Basic Cryptography
Keys and Addresses in Bitcoin

Konstantinos Karasavvas
R&D Blockchain Initiative, University of Nicosia

Outline

Hashing Algorithms

Asymmetric Cryptography

Bitcoin’s Private keys

Bitcoin’s Public keys

Bitcoin’s Addresses

Wallets

Conclusions

Hashing Algorithms

Cryptographic Hash Function (1)

- Function
    - arbitrary block of data -> a fixed-size bit string
    - hash or hash value or digest or digital fingerprint
    - any change to the data will significantly change the hash
- Properties
    - It is deterministic, i.e. the same block of data always returns the same hash
    - It is quick to compute
    - It is close to impossible to generate the block of data from the hash (one-way Fn)
    - A small change to the data will change the hash so that it appears uncorrelated to the old hash value
    - It is close to impossible to find two different blocks of data with the same hash value
- Bitcoin uses the SHA-256 function.
    - 256 bits or 32 bytes long
    - 64 hexadecimal numbers

Cryptographic Hash Function (2)

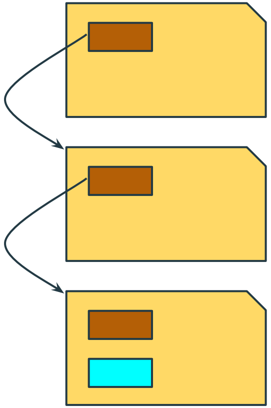
```
$                                                            sha256sum
Bitcoin
deb10ca6fd85a5eba792ea8561da390635242f0c37c376f8eb7d7859adbffca9      -

$                                                            sha256sum
bitcoin
61d520ccb74288c96bc1a2b20ea1c0d5a704776dd0164a396efec3ea7040349d  -
```

- Cryptographic hash functions
  - used in digital signatures, message authentication codes and as ordinary (but more secure) hash functions to index data in hash tables, to uniquely identify files (bittorrent, IPFS), as checksums to detect accidental (or not) corruption of data, etc.

Cryptographic Hash Function (3)

In Bitcoin the SHA-256 hash function is used for:

- Block chaining & Integrity verification
  - When a new block is created its header contains the hash of the previous block, thus *chaining* the two blocks together.
  - Since peers follow the longest chain a Tx with more confirmations (chained blocks on top of the block that contains it) is much more difficult to alter/remove since to change the Tx successfully an attacker has to modify that block as well as all the blocks on top of it (or else the tampering will be detected immediately)
  - This is computationally infeasible to sustain



Cryptographic Hash Function (4)

In Bitcoin the SHA-256 hash function is used for:

- Proof-of-Work (hashcash) cost-function (mining)
    - The Proof-of-Work puzzle involves finding a hash of the new block that is less than a certain value
- Generation of Bitcoin public keys and addresses
    - Used to improve security and privacy
    - To create addresses another hashing algorithm, RIPEMD-160, is also used

Asymmetric Cryptography

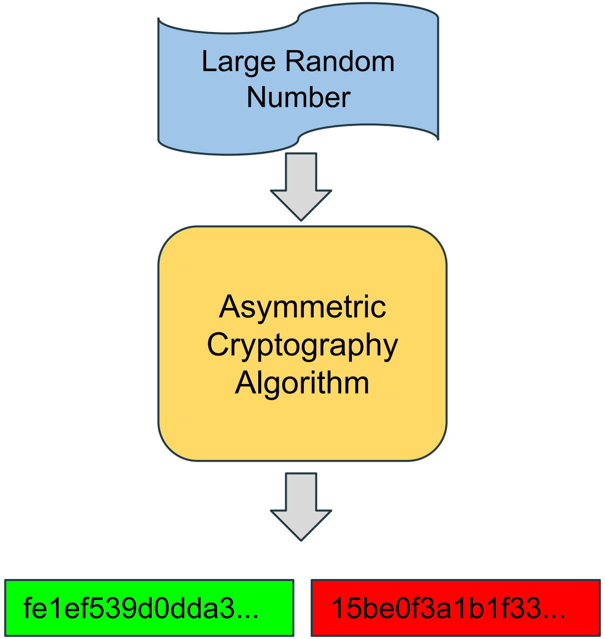
Asymmetric Cryptography

- public key cryptography
  - uses pairs of keys: public and private
  - mathematically related
  - public key can be shared

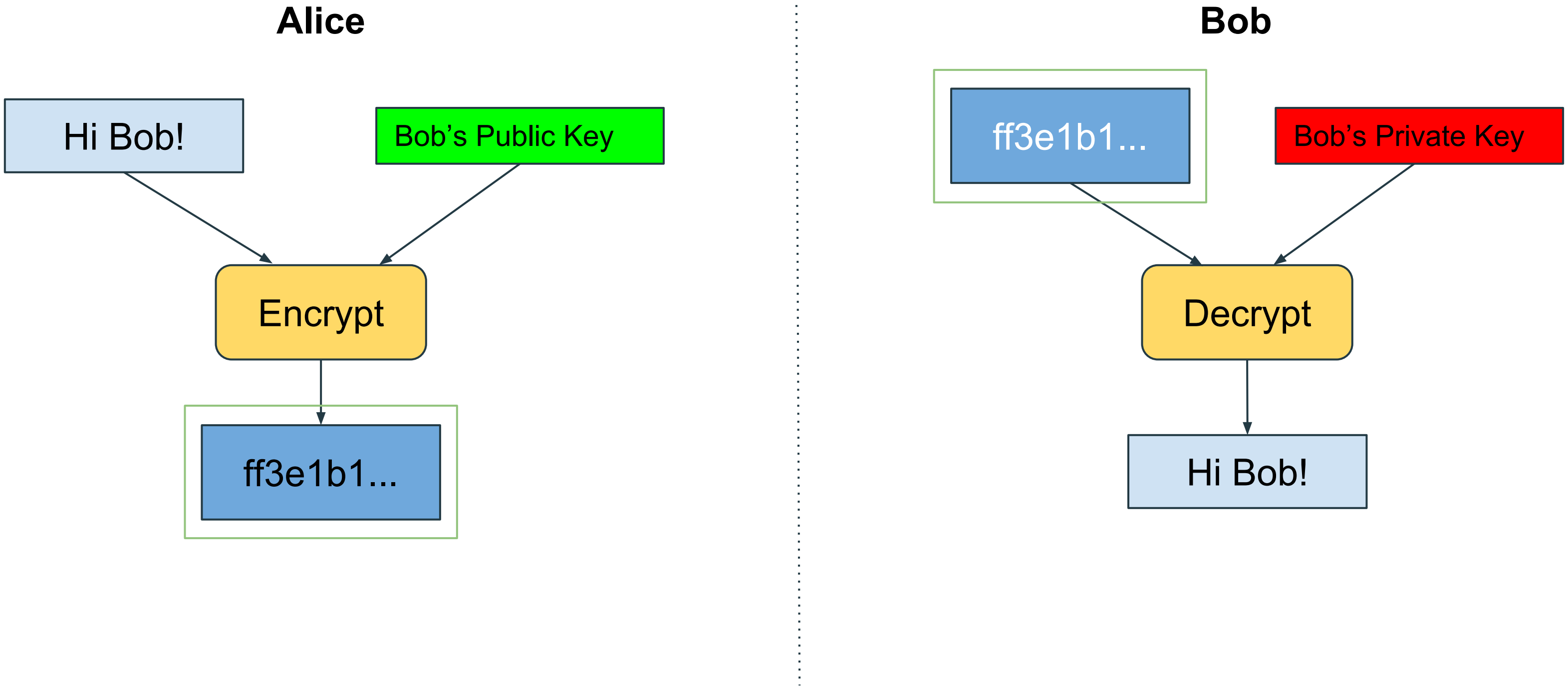
- Encryption

  Alice can *encrypt* a message with Bob’s public key and send it to Bob. Only the owner of the corresponding private key can decrypt and view the message.
- Authentication / Digital Signatures

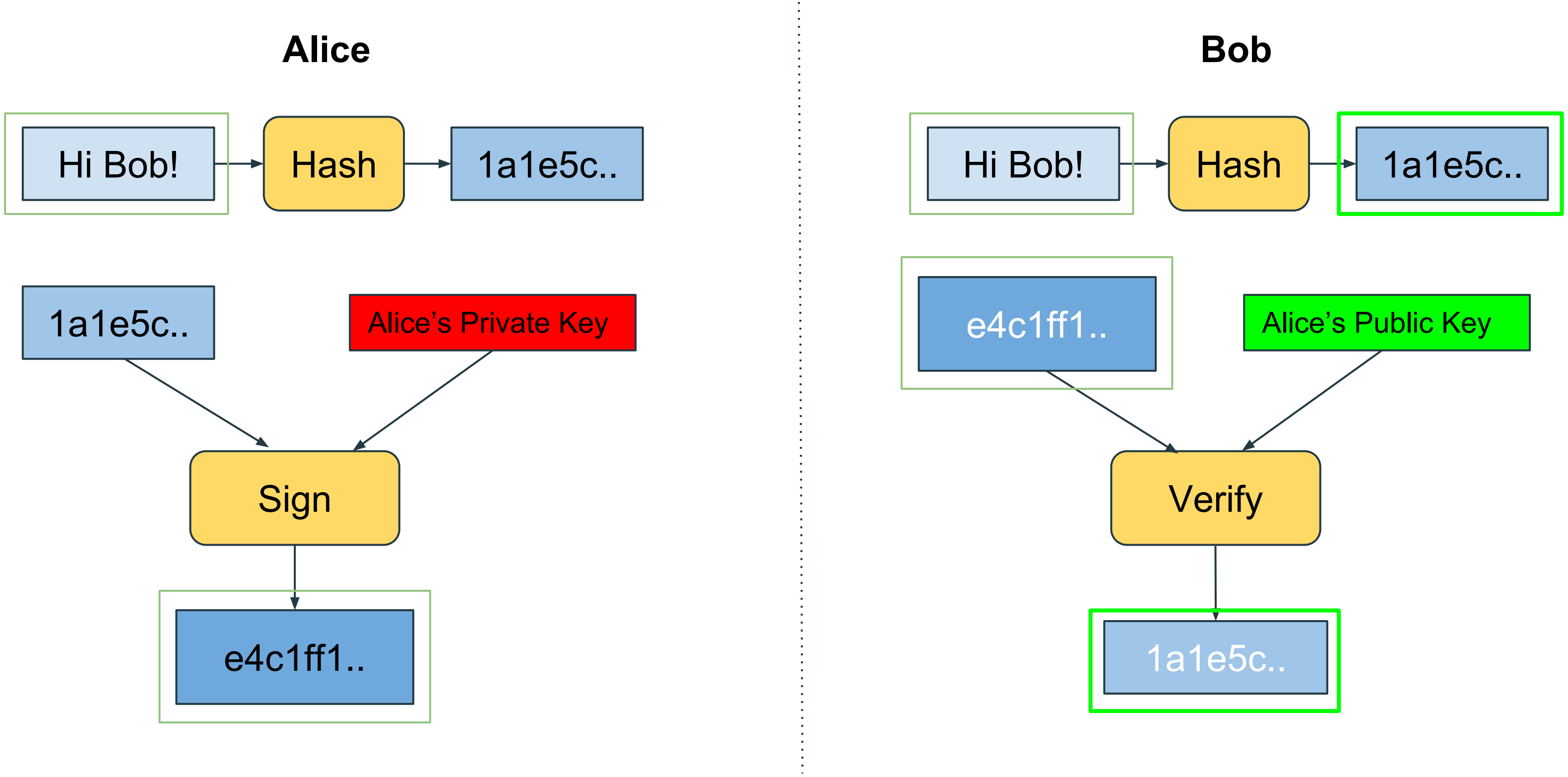
  Alice can *sign* a message using her private key and send it to Bob. Anyone can view the contents by using Alice’s public key, thus ensuring that it was indeed Alice that send the message.



Encryption / Decryption



Digital Signatures / Verification



Public Key Cryptography in Bitcoin

- encryption is not used

- digital signatures are used to sign transactions in order to authenticate that you are the owner of the coins you wish to transfer.

- Bitcoin uses the *Elliptic Curve Digital Signature Algorithm* (ECDSA)
  - elliptic curve parameters used are defined by secp256k1.

- In ECDSA
  - private key can be used to calculate the corresponding public key
  - the address is calculated from the public key

- Keep the private key secure

Bitcoin’s Private keys

Private Keys

- ECDSA private key
  - just a random number consisting of 256 bits
  - nearly all 256-bit numbers can be valid private keys as specified in secp256k1.
- Common formats
  - *Wallet Import Format* (WIF) or a WIF-compressed (WIFC)
  - they use Base58Check encoding of the ECDSA key
- WIF-compressed just adds an extra byte (0x01) at the end of the ECDSA key
  - before the Base58Check encoding
  - specifies whether the public key will be compressed or not
  - default for most wallets

```
$ ./bitcoin-cli dumpprivkey mg6KkpbdyyFkwzCFzmna3yZAUj2yPhoKd
cNg68oFh99vkg5FRkegvx11jq5w9bxzBaDan9ZLUfZeMr4Vn6dii
```

Private Keys: formats (1)

	Mainnet		**Testnet**	
ECDSA HEX	64 digits number		64 digits number	
ECDSA HEX-C	Above number + “01”		Above number + “01”	
	Version Prefix	**Base58 Prefix**	**Version Prefix**	**Base58 Prefix**
WIF	128 | 0x80	5	239 | 0xef	9
WIF-Compressed	128 | 0x80	K or L	239 | 0xef	c

Check it out by entering a testnet private key on “Wallet Details” at: https://www.bitaddress.org or https://www.bitaddress.org?testnet=true

Private Keys: formats (2)

ECDSA in HEX	0dde70823a4bb0ca3bd75a2010e8d5dc091185e73d8b4257a981c695a3eba95b
ECDSA in HEX-C	0dde70823a4bb0ca3bd75a2010e8d5dc091185e73d8b4257a981c695a3eba95b**01**
WIF	91h2ReUJRwJhTNd828zhc8RRVMU4krX9q3LNi4nVfiVwkMPfA9p
WIFC	cN3fHnPVw4h7ZQSRz2HgE3ko69LTaZa5y3JWpFhoXtAke4MiqVQo

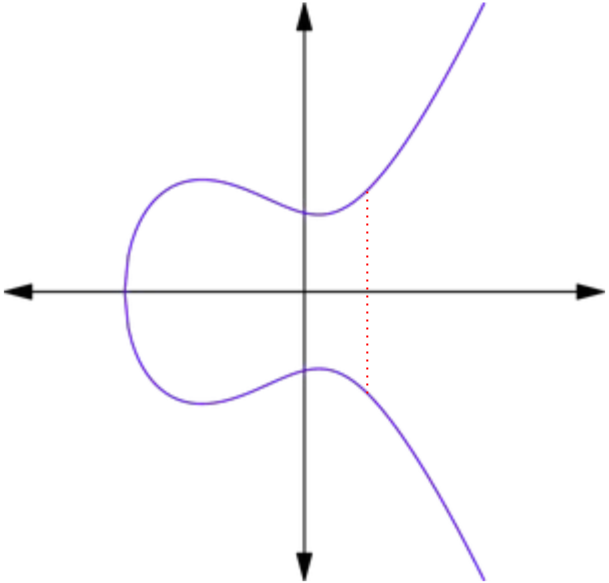
- libbitcoin-explorer provides *bx* tool
  - Decimal prefixes 128 (0x80) and 239 (0xef) are used for mainnet and testnet respectively.

```
$ ./bx base58check-encode --version 239 0dde70823a4bb0ca3bd75a2010e8d5dc091185e73d8b4257a981c695a3eba95b
91h2ReUJRwJhTNd828zhc8RRVMU4krX9q3LNi4nVfiVwkMPfA9p

$ ./bx base58check-encode --version 239 0dde70823a4bb0ca3bd75a2010e8d5dc091185e73d8b4257a981c695a3eba95b01
cN3fHnPVw4h7ZQSRz2HgE3ko69LTaZa5y3JWpFhoXtAke4MiqVQo
```

Bitcoin’s Public keys

Public Keys

- generated from the private key
  - using *elliptic curve multiplication*
  - a one-way function
- the public key is a point P in the curve
  - P = (x,y).
- uncompressed public key is 65 bytes long
  - two points (32 bytes each) concatenated
  - plus a prefix of 0x04
- compressed public key is 33 bytes long
  - one point (32 bytes of the x coordinate
  - plus prefix of 0x02 for y
  - or prefix of 0x03 for -y



Public Keys: example

WIF	91h2ReUJRwJhTNd828zhc8RRVMU4krX9q3LNi4nVfiVwkMPfA9p
Uncompressed Public Key	04c1acdac799fb0308b4b6475ddf7967676759d31484ab55555482472f3bc7c3e7addc4cbba6656a4be4bc6933a6af712b897a543a09c4b899e5f7b943d38108a8
WIFC	cN3fHnPVw4h7ZQSRz2HgE3ko69LTaZa5y3JWpFhoXtAke4MiqVQo
Compressed Public Key	02c1acdac799fb0308b4b6475ddf7967676759d31484ab55555482472f3bc7c3e7

```
$ ./bx wif-to-public 91h2ReUJRwJhTNd828zhc8RRVMU4krX9q3LNi4nVfiVwkMPfA9p
04c1acdac799fb0308b4b6475ddf7967676759d31484ab55555482472f3bc7c3e7addc4cbba6656a4be4bc6933a6af712b897a543a
09c4b899e5f7b943d38108a8

$ ./bx wif-to-public cN3fHnPVw4h7ZQSRz2HgE3ko69LTaZa5y3JWpFhoXtAke4MiqVQo
02c1acdac799fb0308b4b6475ddf7967676759d31484ab55555482472f3bc7c3e7
```

Bitcoin’s Addresses

Addresses

- shared to anyone who wants to sent you money
  - generated from the public key
  - consist of a sequence of characters and digits
  - typically start with 1 for the mainnet and with m or n for testnet.

- an address represents the owner of a private/public pair
  - it can also represent complex scripts (P2SH)

- public key is not shared but rather the address
  - shorter addresses
  - quantum computer resistant

Calculate Address

- To calculate an address we need to apply the following algorithm:

```
version = (1 byte version number)
keyHash = RIPEMD-160( SHA-256( publicKey ) )
data = version + keyHash
dataHash = SHA-256( SHA-256( data))
checksum = (first 4 bytes of dataHash)
address = Base58Encode( data + checksum )
```

- all byte sequences should be treated as big-endian
- to validate a bitcoin address
  - Base58Decode
  - remove the checksum
  - double SHA-256 the reminder
    - the first 4 bytes should be equal to the checksum

Python library example

```
from bitcoin import *

pub =
'04c1acdac799fb0308b4b6475ddf7967676759d31484ab55555482472f3bc7c3e7addc4cbba665
6a4be4bc6933a6af712b897a543a09c4b899e5f7b943d38108a8'
pubc = '02c1acdac799fb0308b4b6475ddf7967676759d31484ab55555482472f3bc7c3e7'

print( pubtoaddr(pub, 111) )
print( pubtoaddr(pubc, 111) )
```

```
$ python addr.py
n2JjAgC6UqFf8DvsZXhWcyNzm8w8YKj7MQ
n42m3hGC52QTChUbXq3QAPVU6nWkG9xuWj
```

Addresses and prefixes

Uncompressed Public Key	04c1acdac799fb0308b4b6475ddf7967676759d31484ab55555482472f3bc7c3e7addc4cbba6656a4be4bc6933a6af712b897a543a09c4b899e5f7b943d38108a8
Uncompressed Address	n2JjAgC6UqFf8DvsZXhWcyNzm8w8YKj7MQ
Compressed Public Key	02c1acdac799fb0308b4b6475ddf7967676759d31484ab55555482472f3bc7c3e7
Compressed Address	n42m3hGC52QTChUbXq3QAPVU6nWkG9xuWj

	Mainnet		Testnet	
	Version prefix	Base58 prefix	Version prefix	Base58 prefix
Bitcoin Address	0 | 0x00	1	111 | 0x6f	m or n
Script Address	5 | 0x05	3	196 | 0xc4	2

Wallets

Wallets

- software that allows us to manage keys and addresses
  - allow to send bitcoins, check balances, create contact lists and other
  - usually a key (i.e. address) is used only once.

- Nondeterministic
  - all private keys are just a randomly generated
  - several private keys are pre-generated and new keys are created if needed
  - backup required when new keys are created

- Deterministic
  - a seed is used to create a master private key
  - the master key is used to create all other private keys
  - you only need to backup the master key

Mnemonic codes and HD Wallets

- Deterministic wallets
  - defined in *Bitcoin Improvement Proposals** (BIPs) 32 and 44.

- Mnemonic codes
  - English words that are used to create the seed of deterministic wallets
  - e.g. by hashing them
  - easier to memorize or write down
  - are defined in BIP-39.

Paper Wallets and Encryption

- Storing offline in a physical document
  - contains only the private key address
  - plus QR codes for convenience

- Paper wallets can be created and printed
  - www.bitaddress.org
  - Bitcoinpaperwallet.com.

- Private key is visible!
  - key can be encrypted using [BIP-38](BIP-38) proposal

- A WIF key begins with ‘5’
- An encrypted WIF begins with ‘6P’

Conclusions

Conclusions

- We introduced basic cryptography required to understand Bitcoin
- We explained what are private keys and public keys in Bitcoin, how they are generated and how addresses derive from them
- We differentiated between deterministic and non-deterministic wallets and explained what paper wallets and mnemonic codes are

Questions ?

Website:	www.kkarasavvas.com
Linkedin:	https://www.linkedin.com/in/kkarasavvas
Twitter:	@kkarasavvas
Email:	kkarasavvas@gmail.com
Bitrated:	https://www.bitrated.com/kostas
Keybase:	https://keybase.io/kkarasavvas