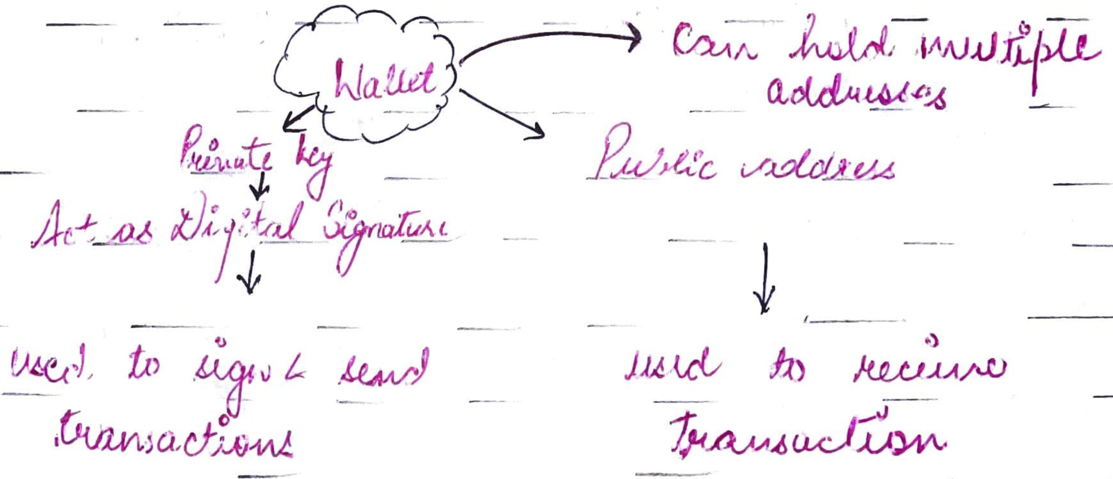
Wallets, Digital Signatures and Protocols

Wallets :→

- A wallet lets users generate the private key and public address.
- The private key is used to send the transaction and public address is used to receive the transacti
- Similar to a digital wallet that allows you to keep cryptocurrencies.

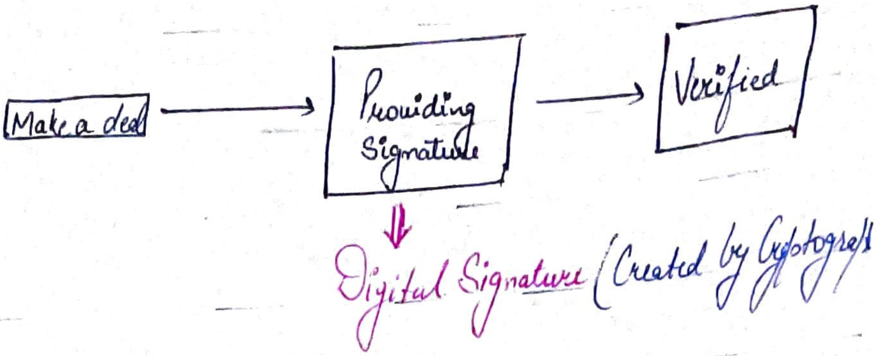
Types of Wallets :→

→ Papper Wallets (Stored on paper)
→ Web Wallets (only present on internet)
→ Mobile Wallets (application on mobile)
→ Desktop Wallets (software on deskstop like Bitcoin)
→ Hardware Wallets (Hardware device storing privatekey)
→ Physical Wallets (Smart Card)


Wallet
Can hold multiple addresses
Private key
Act as Digital Signature
used to sign & send transactions
Public address
used to receive transaction

Digital Signatures :→

Traditional Method :



Make a deal → Providing Signature → Verified

⇓

Digital Signature (Created by Cryptograph

- Digital Signature use cryptography which is most secure.
- The private key is used to sign messages digitally.
- Every transaction is signed by sender using PR key.
- Example :→ SSL

Protocols :→

- Behaviour of a Blockchain depend on it.
- These define the blockchain and its funtionality.

Benefits of Blockchain :

Traditional Method	Blockchain
→ There is single Server (Could be easily attacked)	There are multiple Server. (Very difficult to get hacked)
→ Centralised Control	Decentralised Control
→ Expensive for Security	No extra Security needed
→ Only one body can Verify the changes.	Every user can append over blockchain
→ Not transparent	T Full Transparency
→ Not as confidential	Confidential
→ Slow process (Banking)	Very fast process (Banking)

⊙ Key :→

⊙ It is a string of alphanumeric characters which us to secure our data by encrypting it

Key Generation Algorithm

↙ Symmetric

⊙ Generates only one key and shared by users to encrypte and decrypt the data

⊙ faster than the other

⊙ Difficult to distribute initially.

⊙ Example:→ →AES Algorithm [TLS Service]

→ RC4 Algorithm [Wireless Encryption]

↘ Asymmetric

⊙ A pair of key is generated

→ Private key to encrypt the data while sending it.

→ Public key to decrypt the data and view it.

⊙ Slower as compared to Symmetric

* Private key $\xrightarrow{\text{derives}}$ Public key

Private key ⟵ Public key [NOT POSSIBLE]

⊙ Example ⟶ RSA Algorithm [Access of Servers in cloud]

⟶ ECC Algorithm [Blockchain]

* WALLET IMPORT :→

Used in ECC Algorithm

→ It is a way of encoding private ECDSA key so as to make it easier to copy.

* PRIVATE KEY :→

⊙ generate a signature for each transaction

⊙ Signature is used to confirm transaction that has come from a specific user

⊙ If someone has your private key they can send our currency to themselves.

Example:→ L34ExRFCux QCootE Qe 8 . . . .

PUBLIC KEY :
- Derived from the Private key.
- Can be distributed to Everyone
- 256 Bits long, hash - 160 Bits (Wallet address)

ADDRESSES :→
- representation of Public key.
- One-Way cryptographic hash functions are used to derive it from public key.
- Example:→ the algorithms that are used to generate bitcoin address from public key are Secure Hash Algorithm (SHA-256) and Race Integrity Primitives Evaluation Message Digest 160 [RIPEMD-160].

TRANSACTION:
- Record of data in chronological order
- Stored in Merkle tree.

BLOCKS :
- is a container of data in a blockchain
- Each block has hash of previous block.
- In blockchain there is block is created in a certain time. (BTC → 10mins)
- BLOCK HEADER + NONCE → TWICE SHA 256 = Block Hash

NONCE ↓ Value ko guess krna hotah

→ if guess correctly block is mined ↓ MINING

STRUCTURE Of BLOCK :→

- BLOCK HEADER :→ → contains Metadata information
  ↓
  - Time stamp
  - Protocol information
  - Nonce
  - Difficulty
  - Previous Block hash

⊙Block Identifier → are the hash taken for the transaction, hash of the previous block is used in the construction of the the next block

→ Another way to identify the block is by height (the countery given to its position.

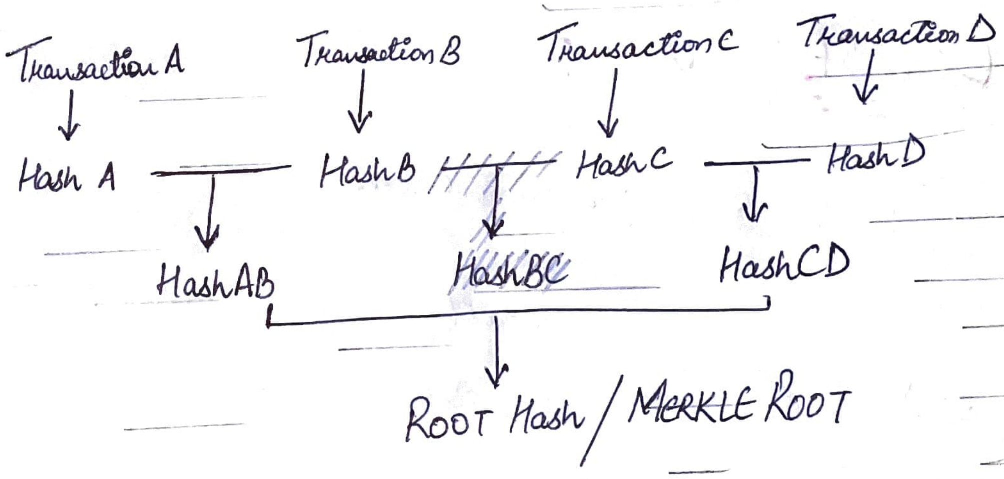
MERKLE TREE → summarizes all transactions in a block (like a digital foot print)

* Merkle trees are created by repeatedly hashing pair of nodes until there is only one hash left which is called the root (hash)

* Each leaf node is a hash of transactional data and each non-leaf node is a hash of its previous hashes.

* Merkle trees are binary and they require even no. of leafs.

* Any change in the transaction → changes the Merkel Root.


Transaction A
Transaction B
Transaction C
Transaction D
Hash A
Hash B
Hash C
Hash D
HashAB
HashBC
HashCD
ROOT Hash / MERKLE ROOT