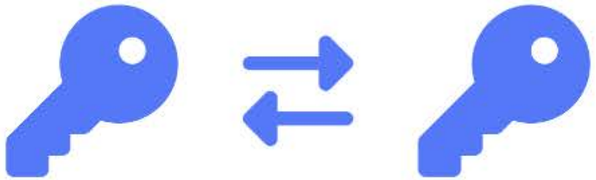
Digital Signatures

alchemy | 2022



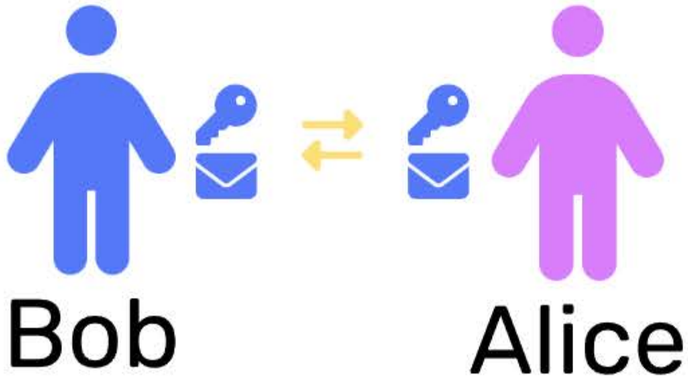
Symmetric Key

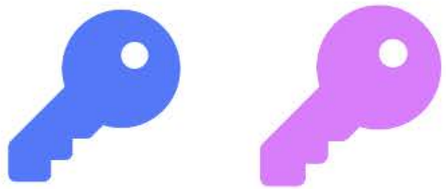
Symmetric Key 🔑 ⇄ 🔑

- Same key on both sides 🔑
- Communicate securely over an unsecured channel 🔐
  - **History**: Military Usage ⚔️
    - **Key**: 324, **Message**: Cat, **Encrypted**: Fcx Each character is moved by 3, 2 and 4 steps respectively.
  - **Example:** AES - Advanced Encryption Standard
  - **Downside**: Both sides must have the key beforehand

Symmetric Key 🔑 🔄 🔑

- Same key on both sides 🔑
- Communicate securely over an unsecured channel 🔐
  - **History**: Military Usage ⚔️
    - **Key**: 324, **Message**: Cat, **Encrypted**: Fcx
  - **Example:** AES - Advanced Encryption Standard
  - **Downside**: Both sides must have the key beforehand





Asymmetric Key

(Public Key Cryptography)

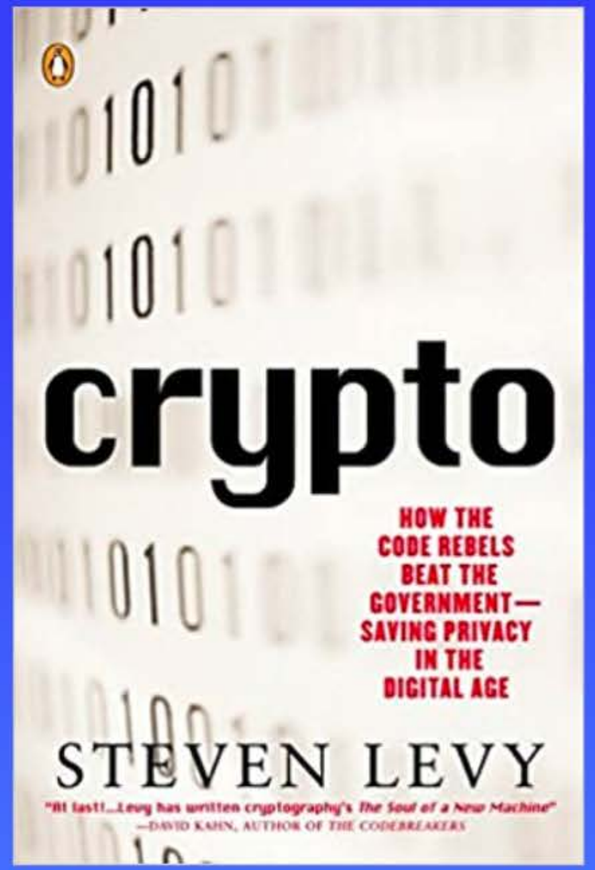
Solve two parties communicate securely without having met beforehand to exchange keys? Examples: RSA and ECDSA.

The RSA algorithm is based on the idea that it's very easy to find the product of two prime numbers, yet extremely difficult to factor out those two prime numbers if you have the product. (https://youtu.be/4zahvcJ9glg)

The ECDSA algorithm uses elliptic curves. It can provide the same level security as other public key algorithms with smaller key sizes, which is the reason it's become quite popular. It is the Digital Signing Algorithm used by Bitcoin, specifically the secp256k1 curve.

Asymmetric Key 🔑🔑

- Whit Diffie had the inspiration to split the key in two ✂️ https://youtu.be/4zahvcJ9glg
- A private/public keypair could:
  1. **Authenticate**: One key signs, (digital signature) the other verifies (by sender's private key / by sender's public key)
  2. **Encrypt**: One key encrypts, the other decrypts (send data: receiver's public key->algo-> / receiver's private key)
- Commonly referred to as **Public Key Cryptography** 💡



1. Public Key Encryption 🔑🔑

- **Bob** provides his **public key**
- **Bob** secures his **private key**
- **Charlie** can encrypt a message only **Bob** can read with **Bob's private key**
- **Eve** could not read this message, only Bob can decrypt it.

1. Public Key Encryption

- **Bob** provides his **public key**
- **Bob** secures his **private key**
- **Charlie** can encrypt a message only **Bob** can read with **Bob's private key**
- **Eve** could not read this message, only Bob can decrypt it.

1. Public Key Encryption

- **Bob** provides his **public key**
- **Bob** secures his **private key**
- **Charlie** can encrypt a message only **Bob** can read with **Bob's private key**
- **Eve** could not read this message, only Bob can decrypt it.

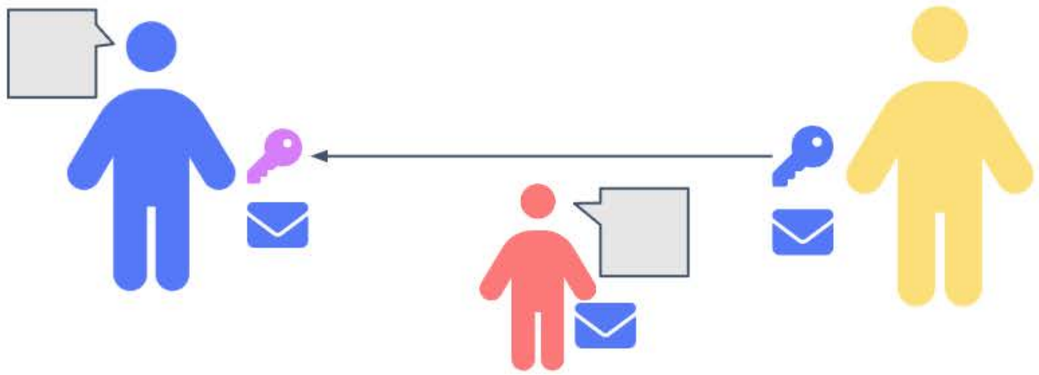
1. Public Key Encryption

Zero Knowledge Proof
Example: I should convince you that some relevant statements is true and I'm able to prove it by without revealing you any details of information or secrets.
Use Case: Instant Messaging, Solve NP class problem
https://youtu.be/fOGdb1CTu5c

- **Bob** provides his **public key**
- **Bob** secures his **private key**
- **Charlie** can encrypt a message only **Bob** can read with **Bob's private key**
- **Eve** could not read this message, only Bob can decrypt it.

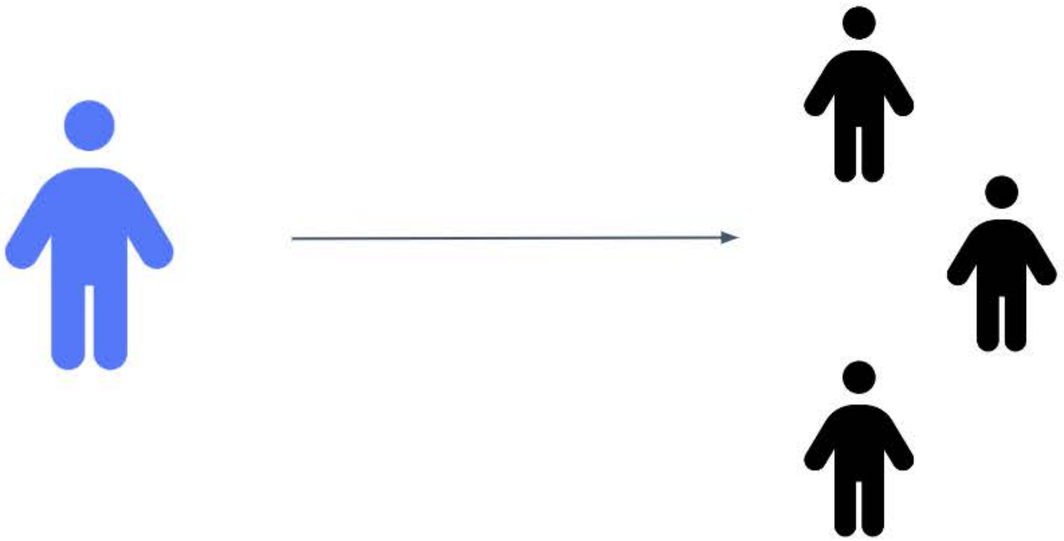
1. Public Key Encryption

- **Bob** provides his **public key**
- **Bob** secures his **private key** make sure no one holding his private key
- **Charlie** can encrypt a message only **Bob** can read with **Bob's private key**
- **Eve** could not read this message, only Bob can decrypt it.



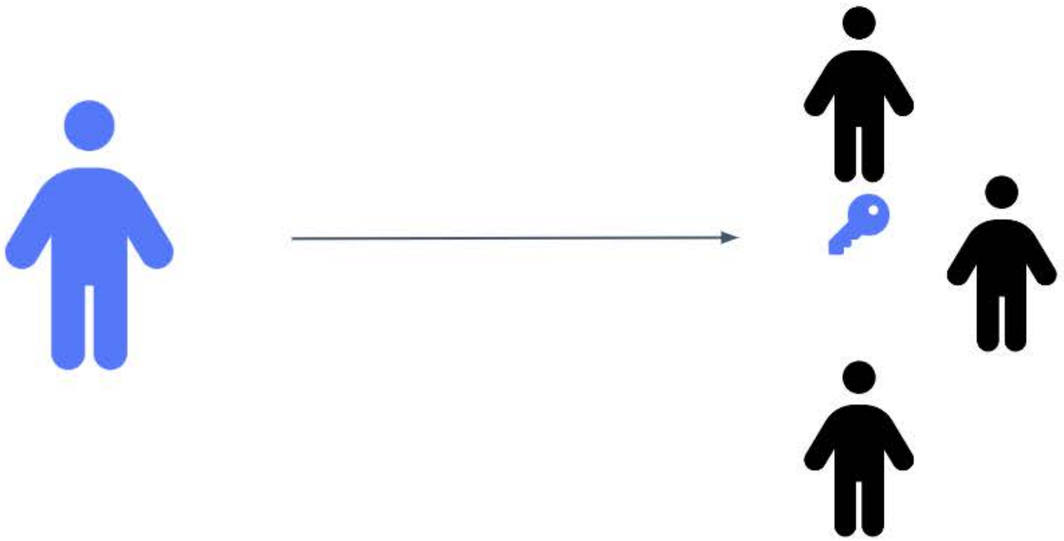
2. Public Key Authentication

- **Bob** provides his **public key**
- **Bob** secures his **private key**
- **Bob** signs a message with his **private key**
- Anyone with **Bob's public key** can verify the message was signed by Bob



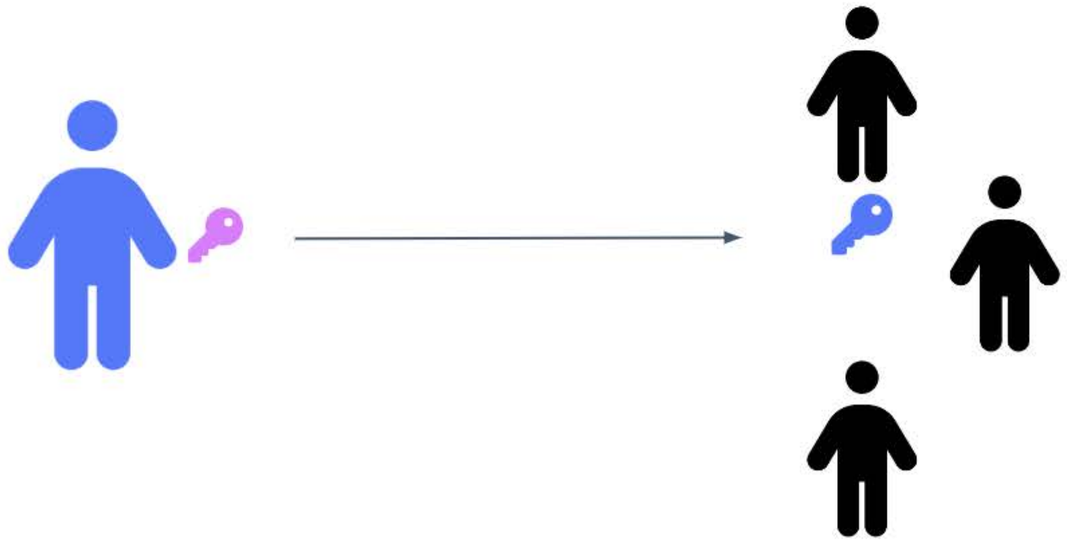
2. Public Key Authentication

- **Bob** provides his **public key**
- **Bob** secures his **private key**
- **Bob** signs a message with his **private key**
- Anyone with **Bob's public key** can verify the message was signed by Bob



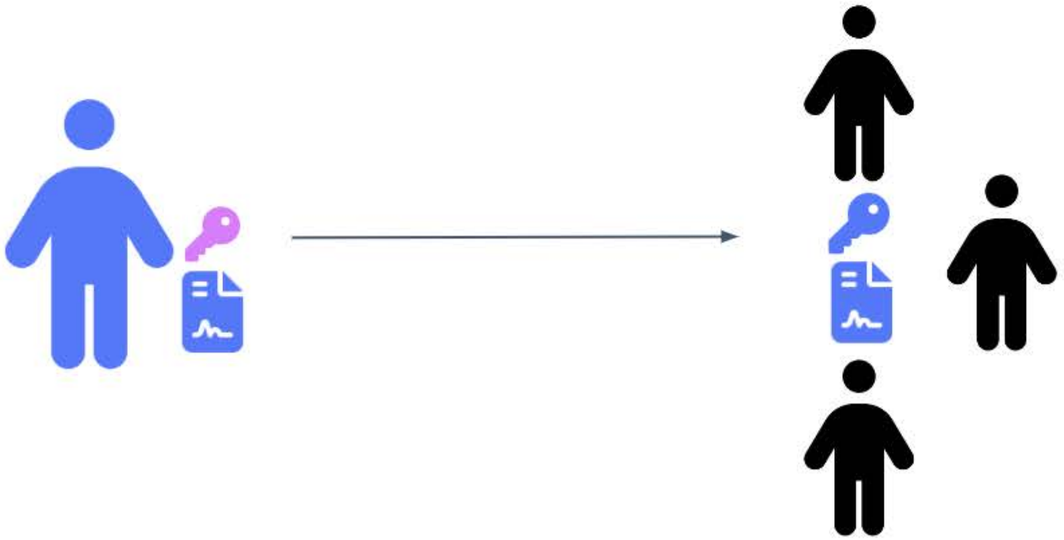
2. Public Key Authentication

- **Bob** provides his **public key**
- **Bob** secures his **private key**
- **Bob** signs a message with his **private key**
- Anyone with **Bob's public key** can verify the message was signed by Bob



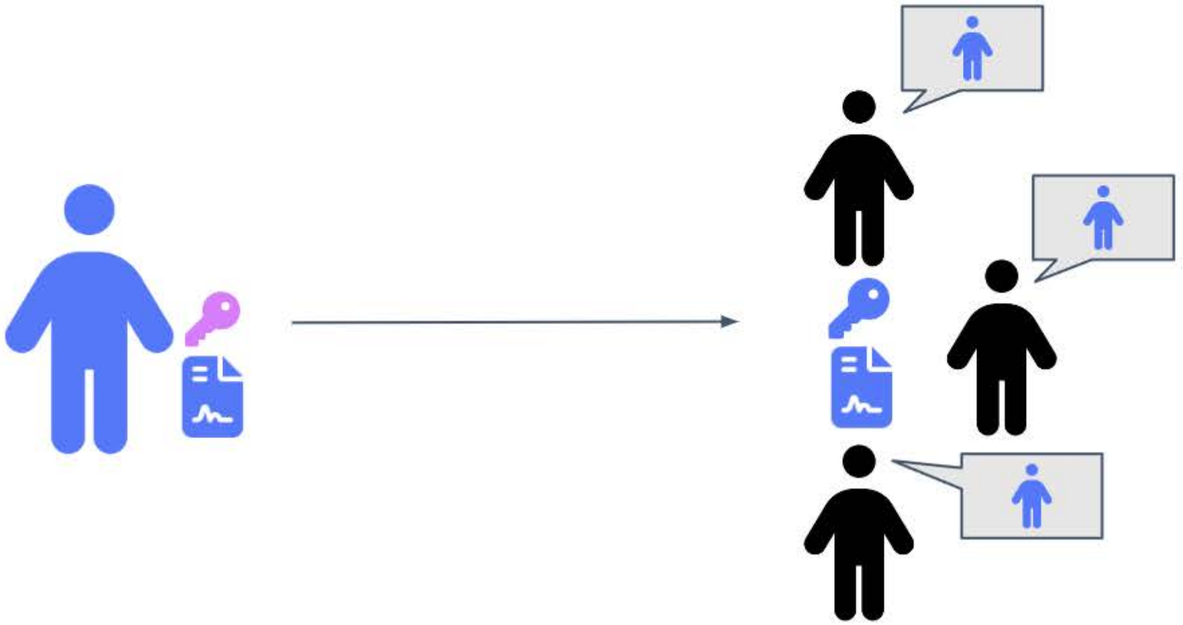
2. Public Key Authentication

- **Bob** provides his **public key**
- **Bob** secures his **private key**
- **Bob** signs a message with his **private key**
- Anyone with **Bob's public key** can verify the message was signed by Bob



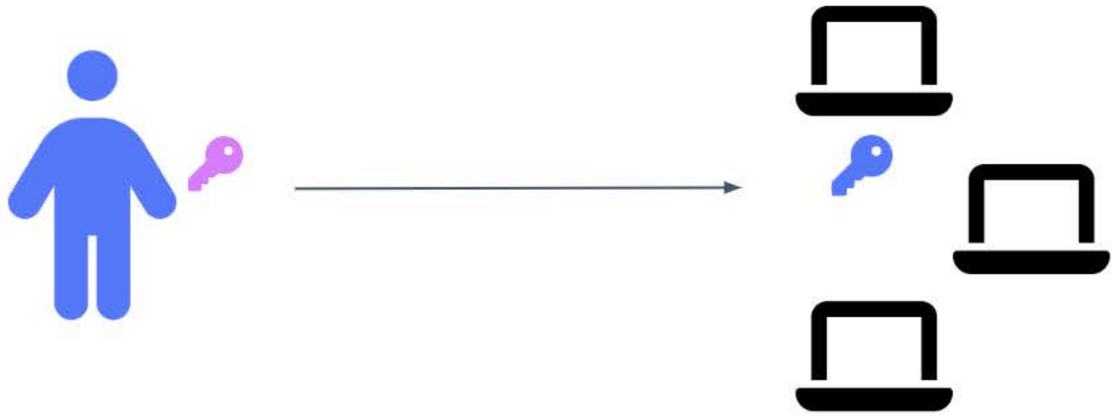
2. Public Key Authentication

- **Bob** provides his **public key**
- **Bob** secures his **private key**
- **Bob** signs a message with his **private key**
- Anyone with **Bob's public key** can verify the message was signed by Bob



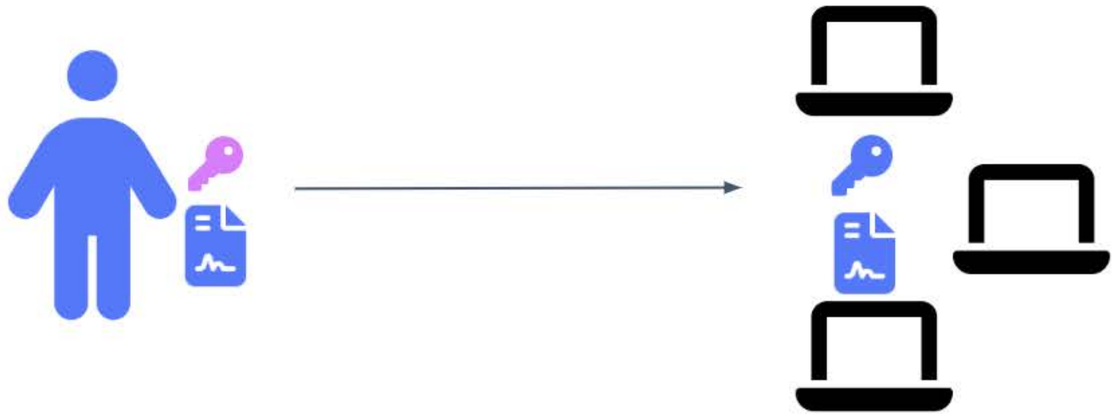
Public Key Cryptography in Web3 🔑🔑

- User signs a transaction with their **private key** 🔑
- User broadcasts the transaction to the blockchain 🌐
- Blockchain nodes recover **public key** from the signature from which the user's address is derived



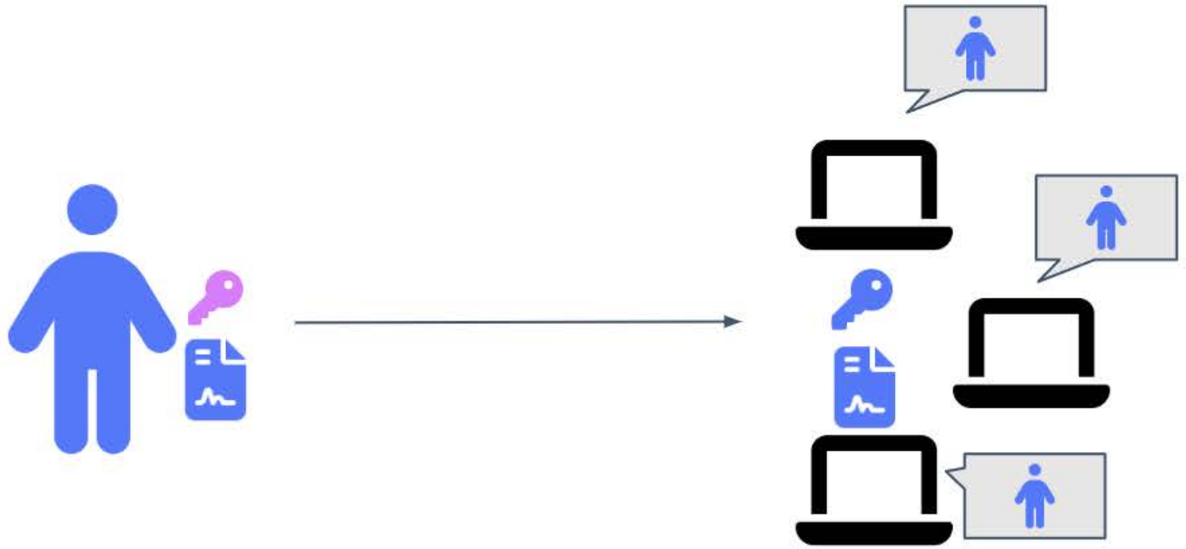
Public Key Cryptography in Web3 🔑🔑

- User signs a transaction with their **private key** 🔑
- User broadcasts the transaction to the blockchain 🌐
- Blockchain nodes recover **public key** from the signature from which the user's address is derived



Public Key Cryptography in Web3 🔑🔑

- User signs a transaction with their **private key** 🔑
- User broadcasts the transaction to the blockchain 🌐
- Blockchain nodes recover **public key** from the signature from which the user's address is derived



Next:

Crypto Time