
## Course II:

# DeFi Primitives

## 4. Joining the World of DeFi

### (ii) Blockchain Tech Big Picture

### b) Addresses

# Tech Big Picture

## Key ingredients

- **Public Address**: This is either identical to the public key (e.g., Ethereum) or a function of the public key (e.g., Bitcoin).

QR code representation of the address
1MZhiFUaJSLpUyrCj8de7d5UMvZLtyuulz

# Public Addresses

- Bitcoin and Ethereum uses Elliptic Curve Digital Signature Algorithm* (ECDSA) for signing transactions.
- Here are the steps.
- We first generate a private key which is 256 bits (64 hex/32 bytes).
- We use ECDSA to derive a 512 bit public key. The private and public keys are known as the “key pair”.
- You can sign transactions with the private key
- Anyone with your public key can verify the signature is valid
- The Bitcoin and Ethereum addresses are linked to these keys



*Specifically, ECDSA sec256k1. See https://en.bitcoin.it/wiki/Secp256k1

# Ethereum

- Generate a key pair
- Public key is 512 bits (128 hex characters/64 bytes)
- Hash with Keccak-256 the public key (64 hex characters/32 bytes)
- Take last 40 hex characters (20 bytes) as your public address
- When prefixed with ‘0x’ it becomes 42 hex characters

For a given private key, $p_r$, the Ethereum address $A(p_r)$ (a 160-bit value) to which it corresponds is defined as the right most 160-bits of the Keccak hash of the corresponding ECDSA public key:

$$A(p_r) = \mathcal{B}_{96..255}\left(\texttt{KEC}\left(\texttt{ECDSAPUBKEY}(p_r)\right)\right) \quad (213)$$

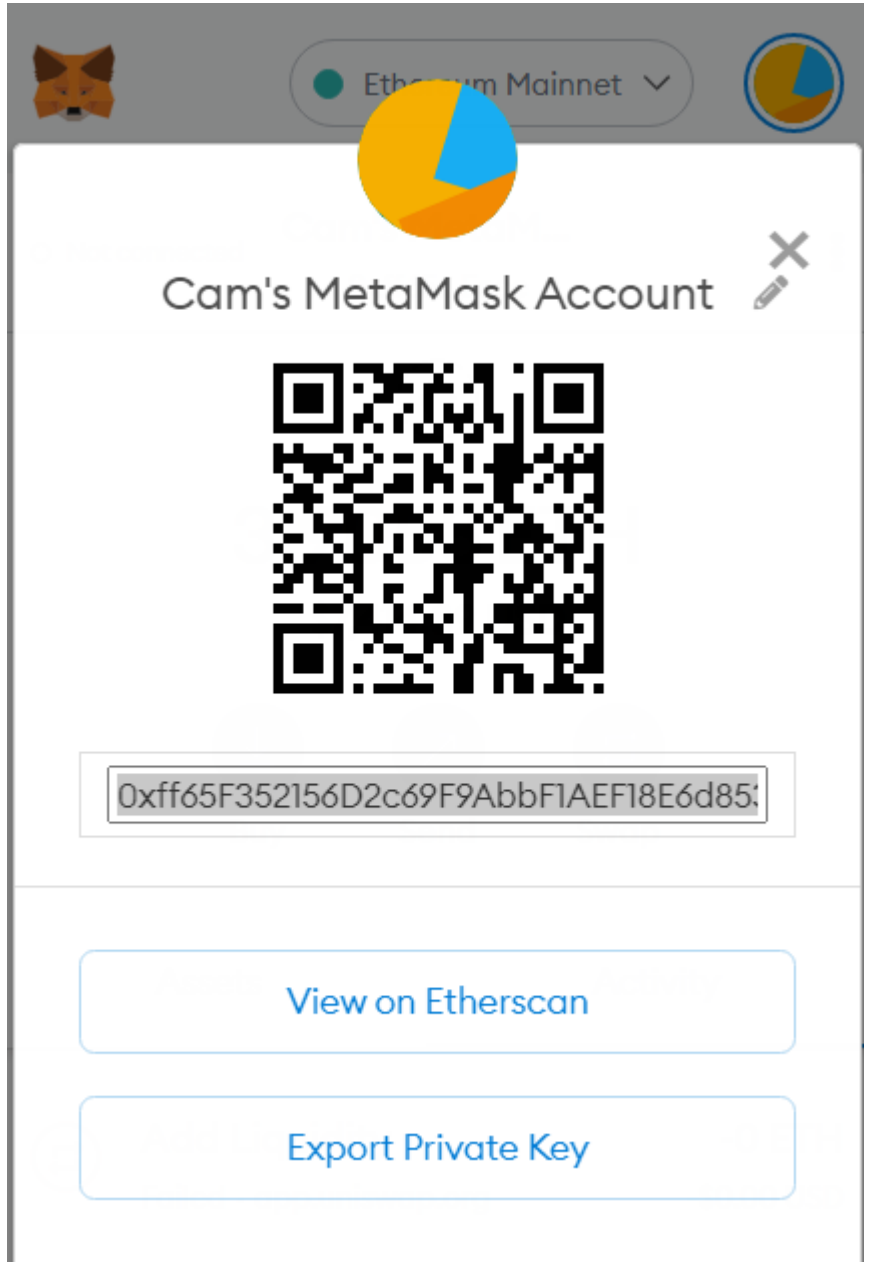
http://gavwood.com/paper.pdf

# Ethereum

- My public address is:

0xff65F352156D2c69F9AbbF1AEF18E6d85314Ecce

0x precedes the public key



https://www.oreilly.com/library/view/mastering-ethereum/9781491971932/ch04.html6

# Ethereum

- Private key

f8f8a2f43c8376ccb0871305060d7b27b0554d2cc72bccf41b2705608452f315

- Public key (04 prepended)

046e145ccef1033dea239875dd00dfb4fee6e3348b84985c92f103444683bae07b83b5c38e5e2b0
C8529d7fa3f64d46daa1ece2d9ac14cab9477d042c84c32ccd0

- Keccak-256 of public key (remove the 04)*

2a5bc342ed616b5ba5732269001d3f1ef827552ae1114027bd3ecf1f086ba0f9

- Last 40 hex (20 bytes) and prepend with 0x

0x001d3f1ef827552ae1114027bd3ecf1f086ba0f9



*In the hash of the public key, it is important to let the hashing algo know the format is “hex”. See https://leventozturk.com/engineering/sha3/

# Bitcoin addresses

- Bitcoin addresses have more steps but the idea is very similar

https://en.bitcoin.it/wiki/Technical_background_of_version_1_Bitcoin_addresses

# Bitcoin addresses

1. Start with private ECDSA key

```
18E14A7B6A307F426A94F8114701E7C8E774E7F9A47E2C2035DB29A206321725
```

2. Take public key generated with it*

```
0450863AD64A87AE8A2FE83C1AF1A8403CB53F53E486D8511DAD8A04887E5B23522CD470243453A299FA9E77237716103ABC11A1DF38855ED6F2EE187E9C582BA6
```

3. Perform SHA-256 on public key

```
600FFE422B4E00731A59557A5CCA46CC183944191006324A447BDB2D98D4B408
```

*65 bytes, 1 byte 0x04, 32 bytes corresponding to x-coordinate; 32 bytes for y coordinate

# Bitcoin addresses

4. Perform RIPEMD-160 hash on the result of SHA-256

`0109667760069 53D5567439E5E39F86A0D273BEE`

5. Add version number byte in front of RIPEMD-160

`00010966776006953D5567439E5E39F86A0D273BEE`

6. Perform SHA-256 on extended RIPEMD-160

`445C7A8007A93D8733188288BB320A8FE2DEBD2AE1B47F0F50BC10BAE845C094`

# Bitcoin addresses

7. Perform SHA-256 on the previous SHA-256

```
D61967F63C7DD183914A4AE452C9F6AD5D462CE3D277798075B107615C1A8A30
```

8. Take first 4 bytes of 2nd SHA-256 (address checksum)

```
D61967F6
```

9. Add 4 checksum bytes to extended RIPEMD-160 in stage 5 (25 byte bitcoin address)

```
00010966776006953D5567439E5E39F86A0D273BEED61967F6
```

# Bitcoin addresses

10. Convert to base58 (upper and lower case letters, numbers, excluding 0,O,I,l)

```
16UwLL9Risc3QfPqBUvKofHmBQ7wMtjvM
```

This is the bitcoin address.