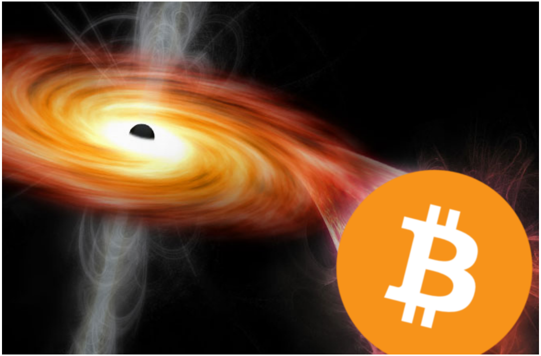
Making Money Disappear with Hash Functions!

Brendan Cordy

!!Con 2016

What is a Bitcoin Address?

`1ELwdsETv4pv1SGvwZ4n2uzXT7bLnR8iVo`



What is Bitcoin?

1v6X...qT74	→	54335
1Ag6...yz93	→	916
1ccV...8kLE	→	0
14rT...u3d5	→	1665

- Addresses are the bitcoin equivalent of account numbers.

Two Problems to Solve

- If the network is anonymous, why can't any user spend money in any account?

Two Problems to Solve

- If the network is anonymous, why can't any user spend money in any account?

  → Public Key Cryptography (ECDSA)

Two Problems to Solve

- If the network is anonymous, why can't any user spend money in any account?

  → Public Key Cryptography (ECDSA)

- What happens when someone spends their money on two things at the same time?

Two Problems to Solve

- If the network is anonymous, why can't any user spend money in any account?

  → Public Key Cryptography (ECDSA)

- What happens when someone spends their money on two things at the same time?

  → Resolved by the Blockchain (Mining)

Base 58

`1ELwdsETv4pv1SGvwZ4n2uzXT7bLnR8iVo`

a b c d e f g h i j k l m n o p q r s t u v w x y z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

0 1 2 3 4 5 6 7 8 9

Base 58

`1ELwdsETv4pv1SGvwZ4n2uzXT7bLnR8iVo`

a b c d e f g h i j k l m n o p q r s t u v w x y z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

0 1 2 3 4 5 6 7 8 9

Base 58

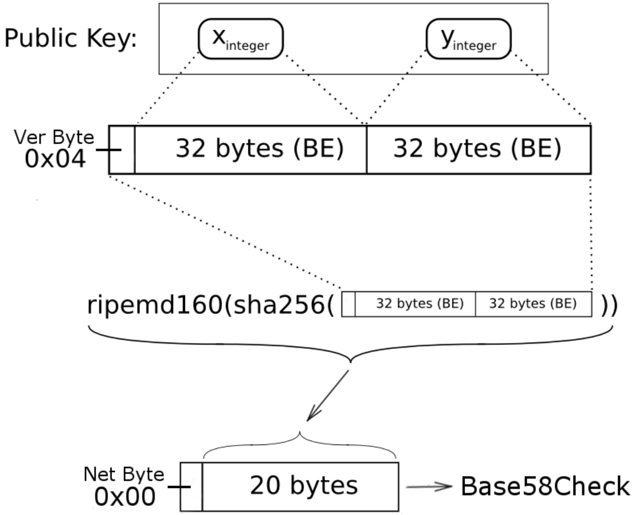
`1ELwdsETv4pv1SGvwZ4n2uzXT7bLnR8iVo`

a b c d e f g h i j k   m n o p q r s t u v w x y z

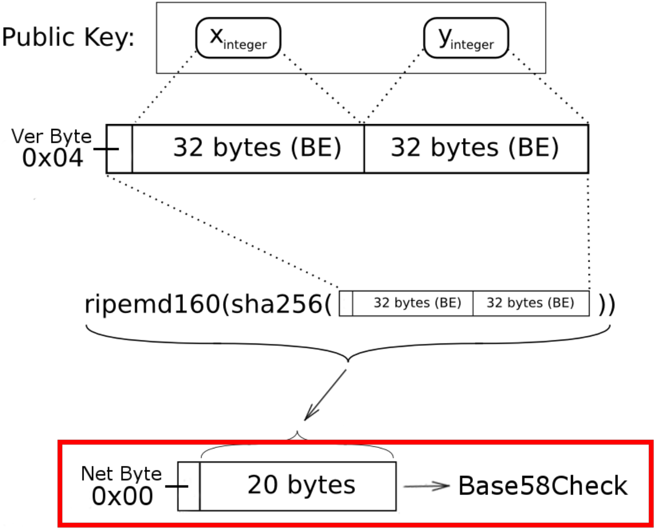
A B C D E F G H   J K L M N   P Q R S T U V W X Y Z

1 2 3 4 5 6 7 8 9

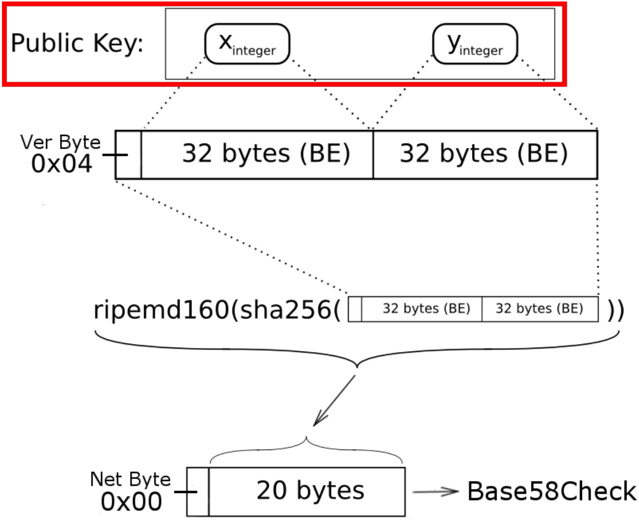
Building Addresses



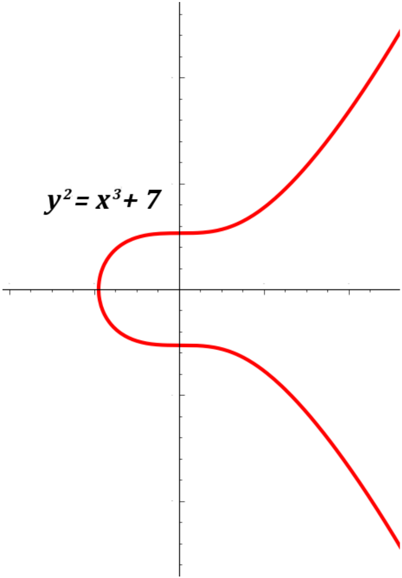
Building Addresses



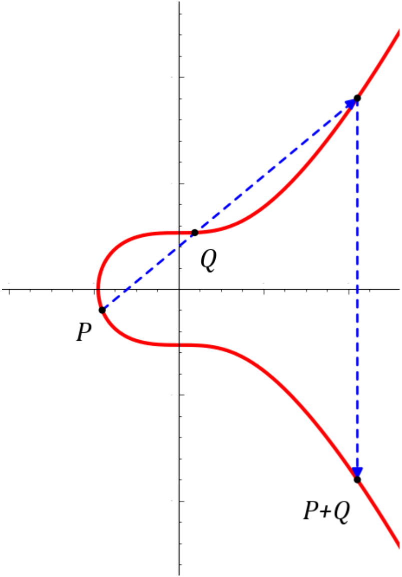
Building Addresses



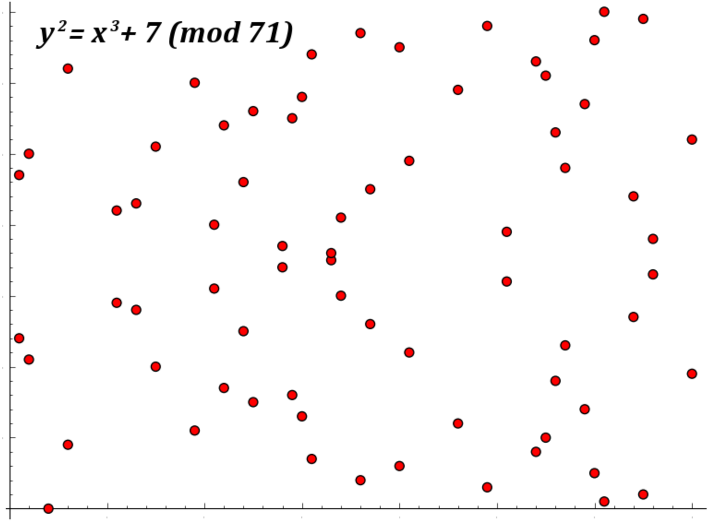
Origins of the Public Key



Origins of the Public Key



Origins of the Public Key



Generating a Public Key

- Given P, pick a random 256-bit s, and let

$$Q = \overbrace{P + P + \ldots + P}^{s \text{ times}} = sP$$

Generating a Public Key

- Given P, pick a random 256-bit s, and let

$$Q = \overbrace{P + P + \ldots + P}^{s \text{ times}} = sP$$

- What if we wanted to find s from P and Q?

$Q = ?P$ ¯_(ツ)_/¯

Generating a Public Key

- Given P, pick a random 256-bit s, and let

$$Q = \overbrace{P + P + \ldots + P}^{s \text{ times}} = sP$$

- What if we wanted to find s from P and Q?

$$Q =? P \qquad \text{¯\\_(ツ)\_/¯}$$

Secret Key → s Public Key → Q

Elliptic Curve Signatures

- From $y^2 = x^3 + 7 \pmod{2^{256} - 4294966319}$ and P we can generate a key-pair (s, Q).

Elliptic Curve Signatures

- From $y^2 = x^3 + 7 \pmod{2^{256} - 4294966319}$ and P we can generate a key-pair (s, Q).
- Using ECDSA, the network can verify that a transaction originated from an individual who knows the value s.

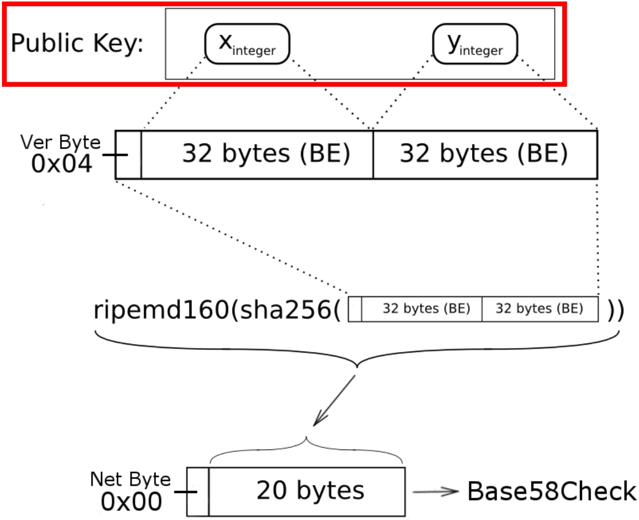
Elliptic Curve Signatures

- From $y^2 = x^3 + 7 \pmod{2^{256} - 4294966319}$ and P we can generate a key-pair (s, Q).

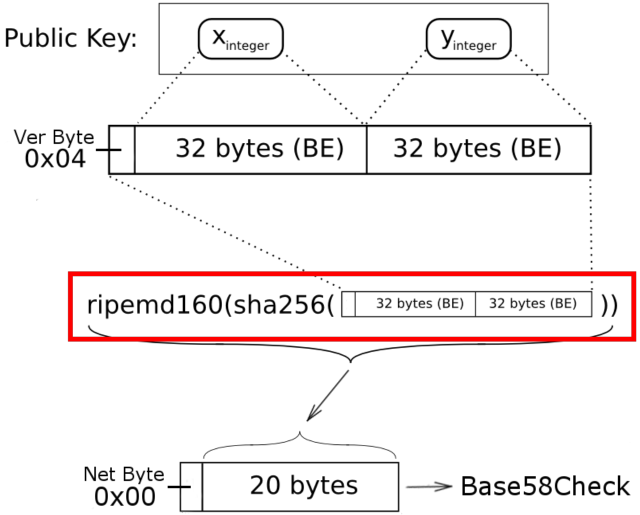
- Using ECDSA, the network can verify that a transaction originated from an individual who knows the value s.

- The secret key s is a PIN used for spending.

Building Addresses



Building Addresses



Cryptographic Hashes

$$sha256(1729) = ? \quad \leftarrow \text{Super Fast!}$$

Cryptographic Hashes

$$sha256(1729) = ? \quad \leftarrow \text{Super Fast!}$$

$$sha256(?) = 1729 \quad \leftarrow \text{Impossible!}$$

Cryptographic Hashes

$$sha256(1729) = ? \quad \leftarrow \text{Super Fast!}$$

$$sha256(?) = 1729 \quad \leftarrow \text{Impossible!}$$

- Impossible to check whether an address was made from a correctly generated keypair.

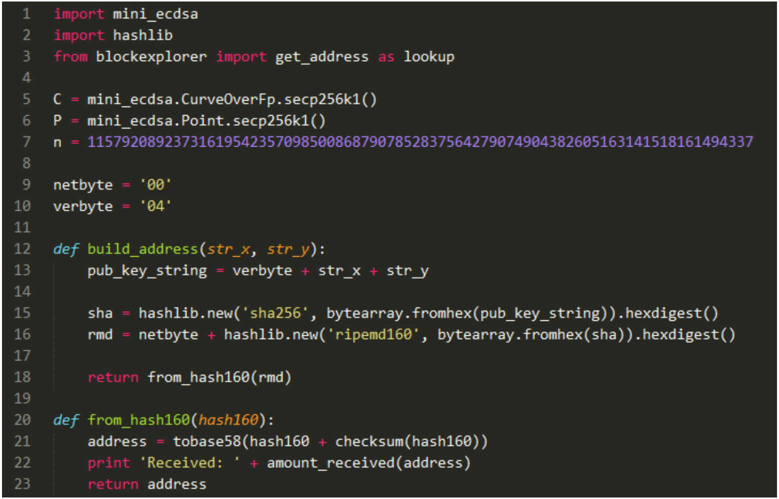
Cryptographic Hashes

$$sha256(1729) = ? \quad \leftarrow \text{Super Fast!}$$

$$sha256(?) = 1729 \quad \leftarrow \text{Impossible!}$$

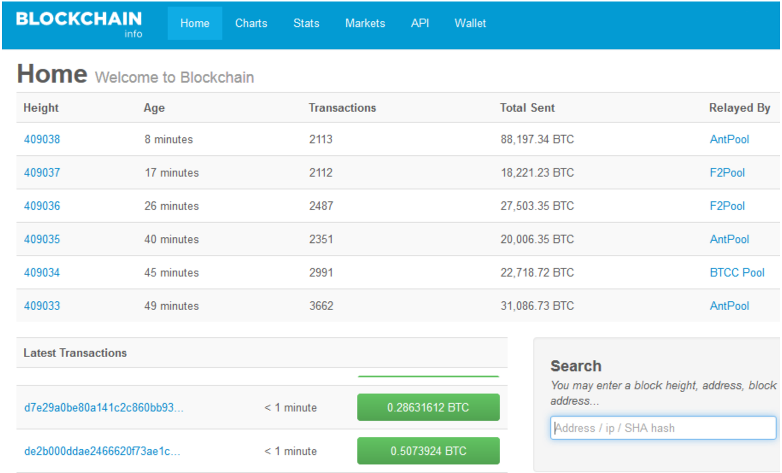
- Impossible to check whether an address was made from a correctly generated keypair.
- Q is not on the curve → No valid PIN!

Counting Money in the Void!



```python
import mini_ecdsa
import hashlib
from blockexplorer import get_address as lookup

C = mini_ecdsa.CurveOverFp.secp256k1()
P = mini_ecdsa.Point.secp256k1()
n = 115792089237316195423570985008687907852837564279074904382605163141518161494337

netbyte = '00'
verbyte = '04'

def build_address(str_x, str_y):
    pub_key_string = verbyte + str_x + str_y

    sha = hashlib.new('sha256', bytearray.fromhex(pub_key_string)).hexdigest()
    rmd = netbyte + hashlib.new('ripemd160', bytearray.fromhex(sha)).hexdigest()

    return from_hash160(rmd)

def from_hash160(hash160):
    address = tobase58(hash160 + checksum(hash160))
    print 'Received: ' + amount_received(address)
    return address
```

Counting Money in the Void!



Home Welcome to Blockchain

Height	Age	Transactions	Total Sent	Relayed By
409038	8 minutes	2113	88,197.34 BTC	AntPool
409037	17 minutes	2112	18,221.23 BTC	F2Pool
409036	26 minutes	2487	27,503.35 BTC	F2Pool
409035	40 minutes	2351	20,006.35 BTC	AntPool
409034	45 minutes	2991	22,718.72 BTC	BTCC Pool
409033	49 minutes	3662	31,086.73 BTC	AntPool

Latest Transactions

d7e29a0be80a141c2c860bb93...	< 1 minute	0.28631612 BTC
de2b000ddae2466620f73ae1c...	< 1 minute	0.5073924 BTC

Search

You may enter a block height, address, block address...

Silly Addresses I

- Build an address from the empty string!

Address	Balance
`1HT7x...K8d4E`	\$31 500

Silly Addresses I

- Build an address from the empty string!

Address	Balance
1HT7x...K8d4E	\$31 500

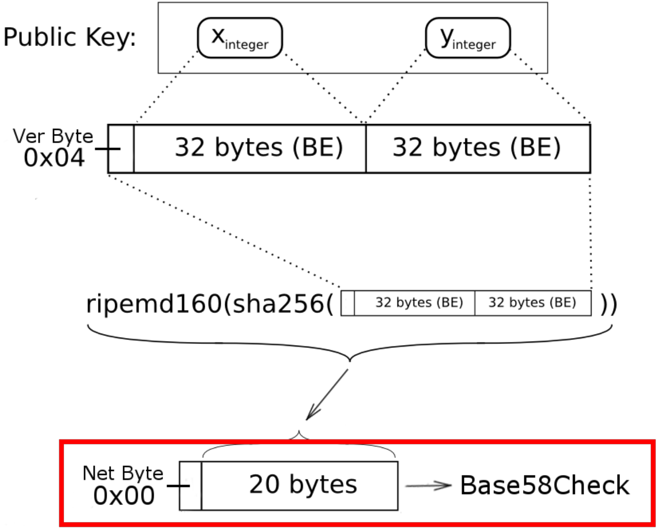
- Build an address from the point (0,0)!

Address	Balance
1FYMZ...YKQxh	\$1 650

Silly Addresses II

- Convert simple hex values to Base 58, and add correct checksums!

Building Addresses



Public Key:

x_{integer}

y_{integer}

Ver Byte 0x04

32 bytes (BE)

32 bytes (BE)

ripemd160(sha256(32 bytes (BE) 32 bytes (BE)))

Net Byte 0x00

20 bytes

Base58Check

Silly Addresses II

- Convert simple hex values to Base 58, and add correct checksums!

Hex String	Address	Balance
000 . . . 000	11111 . . . oLvT2	$22 900
000 . . . 001	11111 . . . Zbvjr	$5
AAA . . . AAA	1GZQK . . . R1zmr	$10
FFF . . . FFF	1QLbz . . . 5j6Qr	$5

The Hashing Debate

- Pros: Compression, Security (kind of...)

The Hashing Debate

- Pros: Compression, Security (kind of...)
- Cons: The void!

The Hashing Debate

- Pros: Compression, Security (kind of...)
- Cons: The void!
- Had Satoshi thought about how bugs will slowly eat away at the number of coins in circulation?

The Hashing Debate

- Pros: Compression, Security (kind of...)
- Cons: The void!
- Had Satoshi thought about how bugs will slowly eat away at the number of coins in circulation?
- Design decisions are hard!

References

Bitcoin: A Peer-to-Peer Electronic Cash System, *Satoshi Nakamoto*.
https://bitcoin.org/bitcoin.pdf, 2008.

Elliptic Curve Cryptography in Practice, *Bos, Halderman, et al*.
https://eprint.iacr.org/2013/734.pdf, 2013.

SEC2: Recommended Elliptic Curve Parameters, *Certicom Research*.
http://www.secg.org/SEC2-Ver-1.0.pdf, 2000

Bitcoin Wiki: Elliptic Curve Public Key to BTC Address Conversion.
https://en.bitcoin.it/wiki/File:PubKeyToAddr.png

Blockchain.info API Library (Python, v1)
https://github.com/blockchain/api-v1-client-python