Writing Smart Contracts
03 Accounts

Peter H. Gruber

Supported by the Algorand Foundation

Algorand Adesses

Private key

- For signing transactions
- A very long number ($\sim$ 77 decimal digits)
- “Master password to account”, “Single factor authentication”

Mnemonic

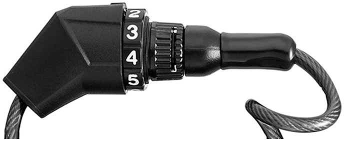
- Human-friendly representation of private key
- List of $2048 = 2^{11}$ words
  - One word represents 11 Bits

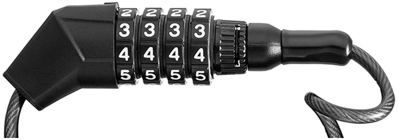
Public key $\sim$ Address

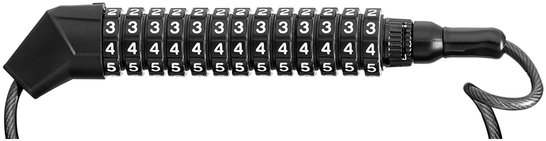
- Identify sender and recipient
- Hash of private key
  - Easy: private $\rightarrow$ public
  - Very hard: public $\rightarrow$ private

Wallet = collection of keys

Which bike lock is harder to crack?



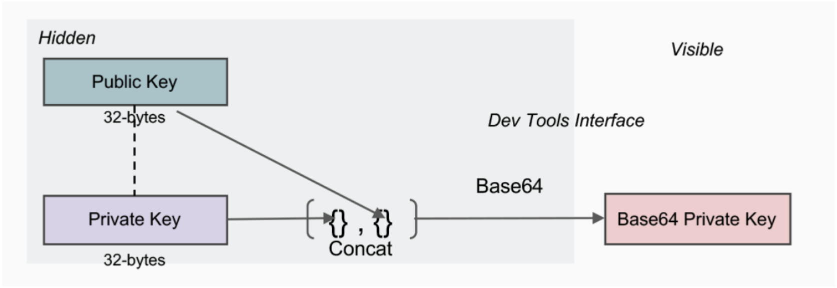




Private Key

- 32 Bytes = 256 Bit $\rightarrow 2^{256} \approx 10^{77}$ different possibilities
- On Algorand
  - Store private key plus public key
  - Encode as numbers/letters/symbols for readability (Base 64)
  - 86 symbols $\times$ 6 Bytes = 516 Bits
  - For developpers only

VwrmAkisLya/0H+HALB13XRpLNGfkoMY4mgUXYL6FURv9FXUJdPt6SrtHbyV5n21AC0ZA65Rtz6CVoCy0O7aqQ==



Mnemonic = Passphrase

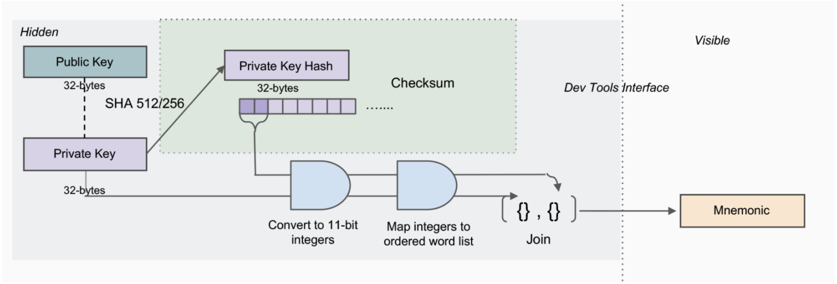
Representation of Private Key for end users

- Encode 256Bit key as word sequence
- List of $2^{11} = 2048$ words
  - Each word $\leftrightarrow$ 11 Bit number

```
0001:  abandon
0002:  ability
...
2047:  zone
2048:  zoo
```

- Algorand mnemonic has 25 words
  - 25 words $\times$ 11 Bits = 275 Bits (extra word is checksum)

```
enough oblige accident setup gap sister magnet lemon axis scale river evidence spray
enrich write myth away mask crucial spend again leaf camera able athlete
```

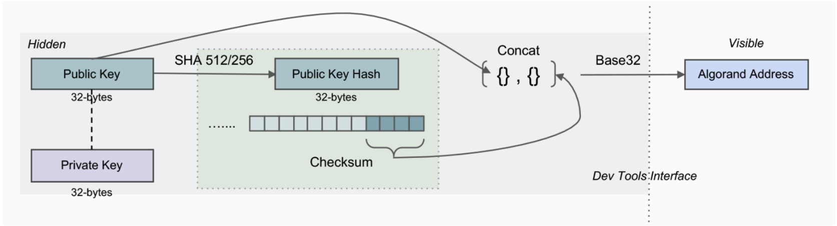


Public Key ∼ Address

From public key to address

- Public key = 256 Bit
- Add hash of 32 Bit length (4 Bytes)
- Encode as numbers/letters for readability (Base 32)
- 58 numbers/letters, 5 Bytes each = 290 Bits > 256+32

```
WSC24MVUSQ32IZYD7FNN54Z44IXWL4X7BOJD6AGFOCHOG4PDFESLZUGLTI
```



An Algorand transaction

```
{
  "txn": {
    "amt":  5000000,
    "fee":  1000,
    "fv":   6000000,
    "lv":   6001000,
    "gen":  "mainnet-v1.0",
    "gh":   "wGHE2Pwdvd7S12BL5FaOP20EGYesN73ktiC1qzkkit8=",
    "note": "SGVsbG8gV29ybGQ=",
    "snd":  "EW64GC6F24M7NDSC5R3ES4YUVE3ZXXNMARJHDCCCLIHZU6TBEOC7XRSBG4",
    "rcv":  "GD64YIY3TWGDMCNPP553DZPPR6LDUSFQOIJVFDPPXWEG3FVOJCCDBBHU5A",
    "type": "pay"
  },
  "sig": "mg8i4gA98pZBFxfgZakscUh6xhdxlqz2NFIAWAe6jL19GMrr40X8XZ0OpOT3X8AwdiBqXlXQ/lslCafEzG12Ag=="
}
```

`fv`, `lv`: first/last valid round
`gh`: Genesis Hash
`sig`: Signature of entire tx object

https://developer.algorand.org/docs/get-details/transactions/transactions/

Life of a transaction

(1) Setup

- Create transaction in Python or (web) app
- Transaction is not yet signed

(2) Sign

- Sender uses private key to sign transaction
- Signature is added in `"sig"` field

(3) Submit

- Send to the blockchain via API or your own participation node

(4) Get accepted

- Other nodes verify signature (using the public key of the sender)
- Consensus decides if a transaction is included in the next block

Accessing the blockchain

Where is the Algorand chain?

- On approx. 1175 nodes (Feb 2024) – one of them at USI
- Up-to-date: `https://metrics.algorand.org`

How large is the Algorand Chain?

- Approx. 1.5TB (Feb 2024)

How can we access the chain?

- Set up our own indexer node
- Access via API
- Explore using web interfaces

Python commands

Transactions

- Local
    1. Prepare/create transaction → `txn`
    2. Sign transaction → `stxn`
    3. Send transaction → `txid`
- On Chain
    4. Verify transaction → `txinfo`

Accounts

- Local
    - Create key pair
- On Chain
    - Get account balance