In US as you login you can just pay you don’t have to give any otp

Some websites like coindcx, wazirx, coinbase, backpack exchange etc

Metamask wallet

Famous phrase **NOT YOUR KEYS NOT YOUR CRYPTO** LOL

How banks do Auth

In traditional banks, you have a username and password that are enough for you to

1. Look at your funds
2. Transfer funds
3. Look at your existing transactions

How Blockchains do auth

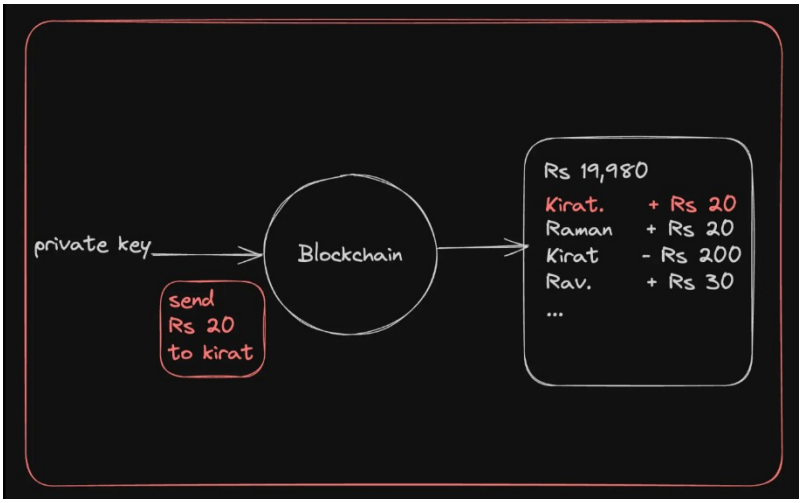
If you ever want to create an account on a blockchain, you need to generate a public-private keypair.

Public private Keypair

A public-private key pair is a set of two keys used in asymmetric cryptography. These two keys have the following characteristics:

Public Key: The public key is a string that can be shared openly.

Private key: The private key is a secret string that must be kept confidential.



Bits and bytes

What is a Bit?

A bit is the smallest unit of data in a computer and can have one of two values: 0 or 1.

Think of a bit like a light switch that can be either off (0) or on (1).

What is a byte?

A byte is a group of 8 bits. It’s the standard unit of data used to represent a single character in memory. Since each bit can be either 0 or 1, a byte can have 2^8 (256) possible values, ranging from 0 to 255

UInt8Array

A better way to represent an array of bytes is to use a UInt8Array in JS

let bytes = new Uint8Array([0, 255, 127, 128]);

console.log(bytes)

Why use UInt8Array over native arrays ?

- They use less space. Every number takes 64 bits (8 bytes). But every value in a UInt8Array takes 1 byte.
- UInt8Array Enforces constraints - It makes sure every element doesn’t exceed 255.

Encodings

Bytes are cool but highly unreadable. Imagine telling someone

Hey, my name is 00101011101010101020

It’s easier to encode data so it is more human readable . Some common encodings include -

1. Ascii
2. Hex

3. Base64
4. Base58

Ascii

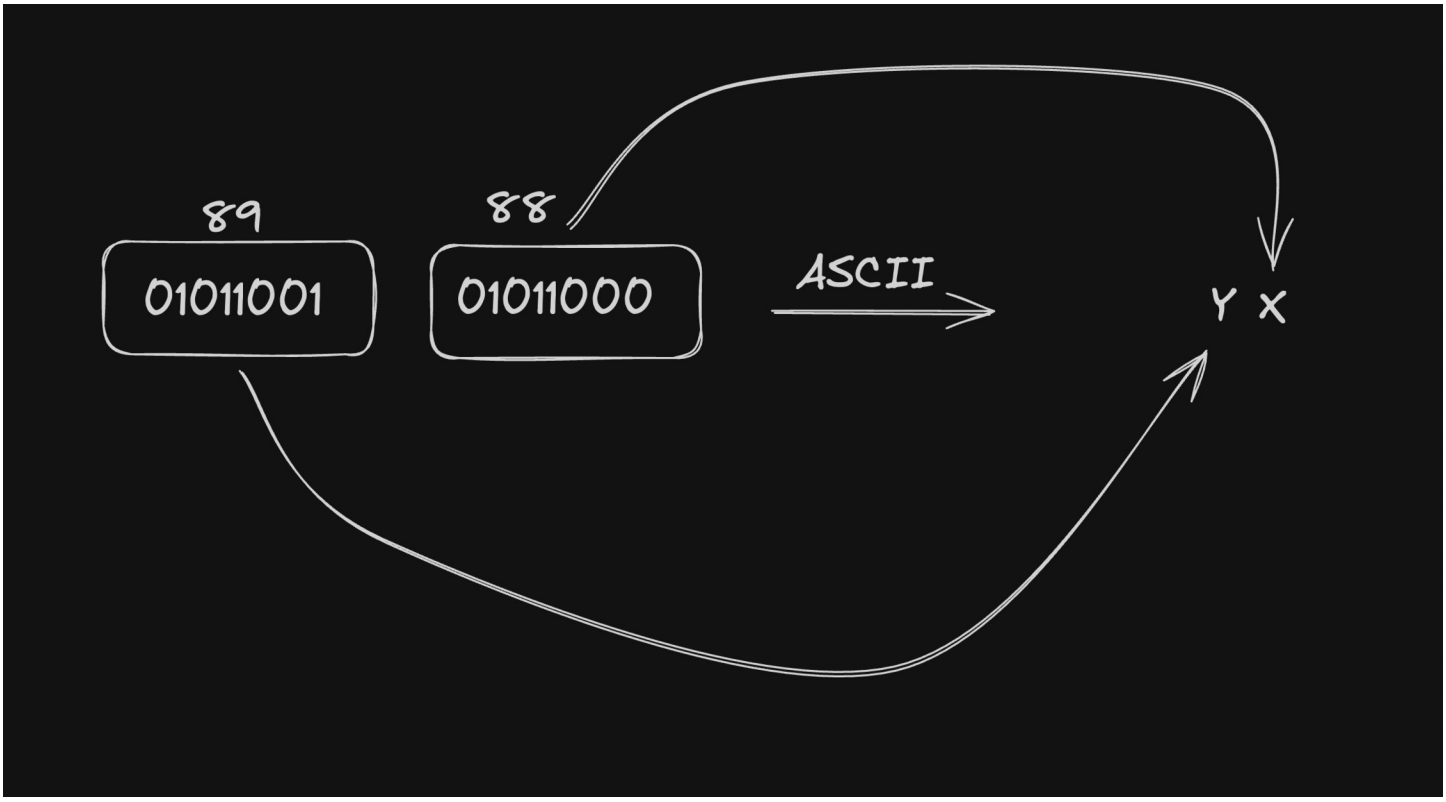
1 character = 7 bits

Bytes to Ascii

Ascii to bytes

UInt8Array to ascii

Ascii to UInt8Array

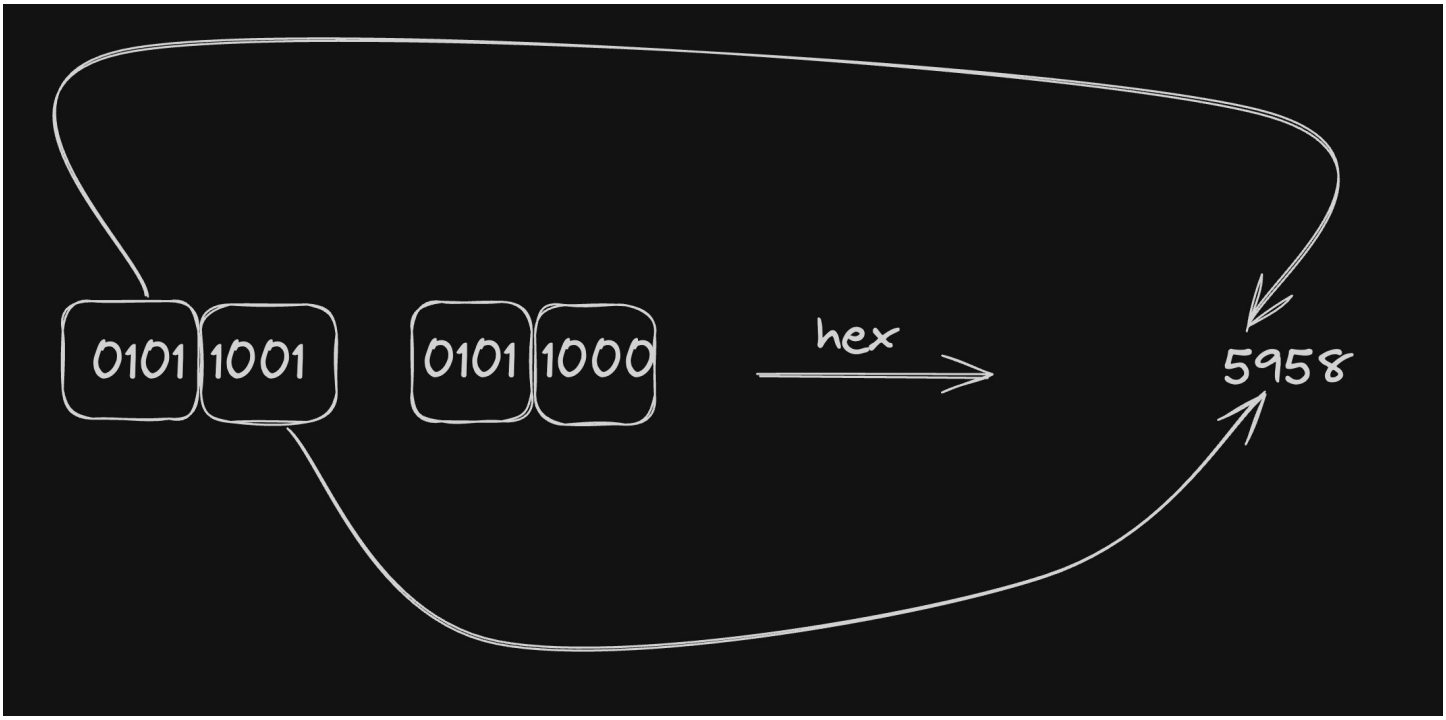


Hex

1 character = 4 bits

A single hex character can be any of the 16 possible values: 0-9 and A-F.

Array to hexHex to array



Base64

1 character = 6 bits

Base64 encoding uses 64 different characters (A-Z, a-z, 0-9, +, /), which means each character can represent one of 64 possible values.

https://www.base64encode.org/

https://www.base64decode.org/

Encode

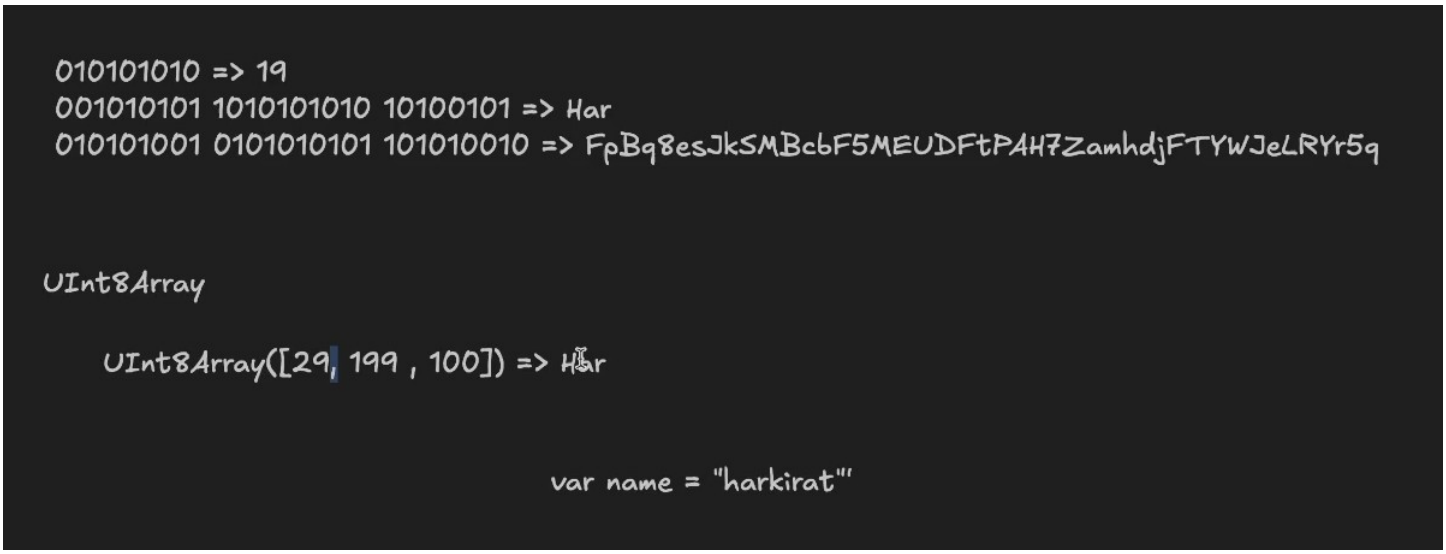
Ascii vs UTF-8

- ASCII uses a 7-bit encoding scheme.
- UTF-8 uses 1 to 4 bytes to encode each character.

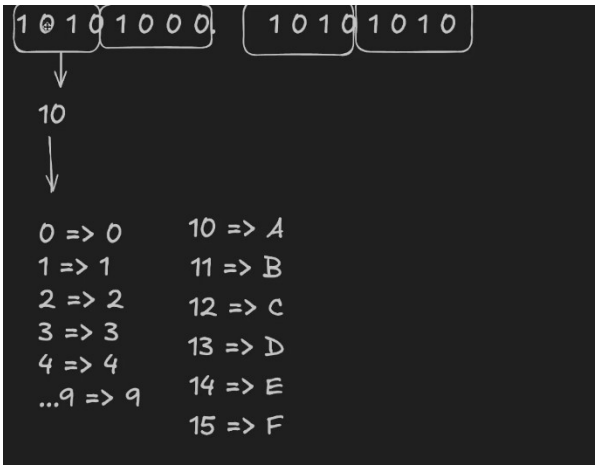
MY LEARNINGS

Uint8array is more efficient in storing data and all

It store the bytes in the form of array

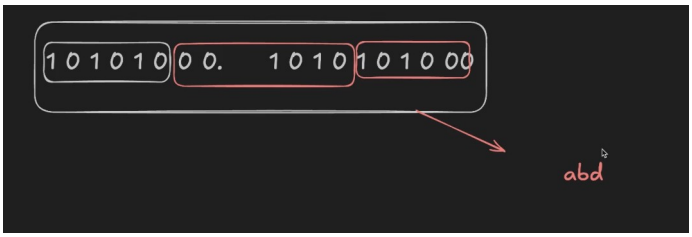


HEX



the final string will be A8AA

BASE64



In this we take like 6 bits at a time to get a answer

Base58

It is similar to Base64 but uses a different set of characters to avoid visually similar characters and to make the encoded output more user-friendly

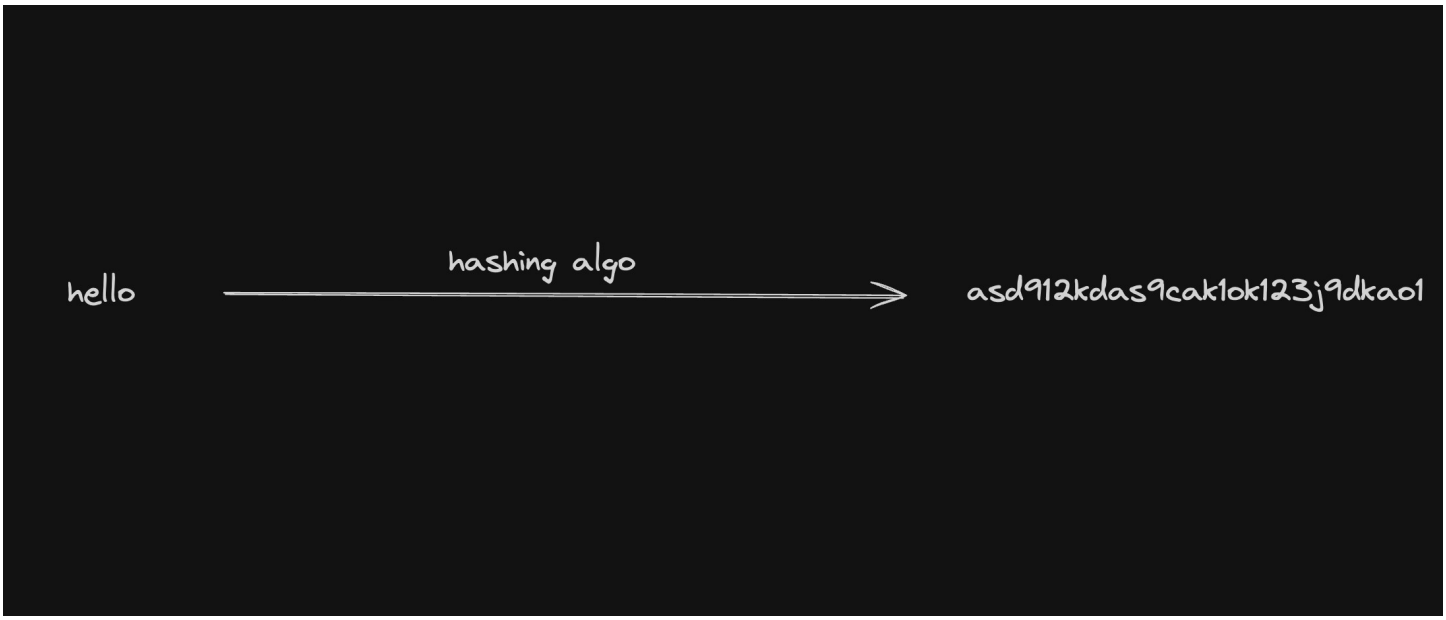
Base58 uses 58 different characters:

- Uppercase letters: A-Z (excluding I and O)
- Lowercase letters: a-z (excluding l)
- Numbers: 1-9 (excluding 0)

- + , /

HASHING

In hashing if you hash any word you cannot get back the original word that’s the power eg: sha256, MD5



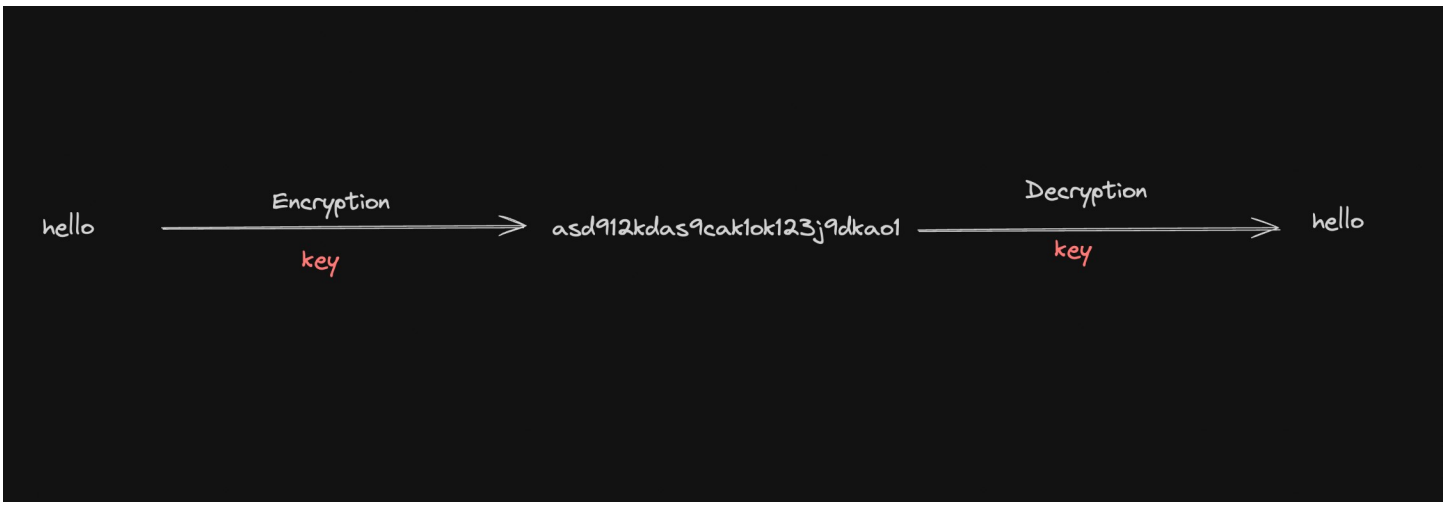
ENCRYPTION

Encryption is the process of converting plaintext data into an unreadable format, called ciphertext, using a specific algorithm and a key. The data can be decrypted back to its original form only with the appropriate key.

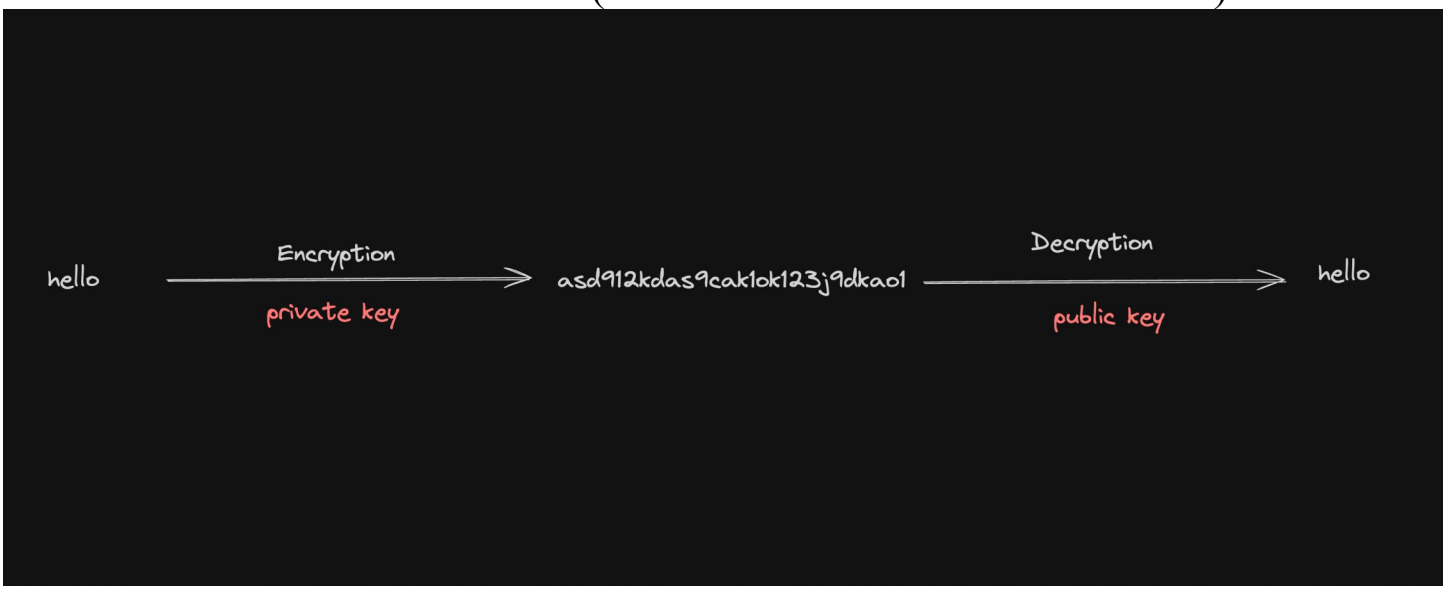
Key Characteristics:

- **Reversible**: With the correct key, the ciphertext can be decrypted back to plaintext.
- **Key-dependent**: The security of encryption relies on the secrecy of the key.
- **Two main types**:
    - **Symmetric encryption**: The same key is used for both encryption and decryption.
    - **Asymmetric encryption**: Different keys are used for encryption (public key) and decryption (private key).

SYMMETRIC ENCRYPTION



ASSYMETTRIC ENCRYPTION (IN BLOCKCHAIN WE USE THIS)



EG: RSA,EDDCSA

THIS IS HOW WE SEND THE DATA IN A BLOCKCHAIN FIRSTLY WE SIGN THAT WE ARE ONLY SEND THE ETH AND THEN THE ETH IS SEND TO THE ANOTHER USER

THE BLOCKCHAIN LATER VERIFIES THAT THE ETH SEND BUT THAT USER ONLY BY USING YOUR PUBLIC KEY

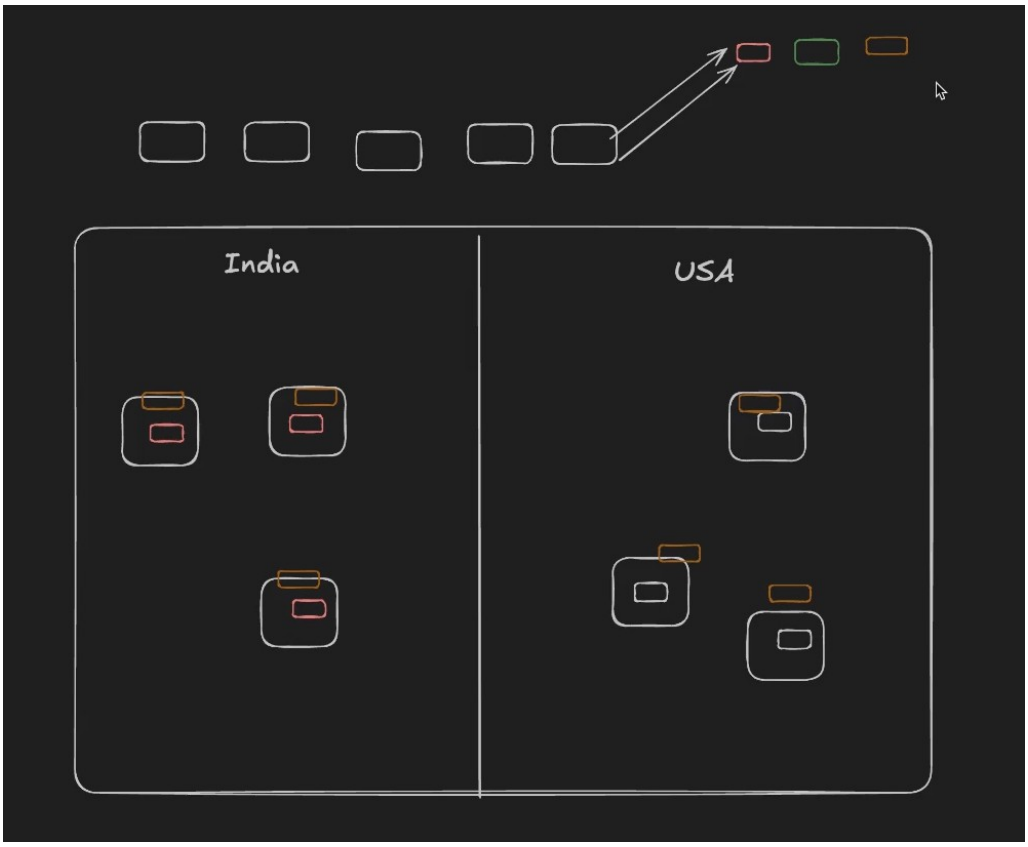
EVERY BLOCK ON THE BLOCKCHAIN HAS A BUNCH OF TRANSACTIONS

EVERY TRANSACTION HAS A SIGNATURE AND THE SIGNATURE IS DONE BY THE PERSON

AND WE CAN CHECK THAT THE TRANSACTION WAS DONE BY THE USER ONLY BY CHECKING IT USING THE PUBLIC KEY

THIS SHOWS HOW TO CREATE A WALLET

https://projects.100xdevs.com/tracks/public-private-keys/Public-Key-Cryptography-9



when the mining is like same again and again eventually it will led to a one chain only

ITS TIME TO LEARN THIS ALL AND REMEMBER

A single character can only be converted into bytes

Uint make sure that the number should be a 8 bit array and if its more than 8 bits it will ignore all those extra bytes

🧠 Core Differences

Feature	Normal Array	Uint8Array
Data Type	Any type	Only numbers (0–255)
Memory Layout	Not contiguous	Contiguous block
Size	Dynamic	Fixed
Performance	Slower for binary	Faster for binary
Use Case	General purpose	Binary data, buffers
Underlying System	High-level JS	Based on `ArrayBuffer`

To convert a character into bits you can use new textEncoder().encode(str)

Encoding its like how are we accessing the code like

0011110011001010

In this you can access it 8(UTF 8) at a time, or 4(Hex) at a time due to which your output will change this is called as encoding

ASCII is like one encoding which has numbers for all the letters and special symbols also

Hexadecimal is like 0-9 its same but at 1010 it will become A it uses 8421 technique

toString(format) converts the number into the given format you want such as 8,16 etc

padStart(2,'0') this means when its converting like any number if that number just has one digit it will add a 0 to start

base64 – uses 6 characters at a time

base58 – makes you life easier by removing the confusing characters

How to transactions work on the blockchain?

User side

1. User first creates a public/private keypair
2. They create a transaction that they want to do (send Rs 50 to Alice). The transaction includes all necessary details like the recipient's address, the amount and some blockchain specific parameters (for eg - latestBlockHash in case of solana)
3. They hash the transaction
4. They sign the transaction using their private key
5. They send the raw transaction , signature and their public key to a node on the blockchain.

Miner

1. Hashes the original message to generate a hash
2. Verifies the signature using the users public key and the hash generated in step 1
3. Transaction validation - The miner/validator checks additional aspects of the transaction, such as ensuring the user has sufficient funds
4. If everything checks out, adds the transaction to the block

How does the transaction goes?

1. Firstly you write a text to whom you want to send the money and how much
2. Then hash it like using sha256
3. Then you sign it

4. Then your signed message and your public key and the original message is submitted on the blockchain and then the miner verifies was this transaction indeed submitted by u?
5. If yes than the miner will introduce your transaction on the block and will start to mine to find the nonce

Simplified version

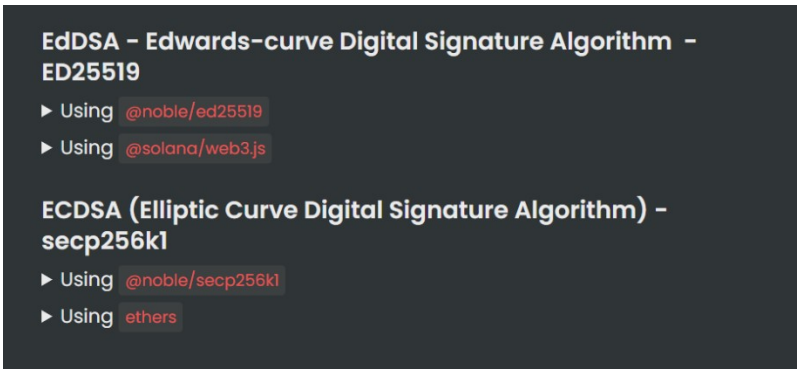
If someone makes a transaction, it must be verified and included in a block by a miner. The miner collects valid transactions and tries different nonce values to solve the block puzzle. The first miner to solve it adds the block and gets rewarded.

```
import { Keypair } from "@solana/web3.js";
import nacl from "tweetnacl";

// Generate a new keypair
const keypair = Keypair.generate(); // static function

// Extract the public and private keys
const publicKey = keypair.publicKey.toBase58();
const secretKey = keypair.secretKey;

// Display the keys
console.log("Public Key:", publicKey);
console.log("Private Key (Secret Key):", secretKey);

// Convert the message "hello world" to a Uint8Array
const message = new TextEncoder().encode("hello world");

const signature = nacl.sign.detached(message, secretKey);
const result = nacl.sign.detached.verify(
  message,
  signature,
  keypair.publicKey.toBytes(),
);
```

enough to make a public private key pair and sign it

nacl is a very famous cryptographic library

This is the 5 steps to be done for the generate solana public private key pair same we can do for eth, bitcoin



Four ways

Overall the meathods is same just the name is different

To make multiple wallets we need to understand HD wallets:

Hierarchical Deterministic (HD) Wallet

Hierarchical Deterministic (HD) wallets are a type of wallet that can generate a tree of key pairs from a single seed. This allows for the generation of multiple addresses from a single root seed, providing both security and convenience.

this single seed or seed phrase is very imp

this is like a recovery key without this if your laptop dies you are done then

Problem

You have to maintain/store multiple public private keys if you want to have multiple wallets.

Solution - BIP-32

Bitcoin Improvement Proposal 32 (BIP-32) provided the solution to this problem in 2012. It was proposed by Pieter Wuilla, a Bitcoin Core developer, to simplify the recovery process of crypto wallets. BIP-32 introduced a hierarchical tree-like

structure for wallets that allowed you to manage multiple accounts much more easily than was previously possible. It’s essentially a standardized way to derive private and public keys from a master seed.

THE MOST IMP THING IS THE SECRET RECOVERY PHASE (SEED) AND FROM WHICH THE IS ACCOUNTS ARE CREATED

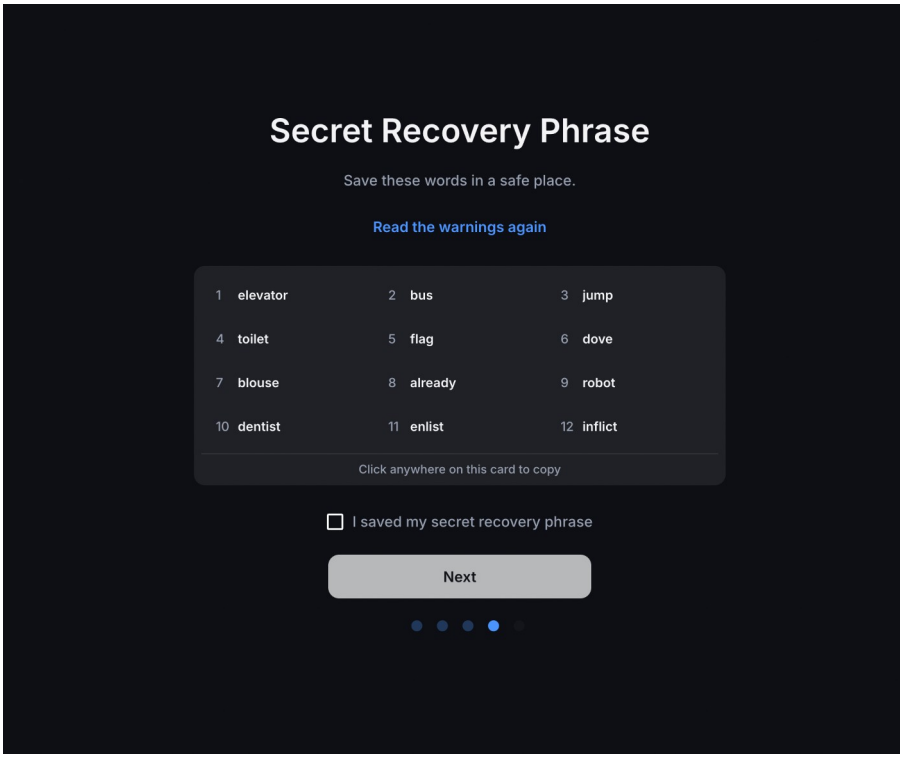
WHAT IS MNEONOMIC?

Mnemonics

A mnemonic phrase (or seed phrase) is a human-readable string of words used to generate a cryptographic seed. BIP-39 (Bitcoin Improvement Proposal 39) defines how mnemonic phrases are generated and converted into a seed.

Ref - https://github.com/bitcoin/bips/blob/master/bip-0039/english.txt

Where this is done in Backpack - https://github.com/coral-xyz/backpack/blob/master/packages/app-extension/src/components/common/Account/MnemonicInput.tsx#L143



SET OF WORDS

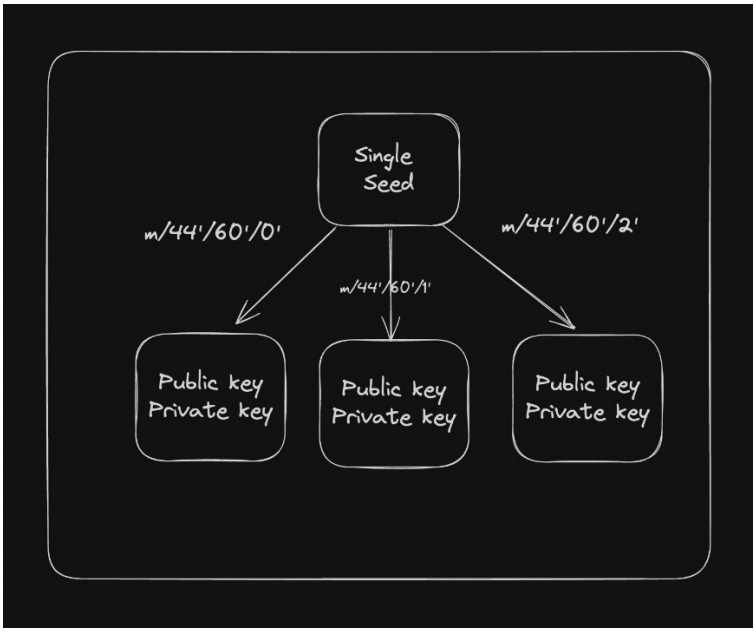
FROM THIS THE SEED IS CREATED AND FROM THAT SEEDS THE ACCOUNTS ARE CREATED

index.ts

```ts
import nacl from "tweetnacl";
import { mnemonicToSeedSync } from "bip39";
import { derivePath } from "ed25519-hd-key";
import { Keypair } from "@solana/web3.js";

const mnemonic =
  "destroy retire candy cotton asset field wheel gate auction borrow join predict";

const seed = mnemonicToSeedSync(mnemonic);

console.log(seed);

for (let i = 0; i < 4; i++) {
  const path = `m/44'/501'/${i}'/0'`; // This is the derivation path
  const derivedSeed = derivePath(path, seed.toString("hex")).key;
  const secret = nacl.sign.keyPair.fromSeed(derivedSeed).secretKey;
  console.log(Keypair.fromSecretKey(secret).publicKey.toBase58());
}
```

```
bigint: Failed to load bindings, pure JS will be used (try npm run rebuild?)
<Buffer 65 80 bf 47 3b 59 b8 f7 4b ad 0b 26 19 72 fb 28 ec 1c 3a 24 56 33 58 a1 77 3b fc c6 0d b3 6d ec ec aa f2 23 23 e5 c9 be 0d 92 28 10 76 e2 76 4e d9 2c ... 14 more bytes>
6BaaWY2NejwQeAq27jc4h7GtuNKqvEHk5eBJnfUZH7Q1
88UoN5QRLcmLxmYqAdwJNtMcniDfZGNN77DWaZZqPU2d
77nyDHKi8GS1DdjhMV6giP6hC1QgseGJtMeHBfFnUfZP
8YFgXmEHtx3JrvQebF8MiXby5zdEBxw5pCBJxyH6Khfx
```

METAMASK ALSO



1. **you have a seed phrase which contain many words out of a mnemonic**
2. **From the seed phrase the wallet is created**
3. **A account is created which contains a public and private key**

Now when you add more wallets it uses a derivation path to create it

which is this m/44’/60’/0’ and its just one letter change for each wallet

- **m**: Refers to the master node, or the root of the HD wallet.
- **purpose**: A constant that defines the purpose of the wallet (e.g., 44' for BIP44, which is a standard for HD wallets).
- **coin_type**: Indicates the type of cryptocurrency (e.g., 0' for Bitcoin, 60' for Ethereum, 501' for solana).
- **account**: Specifies the account number (e.g., 0' for the first account).
- **change**: This is either 0 or 1, where 0 typically represents external addresses (receiving addresses), and 1 represents internal addresses (change addresses).
- **address_index**: A sequential index to generate multiple addresses under the same account and change path.

Now the public key is generated using elliptical cryptography(way to generate public and private keys)

Asymmetric encryption can be done using many algos

Eth and bitcoin uses ecdsa which uses elliptical curve cryptography

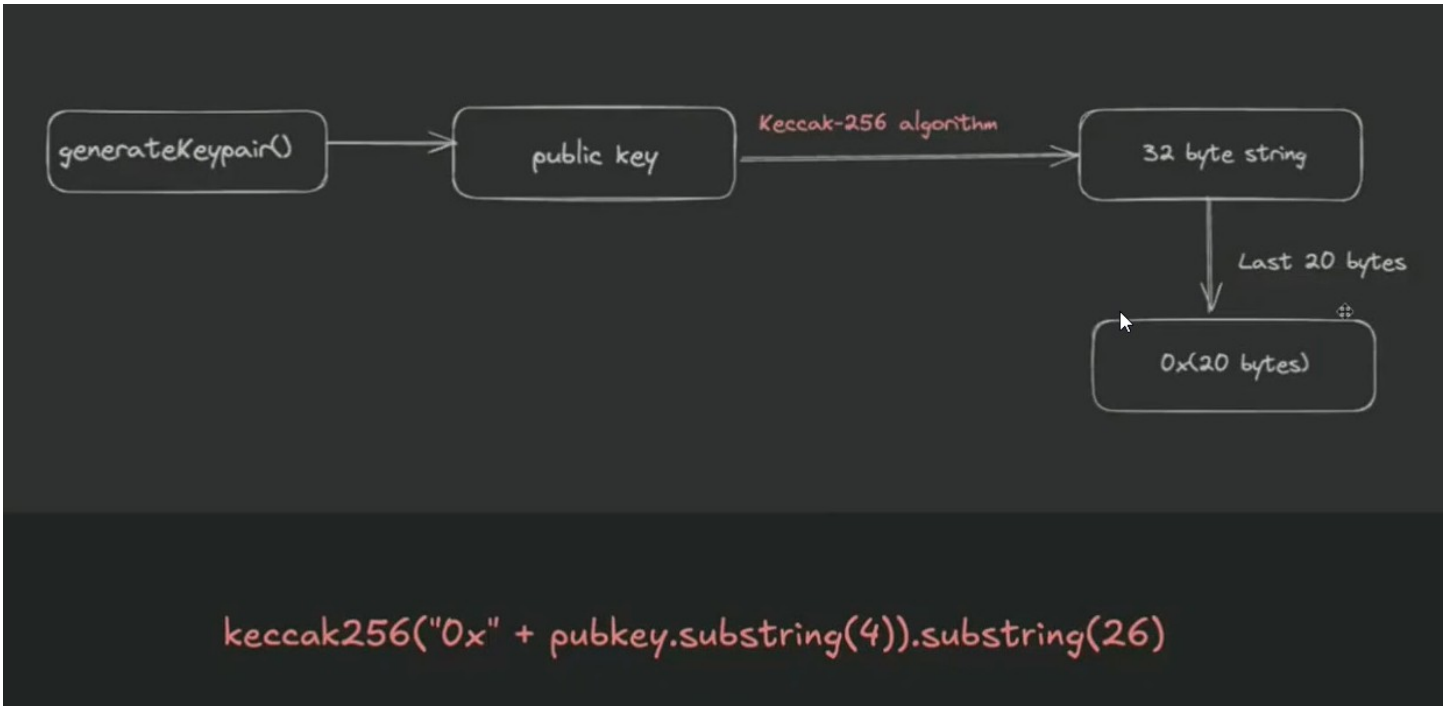
Solana uses eddsa

These also prevents like if we know the public key we wont be able to find the private key

For solana its just like 32 bytes just converted using base58

Eth have some other method to do this

For etherium its done like this




```
keccak256("0x" + pubkey.substring(4)).substring(26)
```