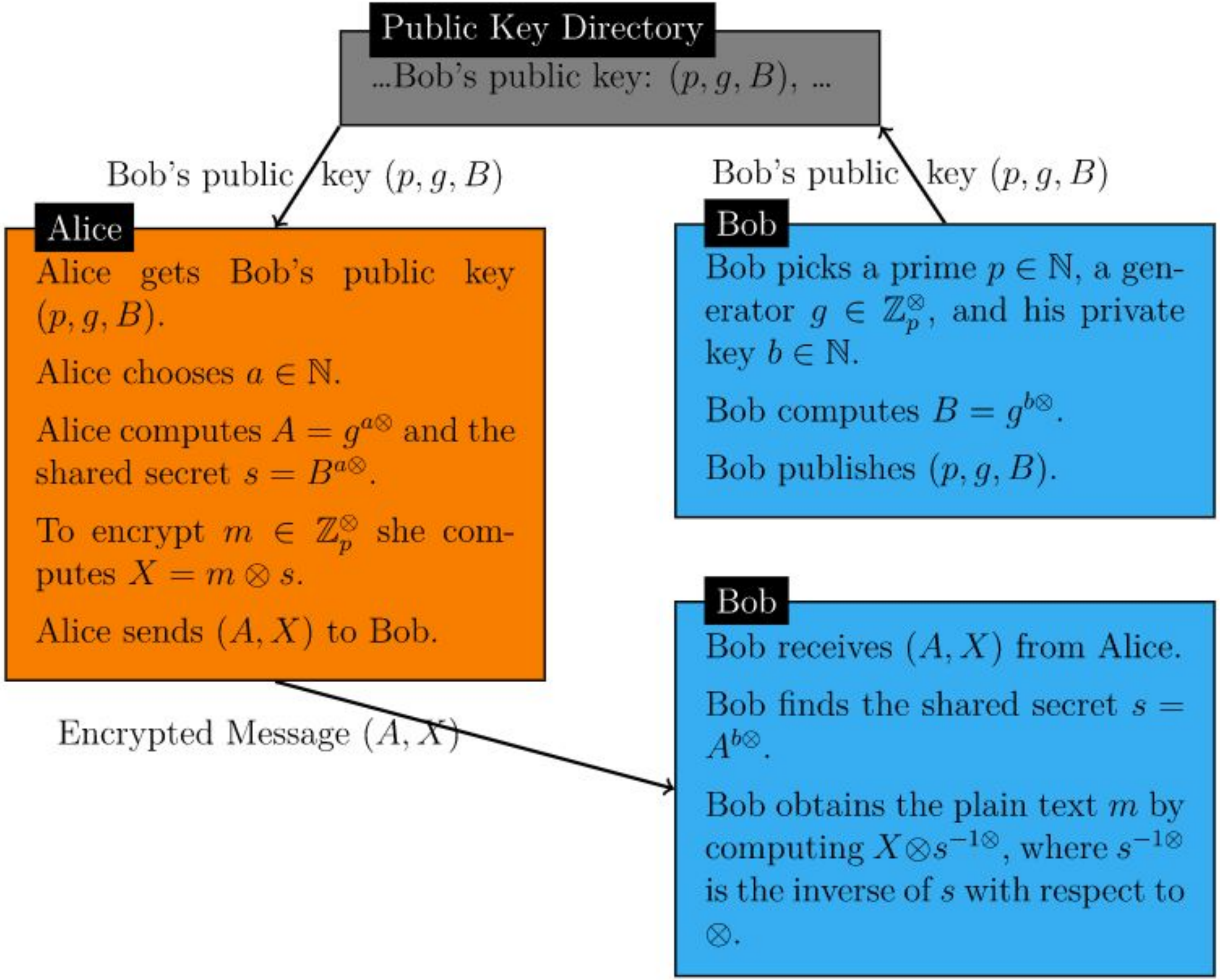
Like RSA, it also uses public and private keys

There are 3 components:

- Key generator
- Encryption algorithm
- Decryption algorithm

YOUR LOGO HERE

The Process



Key Generator

Bob: Key Generator

1) Bob chooses a prime **p** and a generator **g** (any positive integer less than p)
2) Choose a random **b** (any natural number)
    a) **b** is private key
3) Compute B: g^b **mod p**
4) **p, g, B** are the public keys

Bob: Example

1) Bob chooses **p = 29, g=2**
2) **Bob chooses b = 5**
3) **B** = g^b mod p = 2^5 mod 29 = 32 mod 29 = **3**
4) **Public keys: p = 29, g = 2, and B = 3**

Encryption

Alice: Encryption

1) Alice receives Bob's public key: **p, g, B**
2) Choose a random **a** (any natural number)
    a) **a** is private key
3) Compute shared secret **s**:$\mathbf{B^a \bmod p}$
4) Compute $\mathbf{A = g^a}$
5) Encrypt message **m** as **X** $\mathbf{X = (m \cdot s) \bmod p}$
6) Alice sends **(A, X)** to Bob

Alice: Example

1) Alice gets Bob's public keys: **p = 29, g = 2, and B = 3**
2) Chooses secret **a = 4**
3) Compute shared secret $\mathbf{s} = B^a \bmod p = 3^4 \bmod 29 = 81 \bmod 29 = \mathbf{23}$
4) Compute $\mathbf{A} = g^a = 2^4 = \mathbf{16}$
5) Encrypt message **m = 6**: $\mathbf{X} = (6 \cdot 23) \bmod 29 = 138 \bmod 29 = \mathbf{22}$
6) Send **(A, X) = (16,22)** to Bob

Encryption

Bob: Decryption

1) Bob receives **(A, X)** from Alice
2) Compute shared secret **$s = A^b \bmod p$**
3) Find modular inverse of s
4) Decrypt **M: $(X \cdot s^{-1}) \bmod p$**

Bob: Example

1) Receive from Alice **$(A, X) = (16, 22)$**
2) Compute $s = A^b \bmod p = 16^5 \bmod 29 = \mathbf{23}$
3) Find $s^{-1} = 23^{-1} = \mathbf{24}$
$(23 \cdot \mathbf{24}) \bmod 29 = 1$
4) Decrypt **M**
$= (X \cdot s^{-1}) \bmod p = (16 \cdot 24) \bmod 29 = \mathbf{6}$

Now You Try

In breakout rooms:

1) Go to this site: https://mathstats.uncg.edu/sites/pauli/112/HTML/secelgamal.html
2) Work through **Checkpoint 16.3.6 Elgamal with small numbers**
3) Click ***Make Interactive***
4) Comment on Slack your noticings and wonderings
5) What was easy/hard to do?

Pros

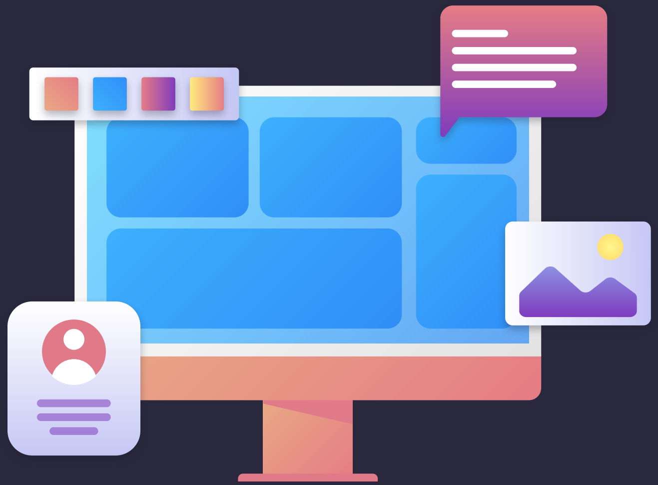
- Encrypting the same text multiple times will result in different ciphertexts every time due to a random private key being chosen every time.
- Elgamal encryption is used in multiple softwares today, such as the free GNU privacy guard and other cryptosystems.
- Faster than RSA in decryption

Cons

- Due to the need for random keys, it is slower than RSA in encryption.
- The ciphertext is much longer than the message because of the encryption process

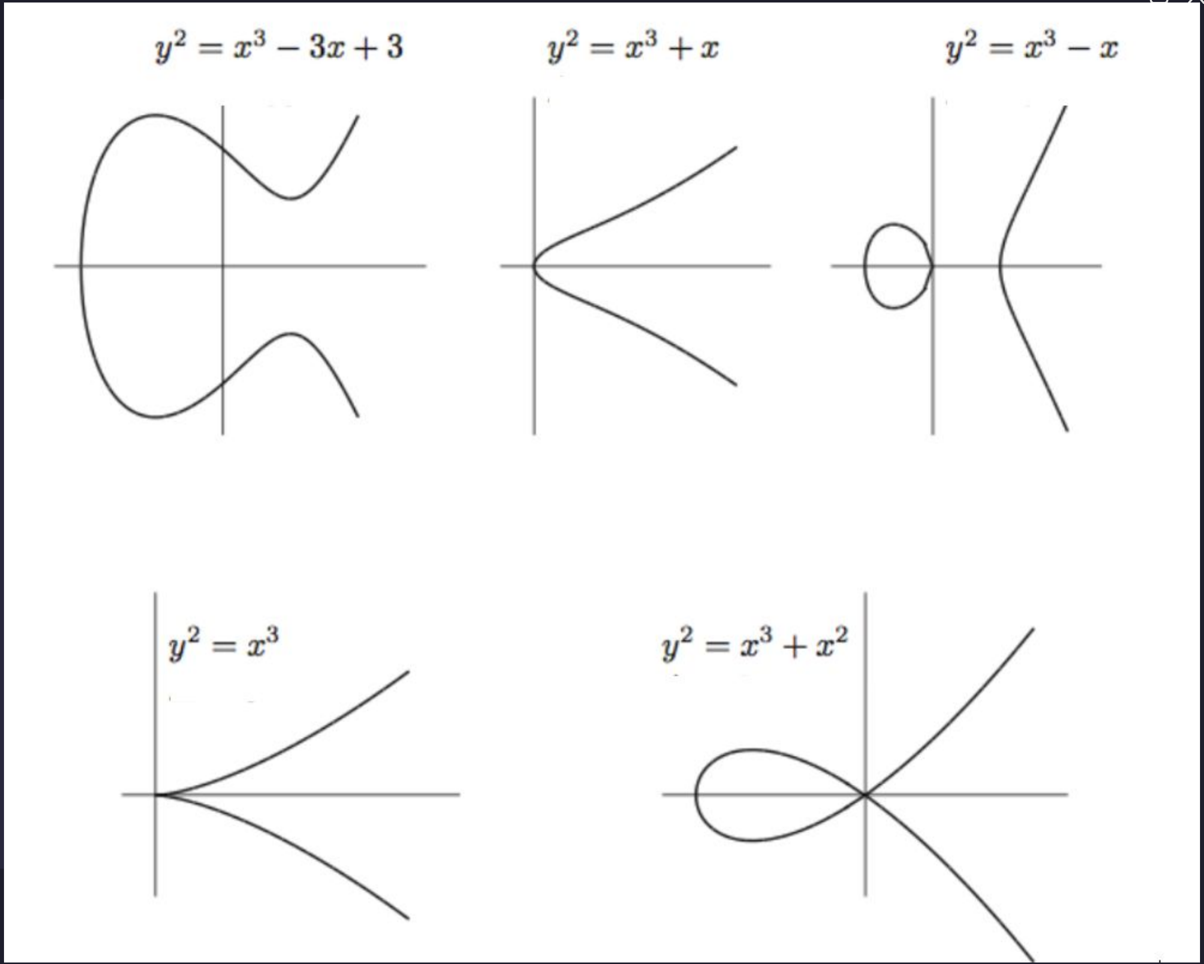
YOUR LOGO HERE

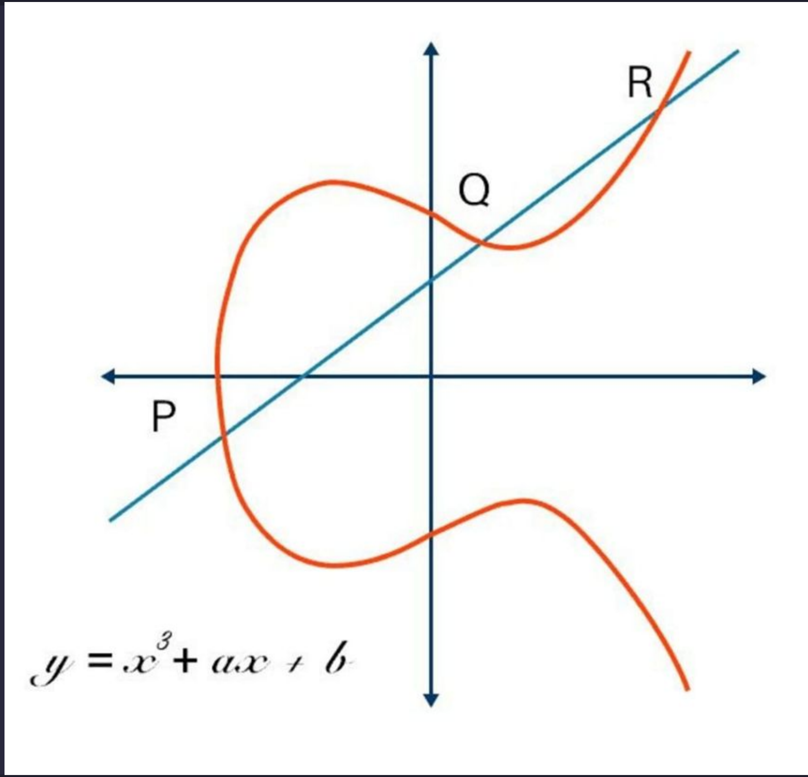
Elliptic Curve Cryptography

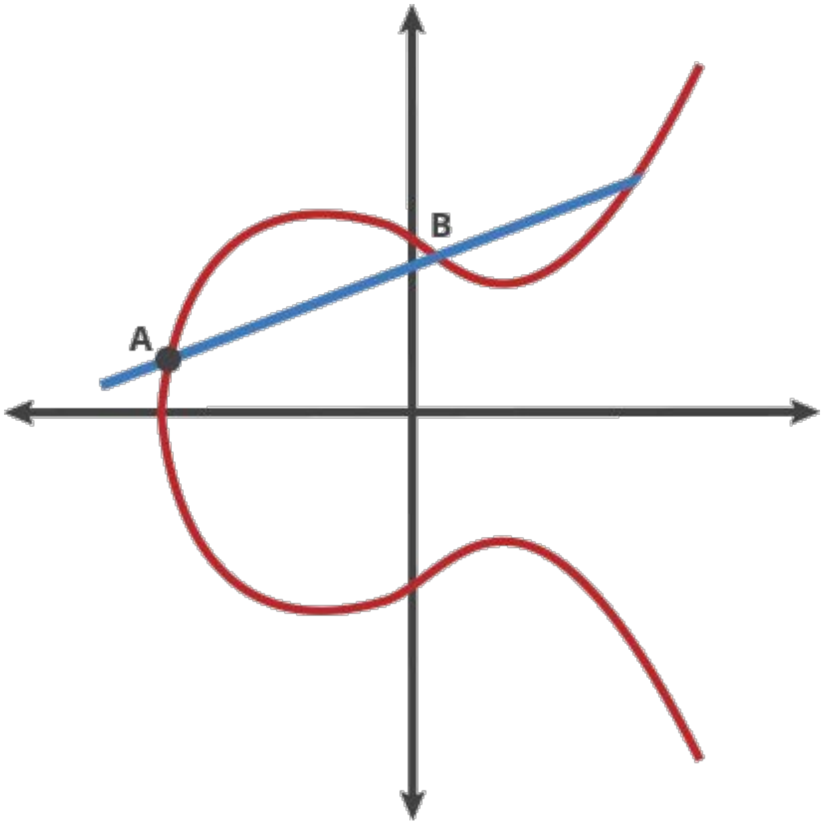


INDEX.HTML

Elliptic Curves




R
Q
P
y = x3 + ax + b


A
B

ECC operations

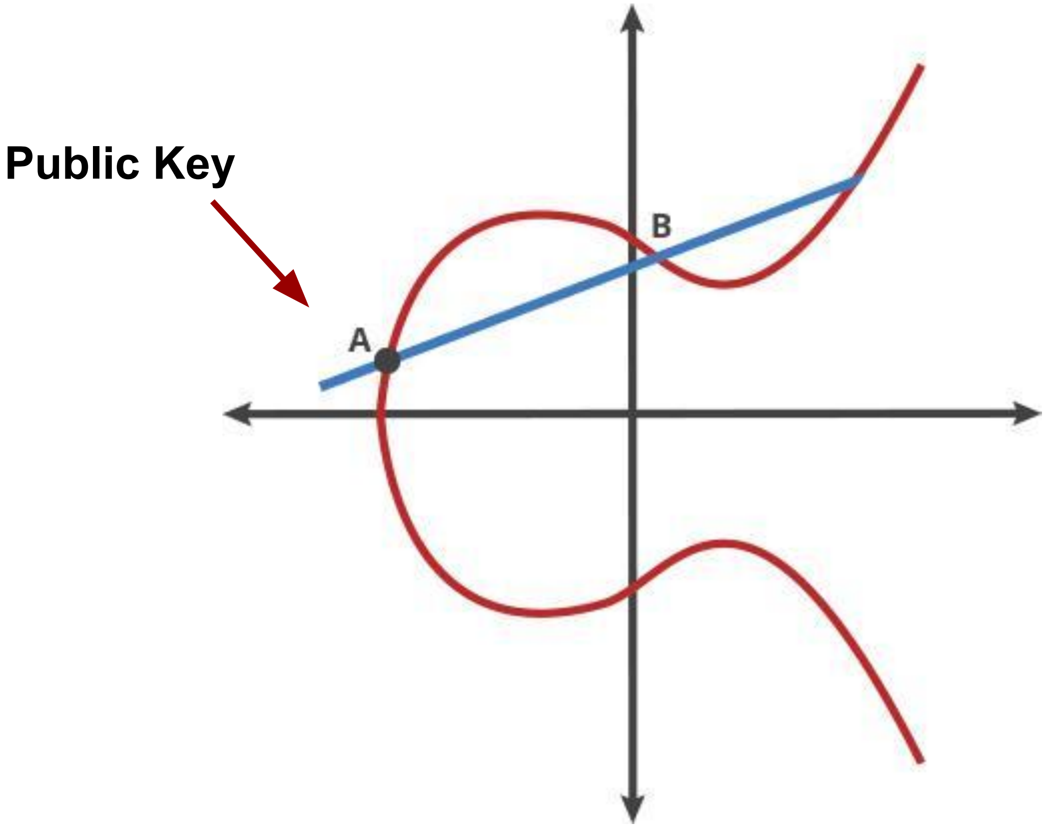
$A + B + C = 0 \Rightarrow A + B = -C \Rightarrow A \text{ dot } B = C$

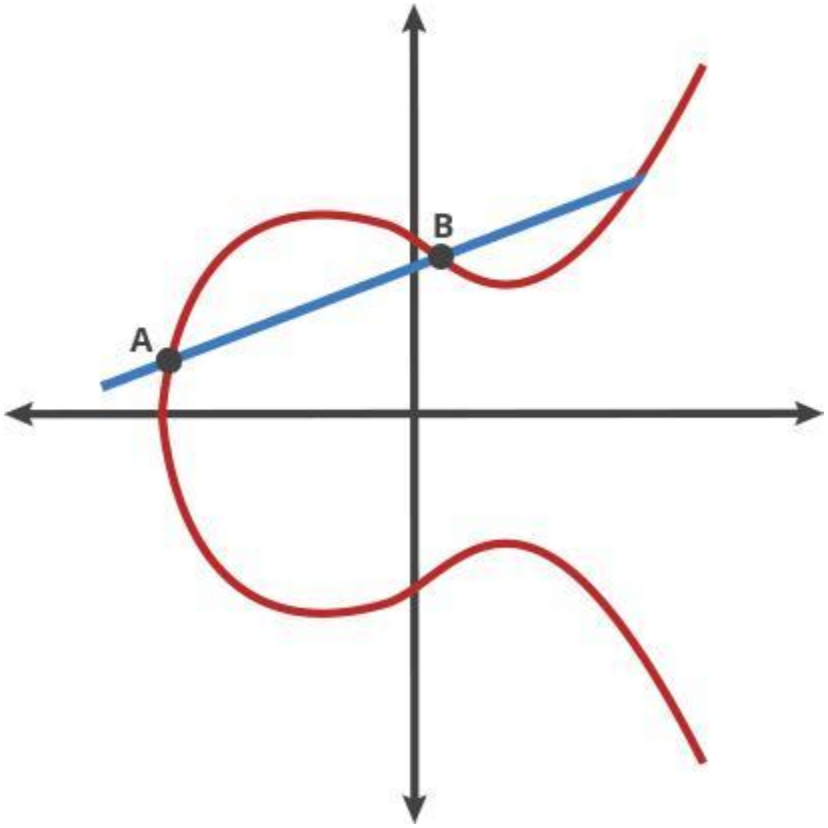
$nA = \underbrace{A + A + A + \ldots + A}_{n \text{ times}}$ $\qquad nA = C$

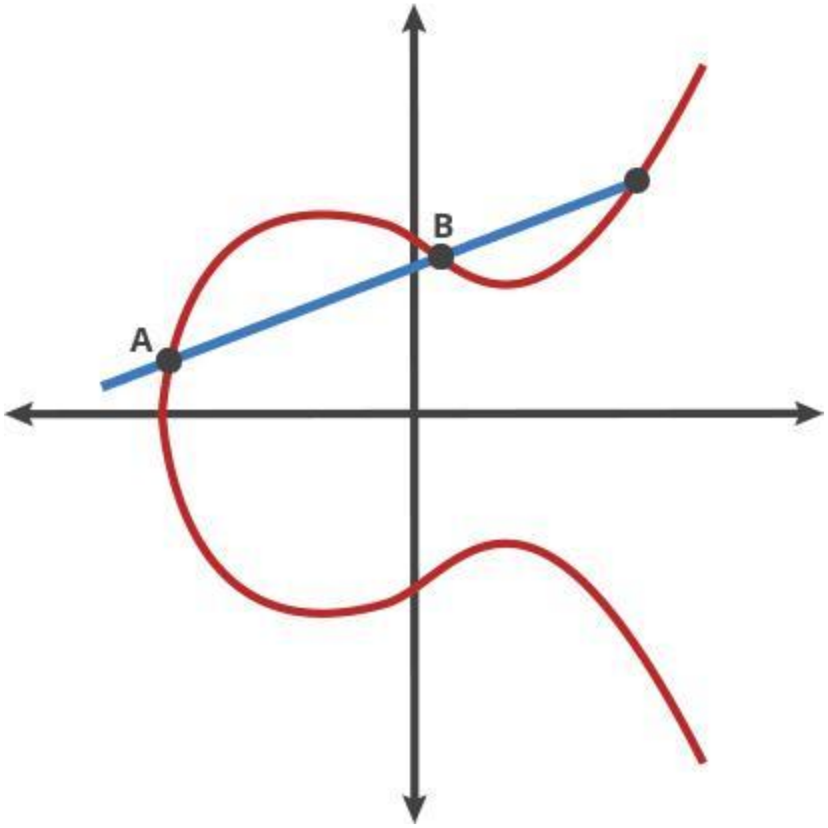
What if we know A and C and need to find n?

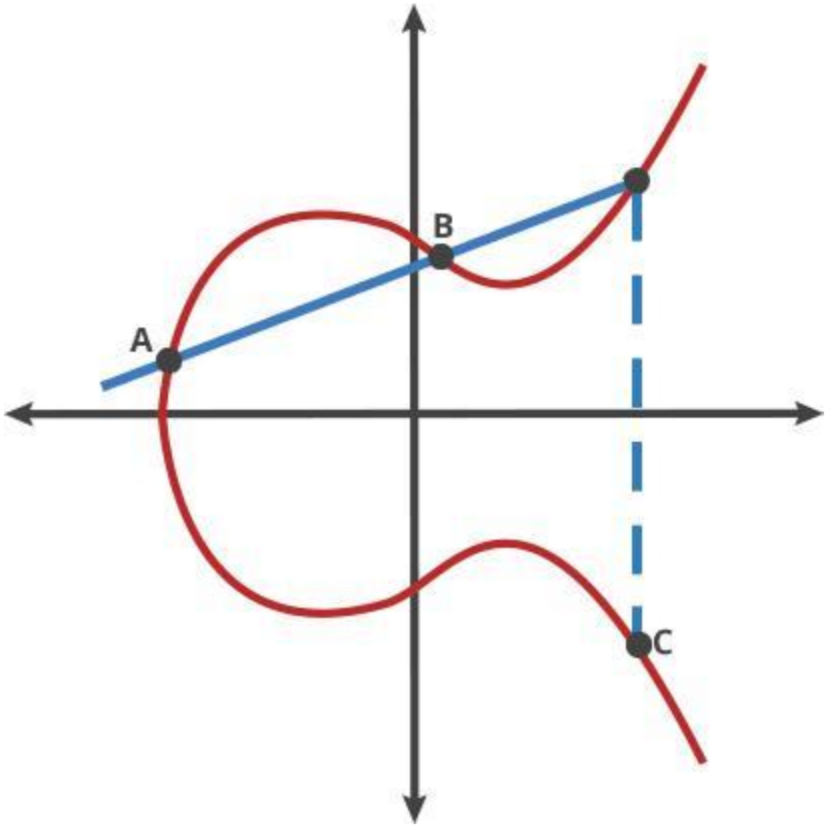
- Logarithmic Problem (conformity with other cryptosystems)

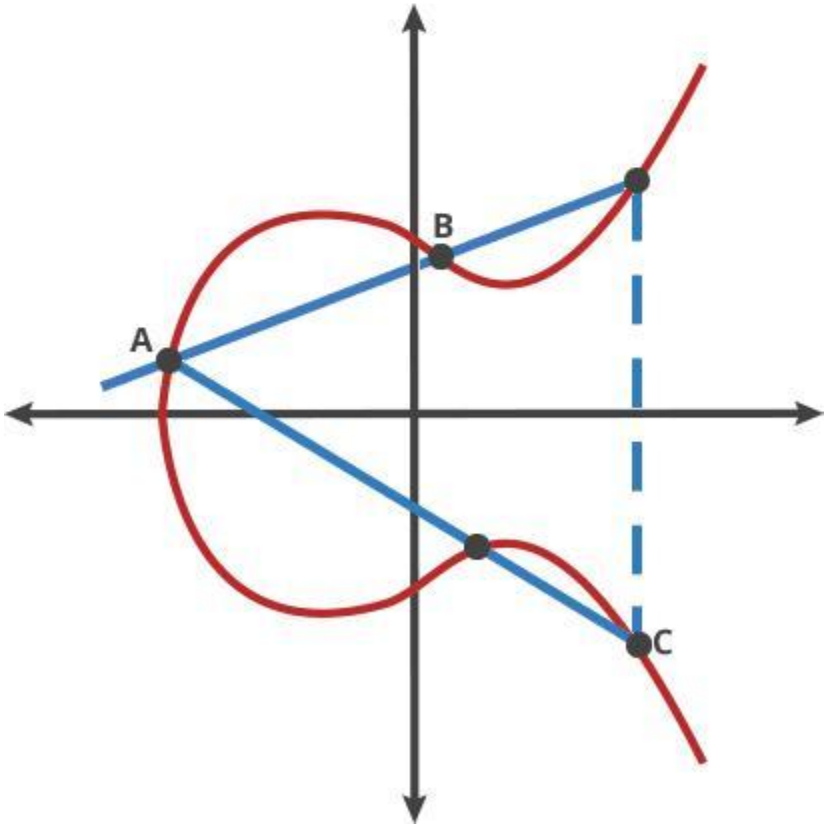
Scalar multiplication remains "easy", while the discrete logarithm becomes a "hard" problem

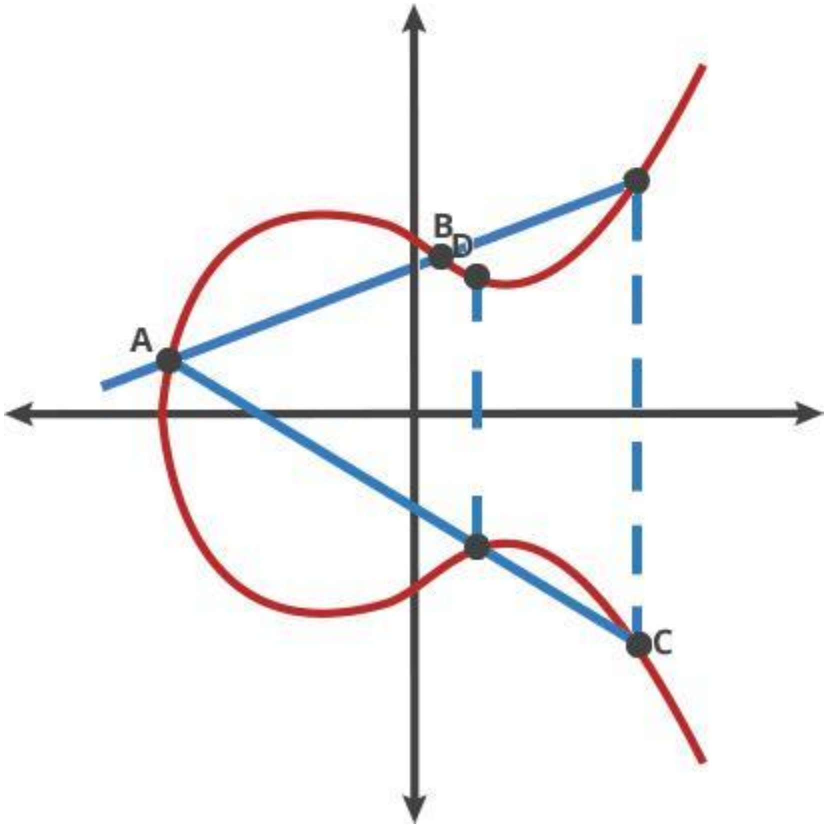

Public Key
B
A

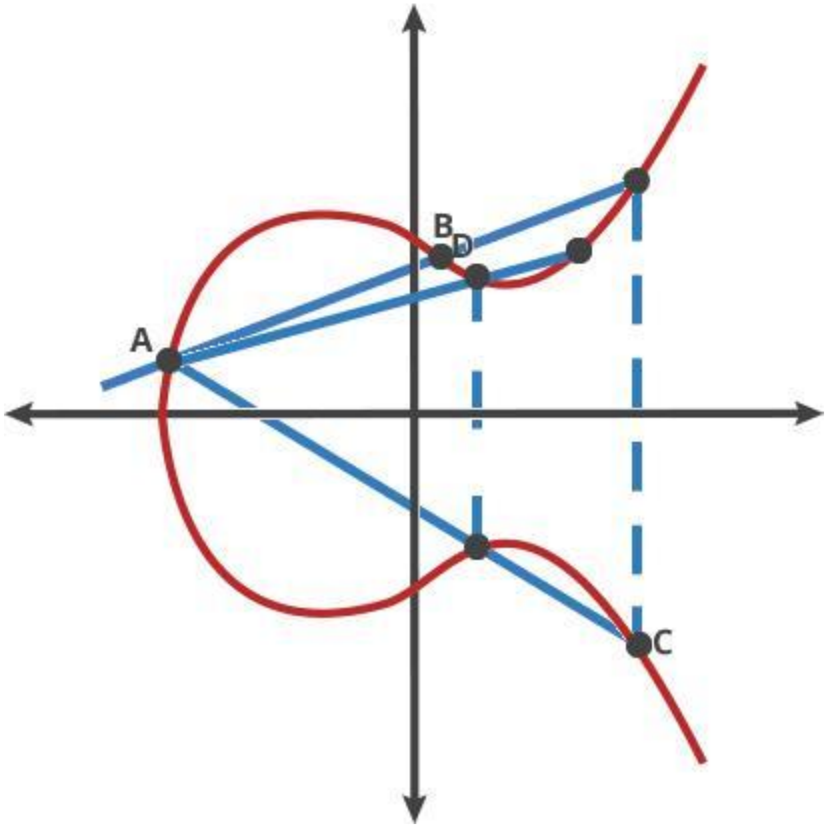

A
B


A
B

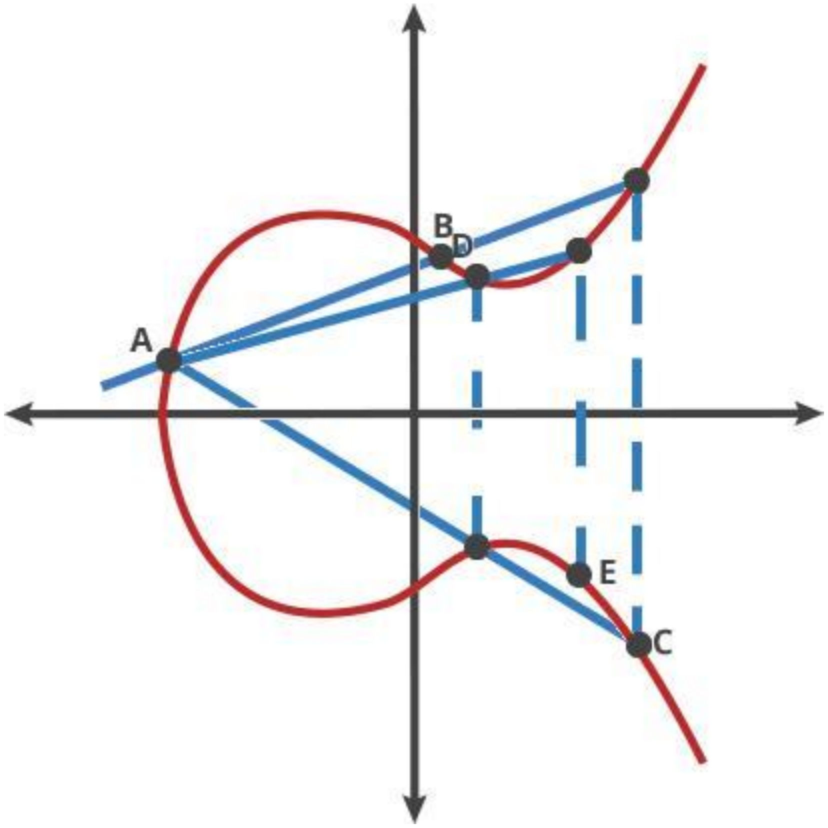

A
B
C


A
B
C

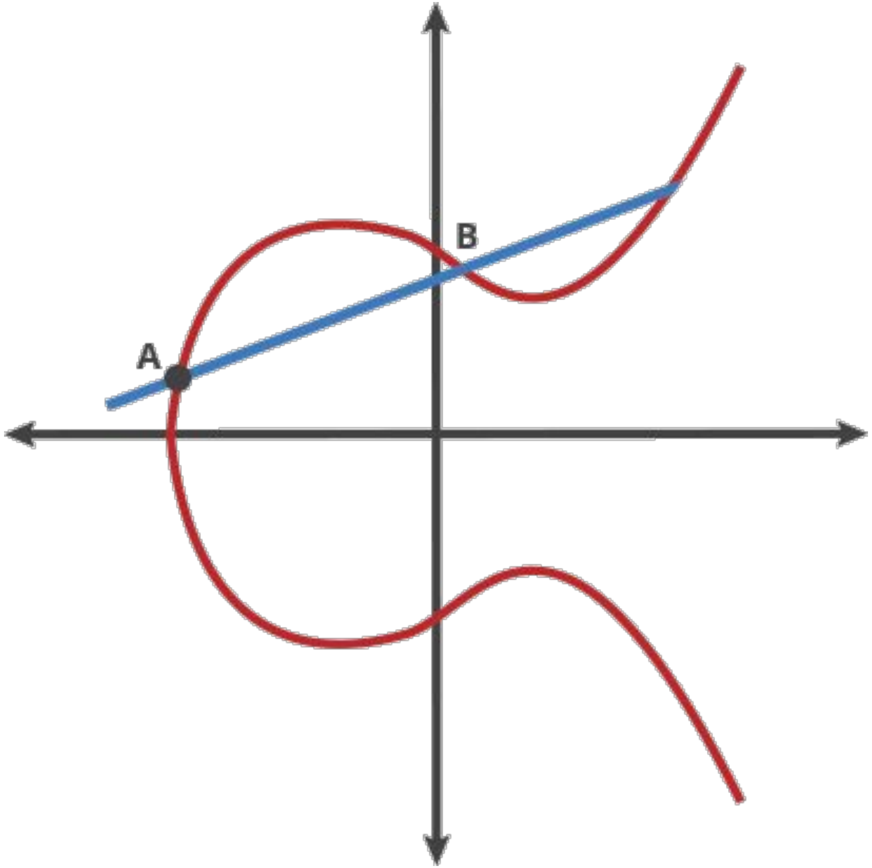

B
D
A
C



A
B
D
C



n - Public Key


B
A

Current use of ECC

- **US government** uses it to protect internal communications
- **Tor project** uses it to help assure anonymity
- mechanism used to prove **ownership of bitcoins**
- provides **signatures in Apple's iMessage** service
- it is used to **encrypt DNS** information with DNSCurve
- it is the preferred method for authentication for secure **Web browsing** over SSL/TLS
- A **growing number of sites** use ECC to provide perfect forward secrecy

https://encrypt.toolpie.com/ - Digital Certificate

Global Security

Compute how much energy is needed to break a cryptographic algorithm and compare that with how much water that energy could boil

Table 1. Intuitive security levels.

security level	volume of water to bring to a boil	bit-lengths: symmetric key	cryptographic hash	RSA modulus
teaspoon security	0.0025 liter	35	70	242
shower security	80 liter	50	100	453
pool security	2 500 000 liter	65	130	745
rain security	$0.082\,\mathrm{km}^3$	80	160	1130
lake security	$89\,\mathrm{km}^3$	90	180	1440
sea security	$3\,750\,000\,\mathrm{km}^3$	105	210	1990
global security	$1\,400\,000\,000\,\mathrm{km}^3$	114	228	2380
solar security	-	140	280	3730

Homework and Async Assignment

https://docs.google.com/document/d/1FD235I7_weh2JbEvBHOe83Jayza1cXfnzgbRtKNd_uQ/copy

SOURCES

https://brilliant.org/wiki/rsa-encryption/

https://www.tutorialspoint.com/cryptography/public_key_encryption.htm

https://www.geeksforgeeks.org/elgamal-encryption-algorithm/

https://mathstats.uncg.edu/sites/pauli/112/HTML/secelgamal.html

https://cs.indstate.edu/~jgrewal/steps.pdf