Cryptography

Cryptography

Cryptography is the science of securing data

Cryptanalysis is the science of analyzing and breaking secure communication.

Classical cryptanalysis involves an interesting combination of analytical reasoning, application of mathematical tools, pattern finding, patience, determination, and luck. Cryptanalysts are also called attackers. Cryptology embraces both cryptography and cryptanalysis.

Cryptography

A message is **plaintext** (sometimes called cleartext). The process of disguising a message in such a way as to hide its substance is encryption. An encrypted message is ciphertext. The process of turning ciphertext back into plaintext is decryption.

A **cipher** (or cypher) is an algorithm for performing encryption or decryption—a series of well-defined steps that can be followed as a procedure.

Cryptography

A cryptosystem is an implementation of cryptographic techniques and their accompanying infrastructure to provide information security services. A cryptosystem is also referred to as a cipher system. The various

components of a basic cryptosystem are as follows −

- Plaintext
- Encryption Algorithm
- Ciphertext
- Decryption Algorithm
- Encryption Key
- Decryption Key

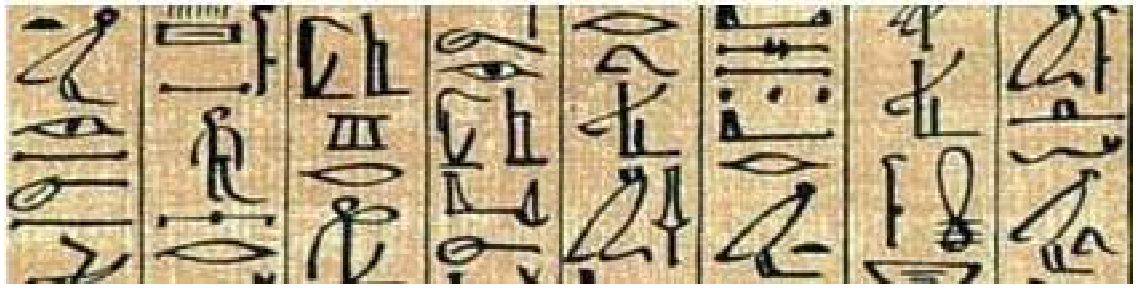
History of Cryptography

As civilizations evolved, human beings got organized in tribes, groups, and kingdoms.
This led to the emergence of ideas such as power, battles, supremacy, and politics.
These ideas further fuelled the natural need of people to communicate secretly with
selective recipient which
in turn ensured the continuous evolution of cryptography as well.

The roots of cryptography are found in Roman and Egyptian civilizations.

History of Cryptography

Hieroglyph
The first known evidence of cryptography can be traced to the use of 'hieroglyph'. Some 4000 years ago, the Egyptians used to communicate by messages written in hieroglyph

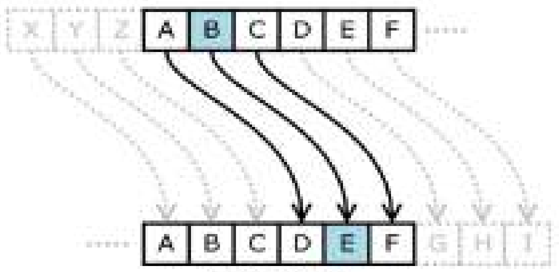


Caesar Shift Cipher, relies on shifting the letters of a message by an agreed number (three was a common choice), the recipient of this message would then shift the letters back by the same number and obtain the original message.

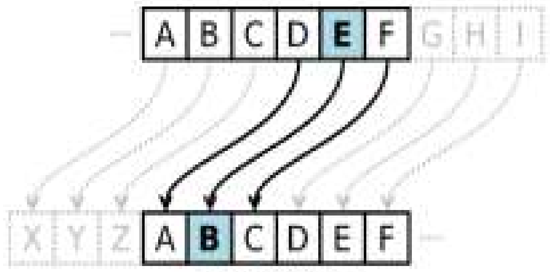
The Caesar cipher is named after Julius Caesar , who used it with a shift of three to protect messages of military significance

History of Cryptography

Caesar Shift Cipher



Encryption



Decryption

PLAINTEXT	:	internet society ghana chapter
CYPHERTEXT	:	lqwhuqhw vrflhwb jkdqd fkdswhu

Cryptography

Goal and Services

Goal: The primary goal of cryptography is to secure important data on the hard disk or as it passes through a medium that may not be secure itself. Usually, that medium is a computer network.

Services: Cryptography can provide the following services:

- Confidentiality (secrecy)
- Integrity (anti-tampering)
- Authentication
- Non-repudiation.

Cryptography

Confidentiality (secrecy)

• Ensuring that no one can read the message except the intended receiver

• Data is kept secret from those without the proper credentials, even if that data travels through an insecure medium

Integrity (anti-tampering)

• Assuring the receiver that the received message has not been altered in any way from the original.

Cryptography

Authentication

- Cryptography can help establish identity for authentication purposes
- The process of proving one's identity. (The primary forms of host-to-host
- authentication on the Internet today are name-based or address-based,
- both of which are notoriously weak.)

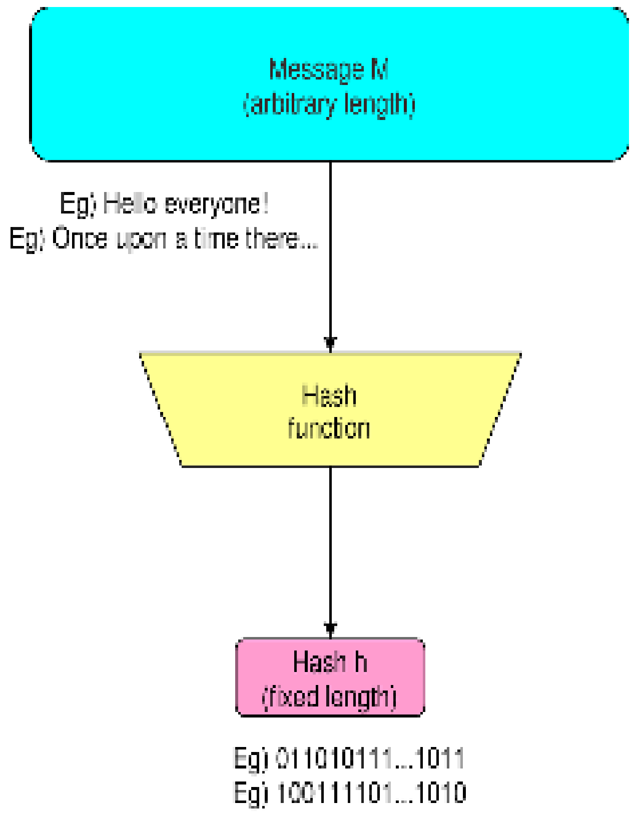
Non-repudiation

- A mechanism to prove that the sender really sent this message

Hash function

A cryptographic hash function is a mathematical algorithm that transforms any input (or 'message') into a fixed-size string of characters, which appears random. This output is known as the hash value, hash code, or simply hash. The important features of a cryptographic hash function include:

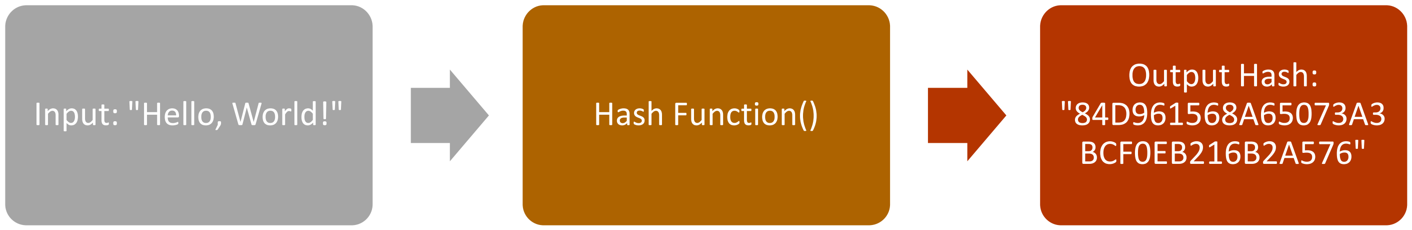
1. Deterministic: The same input will always produce the same output.
2. Quick to Compute: Computing the hash of any input is fast and efficient.
3. Infeasible to Reverse: It's practically impossible to reverse the hash back to the original input.
4. Sensitive to Changes: Changing even a tiny part of the input will significantly change the hash.
5. Collision-Resistant: It's highly unlikely that two different inputs will produce the same output hash.



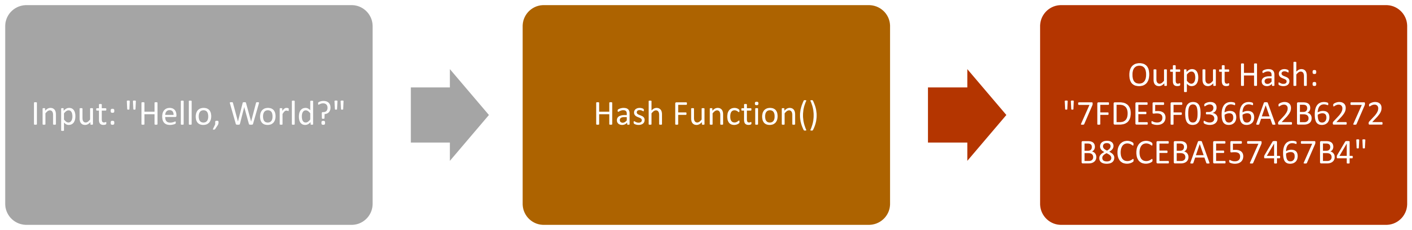
Hash function

Example of a Crypto Hash

Imagine you have a simple phrase, "Hello, World!" and you want to generate a hash for this phrase. Let's use a fictional hash function for this example:



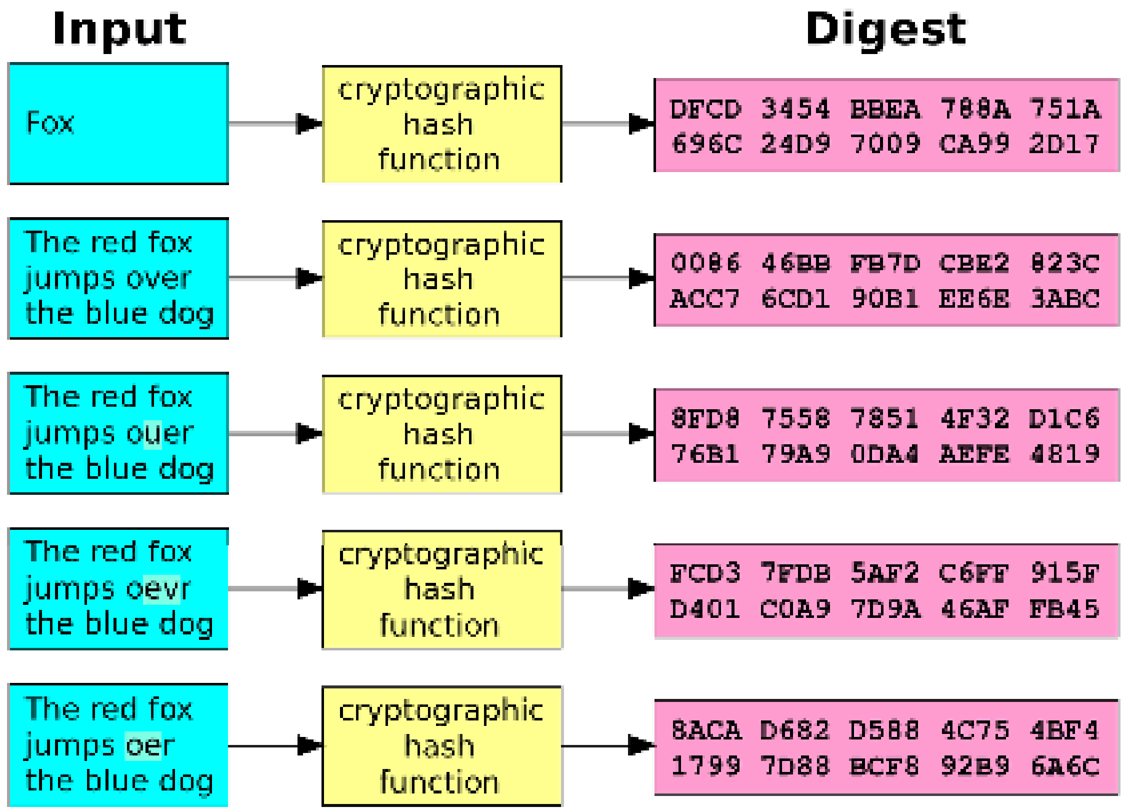
Now, let's see what happens if we change just a single character in the original phrase:



Notice how the output hash is entirely different from the first one, even though we only changed one character in the input. This demonstrates the sensitivity of cryptographic hash functions to input changes.

Desireable properties for a hash function

- Small changes in the input completely change the output
- The change in output should be completely unpredictable
- No better way to find a certain output than simply trying inputs randomly


Input
Digest
Fox
cryptographic hash function
DFCD 3454 BBEA 788A 751A 696C 24D9 7009 CA99 2D17
The red fox jumps over the blue dog
0086 46BB FB7D CBE2 823C ACC7 6CD1 90B1 EE6E 3ABC
The red fox jumps ouer the blue dog
8FD8 7558 7851 4F32 D1C6 76B1 79A9 0DA4 AEFE 4819
The red fox jumps oevr the blue dog
FCD3 7FDB 5AF2 C6FF 915F D401 C0A9 7D9A 46AF FB45
The red fox jumps oer the blue dog
8ACA D682 D588 4C75 4BF4 1799 7D88 BCF8 92B9 6A6C

User:Jorge Stolfi based on Image:Hash_function.svg by Helix84, Public domain, via Wikimedia Commons

Hash function

Practical Uses of Crypto Hashes

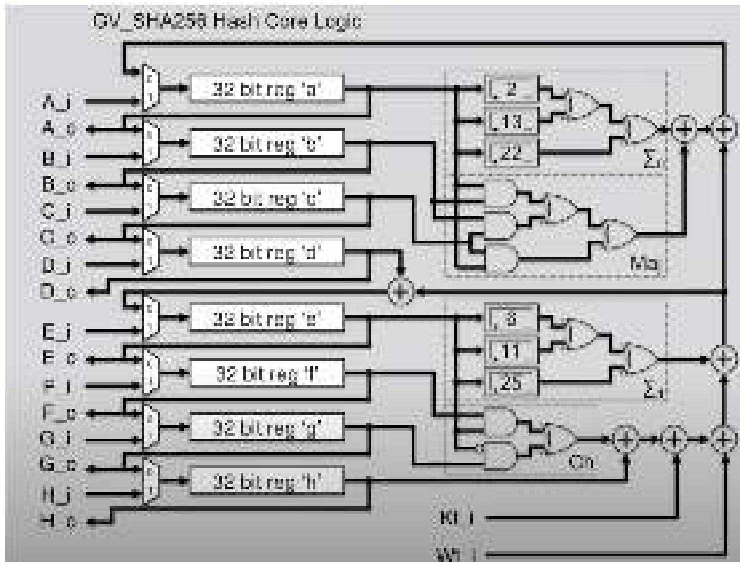
Cryptographic hashes are widely used in various applications, including:

1. Security: Verifying the integrity of data, passwords storage.
2. Blockchain and Cryptocurrencies: Maintaining the integrity and order of transactions.
3. Digital Signatures: Ensuring the authenticity of a digital document.

In summary, cryptographic hash functions play a crucial role in digital security, ensuring data integrity, and underpinning the technology of cryptocurrencies like Bitcoin. They are designed to be fast, deterministic, and nearly impossible to reverse, making them an essential tool in the field of computer science and digital communications.

Popular Hash functions

- SHA256 (part of SHA2 family)
- SHA2, SHA3 family (uses the Keccak algorithm)
- BLAKE2, BLAKE3
- RIPEMD-160 (part of RIPEMD family)
- MD5 (part of MD family)
- SHA1 family
- RIPEMD (original), RIPEMD-128

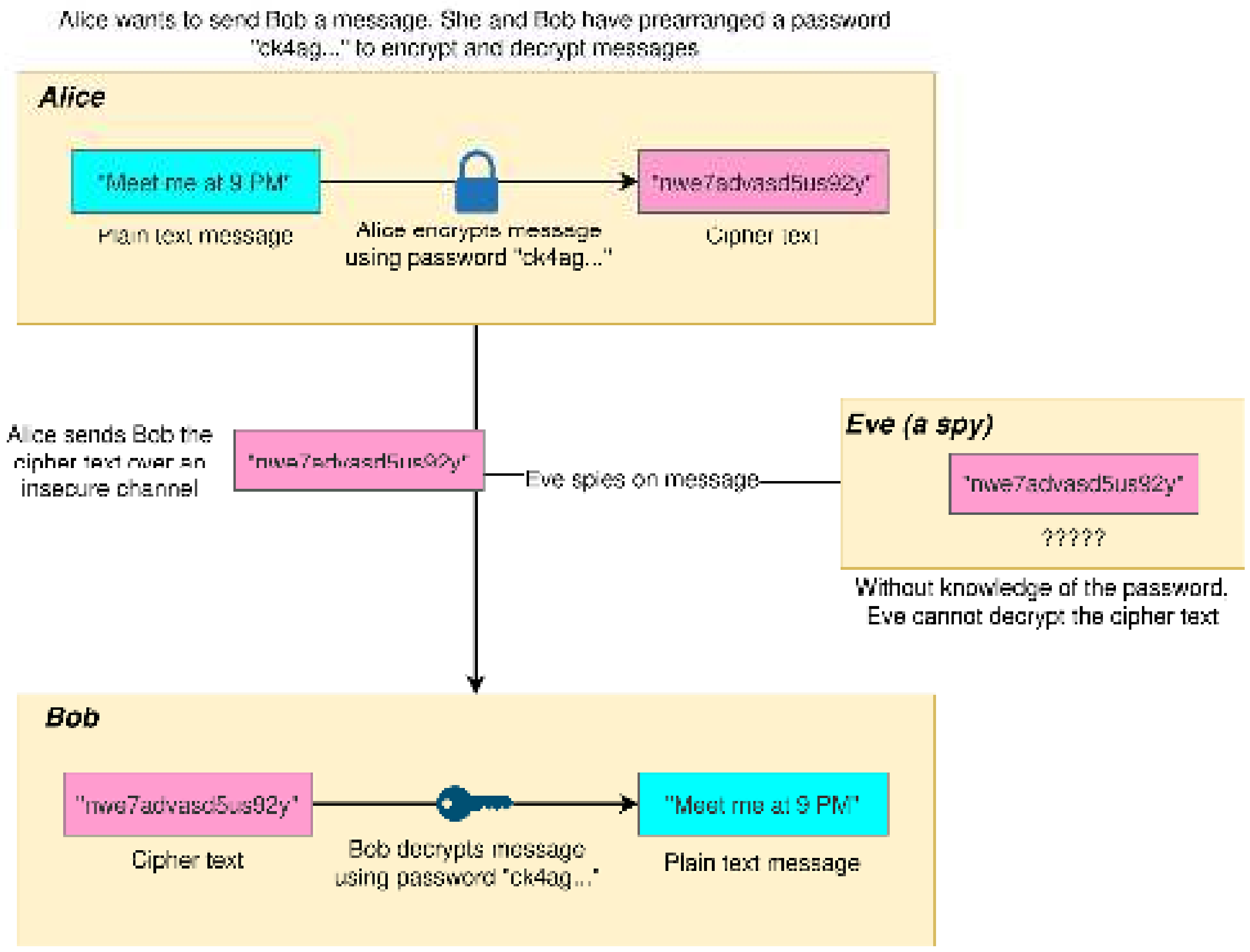


Hash functions in red are known to be ***compromised*** *and are* ***considered unsafe***

Symmetric key encryption

- Both Alice and Bob need a prearranged way to encrypt and decrypt messages
  - Eg. a shared password ("symmetric key")
    - How to securely communicate the key? Chicken and egg problem.

The Key Lock: The lock can be


Alice wants to send Bob a message. She and Bob have prearranged a password "ck4ag..." to encrypt and decrypt messages
Alice
"Meet me at 9 PM"
Plain text message
Alice encrypts message using password "ck4ag..."
"nwe7advasd5us92y"
Cipher text
Alice sends Bob the cipher text over an insecure channel
"nwe7advasd5us92y"
Eve spies on message
Eve (a spy)
"nwe7advasd5us92y"
?????
Without knowledge of the password, Eve cannot decrypt the cipher text
Bob
"nwe7advasd5us92y"
Cipher text
Bob decrypts message using password "ck4ag..."
"Meet me at 9 PM"
Plain text message

Symmetric key encryption

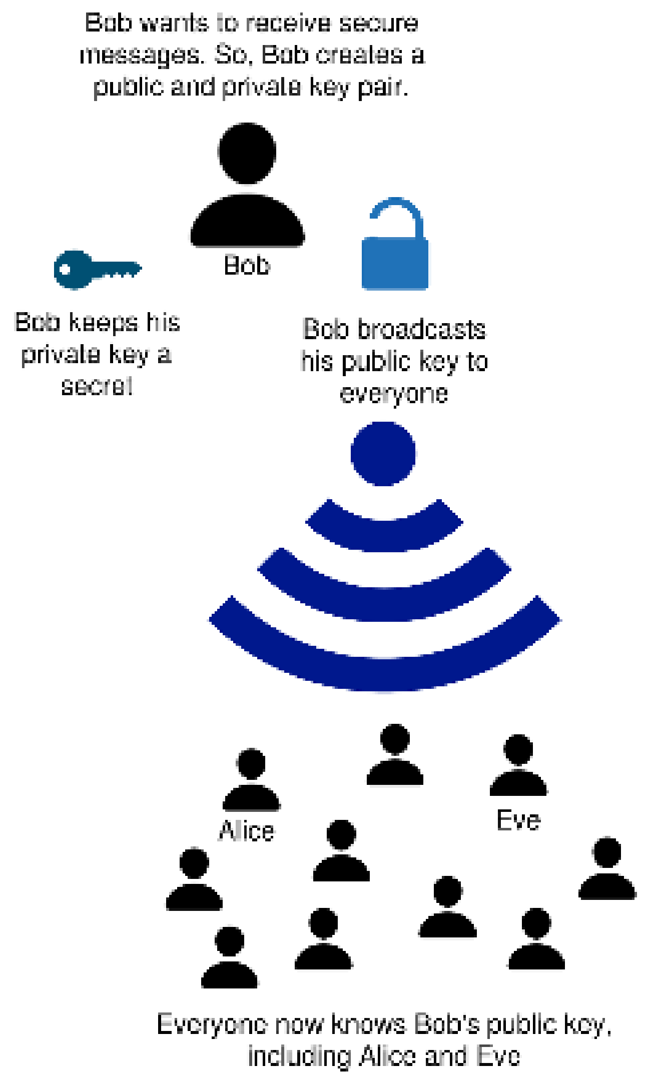
- Both Alice and Bob need a prearranged way to encrypt and decrypt messages
  - Eg. a shared password ("symmetric key")
    - How to securely communicate the key? Chicken and egg problem.

Using Exclusive OR (XOR) in Cryptography	
XOR Logic $\otimes$	Same Bits $0 \otimes 0 = 0$ $1 \otimes 1 = 0$ Different Bits $0 \otimes 1 = 1$ $1 \otimes 0 = 1$
Encrypt	**00110101 Plain text** $\otimes$ **11100011 Secret key** **= 11010110 Ciphertext**
Decrypt	**11010110 Ciphertext** $\otimes$ **11100011 Secret key** **= 00110101 Plain text**

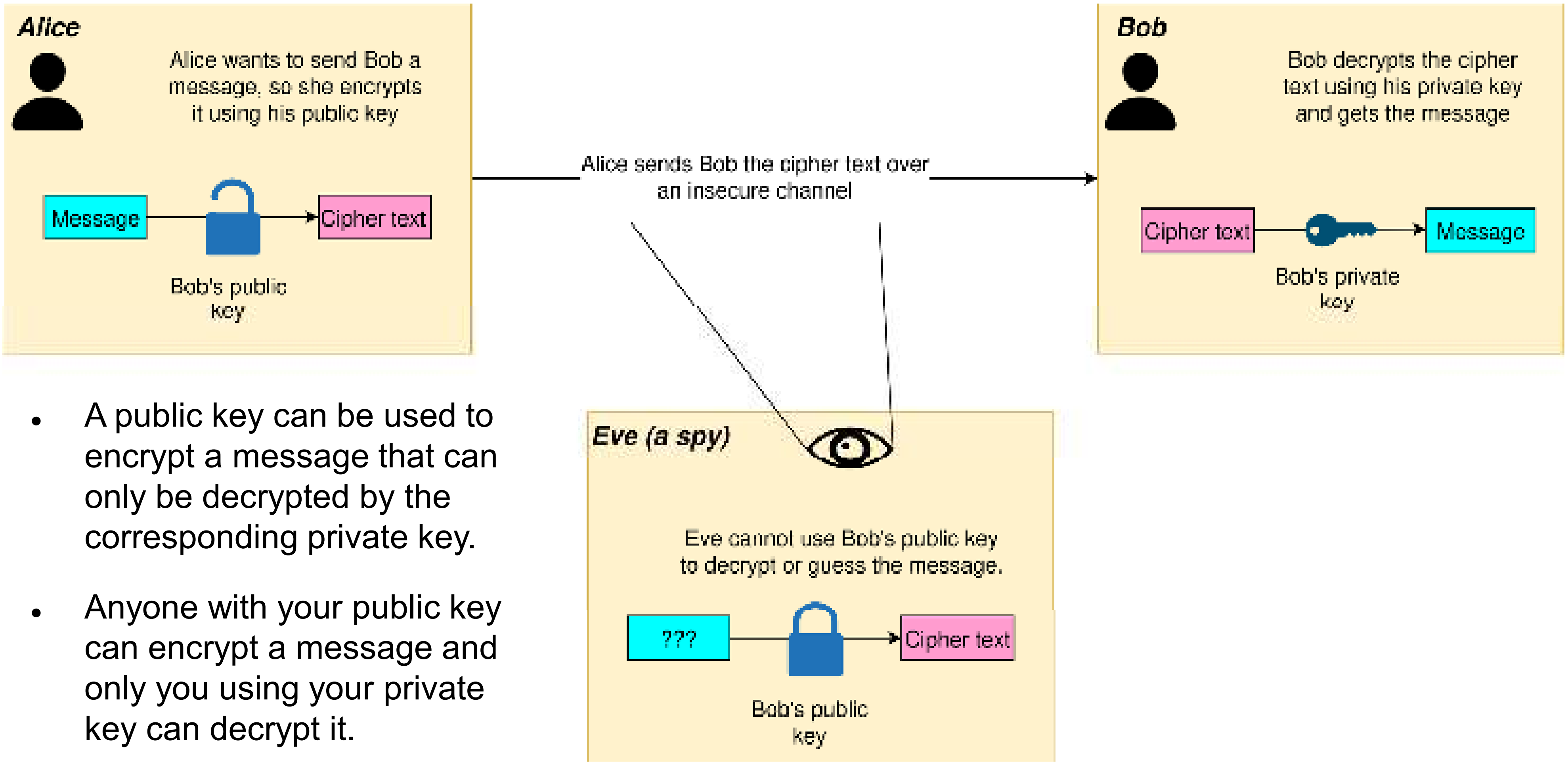
Asymmetric key encryption

Public keys and private keys

- Send all your friends identical locks which only you have the key to. Whenever they want to send you something, they will lock it in a box using your lock, so only you can open it.
- In cryptography, “opened click-locks” are public keys and “keys” are private keys.


Bob wants to receive secure messages. So, Bob creates a public and private key pair.
Bob
Bob keeps his private key a secret
Bob broadcasts his public key to everyone
Alice
Eve
Everyone now knows Bob's public key, including Alice and Eve

Asymmetric key encryption
Public keys and private keys


Alice
Alice wants to send Bob a message, so she encrypts it using his public key
Message
Cipher text
Bob's public key
Alice sends Bob the cipher text over an insecure channel
Bob
Bob decrypts the cipher text using his private key and gets the message
Cipher text
Message
Bob's private key
Eve (a spy)
Eve cannot use Bob's public key to decrypt or guess the message.
???
Cipher text
Bob's public key

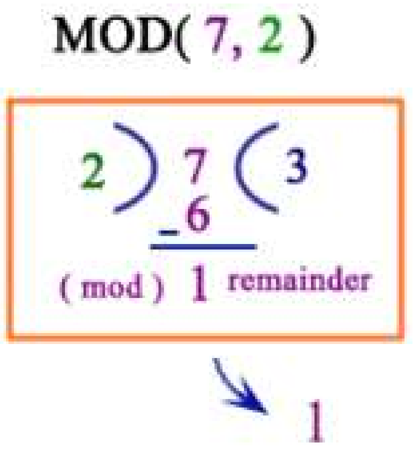
- A public key can be used to encrypt a message that can only be decrypted by the corresponding private key.
- Anyone with your public key can encrypt a message and only you using your private key can decrypt it.

Asymmetric key encryption

- Rivest, Shamir Adleman RSA Algorithm

Encryption
Public keys (5,14)
Text: B → 2
MOD$(2^5, 14) = MOD(32,14) = 4$
Crypto Text: 4 → D

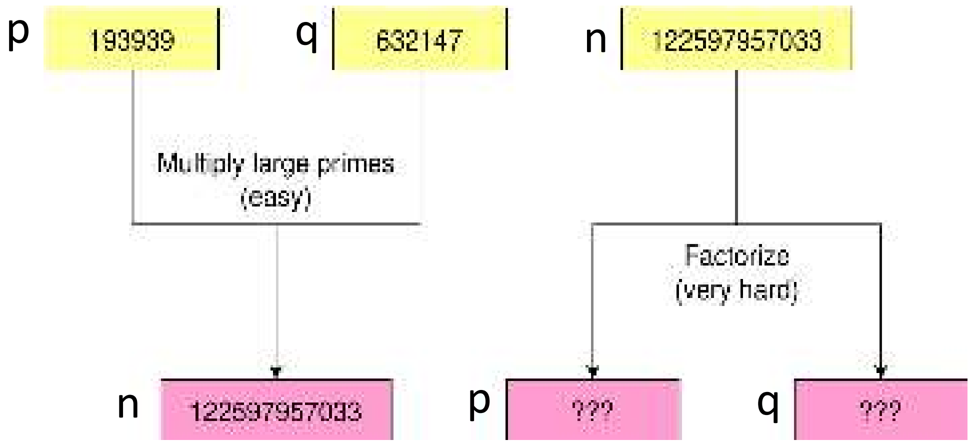
Decryption
Private keys (11,14)
Text: D → 4
MOD$(4^{11}, 14) = MOD(4194304,14) = 2$
Original Msg Text: 2 → B



Factorizing the product of large primes is hard

- *"Can the reader say what two numbers multiplied together will produce the number 8616460799? I think it unlikely that anyone but myself will ever know."*

- William Stanley Jevons, The Principles of Science, 1874

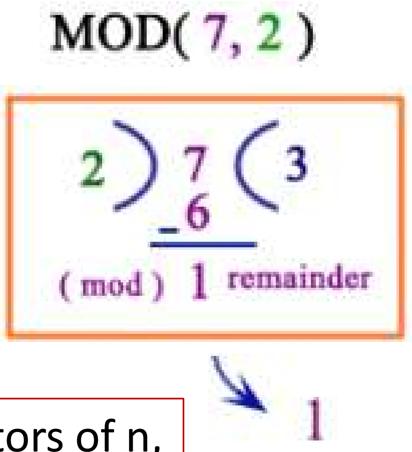


Asymmetric key encryption

Public Key (e, n)

Private Key (d,n)

$\phi(n)=(p-1).(q-1)$
Choose d such that
$MOD(d.e, \phi(n)) = 1$



So to find d you need the factors of n, i.e. p and q

Asymmetric key Generation

Encryption
Public key (5,14)
Text: B → 2
MOD(2^5, 14) = $MOD(32,14) = 4$
Crypto Text: 4 → D

Decryption
Private key (11,14)
Text: D → 4
MOD(4^{11}, 14) = $MOD(4194304,14) = 2$
Original Msg Text: 2 → B

- Rivest, Shamir Adleman RSA Algorithm

How to generate Public key (e, n) and Private key (d, n)?

1. Get two large prime numbers p and q.
2. Find their product n = p . q
3. Find $\phi(n)=(p-1).(q-1)$
4. Choose e such that
   - $1<e<\phi(n)$ and
   - is coprime to n and $\phi(n)$
5. Choose d such that
   - MOD(d.e, $\phi(n)$) = 1

In our case Public key is (5,14) and Private key is (11,14)

1. p=2 and q= 7.
2. Thus, n = 2 x 7 = 14
3. $\phi(n)= 1 \times 6 =6$
4. Choose e such that
   - $1<e<6$ and
   - is coprime to 14 and 6

   - Thus e should be 5. (~~2, 3, 4~~,5)
5. Choose d such that
   - MOD(d.5, 6) = 1

Diffie-Hellman Key Exchange (DHKE)

We use a shared secret key in the symmetric encryption algorithm to encrypt and decrypt messages, but how to distribute the secret key between two parties?

Diffie-Hellman Key Exchange protocol provides a radically new method of distributing cryptographic keys,

- Two parties using DHKE have no prior knowledge of each other.
- DHKE is a method for securely exchanging cryptographic keys over insecure channels like the internet.
- DHKE uses the idea of the one-way function.
- The DHKE is not used for encryption and decryption, instead, it is about creating a shared secret key for encryption and decryption.
- DHKE is not about "symmetric secret key exchange", instead, it is about "creating symmetric secret key" together with exchanging public keys.

Diffie-Hellman Key Exchange (DHKE)

Alice

a $(1<a<p)$

Public

p, g

p is prime

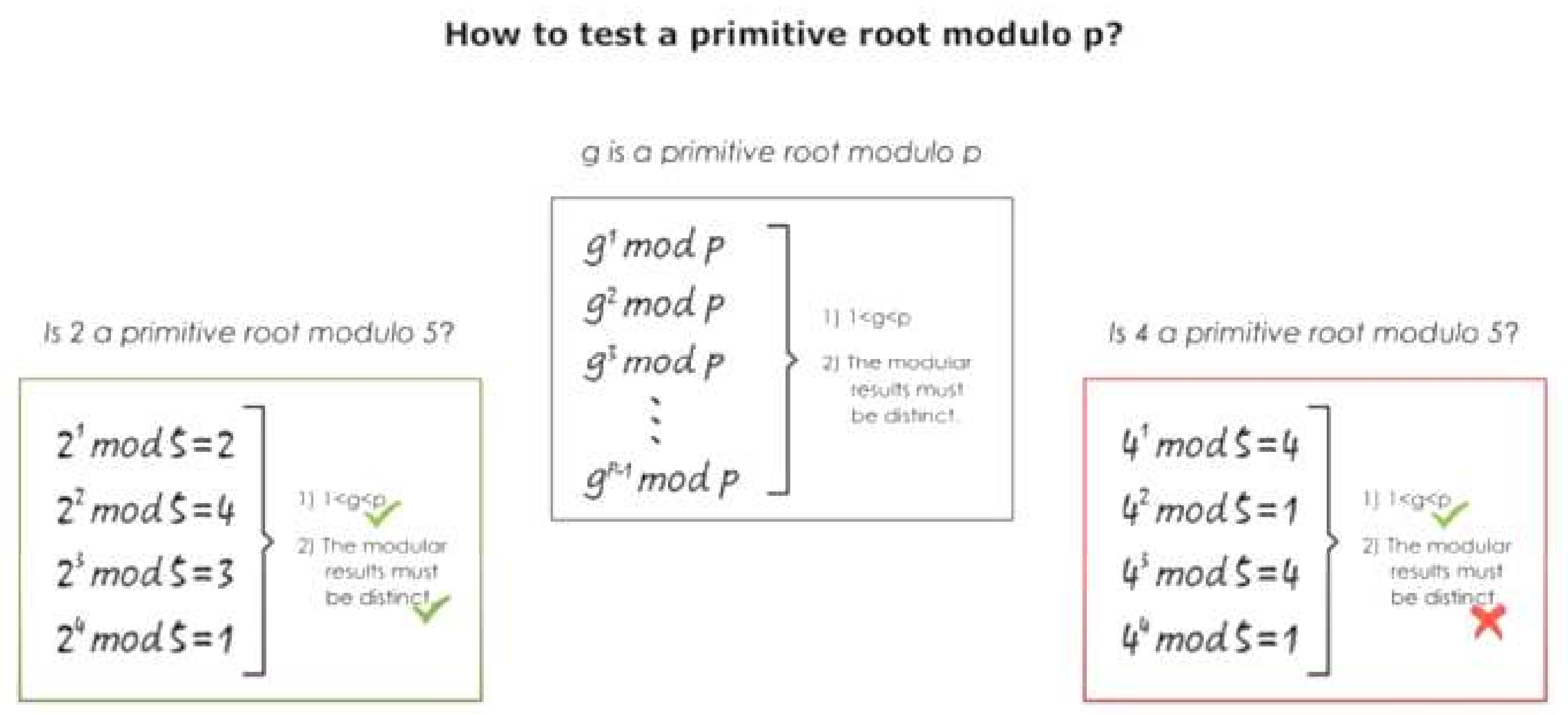
g is a primitive root modulo p

Bob

b $(1<b<p)$

$$\begin{tikzcd}
g^{a} \bmod p \arrow[drr] & & g^{b} \bmod p \arrow[dll] \\
g^{b} \bmod p \arrow[d] & & g^{a} \bmod p \arrow[d] \\
(g^{b} \bmod p)^{a} \bmod p & = & (g^{a} \bmod p)^{b} \bmod p
\end{tikzcd}$$

Diffie-Hellman Key Exchange (DHKE)



How to test a primitive root modulo p?

g is a primitive root modulo p

$$\left.\begin{array}{l} g^1 \bmod p \\ g^2 \bmod p \\ g^3 \bmod p \\ \vdots \\ g^{p-1} \bmod p \end{array}\right\}$$

1) 1<g<p

2) The modular results must be distinct.

Is 2 a primitive root modulo 5?

$$\left.\begin{array}{l} 2^1 \bmod 5 = 2 \\ 2^2 \bmod 5 = 4 \\ 2^3 \bmod 5 = 3 \\ 2^4 \bmod 5 = 1 \end{array}\right\}$$

1) 1<g<p ✓

2) The modular results must be distinct ✓

Is 4 a primitive root modulo 5?

$$\left.\begin{array}{l} 4^1 \bmod 5 = 4 \\ 4^2 \bmod 5 = 1 \\ 4^3 \bmod 5 = 4 \\ 4^4 \bmod 5 = 1 \end{array}\right\}$$

1) 1<g<p ✓

2) The modular results must be distinct ✗

Diffie-Hellman Key Exchange (DHKE)

Public

7, 5

7 is prime

5 is a primitive root modulo 7

Alice

5

(1<5<7)

5^5 mod 7=3

4

$(4)^5$ mod 7=2

=

Bob

2

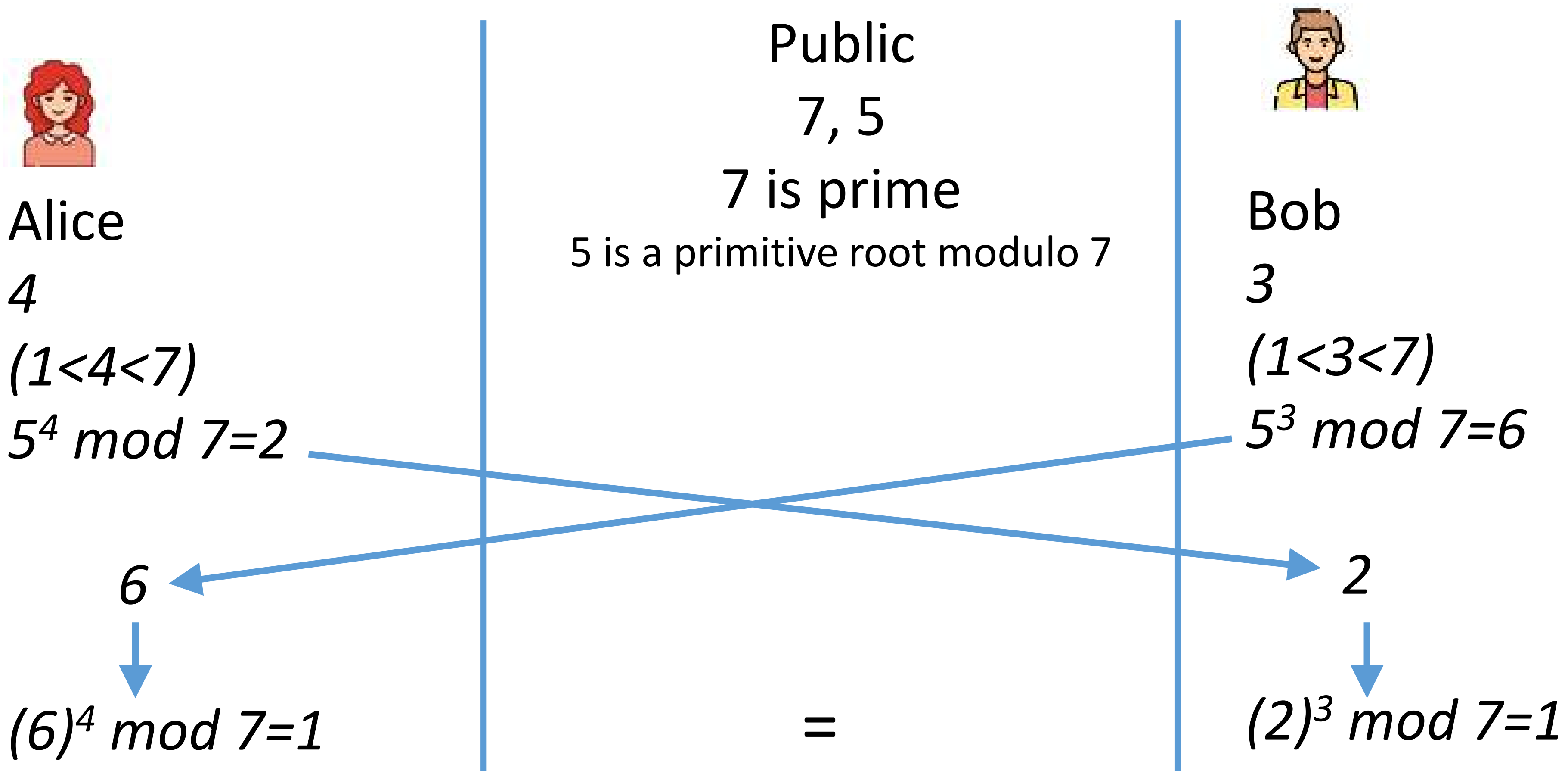
(1<2<7)

5^2 mod 7=4

3

$(3)^2$ mod 7=2

Diffie-Hellman Key Exchange (DHKE)

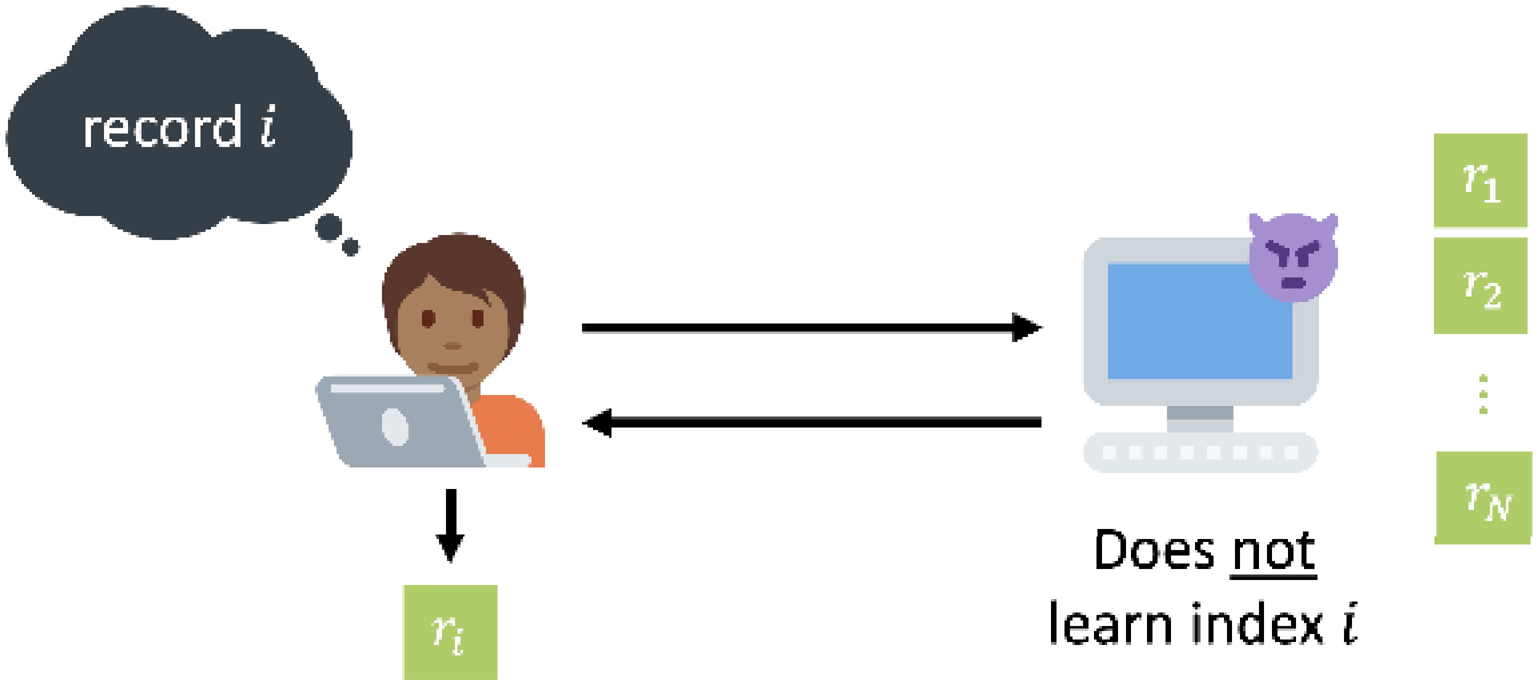


Homomorphic encryption

The challenge is to fetch an article from Wikipedia without revealing to the server which article was fetched.

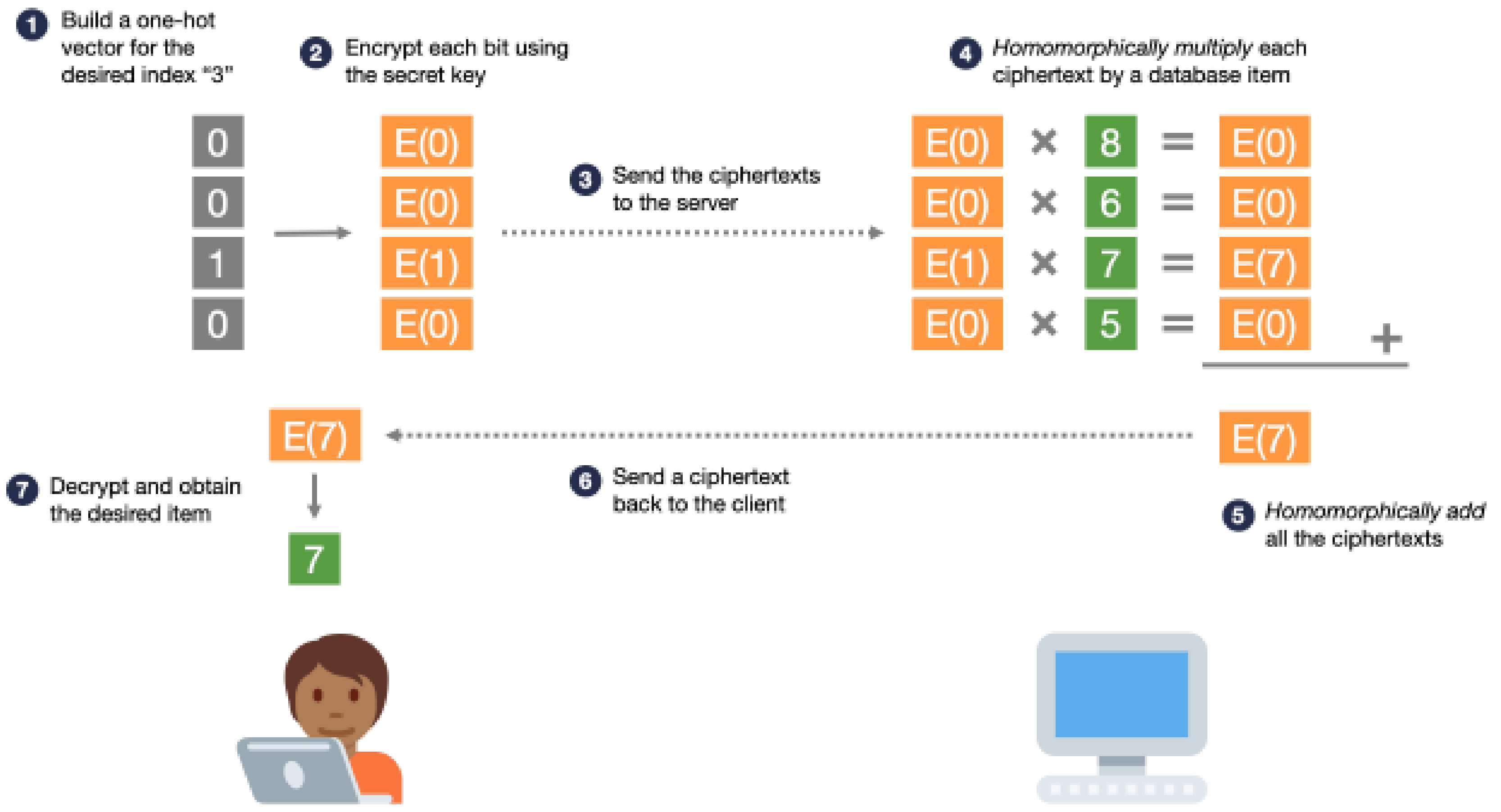
- One naive solution is downloading and storing the entire Wikipedia locally, but this requires too much bandwidth and storage (about 10 GB).
- Another approach is to group articles in "buckets" and send dummy responses along with the real one, but this still leaks some information over time.
- A better solution comes from cryptography, known as Private Information Retrieval (PIR).
- PIR allows a client to retrieve an item from a database without revealing which item was requested, even if the server is malicious.

Homomorphic encryption



https://blintzbase.com/posts/pir-and-fhe-from-scratch/

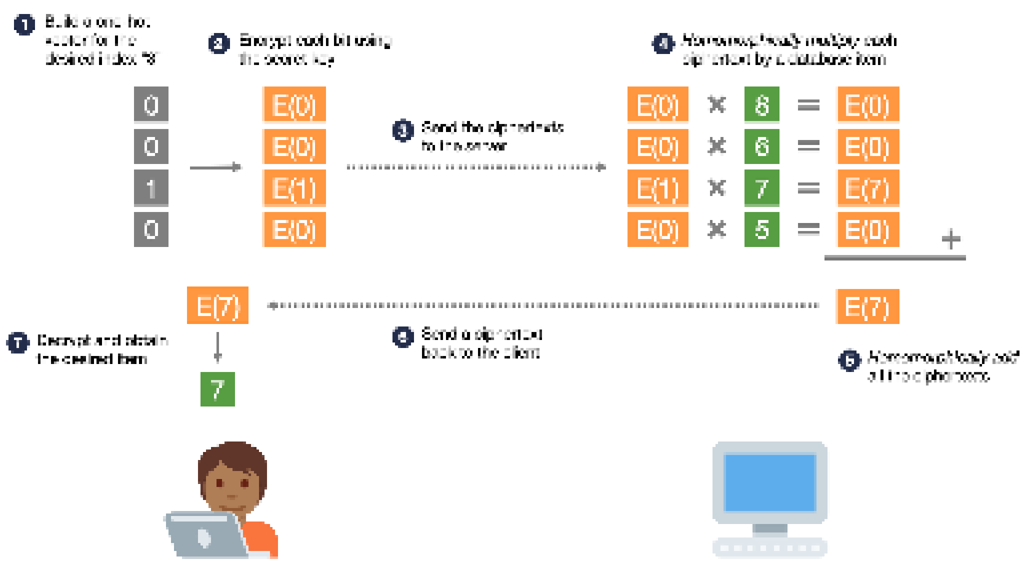
Homomorphic encryption



1 Build a one-hot vector for the desired index "3"
2 Encrypt each bit using the secret key
3 Send the ciphertexts to the server
4 *Homomorphically multiply* each ciphertext by a database item
5 *Homomorphically add* all the ciphertexts
6 Send a ciphertext back to the client
7 Decrypt and obtain the desired item
0
0
1
0
E(0)
E(0)
E(1)
E(0)
E(0) × 8 = E(0)
E(0) × 6 = E(0)
E(1) × 7 = E(7)
E(0) × 5 = E(0)
+
E(7)
E(7)
7

https://blintzbase.com/posts/pir-and-fhe-from-scratch/

Homomorphic encryption


1 Build a one-hot vector for the desired index "3"
2 Encrypt each bit using the secret key
3 Send the ciphertexts to the server
4 Homomorphically multiply each ciphertext by a database item
5 Homomorphically add all the products
6 Send a ciphertext back to the client
7 Decrypt and obtain the desired item
0
0
1
0
E(0)
E(0)
E(1)
E(0)
E(0) × 8 = E(0)
E(0) × 6 = E(0)
E(1) × 7 = E(7)
E(0) × 5 = E(0)
+
E(7)
E(7)
7

The special feature of homomorphic encryption is that it is possible to perform the below two operations:

•**Homomorphic addition**: If the ciphertext c_1 encrypts m_1, and the ciphertext c_2 encrypts m_2,

then

$$c' = c_1 + c_2 \text{ encrypts the value } m_1 + m_2$$

•**Homomorphic plaintext multiplication**: If the ciphertext c_1 encrypts m_1 and we have some other plaintext m_2,

then

$$c' = c_1 * m_2 \text{ encrypts the value } m_1 * m_2$$

Basically, these properties let the server "add" ciphertexts or "multiply them by plaintexts", without learning what they encrypt.

https://blintzbase.com/posts/pir-and-fhe-from-scratch/

Homomorphic encryption

Symmetric cryptography
Examples: AES or ChaCha

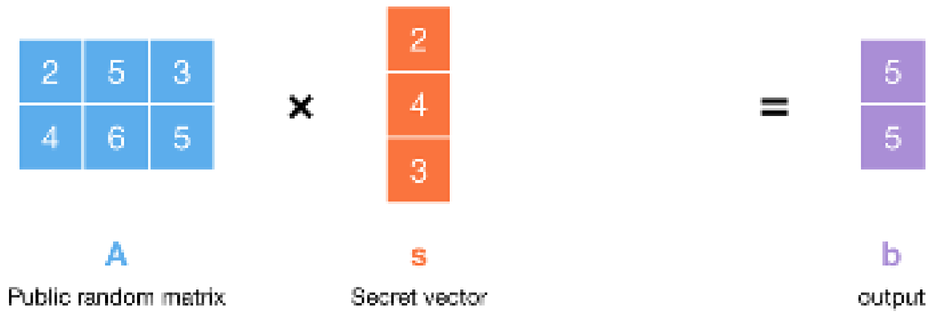
Public-key cryptography
Examples: RSA and elliptic curve cryptography (ECC)

Lattice-based cryptography
Example: Learning with errors (LWE)

Homomorphic encryption

Lattice-based cryptography

Example: Learning with errors (LWE)



Given A and b, it is *easy* to determine s

Assume that q is some prime number.
We write Zq to represent the set of integers modulo q

$x \in Zq$, mean that x is an integer in the range $[0,q-1]$.

We assume that all operations between elements of Zq are modulo q;
so, if we have $x=4 \in Z_7$ and $y=5 \in Z_7$,
then $x+y=2 \in Z_7$ and $x \cdot y=6 \in Z_7$.

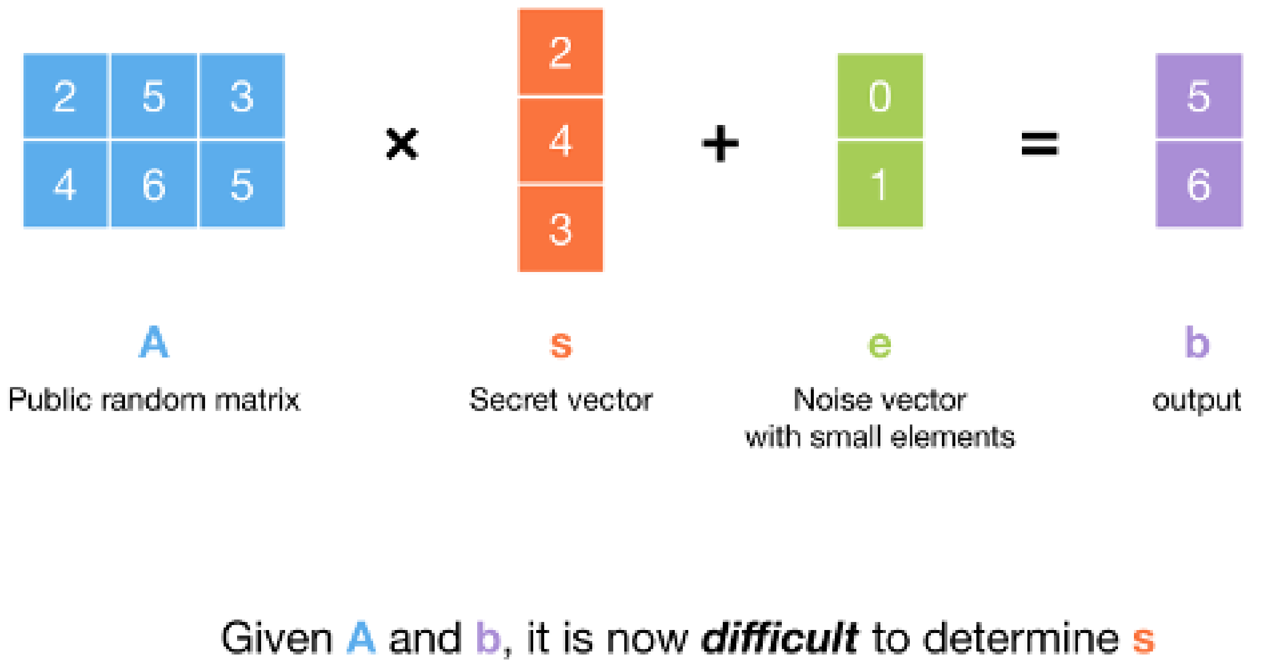
We write Z_q^n to represent an n-element vector, and Z_q^{mxn} to represent an $m \times n$ matrix.

When we say that we "sample" vectors or matrices "at random", we mean that we choose a random number in the range $[0,q-1]$ for each element of the vector or matrix.

Homomorphic encryption

Lattice-based cryptography

Example: Learning with errors (LWE) is the building block for Homomorphic encryption.



2 5 3
4 6 5
×
2
4
3
+
0
1
=
5
6
A
Public random matrix
s
Secret vector
e
Noise vector with small elements
b
output
Given A and b, it is now *difficult* to determine s

Introduction to Cryptocurrency

Definition: Cryptocurrency is a digital or virtual currency that uses cryptography for security, making it difficult to counterfeit.

Decentralization: Unlike traditional currencies, cryptocurrencies operate on a decentralized network of computers using blockchain technology.

Blockchain Technology

Blockchain Defined: A blockchain is a distributed ledger that records all transactions made with a particular cryptocurrency, across a network of computers.

Immutable Records: Once a transaction is recorded, its data cannot be altered without the alteration of all subsequent blocks and network consensus.

How Transactions Work

Initiation: A user initiates a transaction by sending cryptocurrency from their digital wallet to another user's wallet address.

Verification: The transaction is verified by network nodes through a process called mining (in the case of Bitcoin).

Mining Process

Definition: Mining is a consensus mechanism used to validate transactions and add them to the blockchain ledger.

The process steps are:

1. Transactions are bundled into a block.
2. Miners compete to solve a complex mathematical puzzle related to the block.
3. The first miner to solve the puzzle gets to add the block to the blockchain and is rewarded with newly minted cryptocurrency (e.g., bitcoins).

Example: Bitcoin Transaction

Step 1: Alice wants to send 1 Bitcoin to Bob.
Step 2: Alice initiates the transaction from her digital wallet, specifying Bob's wallet address and the amount of Bitcoin to send.
Step 3: The transaction is broadcast to the Bitcoin network and awaits verification.
Step 4: Miners verify the transaction's legitimacy by solving a cryptographic puzzle.
Step 5: Once verified, the transaction is added to a new block on the blockchain.
Step 6: Bob receives the Bitcoin in his digital wallet.
Slide 6: Security Measures

Cryptography: Ensures secure transaction processing and the creation of new coins.
Public and Private Keys: A cryptographic pair that secures transactions. The public key is shared and used to receive funds, while the private key is kept secret and used to sign transactions.
Network Consensus: Transactions are confirmed by consensus among participants in the network, enhancing security and trust.

Block Chain in HR and Ops

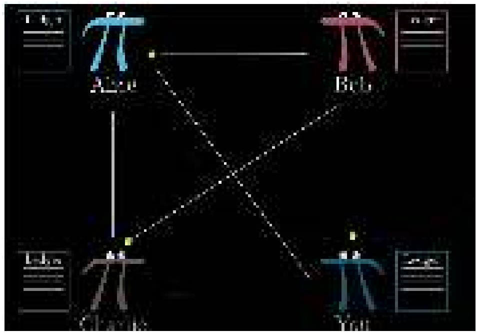
1. A distributed leisure
   - Made of blocks
   - Each block has a list of signed transactions

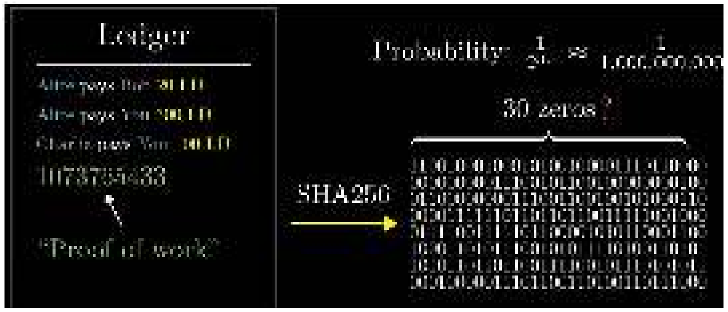
2. Proof of work
   - Hard to generate
   - Easy to verify

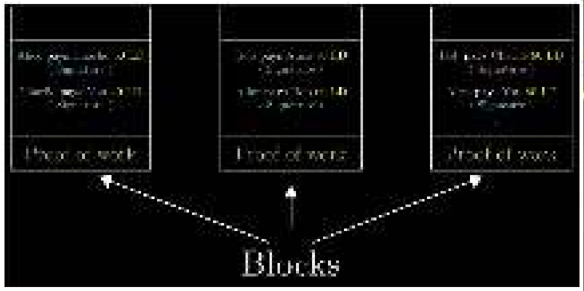
3. The miner puts in effort to mine
   - Gets a reward for each block
   - Also gets a reward for each transaction

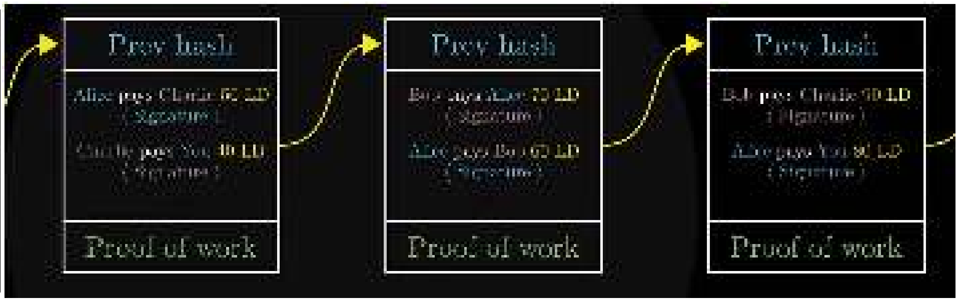
1. The chain
   - The hash of each block is appended to the next block

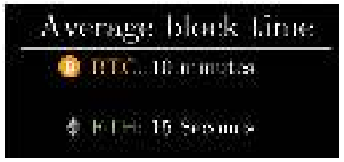
2. The longest chain is the authentic blockchain

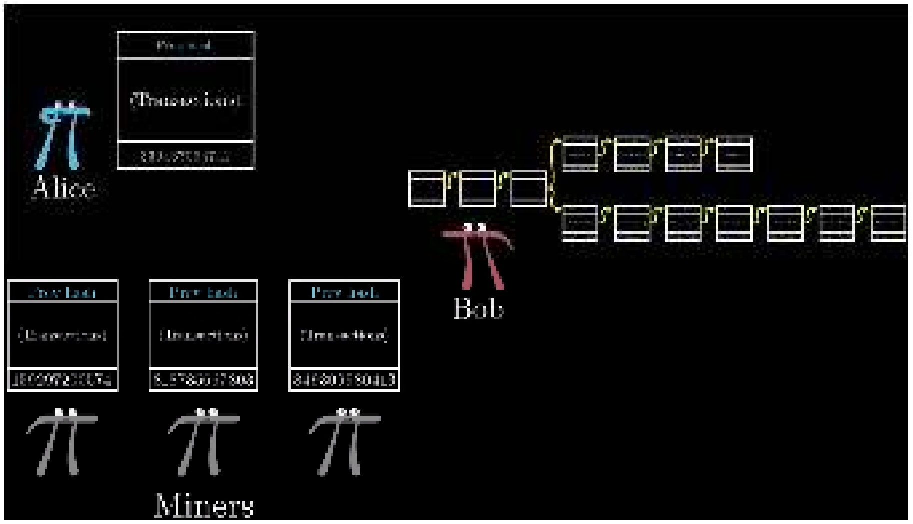



Ledger
1073734433
"Proof of work"
Probability: $\frac{1}{2^{30}} \approx \frac{1}{1{,}000{,}000{,}000}$
30 zeros?
SHA256






Average block time
BTC: 10 minutes
ETH: 15 Seconds



Advantages and Challenges

Advantages:

1. Decentralization reduces the risk of centralized control and fraud.
2. Transparency of transactions, while maintaining privacy.
3. Lower transaction fees compared to traditional banking systems.

Challenges:

1. Scalability issues can lead to slow transaction times and higher costs.
2. Regulatory uncertainty as governments figure out how to deal with cryptocurrencies.
3. Security concerns, despite strong cryptographic protections, include the risk of loss due to hacking or lost private keys.

Cryptocurrency

Cryptocurrency Wallets
Users store their cryptocurrencies in digital wallets, which can be hardware-based or software-based.
These wallets do not store the currency per se but the keys used to access cryptocurrency addresses and sign transactions.

Smart Contracts
Some cryptocurrencies, like Ethereum, enable smart contracts, which are self-executing contracts with the terms of the agreement between buyer and seller directly written into lines of code.

Security
Cryptocurrencies are generally considered secure, but they are not immune to attacks. Wallets can be hacked, and exchanges are vulnerable to breaches.
Users are advised to employ security measures, such as using hardware wallets and enabling two-factor authentication.

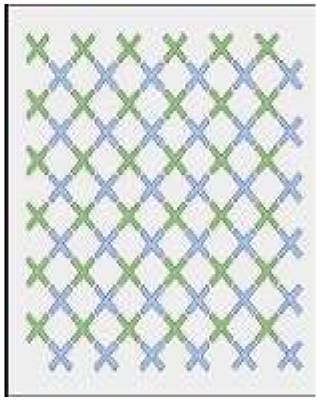
Regulation and Legality
The regulatory environment for cryptocurrencies is evolving and varies significantly from country to country.
Some countries have embraced them, while others have imposed restrictions or outright bans.
Cryptocurrencies represent a complex interplay of technology, economics, and policy. Their operation relies on cryptographic techniques and a consensus mechanism to create a secure, decentralized system for financial transactions.

Non-Fungible Token (NFT) Technology

Non-Fungible Token (NFT) Technology

- **Definition**:
  - Unique digital assets representing ownership of a specific item or piece of content (art, music, videos, virtual real estate, etc.).
  - Cannot be replaced or exchanged on a one-to-one basis (non-fungible).
- **Key Features**:
  - **Unique**: Each NFT has a unique identifier that makes it distinguishable from others.
  - **Indivisible**: NFTs cannot be divided into smaller units.
  - **Ownership**: Ownership of NFTs is recorded on a blockchain, making it verifiable and secure.
  - **Interoperability**: NFTs can be traded across multiple platforms and marketplaces.
  - **Smart Contracts**: They use smart contracts for buying, selling, and transferring, automating transactions without intermediaries.
- **Blockchain Platforms**:
  - Ethereum (most common for NFTs)
  - Binance Smart Chain
  - Flow (used by NBA Top Shot)
- **Use Cases**:
  - Digital art (e.g., Beeple's "Everydays")
  - Gaming assets (e.g., skins, weapons)
  - Virtual real estate (e.g., Decentraland)
  - Collectibles (e.g., CryptoPunks, NBA Top Shot)
- **Challenges**:
  - **Environmental Impact**: Energy consumption of blockchain networks.
  - **Scalability**: Transaction speed and cost can be high.
  - **Regulation**: Unclear legal status in many jurisdictions.
- **Future Trends**:
  - Increased adoption in entertainment and media.
  - Integration with augmented reality (AR) and virtual reality (VR).
  - Evolving marketplace ecosystems for creators and brands.

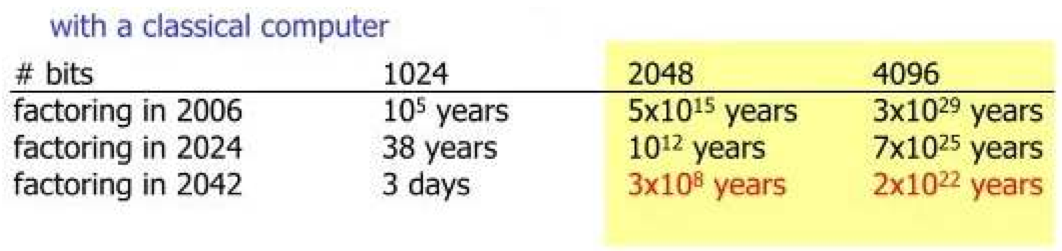
Google Quantum AI lab



‘Bristlecone’ quantum processor

- Gate-based superconducting system
- Qubit linear array technology
- Error in readout (1%), single-qubit gates (0.1%) and two-qubit gates (0.6%)
- Can perform simulations, optimization, and machine learning.

The future of Data Security and cryptography

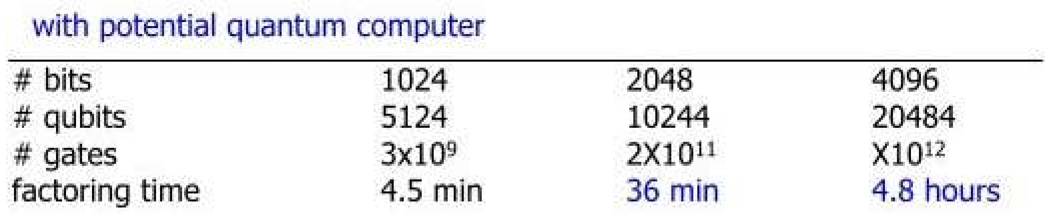


with a classical computer

# bits	1024	2048	4096
factoring in 2006	10^5 years	$5x10^{15}$ years	$3x10^{29}$ years
factoring in 2024	38 years	10^{12} years	$7x10^{25}$ years
factoring in 2042	3 days	$3x10^8$ years	$2x10^{22}$ years

The private key is protected because factorizing is almost impossible

In 2001 IBM factorized 15 in to 3 and 5 using quantum computer with 7 qubits.



with potential quantum computer

# bits	1024	2048	4096
# qubits	5124	10244	20484
# gates	$3x10^9$	$2X10^{11}$	$X10^{12}$
factoring time	4.5 min	36 min	4.8 hours

Quantum computing vs Lab grown Brain cells

Science

Lab-grown brain cells play video game Pong

[illegible]

These 800,000 lab-grown brain cells can play a 1970s video game, [illegible]

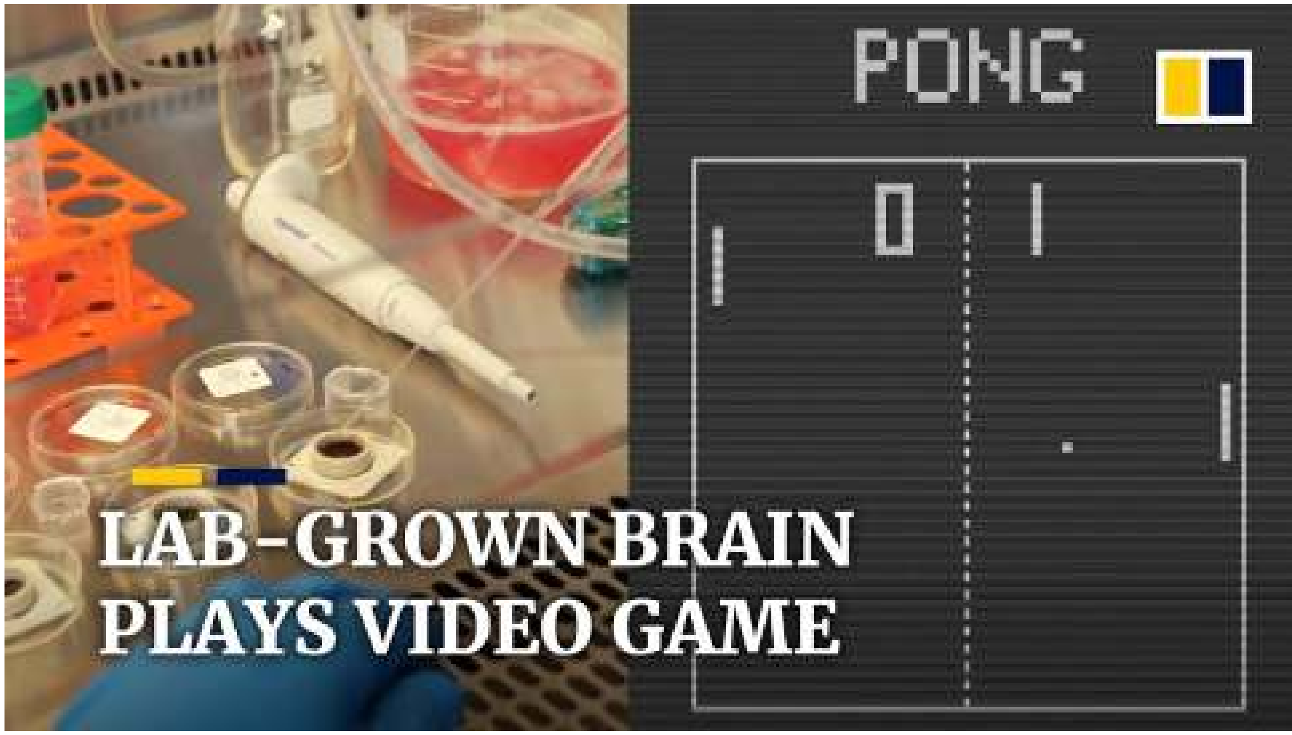
Scientists teach brain cells to play video game Pong

Cells were able to play longer rallies over time, with researchers now planning to see how they react when drunk

Mind

Human brain cells in a dish learn to play Pong faster than an AI

Hundreds of thousands of brain cells in a dish are being taught to play Pong by responding to pulses of electricity – and can improve their performance more quickly than an AI can



History of Cryptography

- Cryptography is the science of encoding and decoding information to keep it secure from unauthorized access. It has a long history dating back to ancient civilizations.
- **Ancient Cryptography:** The earliest forms of cryptography were simple and involved transposition or substitution of characters. For example, the Spartans used a scytale, a tool that involved wrapping a strip of parchment around a cylinder to reveal a message. The Caesar cipher, attributed to Julius Caesar, involved substituting each letter in the plaintext with a letter a fixed number of places down the alphabet.
- **Medieval to Renaissance Cryptography:** During this period, cryptography became more sophisticated. The Arabs developed new methods of encryption and also contributed significantly to the development of frequency analysis, which could break many simple ciphers. In the Renaissance, Europeans like Leon Battista Alberti advanced the art of secret writing with polyalphabetic ciphers, which used multiple alphabets to make decryption harder.
- **The Advent of Mechanical and Electromechanical Devices:** In the 20th century, mechanical and electromechanical devices were developed for more complex encryption. The most famous of these was the German Enigma machine used during World War II, which the Allies managed to break—most notably by teams led by Alan Turing at Bletchley Park.
- **Modern Cryptography:** Modern cryptography has evolved significantly with the advent of computers. It can be divided into two broad categories:

1. **Symmetric Key Cryptography:** Both the sender and receiver share a single key to encrypt and decrypt information. It's fast and efficient for large amounts of data. The Advanced Encryption Standard (AES) is a widely used symmetric encryption standard today.
2. **Asymmetric Key Cryptography (Public Key Cryptography):** This uses a pair of keys, a public key for encryption and a private key for decryption. This solves the problem of key distribution that symmetric key cryptography suffers from. RSA is one of the most common public key systems.

- **Cryptographic Protocols and Practices:** Cryptographic techniques are also used in various protocols to secure communication over the internet, such as SSL/TLS for web traffic, PGP for emails, and various encryption methods for secure file storage.
- **Current Technology and Trends:** Cryptography has grown beyond just encryption and decryption. It now includes hash functions, which are used to verify data integrity and authenticity, and digital signatures, which provide a way to ensure the non-repudiation of a digital document. Blockchain technology uses cryptographic principles to maintain a secure and decentralized ledger for transactions, which is the backbone of cryptocurrencies like Bitcoin.
- Homomorphic encryption is a recent trend that allows computation on encrypted data without needing to decrypt it first. Quantum cryptography is another emerging field, especially important as quantum computing poses a threat to current cryptographic algorithms. Quantum key distribution (QKD) is seen as a potential way to create unbreakable encryption due to the principles of quantum mechanics.
- Overall, cryptography continues to evolve rapidly, aiming to stay ahead of potential security threats while enabling secure communication and data protection in an increasingly digital world.

Use cases for cryptographic hashing

Cryptographic hashing serves as a cornerstone for a multitude of security-related applications due to its ability to produce a unique fixed-size string (the hash) from input data of any size. Here are several use cases for cryptographic hashing:

1. **Data Integrity Checks:** Cryptographic hashes are used to verify the integrity of data during transmission or storage. A file can be hashed and the hash value can be compared with the hash received from the sender or stored alongside the data. If the hashes match, the data is considered to be intact.
2. **Password Storage:** Storing passwords in plaintext is insecure. Instead, systems store a hash of the user's password. When a user logs in, the system hashes the entered password and compares it to the stored hash. This means that even if the database is compromised, the actual passwords are not exposed.
3. **Digital Signatures:** Hashes are used in digital signatures, where a hash of a document is encrypted with a private key to create a signature. The signature can be verified by decrypting it with the corresponding public key and comparing the result with the document's hash.
4. **Blockchain and Cryptocurrencies:** In blockchain technology, each block contains the hash of the previous block, creating a linked chain. This ensures the immutability of the ledger; any change in a block would require re-mining all subsequent blocks. Cryptocurrencies like Bitcoin use cryptographic hashing for their proof-of-work system.
5. **Data Deduplication:** In large storage systems, hashing can identify duplicate pieces of data. By comparing hashes instead of entire files, the system can efficiently detect and eliminate duplicates to save space.
6. **Commitment Schemes:** Hash functions are used in commitment schemes where one party wants to commit to a chosen value while keeping it hidden, to be revealed later. The hash of the value is shared, and since hashes are one-way functions, the original value cannot be derived without additional information.
7. **Checksums for Software Downloads:** Software distributors often provide a hash value along with downloads. Users can hash the downloaded file and compare it to the provided hash to ensure the file has not been tampered with.
8. **Secure Message Digests:** Email and messaging services use cryptographic hashing to create message digests, ensuring that the message has not been altered during transmission.
9. **Certificate Fingerprints:** SSL certificates and other security certificates include a hash, known as a fingerprint, which uniquely identifies the certificate and can be used to verify its authenticity.
10. **Random Number Generation:** Hash functions can be used in the generation of pseudo-random numbers or for hashing-based randomization in various algorithms.
11. **Proof of Space and Proof of Burn:** Cryptographic hashes are used in proof of space, where a service proves it has allocated a certain amount of storage space, and proof of burn, where a cryptocurrency proof is created by "burning" or making unusable a certain amount of tokens by sending them to a verifiable unspendable address.

- Cryptographic hashes are a fundamental tool in securing modern digital systems, providing a reliable and efficient way to ensure data integrity, verify authenticity, and maintain privacy.

compromised cryptographic hash functions

- Several cryptographic hash functions have been compromised or broken over time, either by theoretical vulnerabilities or by practical attacks. Here are some notable examples:

1. **MD5:** The MD5 hashing algorithm has been deemed cryptographically broken and unsuitable for further use. Its vulnerabilities to collision attacks were famously demonstrated by producing two different sets of data with the same MD5 hash. This means attackers can create two different documents that hash to the same value, which is a significant security risk.
2. **SHA-1:** SHA-1 has been shown to be vulnerable to collision attacks, which was practically demonstrated in 2017 when researchers from Google and CWI Amsterdam succeeded in creating two different PDF files with the same SHA-1 hash. Due to these vulnerabilities, its use is now generally discouraged in favor of stronger hash functions like SHA-256.
3. **RIPEMD:** The original RIPEMD (as opposed to RIPEMD-160) was also shown to have vulnerabilities. However, its successors, such as RIPEMD-160, have not yet been compromised and are still considered secure.
4. **HAVAL:** HAVAL is a family of cryptographic hash functions that allows users to choose a hash length and the number of rounds. Certain configurations of HAVAL have been shown to be vulnerable to collision attacks.
5. **LAN Manager Hash:** The LAN Manager Hash (LM hash) was used by Microsoft for password protection in early Windows systems. It was widely criticized for its vulnerabilities and has been replaced in later versions of Windows.
6. **NTLMv1:** NTLMv1 is an older Microsoft authentication protocol that has been broken and is considered insecure. It has been replaced by NTLMv2 in modern systems.

- While these hash functions have been compromised, it's important to note that not all versions or derivatives of the algorithms are necessarily insecure. For example, SHA-256 and SHA-3 are currently considered secure despite SHA-1's weaknesses.
- Due to the advancements in computational power and cryptanalysis techniques, older hash functions may become vulnerable over time. This is why the cryptographic community moves towards newer, more secure algorithms as potential vulnerabilities are discovered.

- While SHA-256 is widely regarded as secure and is extensively used, there are various scenarios where other hashing algorithms might be preferable:

1. **Performance:** Some applications require faster hash computations and can use algorithms that are less computationally intensive than SHA-256. For instance, algorithms like SHA-1 (despite its weaknesses) or BLAKE2 are faster and may be used in contexts where the speed of hashing is critical and the security requirements permit their use.
2. **Resource Constraints:** In constrained environments such as embedded systems or IoT devices, resources like CPU power, memory, and battery life are limited. Lightweight hash functions, designed to minimize resource usage, can be more appropriate in these cases.
3. **Security Profiles:** Different hash functions offer different security features. For example, SHA-3 (based on the Keccak algorithm) provides resistance to length-extension attacks, which is a weakness in SHA-256. Depending on the specific security needs, one might opt for a hash function with particular security properties.
4. **Cryptographic Agility:** Some systems are designed to be cryptographically agile, meaning they can switch between algorithms if one becomes compromised. This requires the use of multiple hash functions and the ability to transition between them as needed.
5. **Proofs and Protocols:** Certain cryptographic protocols or proof systems may require specific properties from a hash function that SHA-256 does not provide, like homomorphic properties or being suited for zero-knowledge proofs.
6. **Standard Compliance:** Some industries and applications must adhere to specific standards that require the use of particular hash functions. For example, the digital signature algorithm (DSA) as specified by the Digital Signature Standard (DSS) uses SHA-1 or SHA-2, but not all applications might require the security level of SHA-256.
7. **Cryptographic Research and Diversity:** Relying on a single algorithm could be risky if a vulnerability in the algorithm is discovered. Using a variety of hash functions can spread the risk. Additionally, research into new algorithms can lead to the development of better and more secure hashing techniques.
8. **Quantum Resistance:** In anticipation of quantum computing, there is ongoing research into post-quantum hash functions that would be secure against an attacker with a quantum computer. SHA-256 is not considered quantum-resistant, so future applications might use different hash functions to guard against this threat.

- In summary, while SHA-256 is robust and suitable for many applications, there are practical, performance, and security considerations that may necessitate the use of alternative hashing functions.

Digital Signature

Three purposes of Digital Signature

1. Authentication: A digital signature gives the receiver reason to believe the message was created and sent by the claimed sender
2. Non-repudiation: With digital signature, the sender cannot deny having sent the message later on
3. Integrity: A digital signature ensures that the message was not altered in transit