Bitcoin Backups

- In Bitcoin, private keys control ownership of funds
- -Private keys → Public Keys → Addresses you send money to

You can generate 1000’s of addresses from a seed of 12-24 words called a mnemonic!

It’s easy to generate one in Python and import it into any modern wallet

Step 1

- Generate entropy (random bits)
- We need 128 bits for a 12 word seed
- 128 bits = 32 bytes

```
entropy = os.urandom(size_bits // 8)
```

Step 2

- SHA-256 hash of the entropy
- Append the first (N/32) bits to the end
- For 128 bits, that’s the first 4 bits

```
hash = hashlib.sha256(entropy)
hash_bits = bitstring.BitArray("0x" + hash.hexdigest())

num_checksum_bits = size_bits // 32

entropy_bits = bitstring.BitArray("0x" + entropy.hex())
checksummed_bits = entropy_bits + hash_bits[0:num_checksum_bits]
```

Step 3

- Split into 11 bit “chunks” (no matter the entropy size)
- Treat these as 11 bit integers
- Use each number as an index on the standard mnemonic word list

```
num_chunks = len(checksummed_entropy) // 11

mnemonic = []
for i in range(0, num_chunks):
     start = i * 11
     end = (i + 1) * 11

     chunk = bitstring.BitArray(checksummed_entropy[start:end])
     word_index = int(chunk.bin, 2)

     word = words[word_index]
     mnemonic.append(word)

return mnemonic
```

Final Product

- Example Mnemonic:

```
salmon cliff inherit physical help type warfare
regular dial photo asset scheme
```

DO NOT USE! Your seed is private.

More Info

- ***Chaintuts.com*** (Articles and Videos)
- ***Github.com/chaintuts*** has a full working version of a Python mnemonic generator
- Join me in **Room 15** @ 10 AM for a Bitcoin + Python Meetup

Bitcoin Backups

- In Bitcoin, private keys control ownership of funds
- -Private keys → Public Keys → Addresses you send money to

You can generate 1000’s of addresses from a seed of 12-24 words called a mnemonic!

It’s easy to generate one in Python and import it into any modern wallet

Step 1

- Generate entropy (random bits)
- We need 128 bits for a 12 word seed
- 128 bits = 32 bytes

```
entropy = os.urandom(size_bits // 8)
```

Step 2

- SHA-256 hash of the entropy
- Append the first (N/32) bits to the end
- For 128 bits, that’s the first 4 bits

```
hash = hashlib.sha256(entropy)
hash_bits = bitstring.BitArray("0x" + hash.hexdigest())

num_checksum_bits = size_bits // 32

entropy_bits = bitstring.BitArray("0x" + entropy.hex())
checksummed_bits = entropy_bits + hash_bits[0:num_checksum_bits]
```

Step 3

- Split into 11 bit “chunks” (no matter the entropy size)
- Treat these as 11 bit integers
- Use each number as an index on the standard mnemonic word list

```
num_chunks = len(checksummed_entropy) // 11

mnemonic = []
for i in range(0, num_chunks):
     start = i * 11
     end = (i + 1) * 11

     chunk = bitstring.BitArray(checksummed_entropy[start:end])
     word_index = int(chunk.bin, 2)

     word = words[word_index]
     mnemonic.append(word)

return mnemonic
```

Final Product

- Example Mnemonic:

```
salmon cliff inherit physical help type warfare
regular dial photo asset scheme
```

DO NOT USE! Your seed is private.

More Info

- ***Chaintuts.com*** (Articles and Videos)
- ***Github.com/chaintuts*** has a full working version of a Python mnemonic generator
- Join me in **Room 15** @ 10 AM for a Bitcoin + Python Meetup