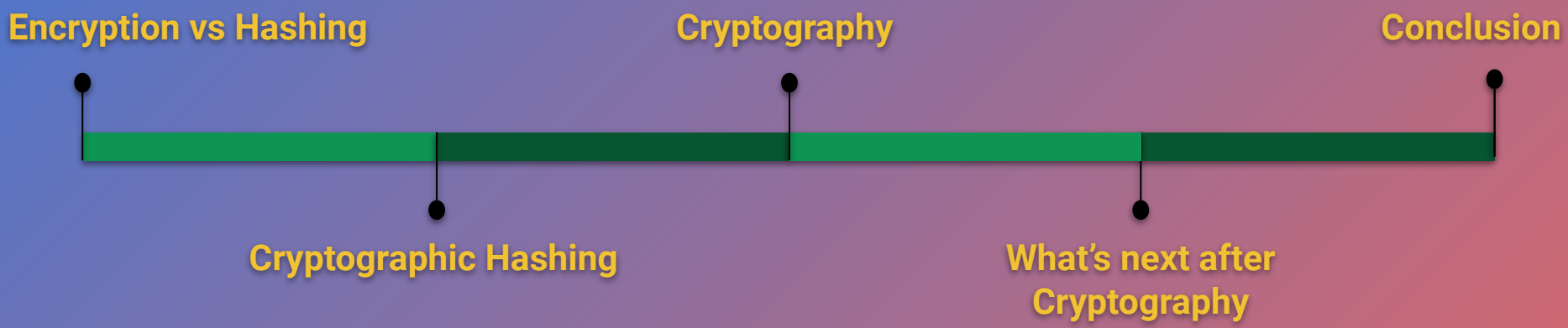
Cryptography

d bnklclq ghwl fsdiug ag dhgowg bklglfdo

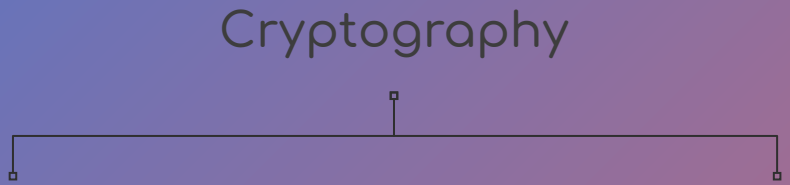
Outline


Encryption vs Hashing
Cryptography
Conclusion
Cryptographic Hashing
What's next after Cryptography

Plaintext - Any text which is human-readable, used in day-to-day work.
You are currently reading a plaintext.
(Plaintext is an “un-encrypted” or “un-hashed” text.)

Key - A key can be used to lock/unlock a physical lock.
Similarly, in Cryptography, a key can be used to
encrypt/decrypt a message.

Difference between Encryption/Decryption and Cryptographic Hashing



Encryption/Decryption

- **Two Way Process:** A message which is encrypted, can be decrypted.
- Key required to decrypt the encrypted text.
- Length of encrypted message depends on length of the plaintext message.
- Used in secure communication, protecting files, etc.

Cryptographic Hashing

- **One Way Process:** A text which is hashed, can never be turned back to the original text.
- No key exist to revert to original text.
- Length of hashed text is fixed irrespective of the length of plaintext.
- Used in hiding passwords in database, authenticate user in websites, signing encrypted message, etc.

Cryptographic Hashing

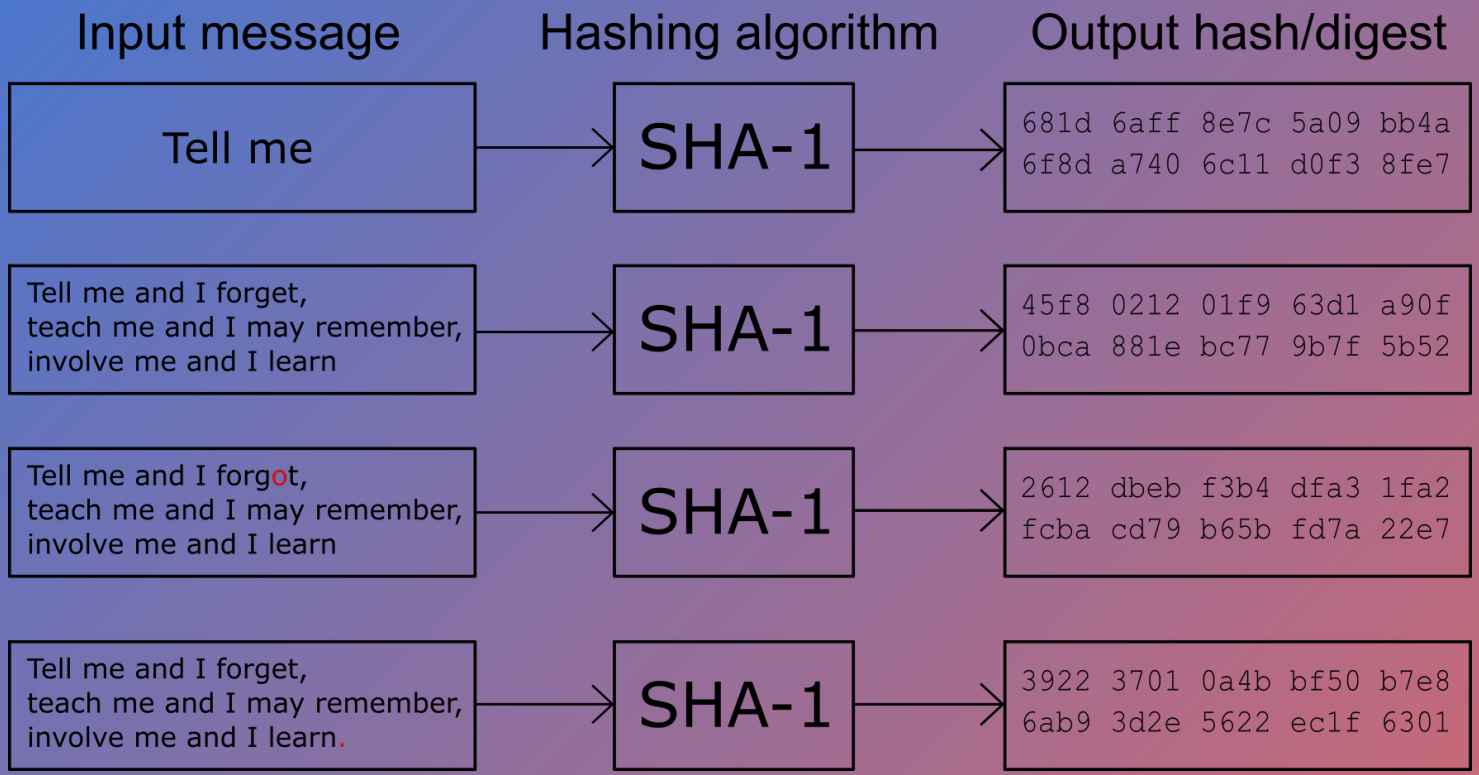
A **Hash function** converts any plaintext to a messy text (called the "*hashed text*") of fixed size. The "*hashed text*" can't be converted back to the original plaintext.

An ideal **Hash function** should have the following properties:

- ➔ The same message should output the same hashed text every time.
- ➔ Hashing any plaintext should be fast to compute.
- ➔ The Hash function should be a "one-way function".
- ➔ Two different messages should never point to the same hashed text (If they do, it is called *Collision*).
- ➔ A slight change in a plaintext message should change the hashed text to such a great extent that the two hashed text can't be correlated to each other (*Avalanche effect*).

  This makes guessing the original plaintext impracticable.

Some well known hashing functions: MD5, SHA family (SHA-0, SHA-1, SHA-2, SHA-3) etc.



- SHA (**S**ecure **H**ashing **A**lgorithm) is a family of hash functions (SHA-1, SHA-2 etc).
- SHA-1 is a function which outputs a 160 bit hash/digest. Here the digests are represented using Hexadecimal numbers (0-9, a-f) to save space. Each hexadecimal number can be made from 4 bits (since $2^4 = 16$), so here, each digest contains 160/4 = 40 characters.

Collision attack

Trying to find two inputs/plaintext producing the same hash value.

E.g: *Hash*(physics12345) = ec1f6301 ; *Hash*(p@$$w0rd)= ec1f6301

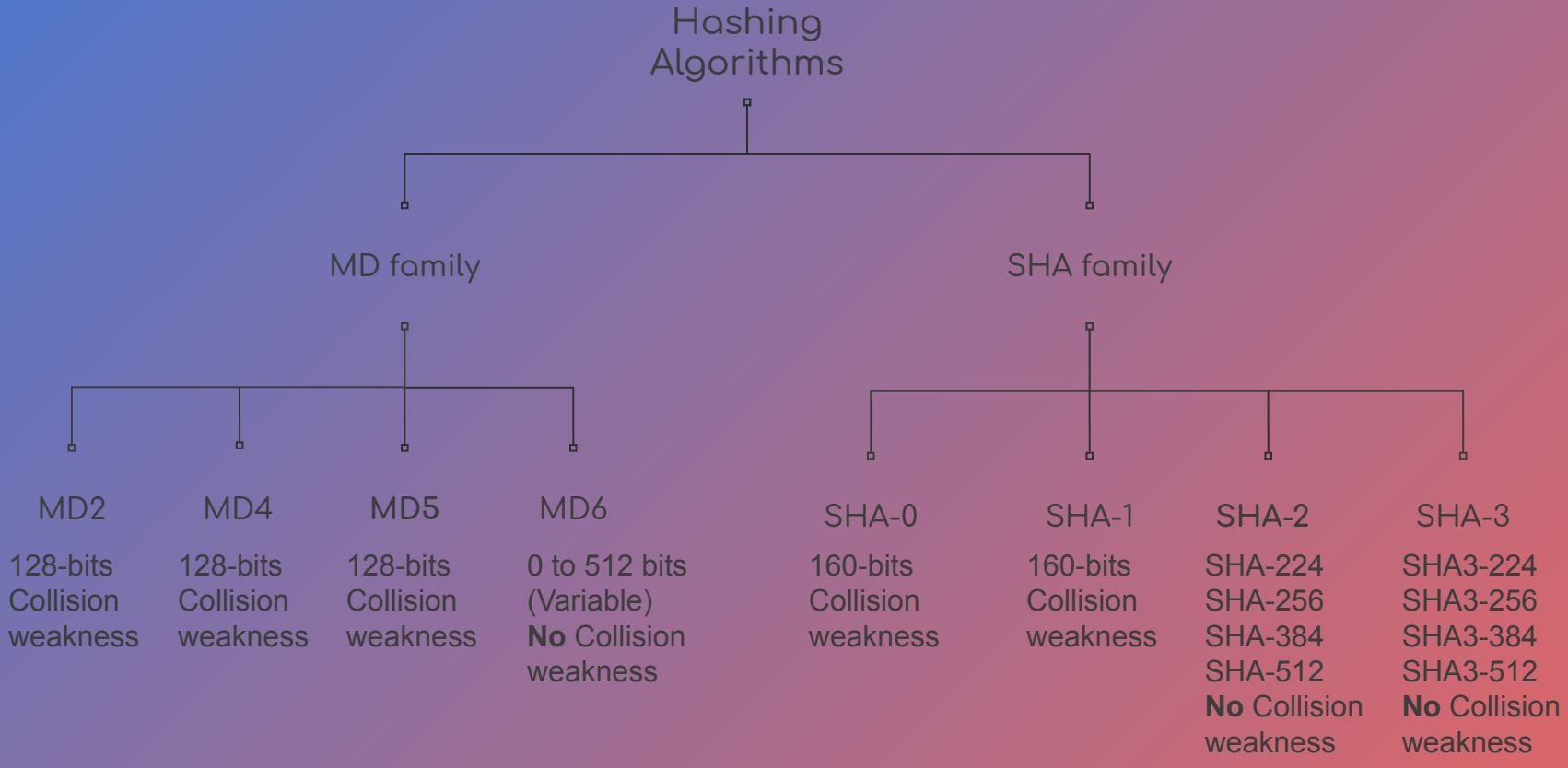
Disadvantages:

➔ Someone else can impersonate you on the internet.
- Log in on your behalf.
- Digitally sign documents on your behalf.
- Make purchases on e-commerce sites on your behalf.

Though the above are very unlikely on most well-known websites, because they use collision-resistant hashing algorithms (SHA-2, SHA-3).

MD5 and SHA-1 are NOT collision-resistant.

Hashing algorithms



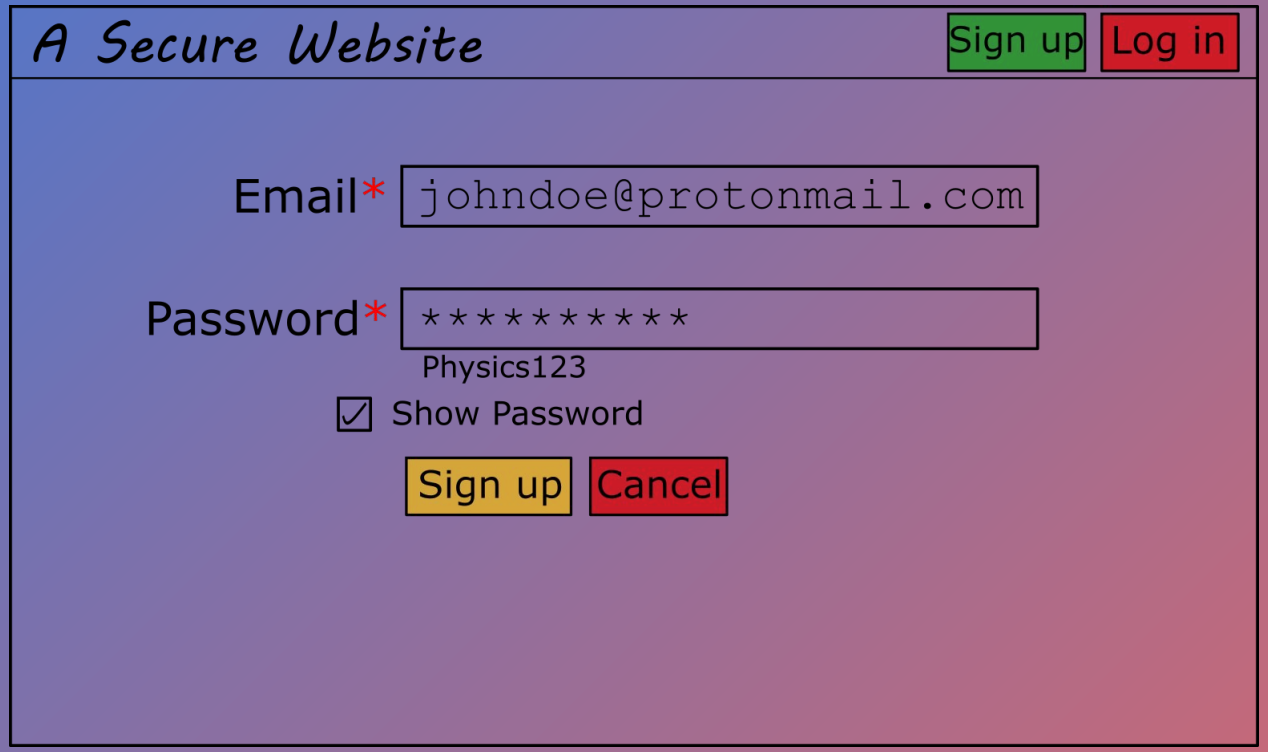
Cryptographic salt

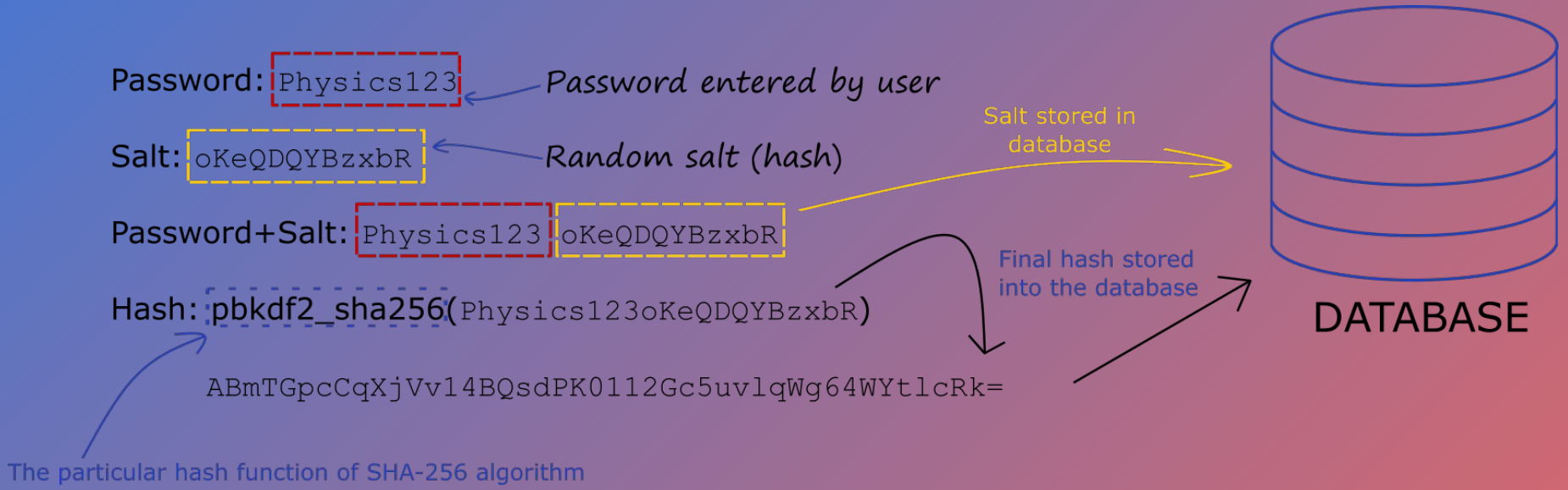
Whenever you create your account on a secure website:

- ➔ You enter your email address and password for the first time.
- ➔ A random hashed value (salt) is appended to your password.
- ➔ Then the whole text is hashed again, and stored into the database.
- ➔ The next time you log into the website, the previously used salt is appended again to your password, and then hashed. If this matches with the hash in the database, you are granted access as yourself in the secure website.

Every user has a unique hashed password in the database, even if multiple users use same password.

Simple hash itself is infeasible to revert. This process makes it more infeasible to crack passwords.


A Secure Website
Sign up
Log in
Email*
johndoe@protonmail.com
Password*
****
Physics123
Show Password
Sign up
Cancel


Password: Physics123
Password entered by user
Salt: oKeQDQYBzxbR
Random salt (hash)
Password+Salt: Physics123 oKeQDQYBzxbR
Salt stored in database
Hash: pbkdf2_sha256(Physics123oKeQDQYBzxbR)
Final hash stored into the database
DATABASE
ABmTGpcCqXjVv14BQsdPK0112Gc5uvlqWg64WYtlcRk=
The particular hash function of SHA-256 algorithm

Cryptography (Encryption & Decryption)

The study of techniques to do secure communication in presence of adversaries

Ancient Cryptography

Some oldest known evidence of cryptography:

- ➔ **1900 BC:** Ancient Egyptian hieroglyphs on the walls of a tomb.
- ➔ **1500 BC:** Clay tablets in Mesopotamia were meant to protect information.
- ➔ **700 - 600 BC:** Scytale cipher used by ancient Greeks (Spartan military).
- ➔ **400 BC - 200 AD:** “Mlecchita vikalpa”, for secret communication between lovers.
- ➔ **58 BC:** First well known cipher (Caesar cipher) was used by Julius Caesar.

Cryptography was used throughout history, including in both the World War I & II.

Some terminologies

Cipher: The un-readable text after encryption is called cipher. Sometimes, the algorithm used to encrypt a plaintext is also called a cipher.

Encryption: The process of converting a plaintext to a cipher-text.

Decryption: The process of converting cipher-text back to plaintext.

Scytale

- ➔ Wrapping the strip around the cylinder reveals the message.
- ➔ Only the strip is transported to the receiver.
- ➔ The recipient must have a rod of same diameter to decrypt the cipher.

  Key: The rod

Iryyatbhmvaehedlurlp

Scytale example

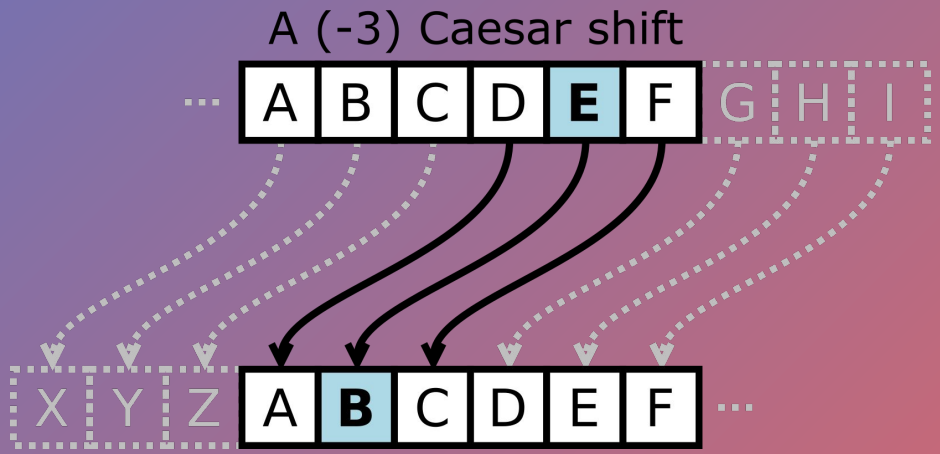
	I	a	m	h	u	
	r	t	v	e	r	
	y	b	a	d	l	
	y	h	e	l	p	

Ciphertext: Iryyatbhmvaehedlurlp

Plaintext: I am hurt very badly help

Caesar shift

- ➔ Shifting the position of each letter in the plaintext by same amount.
- ➔ It is a type of ‘substitution cipher’.
- ➔ Julius Caesar commonly used a +3 shift to encrypt his orders to the military, to carry out surprise attacks.
- ➔ The military men had to do a shift in opposite direction (-3) to reveal the order from Caesar.



- ➔ Also called ROT-3 shift (ROTation by 3).
- ➔ After Caesar, various shifts have been used (0 to 25 or 1 to 26).
- ➔ ROT-13 shift was also popular,
  - ◆ Because there are 26 letters in English alphabet.
  - ◆ To encrypt, shift the position by +13, and to decrypt, shift the position again by +13.
  - ◆ No need to do subtraction in the letter positions. The ‘key’ for encryption & decryption is exactly same.

Ciphertext: Nlpdlc hld delmmpo ehpyej escpp etxpd

Plaintext: Caesar was stabbed twenty three times

I changed position of every letter by +11 places to encrypt the plaintext.

Here, the key is a -11 shift to decrypt the ciphertext.

Breaking the Caesar cipher

There are only two ways to decrypt a cipher.

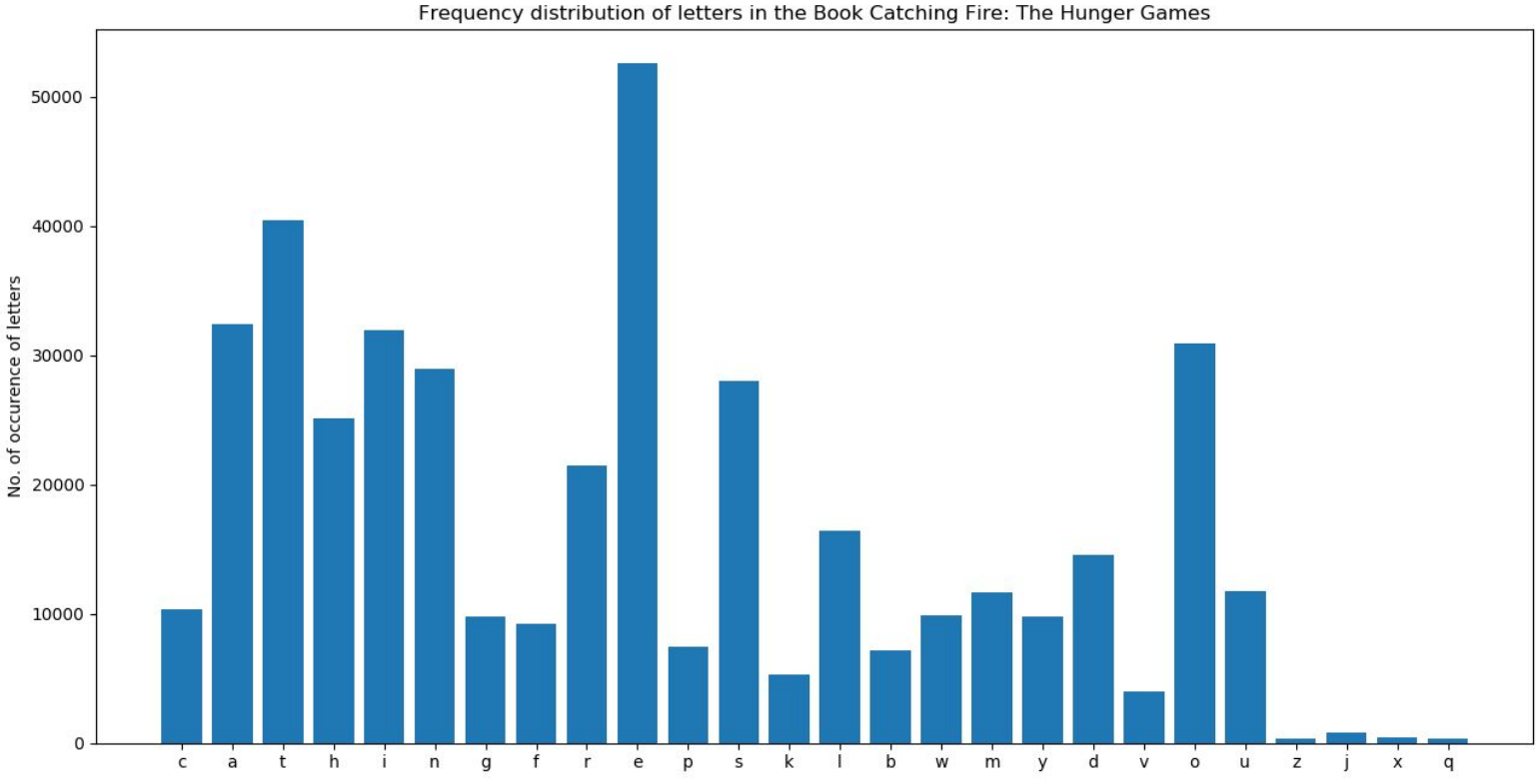
➔ Find the key or guess the key by some analysis.

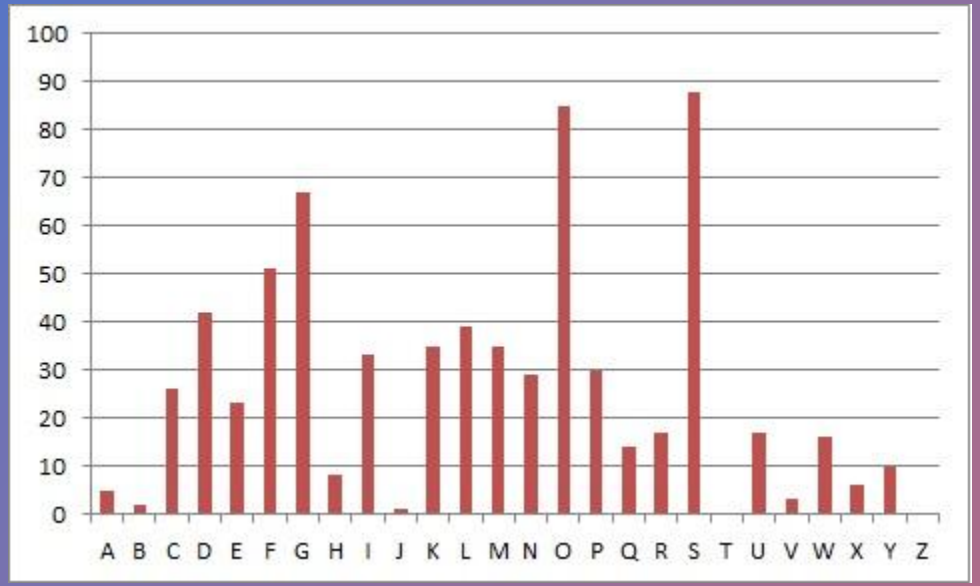
➔ Trying out all possible combinations to turn ciphertext to plaintext.
This is called Brute-Force Attack.

In case of Caesar cipher, the amount of shift can be known by using “Frequency Fingerprint Analysis”.

Frequency Fingerprint Analysis

- ➔ In each alphabet, such as in English alphabet, some letters are used more than the others.
- ➔ There is a frequency distribution of letters.
- ➔ If the Caesar shift ciphertext is somewhat long, it will also leave a distinct frequency fingerprint.
- ➔ We can match the ciphertext fingerprint with that of English alphabet fingerprint, and then can easily know the amount of shift.


Frequency distribution of letters in the Book Catching Fire: The Hunger Games
No. of occurence of letters
0
10000
20000
30000
40000
50000
c a t h i n g f r e p s k l b w m y d v o u z j x q



Frequency Fingerprint of a Caesar cipher

- ➔ We can easily match this fingerprint with the previous one.
- ➔ We can find out that the frequency of S resembles with that of frequency of E.
  So, there is a high chance that this is a (E to S), +14 shift (ROT-14) cipher.

We guessed the key by this method. The more longer the ciphertext, the stronger is the guess.
In an ideal ciphertext, the frequency of all letters should be equal.

Brute Force Attack (Caesar shift)

➔ If we have a ciphertext, apply every possible key to decrypt it.
➔ In Caesar shift cipher, we can apply all 25 possible shifts.

(The 26th shift will return the same ciphertext again!)

Key Space: The set of all possible keys. The more the size of Key space, the more difficult to guess the key and decrypt the message using Brute Force Attack.

➔ In case of Caesar shift, the size of Key Space is 25.

Message Space: The set of all possible messages for a particular plaintext. If I use a plaintext of only 5 lowercase letters, then the size of message space is 26^5.

The size of Message Space also makes it difficult to do Brute Force Attack.

Ciphertext: Nlpdlc hld delmmpo ehpyej escpp etxpd

Shift +1: Mkockb gkc cdkllon dgoxdi drboo dswoc

Shift +2: Ljnbja fjb bcjkknm cfnwch cqann crvnb

Shift +3: Kimaiz eia abijjml bemvbg bpzmm bquma

Shift +4: Jhlzhy dhz zahiilk adluaf aoyll aptlz

... ...

Shift +8: Fdhvdu zdv vwdeehg wzhqwb wkuhh wlphv

Shift +9: Ecguct ycu uvcddgf vygpva vjtgg vkogu

Shift +10: Dbftbs xbt tubccfe uxfouz uisff ujnft

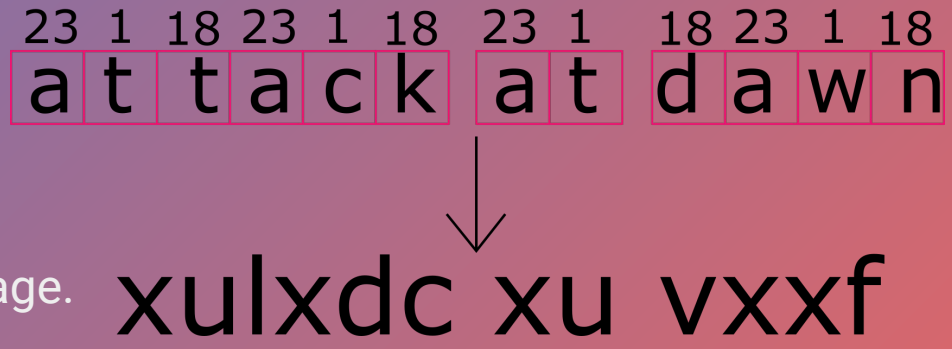
Shift +11: Caesar was stabbed twenty three times ← The plaintext found, the key also known.

Shift +12: Bzdrzq vzr rszaadc svdmsx sgqdd shldr

... ...

Polyalphabetic cipher/Vigenère cipher

- The plaintext we want to encrypt: ‘attack at dawn’.
- First, choose a word/sentence. That will be our key (e.g: war).
- The position of ‘w’, ‘a’, and ‘r’ in the english alphabet is 23, 1 and 18 respectively.
- Write the position no. on top of every letter of plaintext.
- Then, apply the corresponding Caesar shift to each letter to get the ciphertext.
- Do the opposite to decrypt message.



- ➔ Polyalphabetic cipher reduces the frequency fingerprint, but do not flatten it.
- ➔ If the length of the key is not large, it is easy to determine the exact length of the key because of repetition in the distribution.
- ➔ Even if the exact key is not known, the length of the key can determine how many shifts have been applied. In previous example, three Caesar shifts were applied.
- ➔ So, an attacker will do a Brute Force Attack on the ciphertext, applying three Caesar shifts again and again for every three letters.
- ➔ Size of Key space is 26^N, where N is the length of key.
- ➔ Decrypting the ciphertext without key becomes difficult as the length of the key increases.

One-time Pad

- ➔ In this technique, a key is generated randomly.
- ➔ Instead of polyalphabetic key, this key contains random numbers from 1 to 26.
- ➔ According to this key of random numbers, the shifts are applied to each letter of plaintext.
- ➔ The length of the key should be as long as that of plaintext, to avoid any repetition.
- ➔ This method flattens the frequency fingerprint, making it very difficult for computers to decrypt the message using Brute Force attack.
- ➔ Key space is 26^N, where N is the length of the key.
- ➔ Enigma used this technique with some modifications to increase the Key space.

Enigma at a Glance

- Electro-mechanical machine, used to encrypt/decrypt messages.
- 3 rotors out of 5 can be used as 5!/(5! - 3!) = 60 ways
- Each rotor can be set to 26 position = 26^3 = 17576 combinations
- There are 10 wires, which has 20 endings, which can be plugged into any of 26 positions. Considering 2 endings per wire is equivalent, as well the 10 wires themselves, there are, $\frac{26!}{(26-20)! \cdot 2^{10} \cdot 10!} = 150,738,274,937,250$ combinations
- So, in total, there are $\frac{5!}{(5-3)!} \cdot 26^3 \cdot \frac{26!}{(26-20)! \cdot 2^{10} \cdot 10!} = 158,962,555,217,826,360,000$ combinations.
- Nearly 159 million million million combinations!

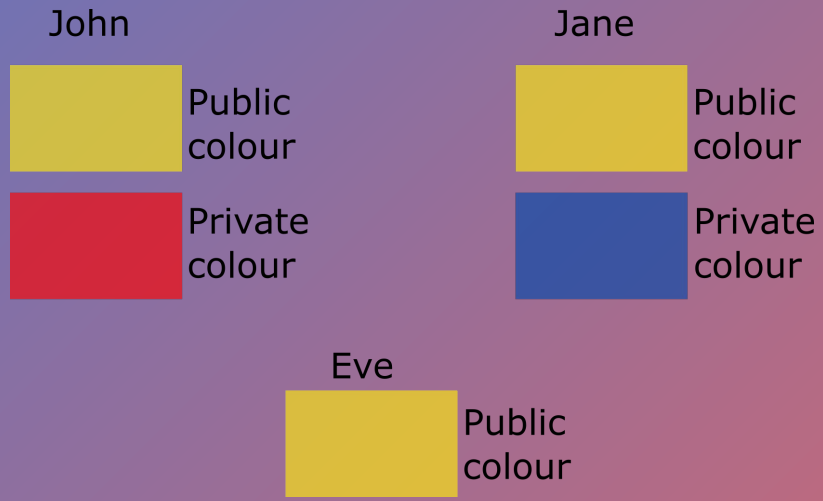
Modern Cryptography

- ➔ Till 1970s, we used symmetric cryptography, where the keys to encrypt and decrypt a message is same, like any physical lock.
- ➔ In early 1977, a new type of cryptography was developed, where the opening and closing keys of the lock need not be same. This is called Asymmetric Cryptography.
- ➔ This technique is also called Public key cryptography, because a public key can be used to encrypt a message and only a private key can decrypt the encrypted message.
- ➔ Anyone with the public key can close the lock, but only those can open the lock, who have the private key.

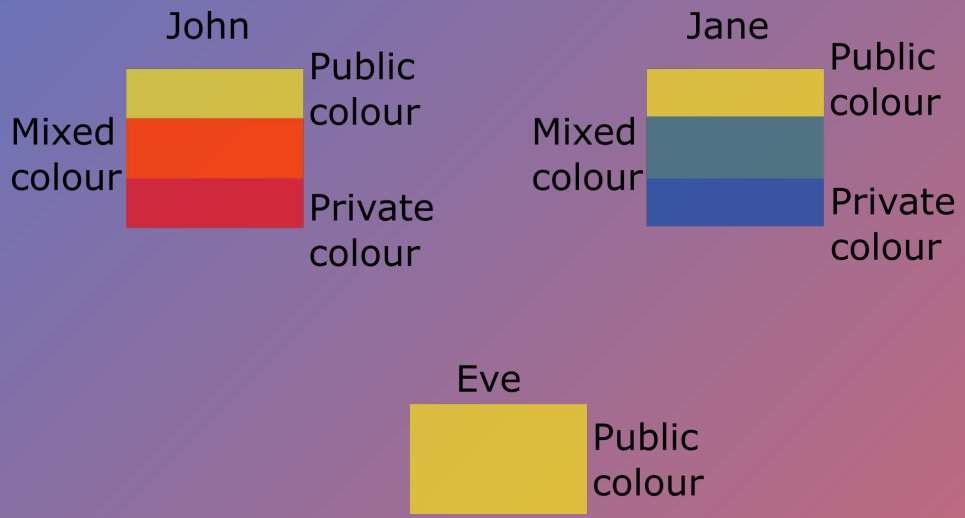
- The algorithm is called RSA behind the surname of its creators.
  - Ron Rivest, Adi Shamir and Leonard Adleman.
- No need to share the private key between those who are communicating. Both have different private keys.

- ➔ Suppose John and Jane have never met with each other.
- ➔ John and Jane will do secured communication using RSA encryption.
- ➔ Their enemy, Eve is intercepting their encrypted messages.

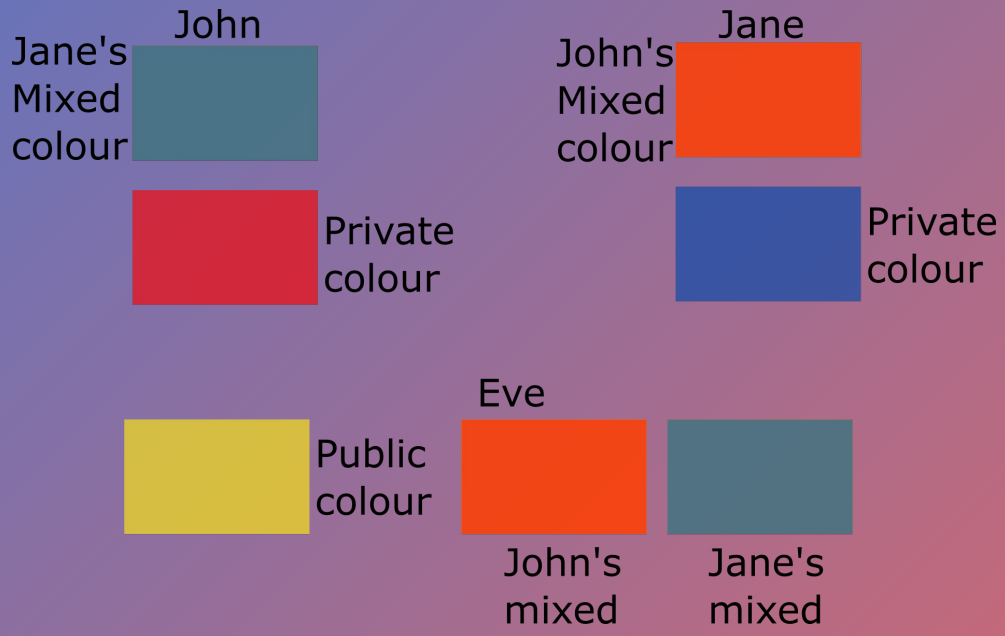
Now understand this with respect to colours.



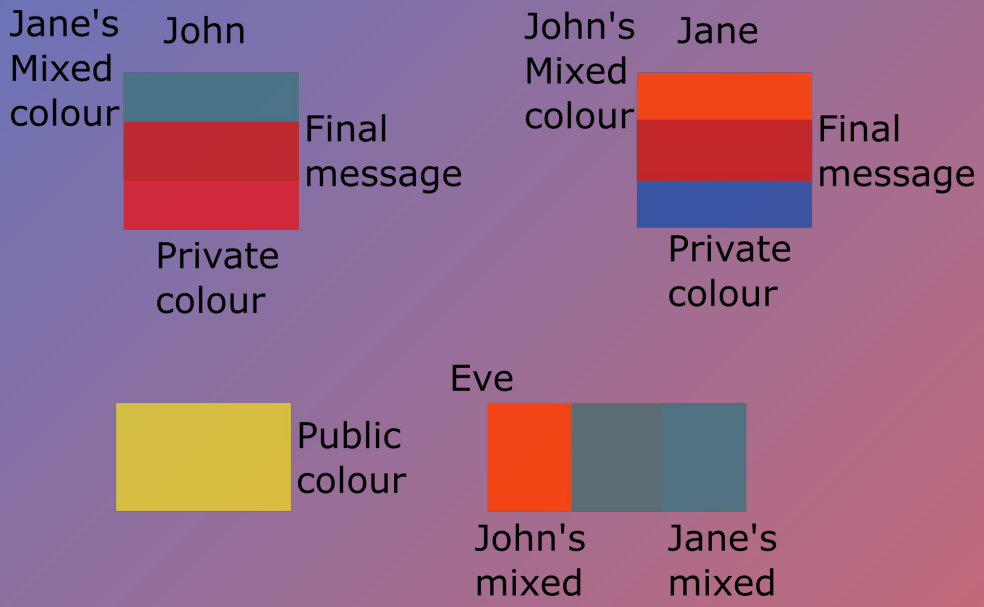
- John and Jane mix their colours. But Eve do not know what their private colours are.
- John and Jane also do not know each other's private colour.



- ➔ After mixing colour, John sends his mixed colour (Orange) to Jane, and Jane sends her mixed colour (Asparagus).
- ➔ Eve can intercept their mixed colours.



- John mix his private colour with Jane's mixed colour. Jane does the same.
- They both get the same final colour. That means they both get the same message, and successfully decrypt the message with different private keys.



- ➔ Eve can never know their private keys by unmixing the colours, because separating mixed colours to their exact previous states is infeasible.
- ➔ It happens so because ordering of mixing doesn't matter. Both John and Jane mixed Yellow + Red + Blue = Final Result.
- ➔ In RSA encryption, the role of "prime numbers", and "factorization" of integers is very important.
  - First, find two large primes p and q, so that $n = pq$.
  - Next, choose a random integer e, relatively prime to n, i.e., (p-1)(q-1)

So, $e = \varphi(n)$

This $\varphi(n)$ is called Euler's phi function.

n	1	2	3	4	5	6	7	8
$\varphi(n)$	1	1	2	2	4	2	6	4

- The pair *(n, e)* is made public (public key), but the factors *p* and *q* are kept secret.
- Suppose John want to send us an encrypted message.
- John will convert his message to numbers (using, a=01, b=02, …, z=26, space=27) and break this message into blocks (*B*) smaller than *n*.
- For each block *B*, John computes an encrypted block *C*, such that

$$C \equiv B^e \bmod n \qquad (C \text{ is ciphertext})$$

- John then send us block *C*, which is encrypted.
- To decrypt the message, we need to find an integer *d* such that

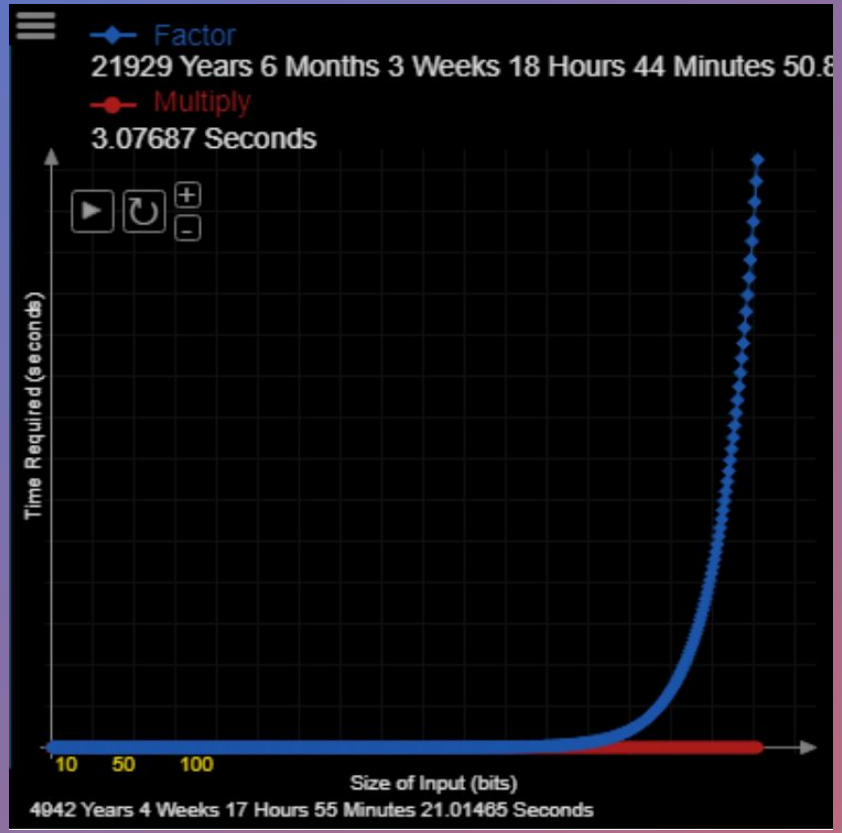
$$ed \equiv 1 \bmod \varphi(n).$$

- The pair *(n, d)* is the private key. Once the pair is found, all records of prime factors *p* and *q* of *n* should be destroyed.

- Now for each encrypted block C we just calculate,

$$B \equiv C^d \bmod n \qquad \text{(Decryption)}$$

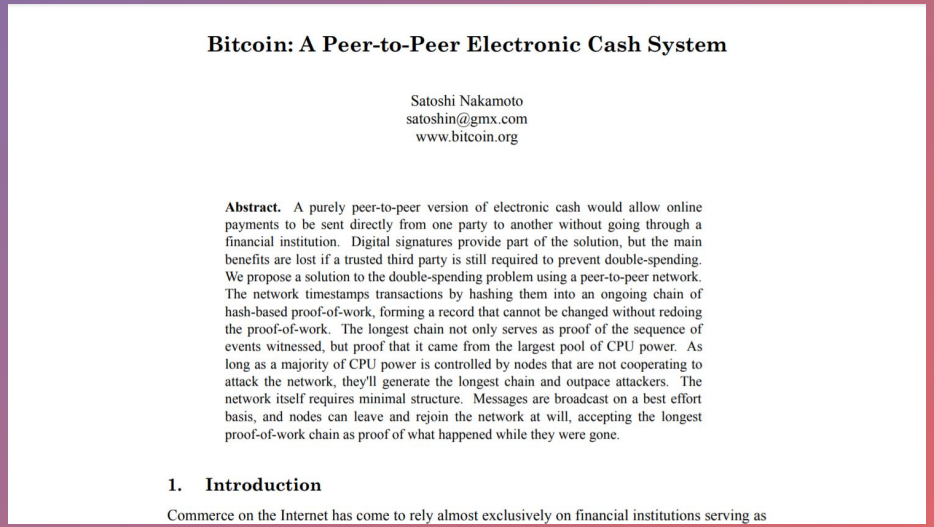
- There is no known way to break RSA encryption.
  - To break RSA, we need to find the prime factors of *n*.
  - n should be large enough to prevent the factorization by computer.
- A computer can multiply large numbers very fast, but, the time required to find the prime factors of a number grows exponentially as the number of bits increases.
- So, it is infeasible to break RSA.
- The current standard for size of the keys is between 2048 to 4096 bits.
- If computers become powerful enough to factorize numbers, then the key sizes are increased to outsmart computers.


Factor
21929 Years 6 Months 3 Weeks 18 Hours 44 Minutes 50.8
Multiply
3.07687 Seconds
Time Required (seconds)
10
50
100
Size of Input (bits)
4942 Years 4 Weeks 17 Hours 55 Minutes 21.01465 Seconds

Crypto currencies

- ➔ A scientist (or group of scientists) wrote a White paper in 2008 about Bitcoin.
- ➔ Decentralized system, without any central authority.
- ➔ Crypto currency is validated by Blockchain, which is a list of transactional records (blocks) secured with cryptography.

- ➔ There are more than 5000 types of crypto currencies as of now.



Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

1. Introduction

Commerce on the Internet has come to rely almost exclusively on financial institutions serving as

What’s next after Cryptography

➔ **Steganography:** Hiding texts in plain sight without anyone noticing.

E.g: Hiding texts in images/videos,

Hiding texts in audio files.

Steganographic Image having a message
“ This is a very Secret Message”

A whole book can be hidden in steganographic image/video/audio.

Steganography can be used along with cryptography: First, encrypt the text, then hide it in image.

Summary

- ➔ How hashing differs from encryption/decryption?
- ➔ Hashing Algorithms and Collision Attacks.
- ➔ Ancient and Modern Cryptography.
- ➔ Frequency fingerprint analysis and Brute force attacks.
- ➔ Asymmetric/Public-private key cryptography (RSA encryption).
- ➔ Digital currencies
- ➔ Steganography

References

- MD5 Collision Demo (https://www.mscs.dal.ca/~selinger/md5collision/)
- Stack Overflow - Matching hashed passwords (https://stackoverflow.com/questions/57850716/how-to-check-match-for-salted-and-hashed-password)
- Cryptography Stack Exchange - Possible Enigma Settings (https://crypto.stackexchange.com/questions/33628/how-many-possible-enigma-machine-settings)
- Wikipedia - History of Cryptography (https://en.wikipedia.org/wiki/History_of_cryptography)
- Wikipedia - Largest Prime No. (https://en.wikipedia.org/wiki/Largest_known_prime_number)
- Wikipedia - SHA1 (https://en.wikipedia.org/wiki/SHA-1)
- YouTube - Steganography (Computerphile) (https://www.youtube.com/watch?v=TWEXCYQKyDc)

* Three free to use historic images from Wikipedia has been used. Rest all illustrations were drawn using Inkscape software.
* Anyone can use my drawn images freely without my permission.

Further reading

- Simon L. Singh, *The Code Book (1999)*, 1st Edn.
- J. Pelzl, C. Paar, *Understanding Cryptography: A Textbook for Students and Practitioners (2009)*, 1st Edn.
- Less Simple Example of RSA encryption (https://www.di-mgt.com.au/rsa_alg.html#lesssimpleexample)
- Wikipedia - Quantum Logic Gate (https://en.wikipedia.org/wiki/Quantum_logic_gate)
- Wikipedia - Blockchain (https://en.wikipedia.org/wiki/Blockchain)
- Bitcoin whitepaper (2008) (https://bitcoin.org/bitcoin.pdf)
- Towards Data Science - Steganography (https://towardsdatascience.com/hiding-data-in-an-image-image-steganography-using-python-e491b68b1372)

Resources

- Code of my website: https://github.com/PhysicistSouravDas/scholarbit
- Code for this presentation: https://github.com/PhysicistSouravDas/Talks/Cryptography
- Online Tools for Encryption/Decryption:
  - https://www.dcode.fr
  - https://cryptii.com/pipes/caesar-cipher
- Khan Academy Cryptography videos: https://www.khanacademy.org/computing/computer-science/cryptography/
- Online Enigma Simulator: https://piotte13.github.io/enigma-cipher/
- Colour Mixing Tool: https://trycolors.com/

Thank You

Appendix

SHA-1 algorithm

One iteration within the SHA-1 compression function:

A, B, C, D and E are 32-bit words of the state;

F is a nonlinear function that varies;

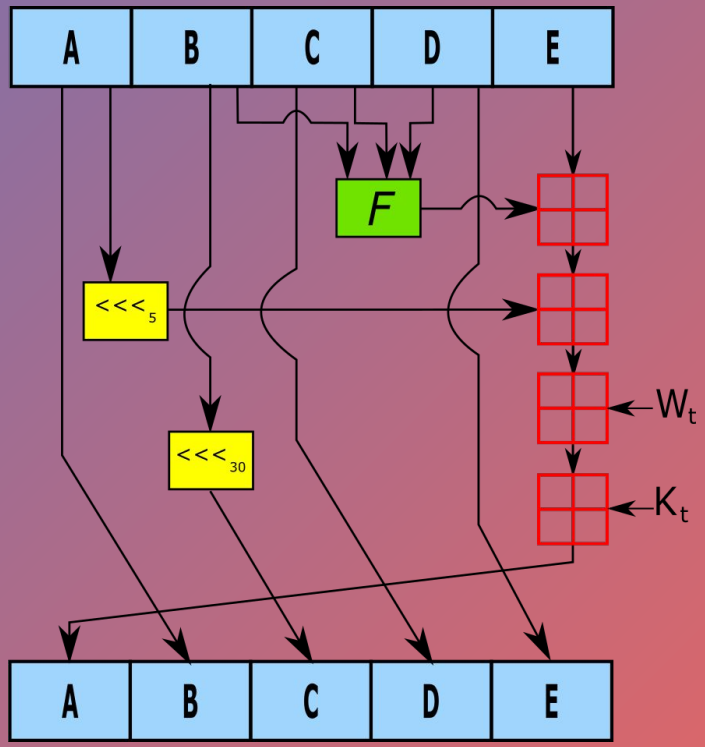
$>>>_n$ denotes a left bit rotation by *n* places;

n varies for each operation;

W_t is the expanded message word of round t;

K_t is the round constant of round t;

Red square boxes denotes addition modulo 2^{32}.



See implementation in Python 3: https://github.com/ajalt/python-sha1/blob/master/sha1.py

Example of a simple one-way function

* But remember that this is not an ideal one-way function. It is just for simple understanding.

Suppose the plaintext is 2021. First, let's convert it to binary number system.

$$(2021)_{10} = (11111100101)_2$$

The length of our message is 11 bits. So now, we will choose a random 11 bit binary number. By tossing a coin, and assigning heads to 1 and tails to 0, we got $(00011110111)_2$.

Then, we will perform XOR operation between our plaintext and the random number.

$$(11111100101)_2 \oplus (00011110111)_2 = (11100010010)_2$$

If someone has just the hashed value $(11100010010)_2$, can they determine our original plaintext? That's how one-way function works. But don't assume what we did here as a perfect algorithm. It was just for understanding purpose, because this is not collision-resistant, nor the same plaintext will return the same result again and again.

Quantum Computer

- The building blocks are Quantum Logic Gates
- The basic unit of data is Qubit, which can be
  - Either a value of 0
  - Either a value of 1
  - Or a superposition state of 0 and 1
- But when Qubit is measured, they return either 0 or 1, not a superposed state.
- Real physical quantum computers do not exist till now.
- Quantum computations are simulated using quantum circuits consisting of Logic Gates.
- A real Quantum Computer will have the ability to factorize large numbers faster, hence, they may break RSA encryption of current standards (upto 4096 bits).

Create your own Quantum Circuits on IBM cloud: https://quantum-computing.ibm.com/

Quantum Logic Gates

- Pauli Gates
  - Pauli X (X): Equivalent of a NOT gate in classical computer
  - Pauli Y (Y)
  - Pauli Z (Z)
- Hadamard Gate (H)
- Phase-shift Gate (S, P)
- Swap Gate
- Toffoli Gate (CCNOT, CCX, TOFF)

Create your own Quantum Circuits on IBM cloud: https://quantum-computing.ibm.com/