Week 3 - Creating a web based wallet, RPCs [16 Aug 2024]

> **Timing** - Every Friday 8:00pm to 10:00pm IST
> Class Slides - Creating a web based wallet, RPCs
> Class Video - Take your development skills from 0 to 100 and join the 100xdevs community
> Assignment - Creating a web based wallet

> **Syllabus** - Notion – The all-in-one workspace for your notes, tasks, wikis, and databases. (100xdevs.com)

▼ Topics Covered So Far

- **Hashing:** SHA-256
- **Encryption:** EDDSA, ECDSA, Public and Private Keys
- **HD Wallets:** Hierarchical Deterministic Wallets

▼ Cohort Student’s assignment submissions [Examples]

- Some participants have built a web-based wallet.
- **Example Web Wallets:**
  - Wallet-Kosh
  - Creepy Wallet
  - GitHub: ETH Wallet Generator

Keccak-256

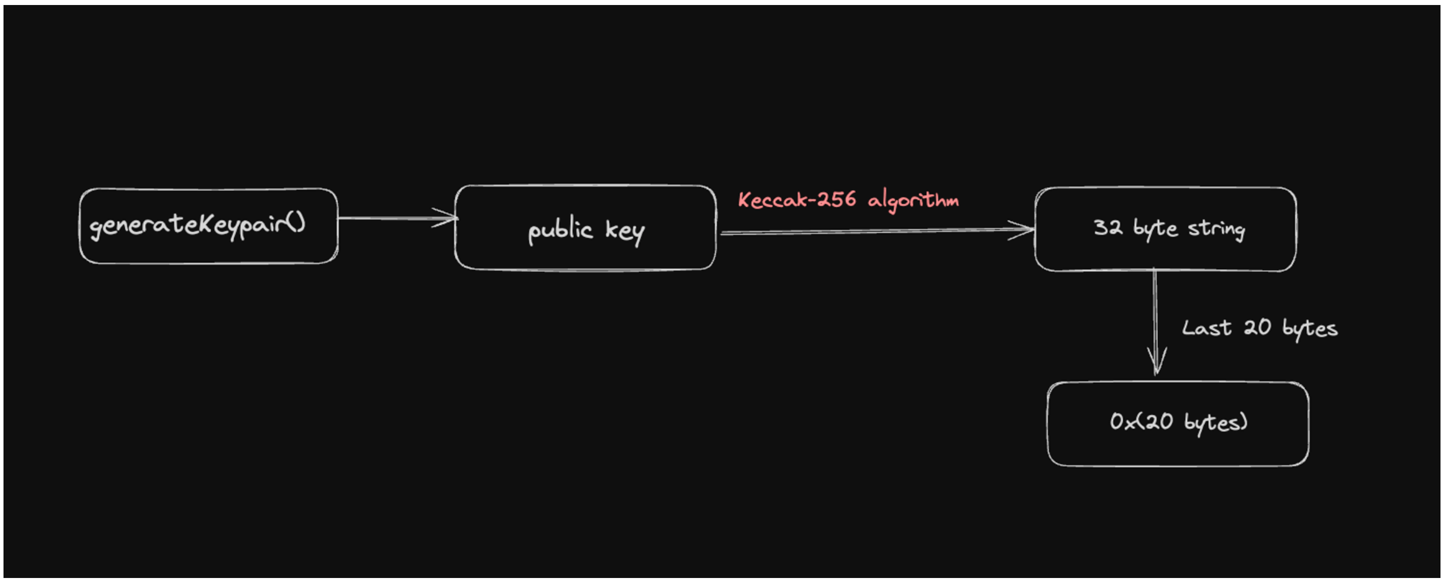
- **Keccak-256** is a cryptographic hash function used extensively in Ethereum.
- **Try it out here:** Keccak-256 Online Tool

Properties

- **Collision Resistance:** Designed to make it extremely difficult to find two inputs that produce the same hash. While collisions are theoretically possible, they are extremely unlikely.

- **Pre-Image Resistance:** Nearly impossible to reverse the hash to find the original input, although brute-force attacks are still possible.
- **Key Length:** Outputs a 256-bit hash value, which makes brute-force attacks difficult.
- **Implementation:** Ensure the Keccak-256 implementation is secure and up to date.

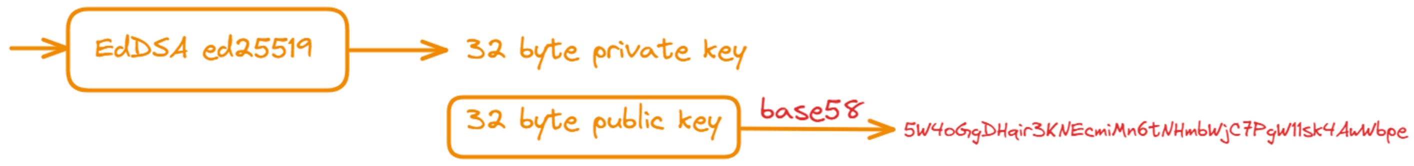
Ethereum (ETH)



Ethereum Addresses

- **Public Address:** 20 bytes (e.g., `0x8BCd4591F46e809B15A490F5C6eD031FDDE0bee0`)
- **Generation Process:**
    1. Generate a public key using elliptic curve cryptography.
    2. Hash the public key using Keccak-256.
    3. Take the last 20 bytes of the hash.
    4. Convert to hexadecimal and prefix with '0x'.
- **References:**
    - Backpack Implementation
    - Ethers.js Implementation

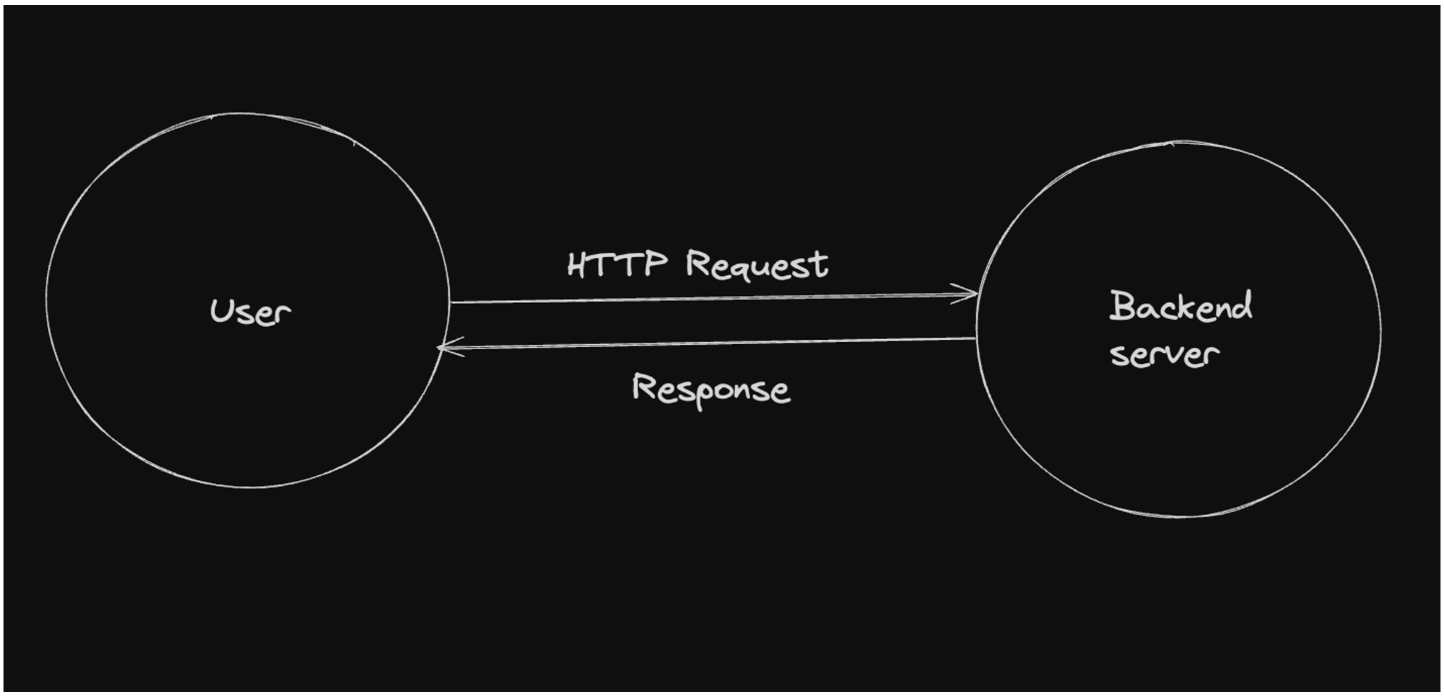
Solana (SOL)



Solana Addresses

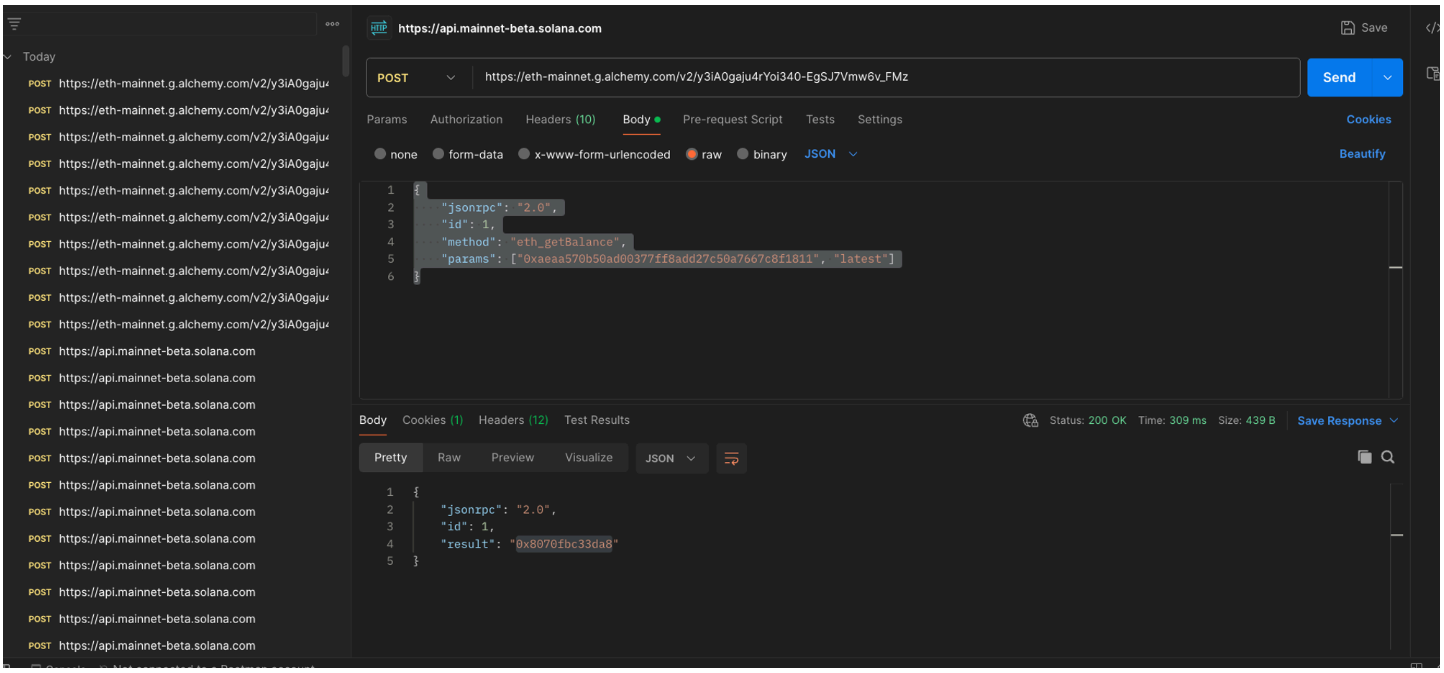
- **Public Keys:** 32 bytes (e.g., `5W4oGgDHqir3KNEcmiMn6tNHmbWjC7PgW11sk4AwWbpe`)
- **Note:** Unlike Ethereum, Solana addresses do not require hashing/chopping.

Frontend vs Backend



- **Backend Servers:** Run your backend logic.
- **Frontend:** Interacts with backend servers via HTTP requests.
- **Example of a traditional backend request:** JSONPlaceholder API

- **Postman:** Allows sending requests to backend servers without using a browser.



POST https://eth-mainnet.g.alchemy.com/v2/y3iA0gaju4rYoi340-EgSJ7Vmw6v_FMz

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "eth_getBalance",
    "params": ["0xaeaa570b50ad00377ff8add27c50a7667c8f1811", "latest"]
}
```

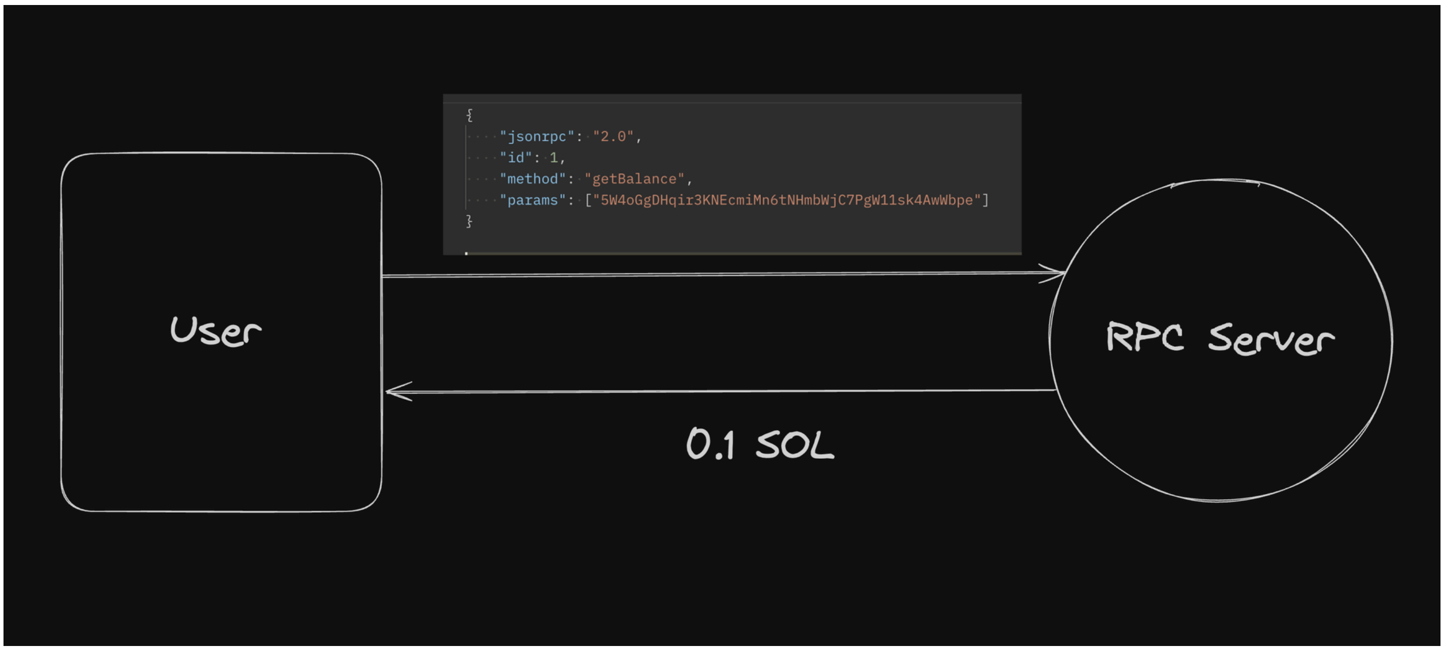
Status: 200 OK Time: 309 ms Size: 439 B

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "result": "0x8070fbc33da8"
}
```

Something similar happens on block explorers as well. When you query an address it sends a request on blockchain network.

1. Solscan
2. Ethereum (ETH) Blockchain Explorer (etherscan.io)

RPC & JSON-RPC



```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "getBalance",
    "params": ["5W4oGgDHqir3KNEcmiMn6tNHmbWjC7PgW11sk4AwWbpe"]
}
```

- **JSON-RPC:** A remote procedure call (RPC) protocol encoded in JSON. Allows client-server communication over a network.
- **Use Cases:**
    - Sending transactions to the blockchain.

- Fetching blockchain data (e.g., balances).

💡 An RPC (Remote Procedure Call) server is not inherently part of the blockchain network itself, nor does it participate in staking or consensus mechanisms

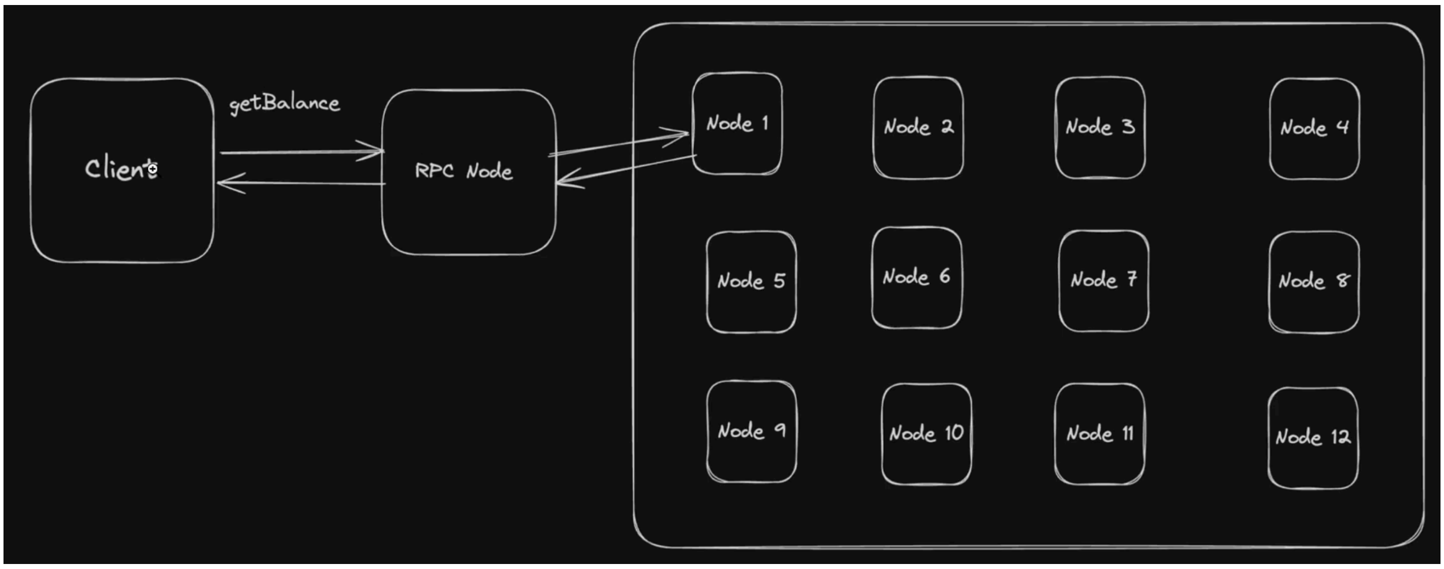
Additional RPC Methods

- GRPC
- TRPC

💡 JSON-RPC Specification

- Read the JSON-RPC Specification

RPC Server



- **RPC Server:** Allows external clients to interact with the blockchain network. It acts as an intermediary between the blockchain and external applications.
- It is easy to scale an RPC server.
- Typically, it's best to use an RPC server, but it's also possible to create your own—though this approach is generally not recommended.
- **Important Note:** An RPC server does not participate in staking or consensus mechanisms.

Common RPC Providers

- Quicknode - QuickNode - Blockchain infrastructure powering secure, decentralized innovation.
- Alchemy - Alchemy - the web3 development platform

- Helius - Helius - Solana's Leading RPC & API Platform
- Infura - Web3 Development Platform | IPFS API & Gateway | Blockchain Node Service (infura.io)

Common RPC Calls

Solana

▼ Get Account Info:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "getAccountInfo",
  "params": ["Eg4F6LW8DD3SvFLLigYJBFvRnXSBiLZYYJ3KEePDL95Q"]
}
```

▼ Get Balance:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "getBalance",
  "params": ["Eg4F6LW8DD3SvFLLigYJBFvRnXSBiLZYYJ3KEePDL95Q"]
}
```

▼ Get Transaction Count:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "getTransactionCount"
}
```

Ethereum

▼ Get Balance:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "eth_getBalance",
  "params": ["0xaeaa570b50ad00377ff8add27c50a7667c8f1811", "latest"]
}
```

▼ Get Latest Block:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "eth_blockNumber"
}
```

▼ Get Block by Number:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "eth_getBlockByNumber",
  "params": ["0x1396d66", true]
}
```

Wei and Lamports

- When working with financial applications, we avoid storing user balances in decimals to prevent floating-point errors.
- Instead, we use smaller units, such as wei for ETH or lamports for SOL, to accurately represent and display the balance.

Ethereum (ETH)

- **Wei:** The smallest unit of Ether.
  - **Value:** 1 Ether (ETH) = 10^18 Wei.
- **Gwei:** A larger unit of Ether, commonly used in gas prices.
  - **Value:** 1 Ether = 10^9 Gwei

Solana (SOL)

- **Lamports:** The smallest unit of SOL.
  - **Value:** 1 SOL = 10^9 Lamports
  - **Example:**

```
const { LAMPORTS_PER_SOL } = require("@solana/web3.js")

console.log(LAMPORTS_PER_SOL)
```