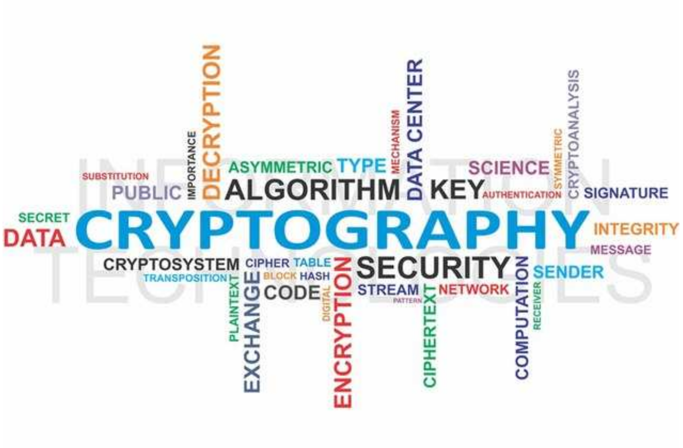
Cryptography review for Blockchain

R. R. Brooks – rrb@acm.org
Clemson University, Electrical and Computer Engineering

July, 2019

Blockchain – Relevant cryptography



□ They are called **Crypto**-currencies

□ We will show you what you need to know to understand and *program* them.

□ Programming will use OpenSSL function calls.

□ Will look at how to compile and use the functions.

□ Uses Elliptic curve cryptography.

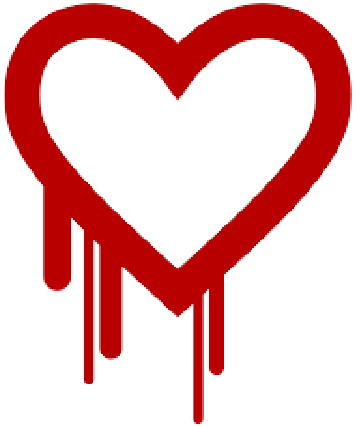
□ BTC uses curve Secp256k1

□ Why that one? I niether know, nor really care.

□ Main tools to understand are hashing and signing.

□ We will explain.

openSSL



- □ Result of NSA and NBS (now NIST) jonit initiative stated in 1986.
- □ SSL == Secure sockets layer. Netscape. v2.0 1995, but first usable version was v3.0 in 1996.
- □ TLS v1.0 is an update to SSL 3.0
- □ OpenSSL is a TLS implementation. Mainly volunteer.

gcc compile line

- □ gcc -g -Og -o program program.c -lssl -lcrypto
- □ -g *outputs symbols for gdb*
- □ -Og *optimizations that do not interfere with debugging*
- □ ssl and crypto libraries included

Debugging

- gdb program *Command line*
- For IDE style gdb in emacs:
  - $ gdb
  - $ gdb-many-windows
  - $ gdb-display-memory-buffer
  - in *gud-data*
    - break main
    - r
    - s

Basic functionality

- □ Cryptology $==$ Cryptography $+$ Cryptanalysis
- □ Cryptography $==$ encryption $+$ decryption
- □ encryption $==$ clear text $\bigoplus$ key $-- >$ cipher text
- □ decryption $==$ cipher text $\bigoplus$ key $-- >$ clear text
- □ Cryptanalysis $==$ cipher text $-- >$ clear text $\sim$ key

Hash

□ Map variable size input to a fixed length number

- Mapping has to be consistent.
- Mapping has to be unpredictable (one way).
- Mapping has to *avoid* collisions.
- Speed of hash function iis usually important.

Symmetric key

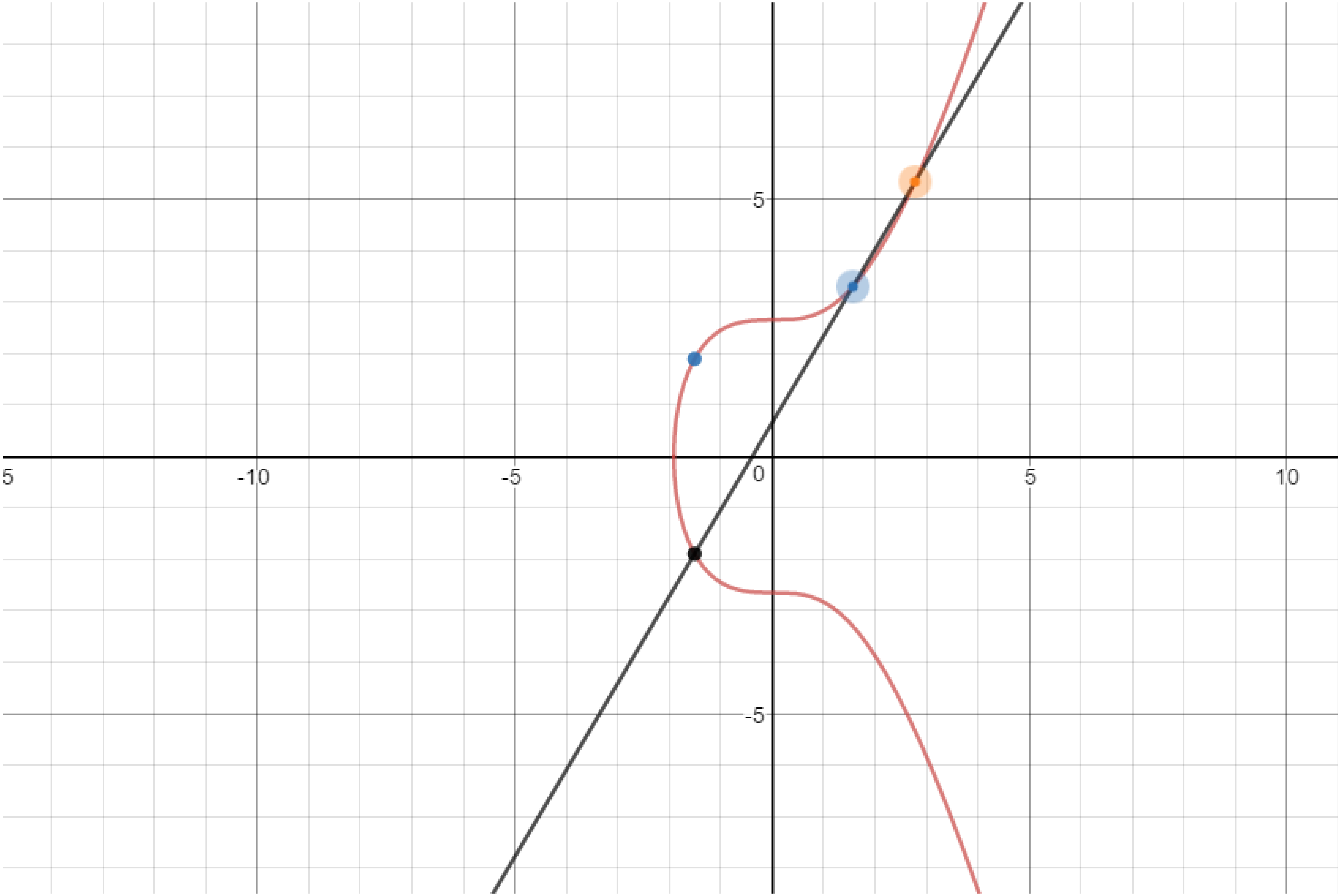
□ Symmetric key has one key.

- Same key encrypts and decrypts.
- Both sides have to share the key.
- Makes key management difficult/important.
- Symmetric key crypto usually faster, easier to do in HW.
- If you use large enough keys, symmetric systems are post-quantum resilient

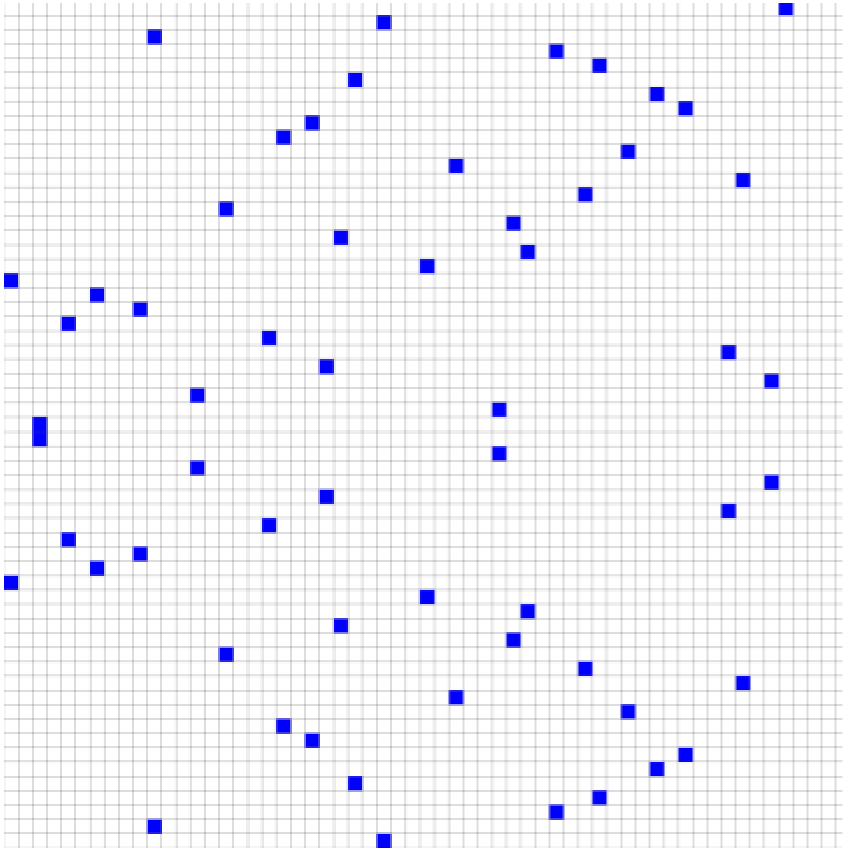
Asymmetric key

- □ Asymmetric key has 2 key.
  - Public key is shared with others, private key is never shared.
  - If public (private) key encrypts, then private (public) key decrypts.
  - Used for either communications security, or showing source.
  - Post-quantum fears are mainly related to public key crypto.
  - RSA is starting to be out of date, public key crypto now mainly elliptic curve crypto.
  - Blockchain systems mainly use elliptic curve.
- □ Common systems, like ssh and tls/ssl, use public key crypto to exchange a symmetric key for the session. Data encrypted using symmetric key.

Secp256k1 – $y^2 = x^3 + 7$



$modulo(2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1)$



Elliptic curve

□ Hardness of *elliptic curve discrete logarithm*[1]

□ Given a curve and a point at infininty,

□ You can pick two points on the curve (public, private),

□ So that, for any value you can encode it with private (public),

□ Combining that result with private (public) gives the orignal,

□ Nothing in particular ties public $< - - - - - - >$ private.

□ Overly simplified, probably not quite right, embarassing

□ But as much of an explanation as you will need for this course.

[1] https://blog.cloudflare.com/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptogr

Key generation – commmand line

- □ List all elliptic curves: openssl ecparam -list_curves
- □ Command line.
  - – Generate key pair:
  - – *Privacy Enhanced Mail* – PEM format de facto standard format for crypto keys
  - – openssl ecparam -genkey -name secp256k1 -noout -out ec256-key-test.pem
  - – Extract public key:
  - – openssl ec -in ec256-key-test.pem -pubout -out ecpubkey.pem
  - – *Distinguished encoding rules* DER binary format is tag, length, value format
  - – openssl ec -in ec256-key-test.pem -pubout -outform DER -out ecpubkey.der

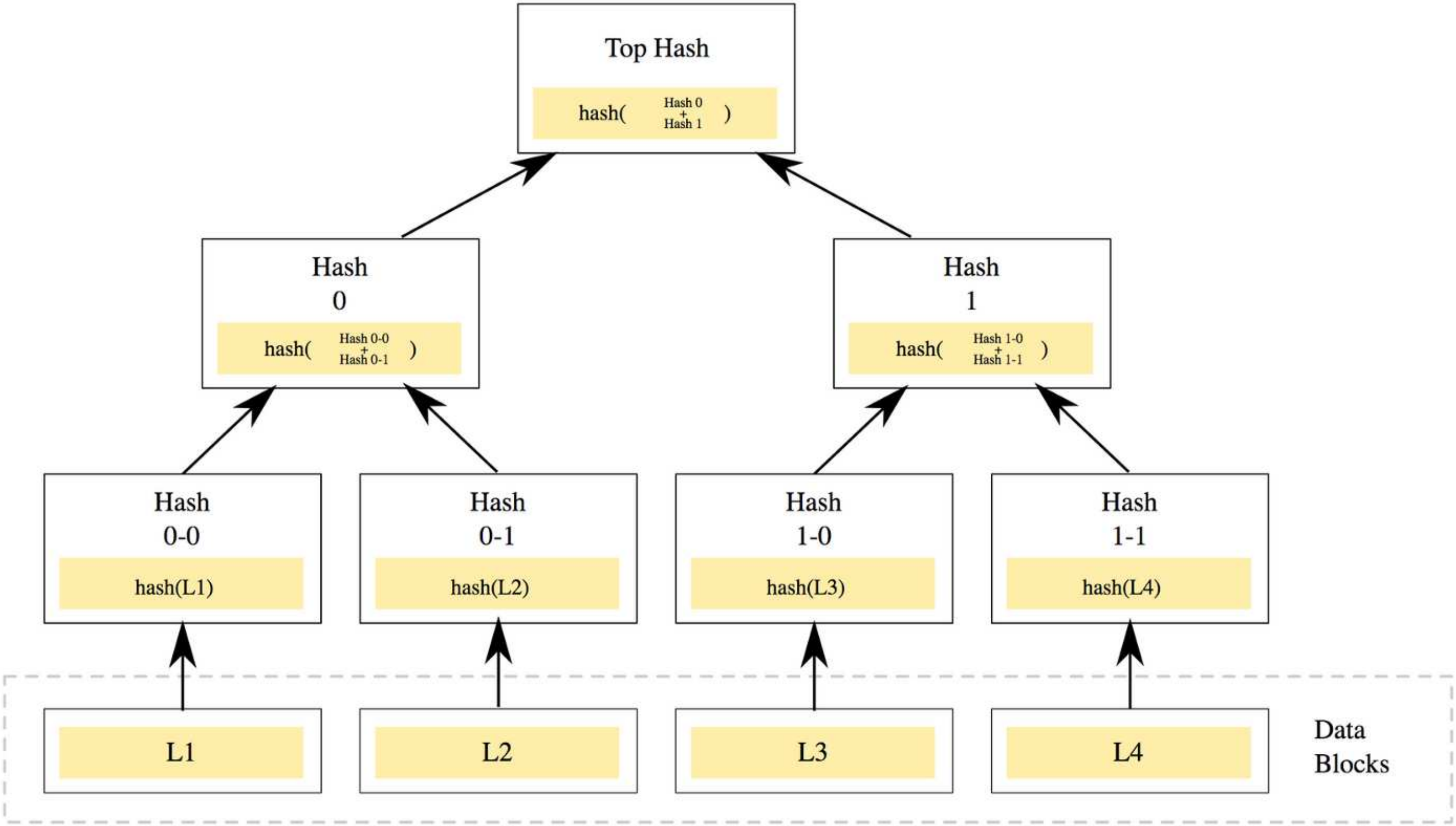
Key generation – C program

□ Look at example program

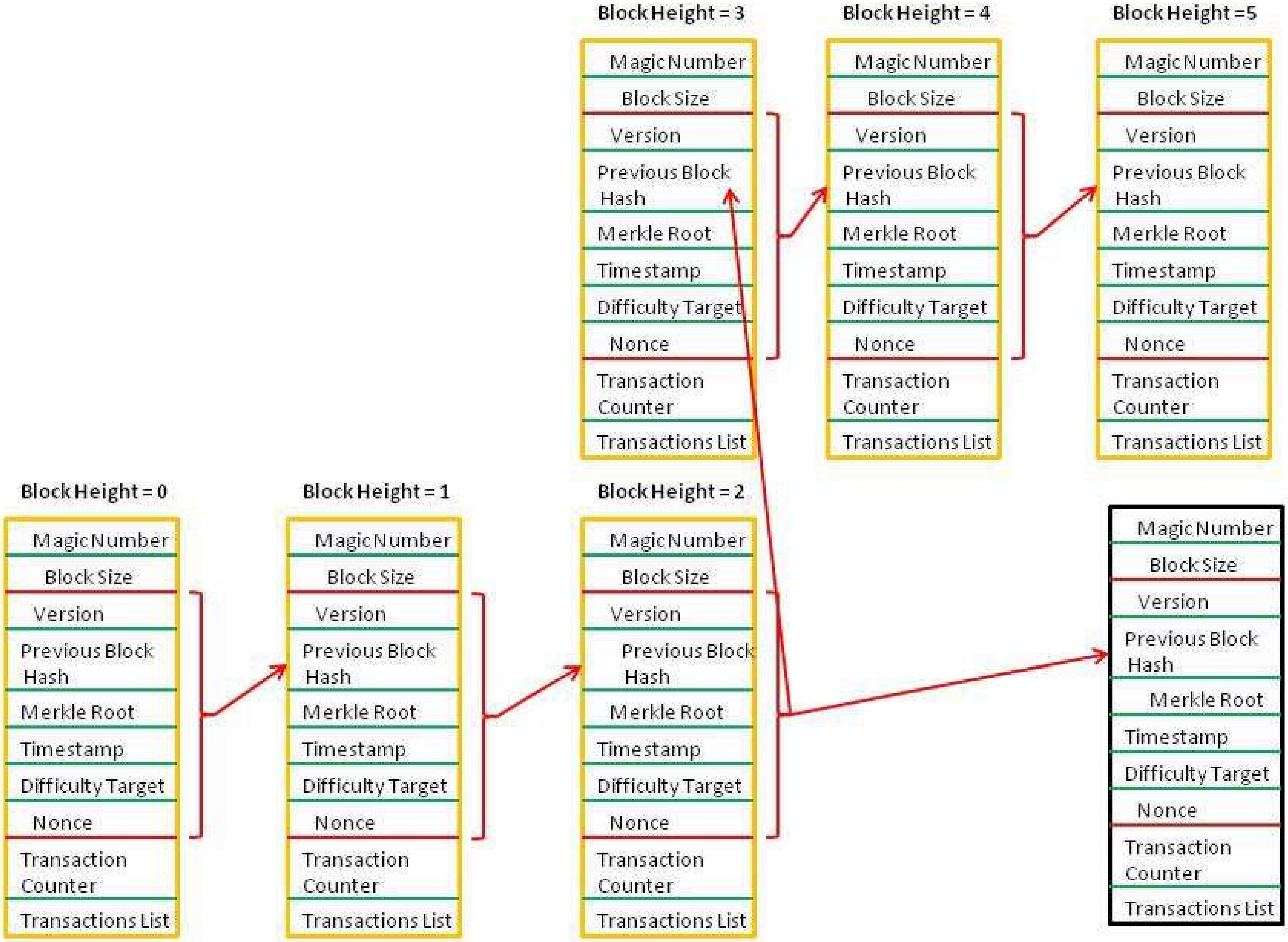
Transaction

- □ `https://www.oreilly.com/library/view/mastering-bitcoin/9781491902639/ch05.html`

Merkle tree


Top Hash
hash(Hash 0 + Hash 1)
Hash 0
hash(Hash 0-0 + Hash 0-1)
Hash 1
hash(Hash 1-0 + Hash 1-1)
Hash 0-0
hash(L1)
Hash 0-1
hash(L2)
Hash 1-0
hash(L3)
Hash 1-1
hash(L4)
L1
L2
L3
L4
Data Blocks

BTC Block



Block Height = 3

Magic Number
Block Size
Version
Previous Block Hash
Merkle Root
Timestamp
Difficulty Target
Nonce
Transaction Counter
Transactions List

Block Height = 4

Magic Number
Block Size
Version
Previous Block Hash
Merkle Root
Timestamp
Difficulty Target
Nonce
Transaction Counter
Transactions List

Block Height =5

Magic Number
Block Size
Version
Previous Block Hash
Merkle Root
Timestamp
Difficulty Target
Nonce
Transaction Counter
Transactions List

Block Height = 0

Magic Number
Block Size
Version
Previous Block Hash
Merkle Root
Timestamp
Difficulty Target
Nonce
Transaction Counter
Transactions List

Block Height = 1

Magic Number
Block Size
Version
Previous Block Hash
Merkle Root
Timestamp
Difficulty Target
Nonce
Transaction Counter
Transactions List

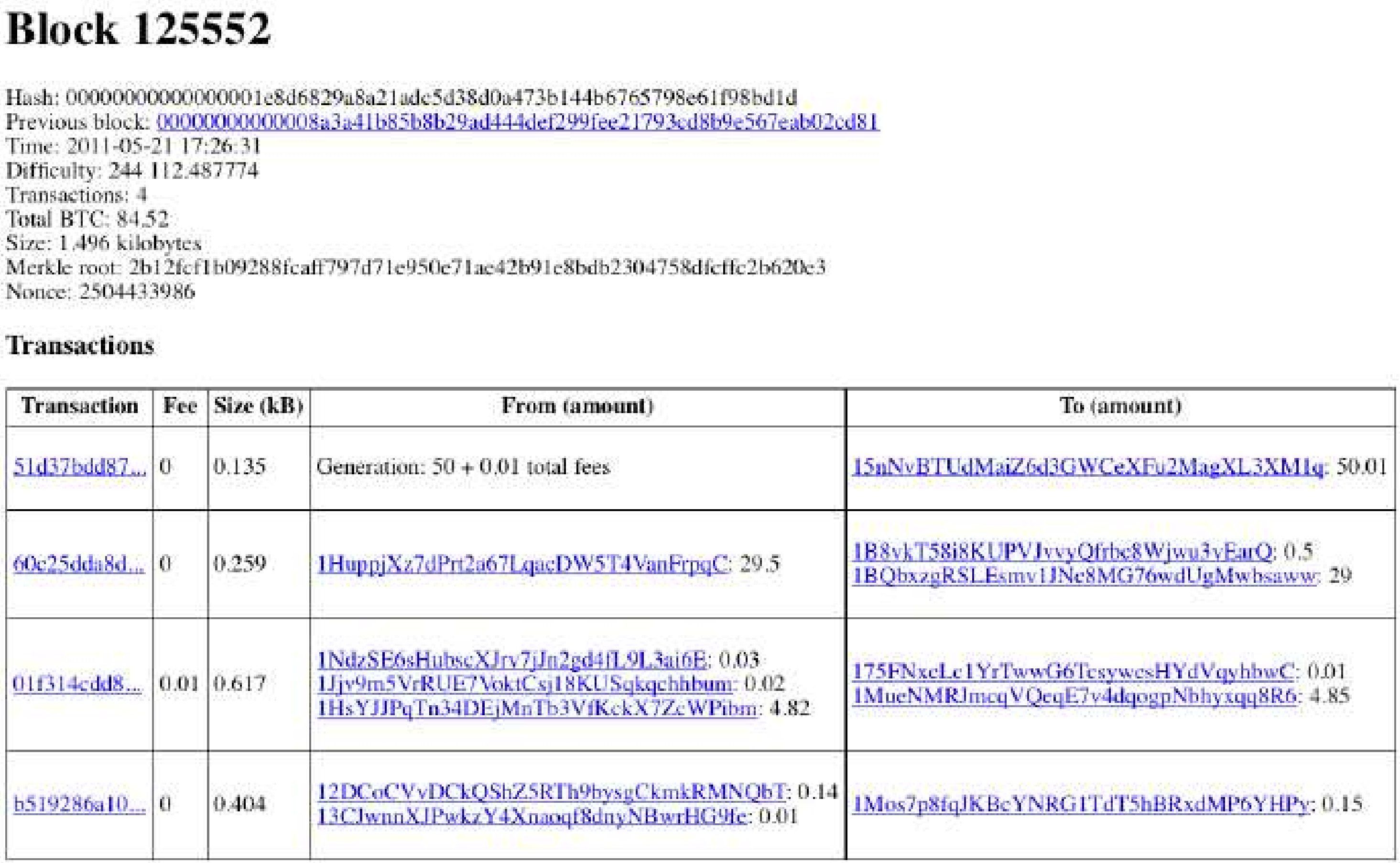
Block Height = 2

Magic Number
Block Size
Version
Previous Block Hash
Merkle Root
Timestamp
Difficulty Target
Nonce
Transaction Counter
Transactions List

Magic Number
Block Size
Version
Previous Block Hash
Merkle Root
Timestamp
Difficulty Target
Nonce
Transaction Counter
Transactions List

BTC Block Example

Introduction
OpenSSL
Cryptography
Public key
Elliptic Curve
Blockchain
Transaction
Merkle tree
BTC Block
BTC Block
▷ Example



Block 125552

Hash: 00000000000000001e8d6829a8a21adc5d38d0a473b144b6765798e61f98bd1d
Previous block: 00000000000008a3a41b85b8b29ad444def299fee21793cd8b9e567eab02cd81
Time: 2011-05-21 17:26:31
Difficulty: 244 112.487774
Transactions: 4
Total BTC: 84.52
Size: 1.496 kilobytes
Merkle root: 2b12fcf1b09288fcaff797d71e950e71ae42b91e8bdb2304758dfcffc2b620e3
Nonce: 2504433986

Transactions

Transaction	Fee	Size (kB)	From (amount)	To (amount)
51d37bdd87...	0	0.135	Generation: 50 + 0.01 total fees	15nNvBTUdMaiZ6d3GWCeXFu2MagXL3XM1q: 50.01
60c25dda8d...	0	0.259	1HuppjXz7dPrt2a67LqacDW5T4VanFrpqC: 29.5	1B8vkT58i8KUPVJvvyQfrbc8Wjwu3yEarQ: 0.5 1BQbxzgRSLEsmv1JNe8MG76wdUgMwbsaww: 29
01f314cdd8...	0.01	0.617	1NdzSE6sHubscXJrv7jJn2gd4fL9L3ai6E: 0.03 1Jjv9m5VrRUE7VoktCsj18KUSqkqchhbum: 0.02 1HsYJJPqTn34DEjMnTb3VfKckX7ZcWPibm: 4.82	175FNxcLc1YrTwwG6TcsywcsHYdVqyhbwC: 0.01 1MueNMRJmcqVQeqE7v4dqogpNbhyxqq8R6: 4.85
b519286a10...	0	0.404	12DCoCVvDCkQShZ5RTh9bysgCkmkRMNQbT: 0.14 13CJwnnXJPwkzY4Xnaoqf8dnyNBwrHG9fe: 0.01	1Mos7p8fqJKBcYNRG1TdT5hBRxdMP6YHPy: 0.15