The cryptography of Bitcoin

Dr Douglas Stebila

School of Mathematical Sciences &
School of Electrical Engineering and Computer Science

Queensland University of Technology

20 May 2014

http://www.douglas.stebila.ca/research/presentations/

Overview of Bitcoin

What is Bitcoin?

Bitcoin is a decentralized distributed system for establishing a public ledger of transactions.

Basic idea

1. There's a public ledger that everyone can read with everyone's balance.
2. Alice wants to pay Bob 3 units.
3. Alice requests to put a transaction in the ledger saying "Alice pays Bob 3 units."
4. The maintainer of the ledger checks (a) that Alice has big enough balance and (b) that Alice really made the request, then records the transaction in the ledger.
5. Bob now has a higher balance.

Problems with the basic idea

No anonymity

- Use public keys rather than names.
- Use transaction references rather than accounts.

How to verify someone has authorization to spend from Alice's account?

- Use digital signatures to demonstrate ownership of currency from previous transaction.

Who maintains the ledger?

- Distributed ledger: incentivize community to maintain.

Transaction

"Alice pays Bob 3 units."

"Alice transfers control of 3 units to Bob."

Input:

- Previous transaction ID.
- Public key used in previous transaction.
- Digital signature using based on previous transaction's public key.

Output:

- Bob's address
- # of units
  - Bitcoin address = hash of public key
- Should include own address to "make change"

Transaction

Input: transaction 24d89c02e7ba1

public key 3048c9d000a11789ed	signature 9b8d910afa0b0476c

Output:

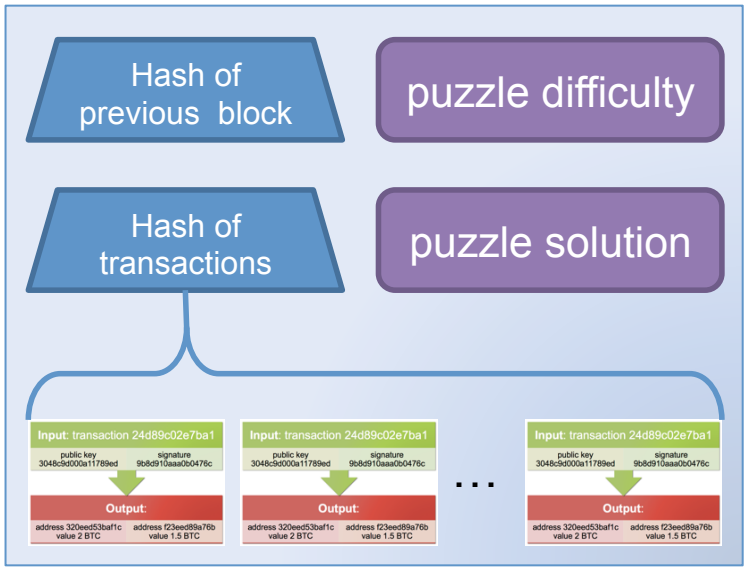
address 320e1d53baf1c value 2 BTC	address f23ea089a76b value 1.5 BTC

Block

Header

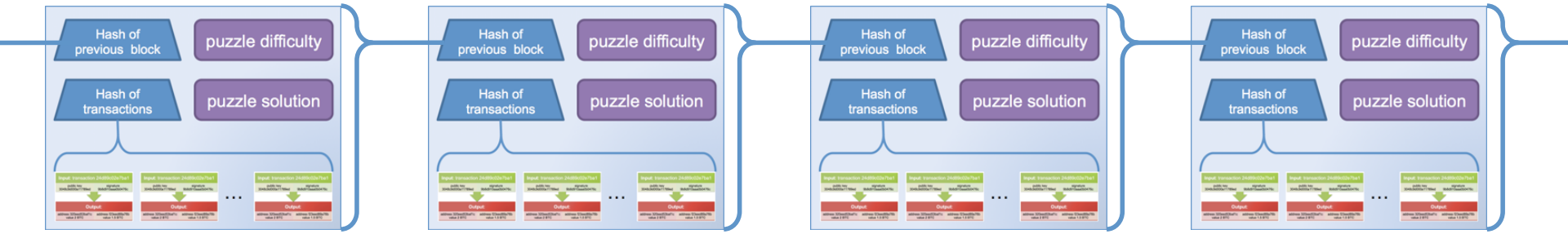
+

a list of transactions



Blockchain

A sequence of blocks = ledger of transactions



Which blockchain?

Blocks form a tree.

- Could have forks in the tree.
- Only the longest chain is considered to be valid by the community.

Adding blocks to the chain

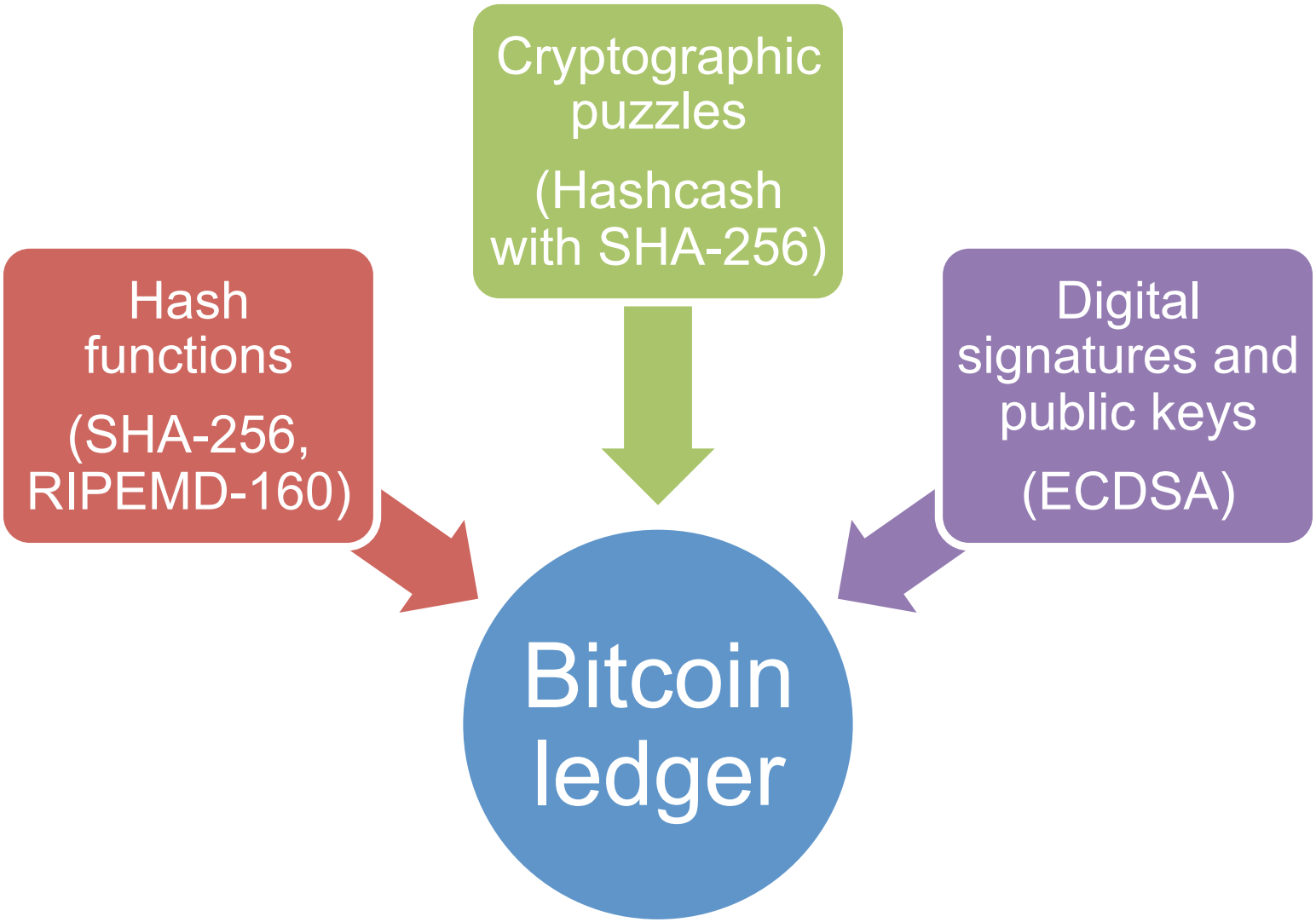
A block can only be added to the blockchain if the hash of the block is small.

- Users try to generate a block with a small hash.
  - ("cryptographic puzzle")

- Updating the blockchain requires work but maintains the public ledger.

- Motivation: whoever constructs the block includes one transaction paying themselves 25 BTC ("**mining**")

Cryptographic ingredients



Hash functions

Hash functions

$$H : \{0,1\}^* \rightarrow \{0,1\}^{\lambda}$$

A public function *H* that is

- fast and easy to compute
- takes as input arbitrary-length binary strings
- outputs a message digest of fixed length

Security properties of hash functions

Collision-resistant

It should be hard to find any two different inputs x_1 and x_2 such that

$H(x_1) = H(x_2)$.

One-way (preimage resistant)

Given a value y, it should be hard to find any input x such that

$H(x) = y$.

Second-preimage resistant

Given an input x_1, it should be hard to find a different input x_2 such that

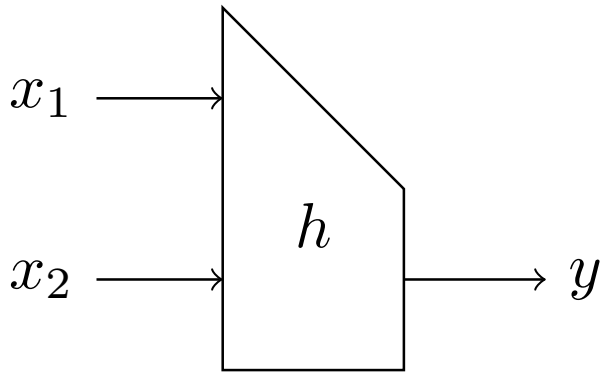
$H(x_1) = H(x_2)$.

Building cryptographic hash functions

Cryptographic hash functions need to take arbitrary-sized input and produce a fixed size output.

Idea: use a fixed-size **compression function** applied to multiple blocks of the message.

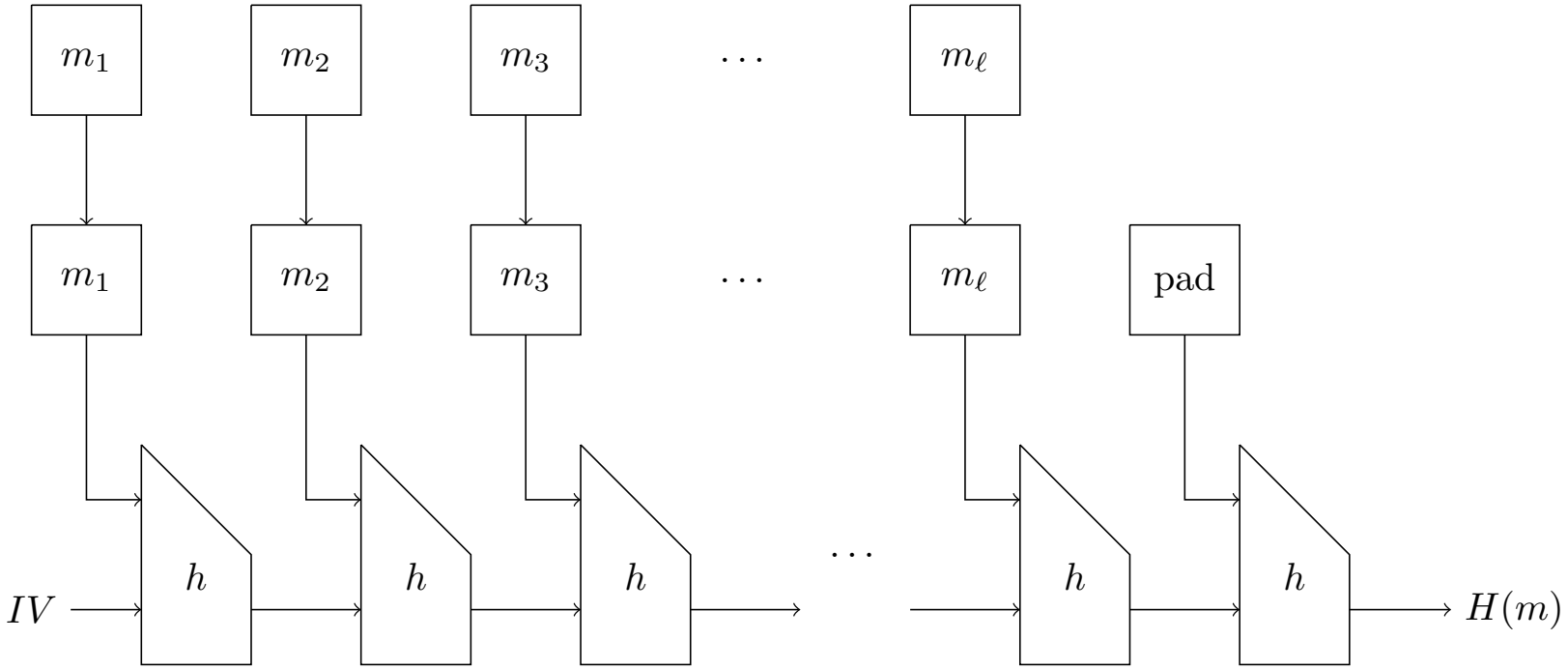
Compression function



$$h : \{0,1\}^\lambda \times \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda$$

Merkle–Damgård construction

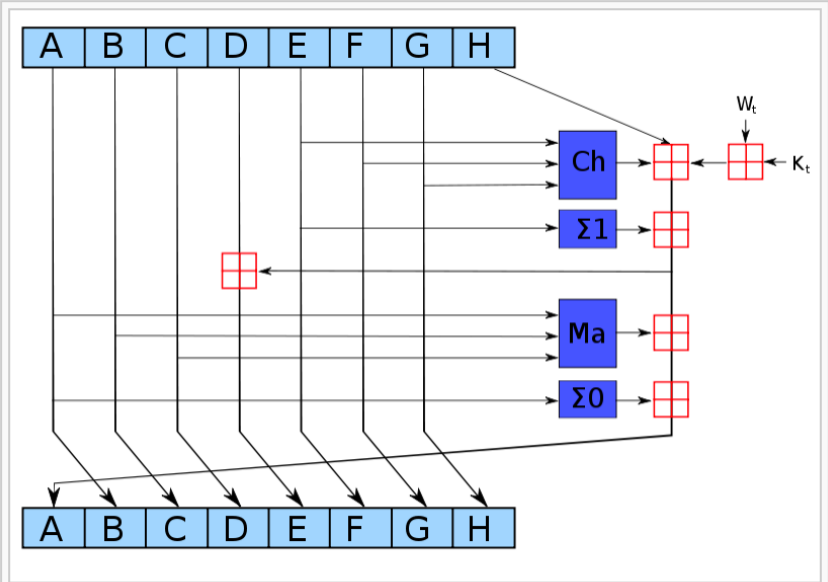
- Break message *m* into λ-bit blocks $m_1 \| m_2 \| \ldots \| m_\ell$
- Add padding.
- Input each block into compression function *h* along with chained output; use standardized initialization vector *IV* to get started.



SHA-256

- Part of the SHA-2 family standardized by NIST in 2001.
- Merkle–Damgård construction.
- Compression function is 64 iterations of function at right.

- No known attacks on SHA-256 (yet) but progress on simplified / reduced-round versions.



One iteration in a SHA-2 family compression function. The blue components perform the following operations:

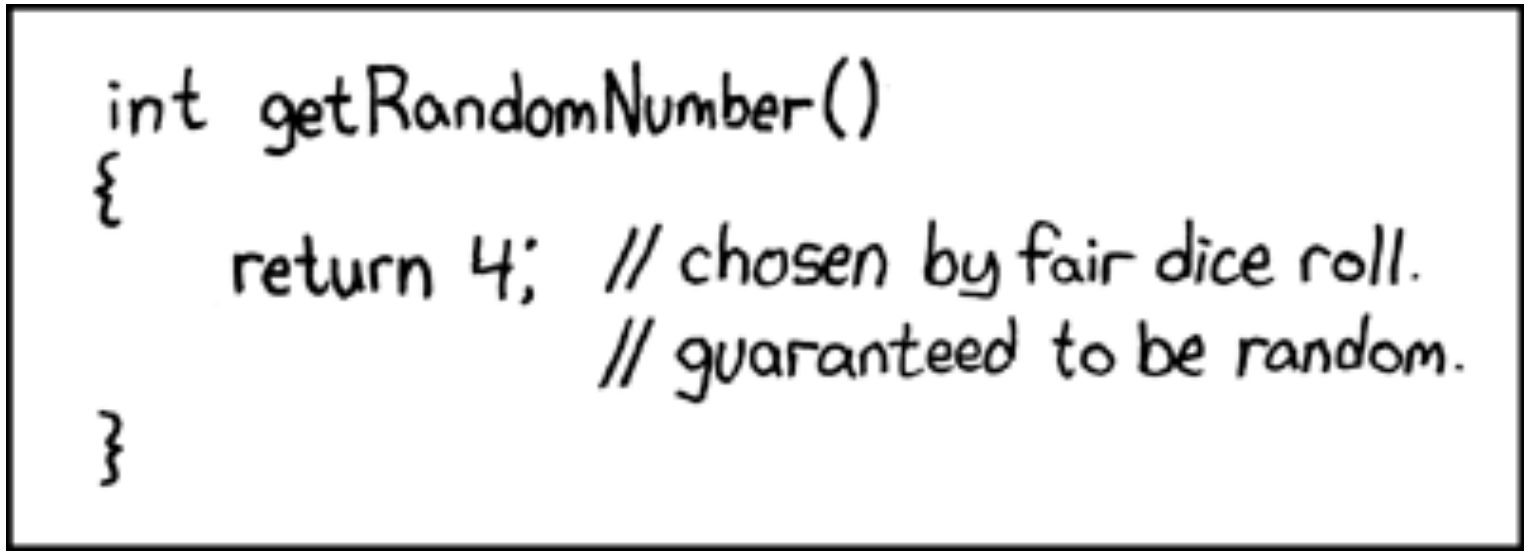
$$\mathrm{Ch}(E,F,G) = (E \wedge F) \oplus (\neg E \wedge G)$$
$$\mathrm{Ma}(A,B,C) = (A \wedge B) \oplus (A \wedge C) \oplus (B \wedge C)$$
$$\Sigma_0(A) = (A \ggg 2) \oplus (A \ggg 13) \oplus (A \ggg 22)$$
$$\Sigma_1(E) = (E \ggg 6) \oplus (E \ggg 11) \oplus (E \ggg 25)$$

The bitwise rotation uses different constants for SHA-512. The given numbers are for SHA-256. The red ⊞ is addition modulo 2^{32}.

http://en.wikipedia.org/wiki/SHA-2

Randomness

- SHA-256 is not random: it is a deterministic function.
- Does it "look random"?
- How can we tell if a function is random?



http://xkcd.com/221/

Pseudorandomness

"Avalanche effect": changing 1 bit of the input should change around half of the output bits.

Golomb's postulates for sequences.

- Assuming SHA-256 is "random" is a stronger assumption than assuming it's collision-resistant / one-way / second-preimage-resistant.
- No known attacks distinguishing SHA-256 from random.

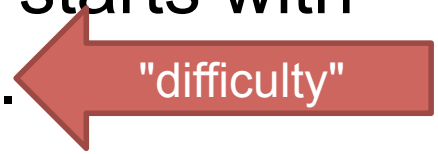
Cryptographic puzzles

Cryptographic puzzle

A "moderately hard" computational task.

Example:

- Let H be a hash function with 256 bits of output.
- Find a value x such that $H(x)$ starts with 32 zeros.



Analysis:

- Assume H is a random function (output bits are independent and identically distributed).
- Then for each different input x and each i, the probability that the ith bit of $H(x)$ is zero is ½.
- The probability that the first 32 bits of $H(x)$ are all zero is $1 / 2^{32}$.
- Need to try about 2^{31} different x values on average to find a satisfying value.

Hashcash cryptographic puzzle

Example:

- Let *H* be a hash function with λ bits of output.
  - Interpret output as an integer between 0 and $2^{\lambda}-1$
- Let *s* be a string.
- Let *t* be an integer.

- Find a value *x* such that $H(s \,||\, x) \leq t$.

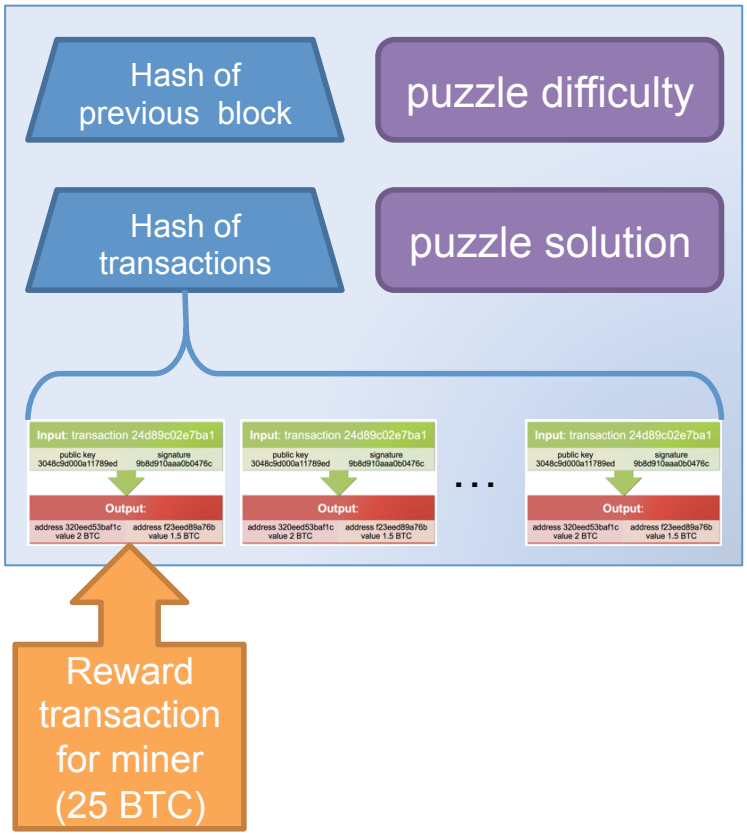
Puzzles in Bitcoin

Every miner is trying to construct a block header where

$H(H(\text{block header} \,||\, \text{solution})) \leq \text{difficulty target}$

H = SHA-256

Keep trying random solutions until one works



Bitcoin mining

Difficulty target adjusted every 2 weeks so that average block generation time is 10 minutes.

Current mining rate:

- 75.7 quadrillion (approx. 2^{56} hashes) per second
  - http://blockchain.info/stats, 2014/05/19

Mining pools

Since finding the solution to a new block is so unlikely individually, miners work together in pools.

If anyone in the pool finds the solution to the puzzle, the whole pool shares the reward.

How to split the reward?

- Just like Bitcoin mining, but with a higher difficulty target
- Pool miners submit whenever they find a hash less than the pool difficulty target
- Even if it's not a valid Bitcoin block, it still demonstrates that you are working hard
- Reward split based on number of submitted hashes

scrypt

An alternative cryptographic puzzle used in other cryptocurrencies e.g. Litecoin.

Bitcoin's cryptographic puzzle is computationally bound.

- Easy to run on low memory GPUs or small custom ASICs.

scrypt is memory-bound.

- Needs large amount of memory.
- Won't work well on GPUs.
- Expensive to build custom ASICs.

Digital signatures

Message authentication

How can we be sure Alice really sent a message?

Symmetric message authentication codes:

- Alice and Bob share a secret key k
- Alice computes $t = \text{MAC}(k, m)$
- Alice sends (m, t)
- Bob checks if $t = \text{MAC}(k, m)$

Problem: how do Alice and Bob share a secret key in the first place?

Problem: How can anyone publicly verify the authentication?

- secret key cryptography
- public key cryptography

Digital signatures

Key generation:

Alice generates a pair of related keys:

- verification key *vk*
  - published in a phone book / transaction record
- signing key *sk*
  - kept secret by Alice

Sign(*sk*, *m*):

Alice uses her signing key *sk* to generate a signature σ

Verify(*vk*, *m*, σ):

Anyone can use Alice's verification key *vk* to check if σ corresponds to *m*

Security goals of digital signatures

Key recovery

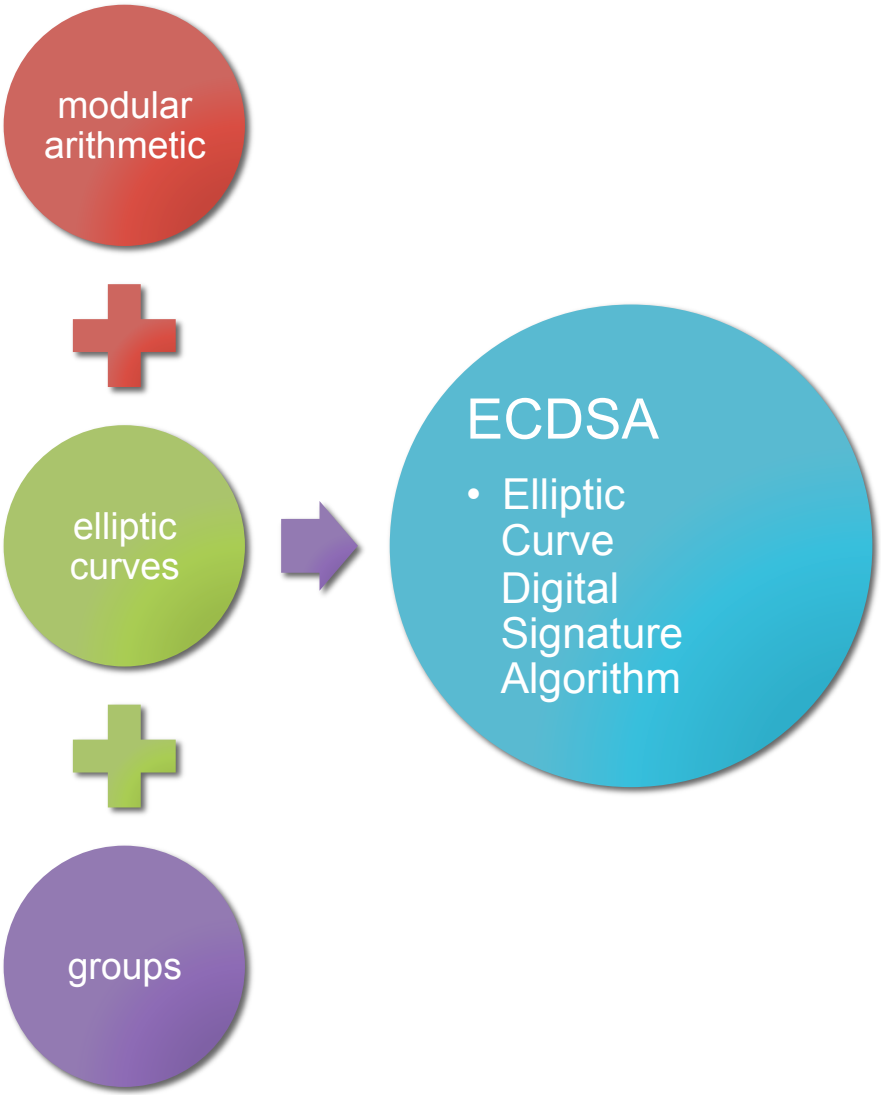
It should be hard compute Alice's signing key *sk* given just her verification key *vk*.

Unforgeability

It should be hard to forge a new valid message-signature pair, given Alice's verification key.

- Forged message doesn't have to be meaningful.
- Even given copies of other signatures.
- Even if attacker can choose which messages are signed.

Building a digital signature scheme

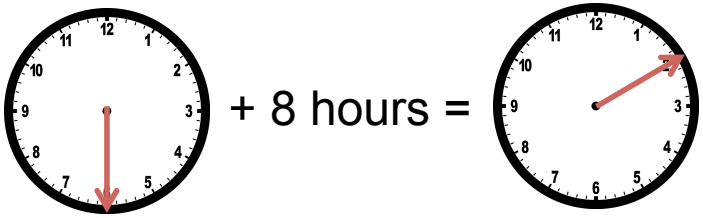


Modular arithmetic

"Clock" arithmetic

Example:

6 o'clock + 8 hours
= 14 o'clock
= 2 o'clock



Modular arithmetic

m: modulus

$r = a \bmod m$

- *r*: the remainder you get when you divide *a* by *m*

Example:

- $14 \bmod 12 = 2$
- $6 + 8 \bmod 12 = 2$
- $2 \times 7 \bmod 12 = 2$

Modular exponentiation

Let g, x, and m be positive integers.

g^x mod m represents multiplying g by itself mod m for x times

- Can compute g^x mod m efficiently even for very large (500+ digit) values using square-and-multiply algorithm.

Discrete logarithm problem

DLP for mod. exp.

1. Let g and m be positive integers.
2. Let x be picked randomly from 0 to $m-1$.
3. Compute $y = g^x \bmod m$.
4. Given (g, m, y), find x.

Difficulty

Intuitively, DLP for modular exponentiation is hard because mod m makes things wrap around in an "unpredictable" way.

Primitive roots

Exponentiation mod 7

g	g^2	g^3	g^4	g^5	g^6
1	1	1	1	1	1
2	4	1	2	4	1
3	2	6	4	5	1
4	2	1	4	2	1
5	4	6	2	3	1
6	1	6	1	6	1

Primitive roots

Notice that some values of g generate all the values from 1 to $m-1$.

Such g are called generators or primitive roots.

Abelian groups

("because making things abstract makes them better")

An abelian group $(G, \times)$ is a set G and an operation $\times$ such that:

- $\times$ is associative:
  $a \times (b \times c) = (a \times b) \times c$
- $\times$ has an identity 1 such that
  $1 \times a = a = a \times 1$
- $\times$ has inverses: every a has a b such that
  $a \times b = 1$
- $\times$ is commutative:
  $a \times b = b \times a$

A cyclic group of order q is a group G that has a generator g such that g, g^2, g^3, g^4, …, g^{q-1} is exactly the set of elements of G.

Example:

- integers modulo a prime with multiplication are an abelian group

Digital signatures from abelian groups

Let g be the generator of a cyclic group of prime order q.

Let H be a hash function.

Key generation:

- pick x randomly between 0 and q−1
- verification key: $vk = g^x$
- signing key: $sk = x$

Sign(sk, m):

- pick k randomly between 0 and $q-1$
- $r = g^k \bmod q$
- $s = k^{-1}(H(m)+xr) \bmod q$
- signature: $\sigma = (r, s)$

Verify(vk, m, σ):

- $w = s^{-1} \bmod q$
- $a = H(m) \times w$
- $b = r \times w \bmod q$
- $v = g^a \times y^b \bmod q$
- valid if $v = r$

Attacking the signature scheme

Key generation:

- pick x randomly between 0 and $q-1$
- verification key: $vk = g^x$
- signing key: $sk = x$

Sign(sk, m):

- pick k randomly between 0 and $q-1$
- $r = g^k \bmod q$
- $s = k^{-1}(H(m)+xr) \bmod q$
- signature: $\sigma = (r, s)$

Verify(vk, m, σ):

- $w = s^{-1} \bmod q$
- $a = H(m) \times w$
- $b = r \times w \bmod q$
- $v = g^a \times y^b \bmod q$
- valid if $v = r$

If you could find x given g^x, you could recover the signing key and forge signatures.

- "discrete logarithm problem"

If you could find $m_1 \neq m_2$ such that $H(m_1) = H(m_2)$, then you could confuse a signature for m_1 as a signature for m_2.

- collision-resistance of H

Digital signatures from abelian groups

DSA: Modular arithmetic

- Group is integers modulo a prime p
- For high security, need:
  - $p \approx 2^{2048}$
  - public keys are 2048 bits long
  - signatures are 4096 bits long

ECDSA: Elliptic curves

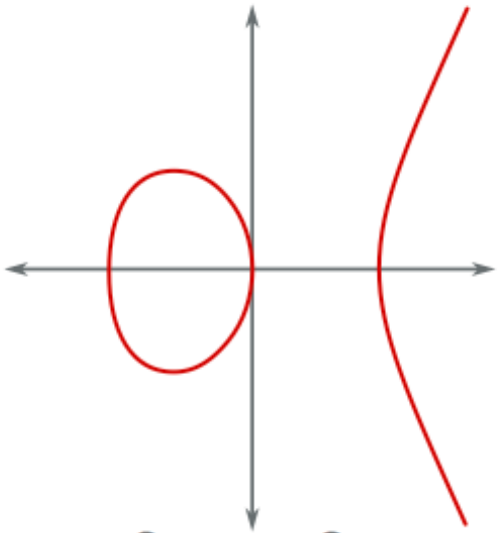
- Group is set of points on a discrete elliptic curve
- For high security, need:
  - **256-bit curve**
  - public keys are 257 bits long
  - signatures are 512 bits long
  - ECDSA is faster and has smaller values for same level of security

Elliptic curve

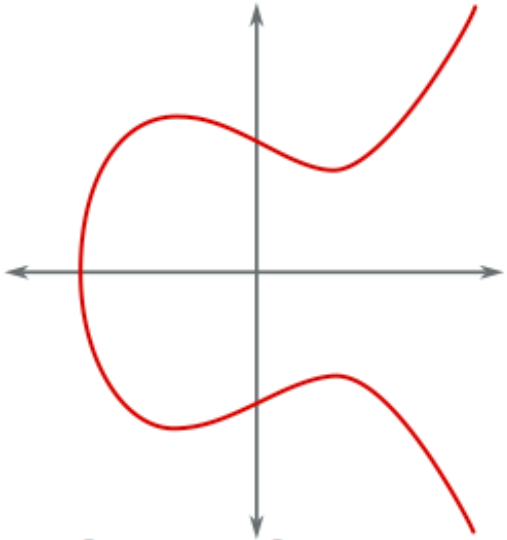
An **elliptic curve over the reals** is the set of real points (x, y) satisfying an equation of the form

$$y^2 = x^3 + ax + b$$

for fixed real numbers a and b.



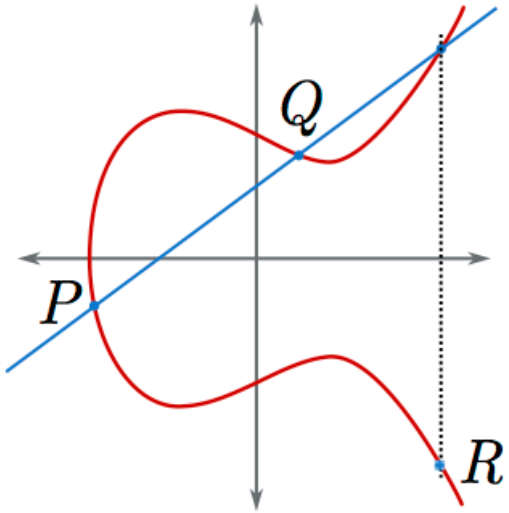
$y^2 = x^3 - x$



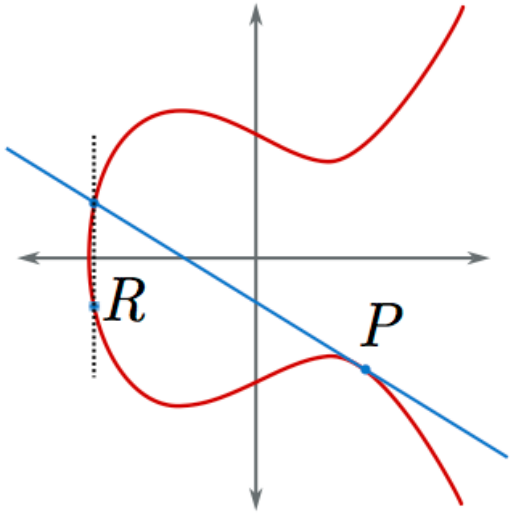
$y^2 = x^3 - x + 1$

Elliptic curve points as a group

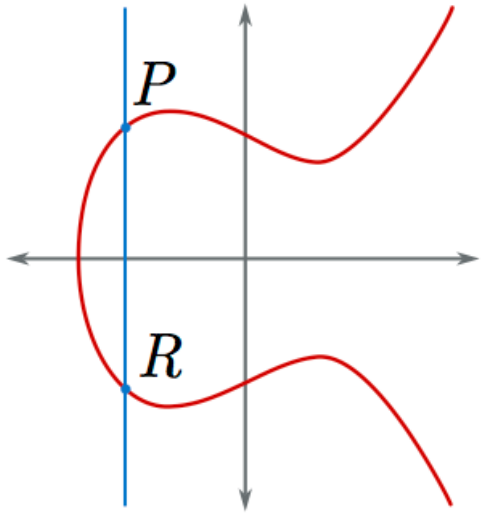
- G = set of points on the curve
- operation = "point addition"
- can make equations from following geometric intuition



$P + Q = R$



$2P = R$



$P + R = O$

Elliptic curve scalar-point multiplication

- Let P be a point on the curve.
- kP represents adding P to itself k times.

Multiplicative groups	Additive groups
multiplication: $g \times h$	addition: $P + Q$
squaring: g^2	doubling: $2P$
exponentiation: g^x	scalar-point multiplication: kP
square-and-multiply algorithm	double-and-add algorithm

Discrete logarithm problem

Multiplicative groups

1. Let g be a generator of a cyclic group of prime order q.
2. Let x be picked randomly from 0 to $q-1$.
3. Compute $y = g^x$.
4. Given (g, q, y), find x.

Additive groups

1. Let P be a generator of a cyclic group of prime order q.
2. Let k be picked randomly from 0 to $q-1$.
3. Compute $Q = kP$.
4. Given (P, q, Q), find k.

Difficulty of DLP

Best known algorithm for DLP that works in every group:

- Pollard's rho algorithm ≈ sqrt(q) operations

The properties of some groups make it easier.

- **mod *p***: number field sieve
  - $p \approx 2^{2048}$ & $q \approx 2^{210}$ => 105-bit security

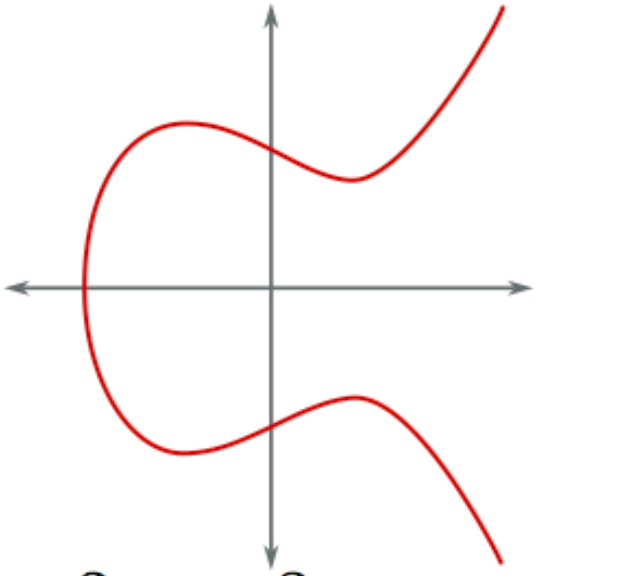
Elliptic curve groups:

- nothing better than sqrt(q)
  - $q \approx 2^{256}$ => 128-bit security

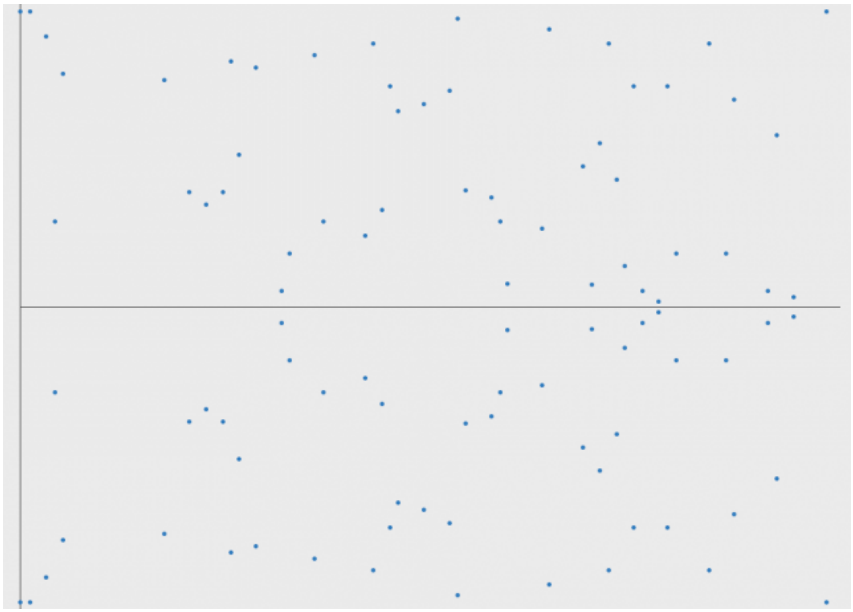
Elliptic curves over prime fields

Use modular arithmetic instead of real numbers:

$$y^2 = x^3 + ax + b \pmod{p}$$



$$y^2 = x^3 - x + 1$$



$$y^2 = x^3 - x + 1 \pmod{97}$$

http://arstechnica.com/security/2013/10/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography/2/

Digital signatures in Bitcoin

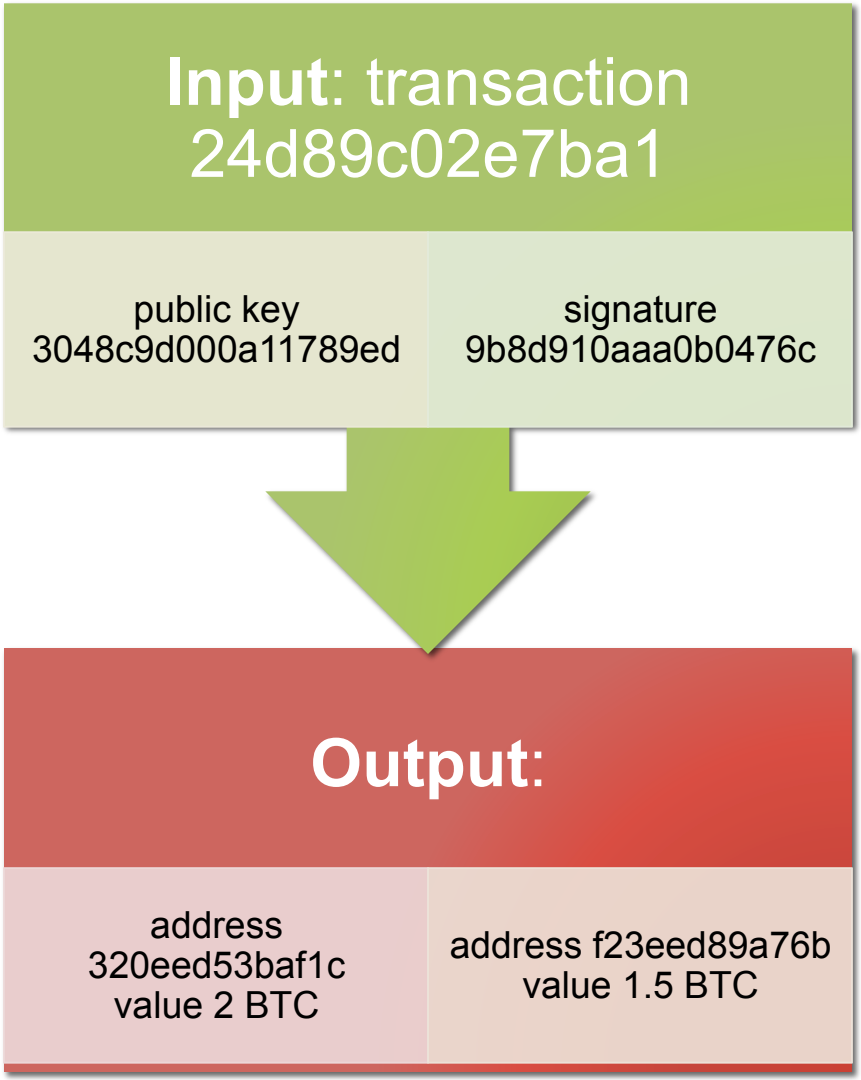
Elliptic curve digital signature algorithm using the NIST p256 elliptic curve group.

- mod $p \approx 2^{256}$

Best known algorithm for forging signatures takes about 2^{128} operations.

- $\approx 2^{48} \approx 10^{24}$ years for 10 million 4GHz computers
- universe is $\approx 10^{10}$ years old

Bitcoin transaction



Public key:

- ECDSA public verification key used in address from previous transaction

Signature:

- signature of transaction using corresponding ECDSA private signing key

Bitcoin address:

RIPEMD-160(

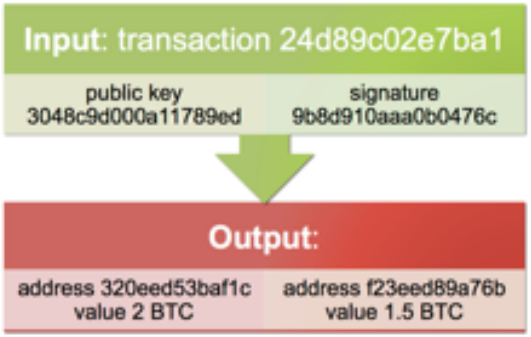
SHA-256(ECDSA public key)

)

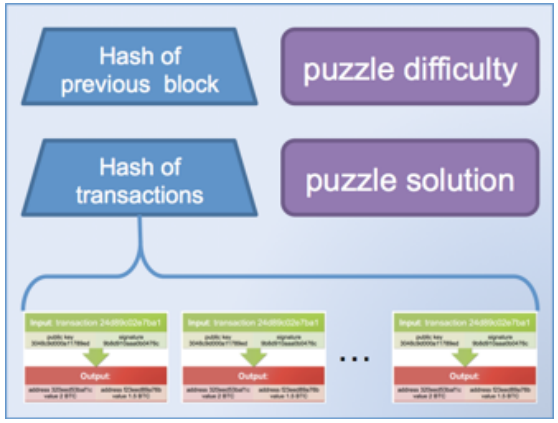
Recap

Cryptographic parts of Bitcoin ledger

Transactions



Blocks



Blockchain

Digital signatures for transaction approval (ECDSA)

Hashed public keys for addresses

Hash used to collect transactions together

Cryptographic hash puzzle required to make block valid (Hashcash SHA-256)

Hash used to chain transactions together (SHA-256)

Only blocks in longest chain considered valid

Breaking Bitcoin via cryptography

Forge transactions

Breaking elliptic curve discrete logarithm with classical computers needs mathematical breakthrough.

- Become a mathematical supergenius.

Quantum computers can easily break ECDLP.

- "Just" need to build a quantum computer.

Mine faster

Figure out how to break partial preimage resistance / pseudorandomness of SHA-256.

- Would break lots of other stuff on the Internet.

Further reading

Bitcoin

Original paper by Satoshi Nakamoto:
https://bitcoin.org/bitcoin.pdf

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

Hash functions
Digital signatures

Handbook of Applied Cryptography:
http://cacr.uwaterloo.ca/hac/

Cryptography by Nigel Smart:
http://www.cs.bris.ac.uk/~nigel/Crypto_Book/

Puzzles

Original Hashcash paper by Adam Back:
http://www.hashcash.org/papers/hashcash.pdf

Elliptic curves

http://arstechnica.com/security/2013/10/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography/

Further study

Queensland University of Technology

MXB251
Number Theory and Abstract Algebra

- mathematics leading up to elliptic curves
- annually starting 2015
- assumes first-year discrete mathematics background

INB355 / INN355
Cryptology and Protocols

- introduces major areas of symmetric and public key cryptography
- annually in semester 2
- no mathematics background assumed