



UNIVERSITY OF NICOSIA
ΠΑΝΕΠΙΣΤΗΜΙΟ ΛΕΥΚΩΣΙΑΣ

DFIN-511

Introduction to Digital Currencies

Session 5

Bitcoin in practice Part 2

**Bitcoin Core (Bitcoind and Bitcoin-Qt),
constructing a transaction, mining**

Objectives of Session 5

- Understanding Bitcoin Core and how it works
- Understand the functionality of Bitcoin Core
- Get more familiar with the mining process and mining pools



This is the last of the three introductory sessions on the technical aspects around the emerging field of digital currencies. Exploring transactions over the basic Bitcoin client is the goal of this session, along with learning more on mining and mining pools.

In the next sessions we'll explore alternative currencies (altcoins) and alternate uses of the blockchain (2.0 protocols) and then switch back to more business oriented subjects, under the view of digital currencies.

Agenda

1. Bitcoin Core (Bitcoin/Bitcoin-Qt)
2. Bitcoin core functionality
3. Mining and mining pools
4. Segregated Witness
5. Conclusions
6. Self-Assessment Exercises and Further Reading

1. Bitcoin Core (Bitcoin/Bitcoin-Qt)

Bitcoin

Bitcoin was the first client in the history of Bitcoin and it is a *full client* that implements the Bitcoin protocol. It is operated from the command line and can be used to send remote procedure call (RPC)-based commands.

Bitcoin was available for most existing popular operating systems, including Windows, Linux, and MacOS. In addition, it could also be adapted or extended by building from its source code that is found in the Bitcoin *github* repository (<https://github.com/bitcoin/bitcoin>).

In order to install and operate bitcoin you needed experience with the command line. More advanced command line operations within Bitcoin Core will be covered later in other modules of the MSc program.

Bitcoin Core -> Bitcoin-Qt

Bitcoin-Qt (now known as Bitcoin Core) is the third Bitcoin client, developed by Wladimir J. van der Laan and based on the original reference code by Satoshi Nakamoto.

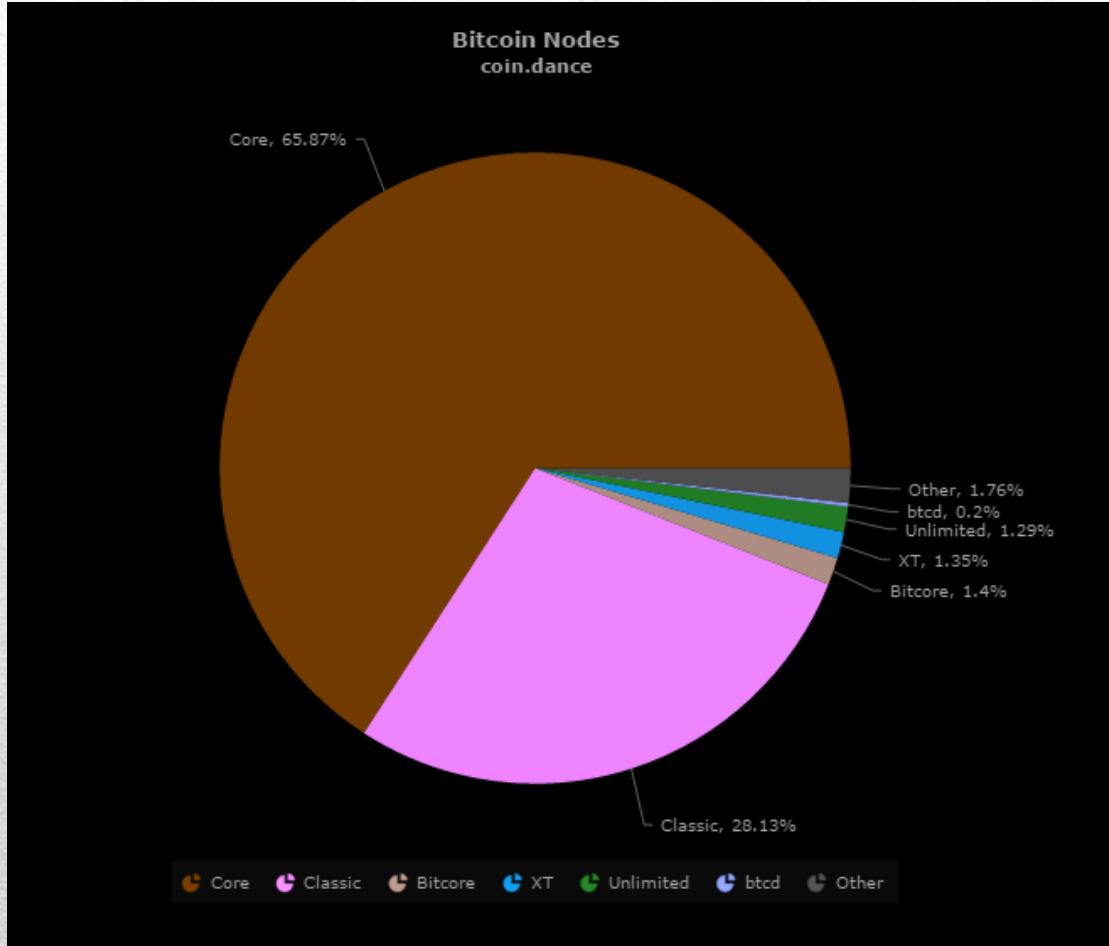
Bitcoin-Qt has been *bundled* with bitcoind since version 0.5 and can be considered to be a Graphical User Interface (GUI) front-end to bitcoind.

Besides Bitcoin Core, other implementations also exist, including :

- **Bitcoin Unlimited**, which is a fork of the Bitcoin Core reference client with the intention of providing a voice to all stakeholders in the Bitcoin ecosystem. The project seeks to remove existing practical barriers to stakeholders expressing their views in these ways. More information [here](#).
- **Bitcoin Classic**, according to the it's developers support the view that "The data shows consensus amongst miners for an immediate 2 MB increase, and demand amongst users for 8 MB or more. We are writing the software that miners and users say they want. We will make sure that it solves their needs, help them deploy it, and gracefully upgrade the bitcoin network's capacity together." Developers include Gavin Andresen and Jeff Garzik. More information [here](#).

Node Distribution

At present, node distribution appears like this :



Top 6 user agents with their respective number of reachable nodes.

RANK	USER AGENT	NODES
1	/Satoshi:0.11.2/	1595 (22.43%)
2	/Satoshi:0.12.0/	1463 (20.57%)
3	/Classic:0.12.0/	1430 (20.11%)
4	/Classic:0.11.2/	524 (7.37%)
5	/Satoshi:0.11.0/	401 (5.64%)
6	/Satoshi:0.11.1/	243 (3.42%)

Sources: coin.dance and bitnodes.21.co

2.Bitcoin core functionality

Bitcoin Installation

You can download the appropriate Bitcoin installer from <https://bitcoin.org/en/download> or you can find more options [here](#)

- For Windows, Mac OSX and Ubuntu Linux the installation is straight forward.
- For different Linux flavors compilation may be required.
If your Linux-based machine runs an appropriate graphical desktop environment and the necessary software development pre-requisites are installed, then both bitcoind and Bitcoin-Qt can be compiled on it.

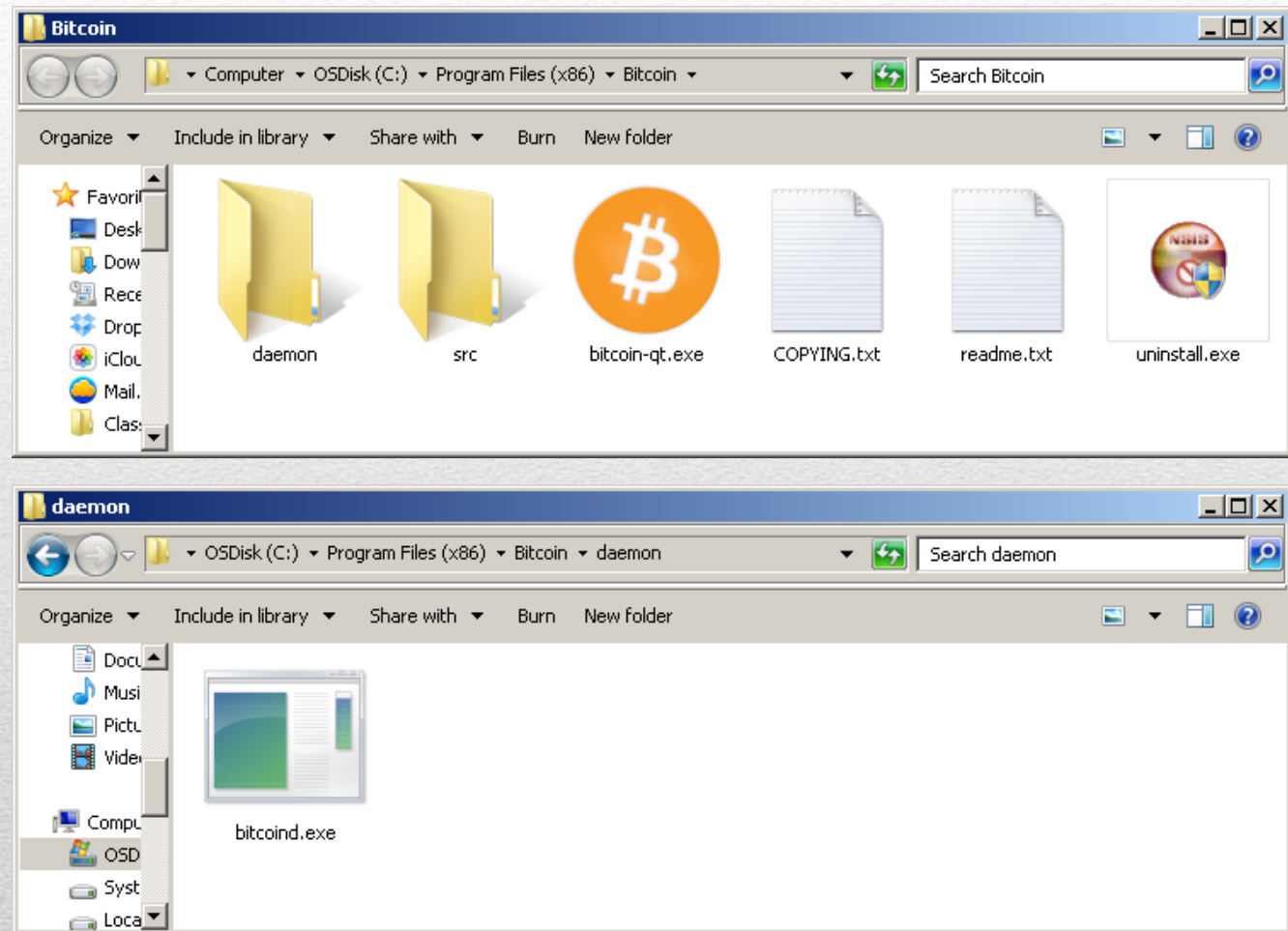
After installation you will have to wait until the initial synchronization of the entire Bitcoin blockchain is done, which may take, depending on your bandwidth and the number of connected Bitcoin nodes, several days.

Bitcoin Core

Demonstration in Windows:

After successful installation, a “Bitcoin” folder is created under “Program files”. Bitcoin-Qt is installed in this folder.

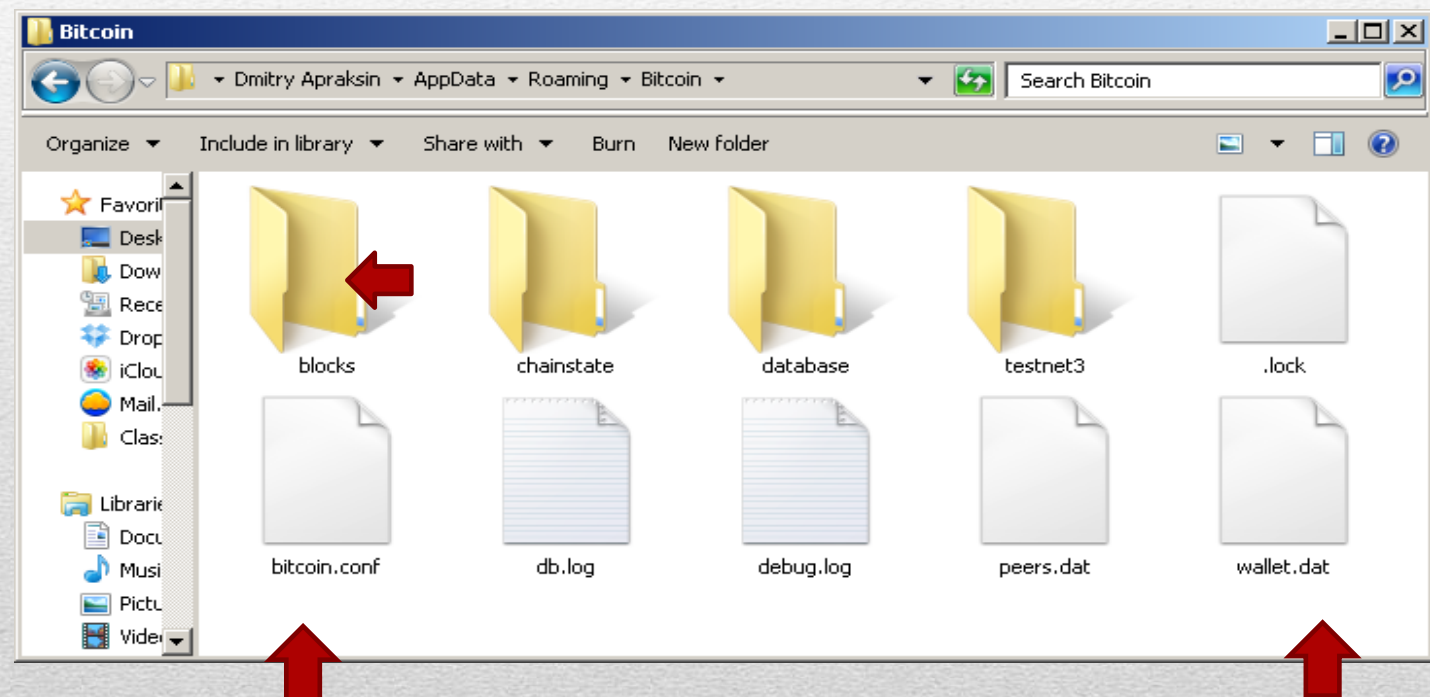
The Bicoind program is found in the “daemon” subfolder. Bitcoin-Qt uses this program to communicate with the Bitcoin network.



Bitcoin- Core

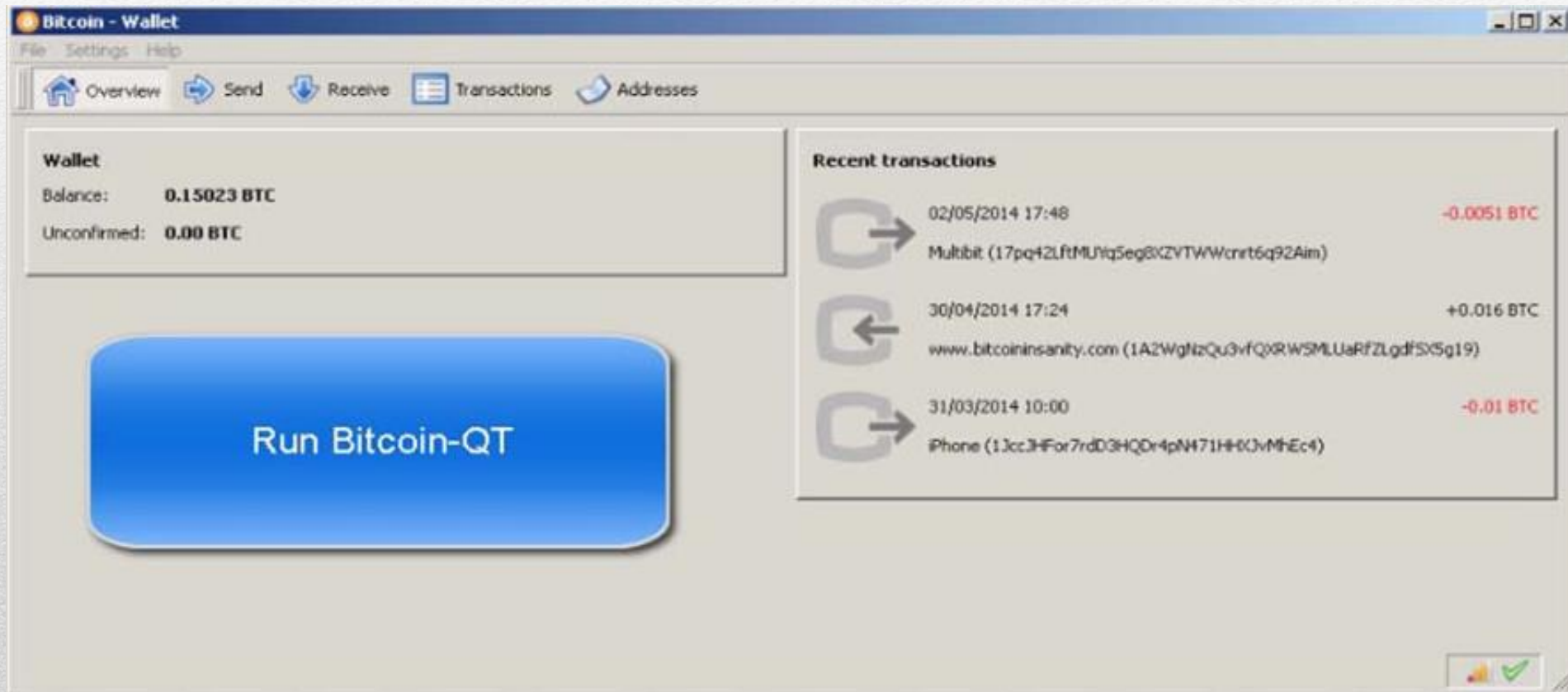
Continuing with the Windows demonstration, the “Bitcoin” folder in a user’s “AppData” folder in Windows is very important because it stores:

- The Bitcoin configuration file
- Your Bitcoin wallet (the wallet.dat file)
- The “blocks” folder which stores a full copy of the Bitcoin blockchain, whose current size is about 44GB



Bitcoin – Command Line Interface (CLI)

Bitcoin offers a number of commands to use from the command line. The same commands can be used through the Bitcoin-Qt GUI from the debug window. Short video demo below:



Command Line Interface (CLI)

The complete reference to the Bitcoin client Application Programming Interface (API) can be found here: https://en.bitcoin.it/wiki/Original_Bitcoin_client/API_calls_list

Using the command line interface you can:

- Get information about the status of the Bitcoin network
- Manage your wallet
- Explore and decode transactions
- Explore blocks
- Create, sign and submit transactions with unspent outputs

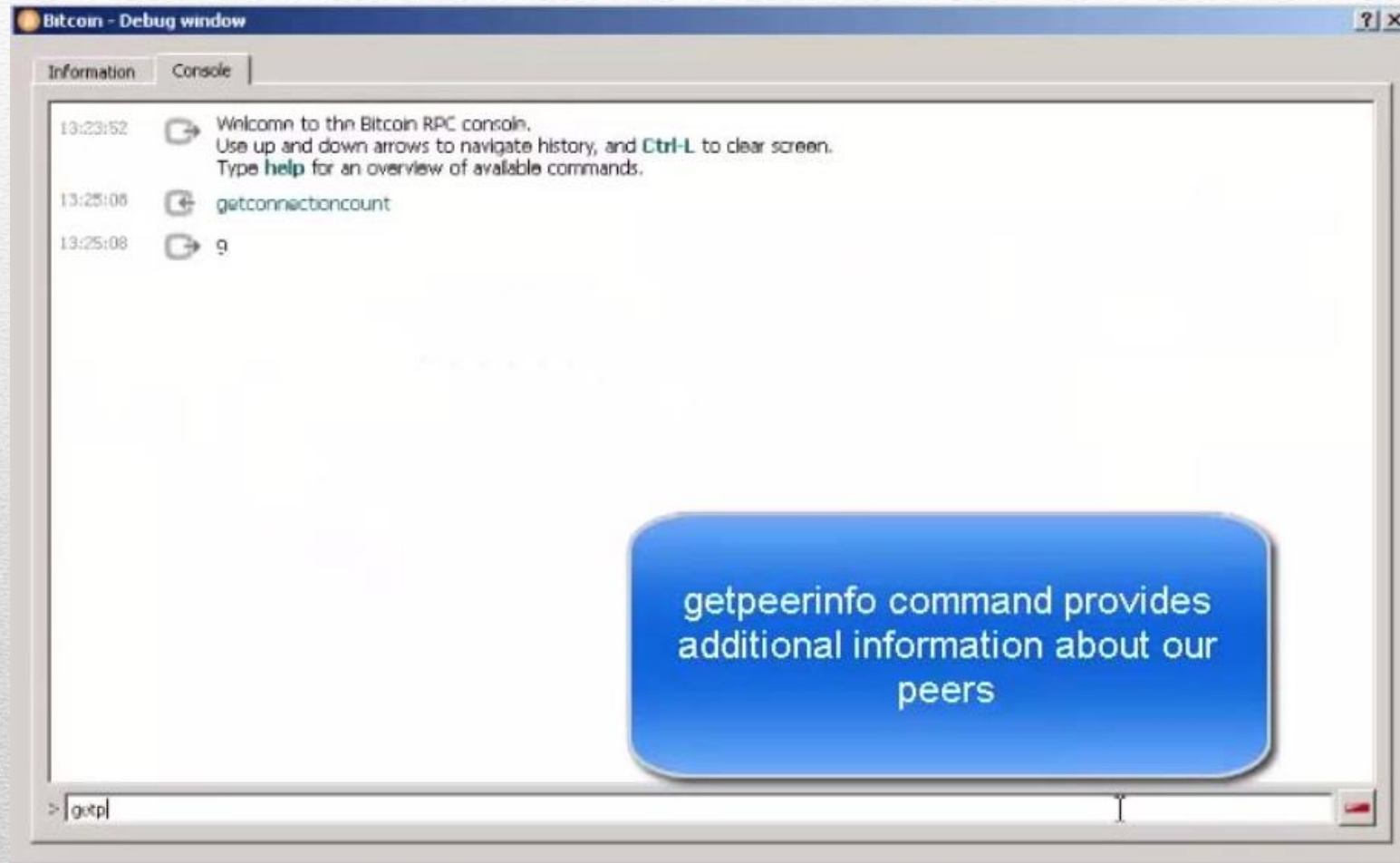
Bitcoin Core – Getting network information

Commands to use:

Command	Description
getconnectioncount	Returns the number of connections to other nodes.
getpeerinfo	Returns data about each connected node.
getdifficulty	Returns the proof-of-work difficulty as a multiple of the minimum difficulty.
getblockcount	Returns the number of blocks in the longest block chain.
getmininginfo	Returns an object containing mining-related information: <ul style="list-style-type: none"> • blocks • currentblocksize • currentblocktx • difficulty • errors • generate • genproclimit • hashespersec • pooledtx • testnet
getgenerate	Returns true or false whether bitcoind is currently generating hashes

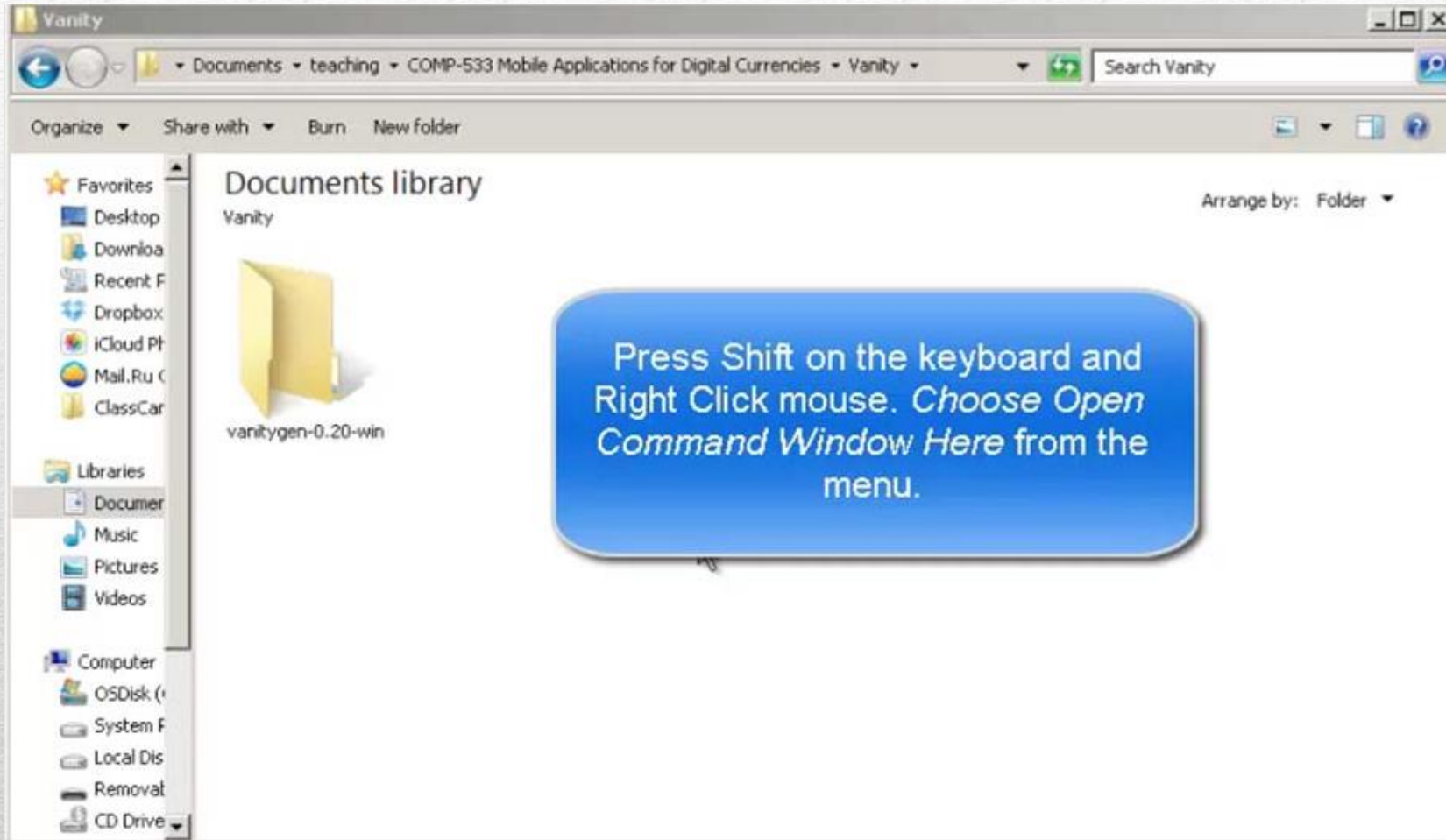
Getting Bitcoin network information

Video illustration below :



Managing your wallet (CLI). Vanitygen

Video illustration below :



Managing your wallet (CLI).

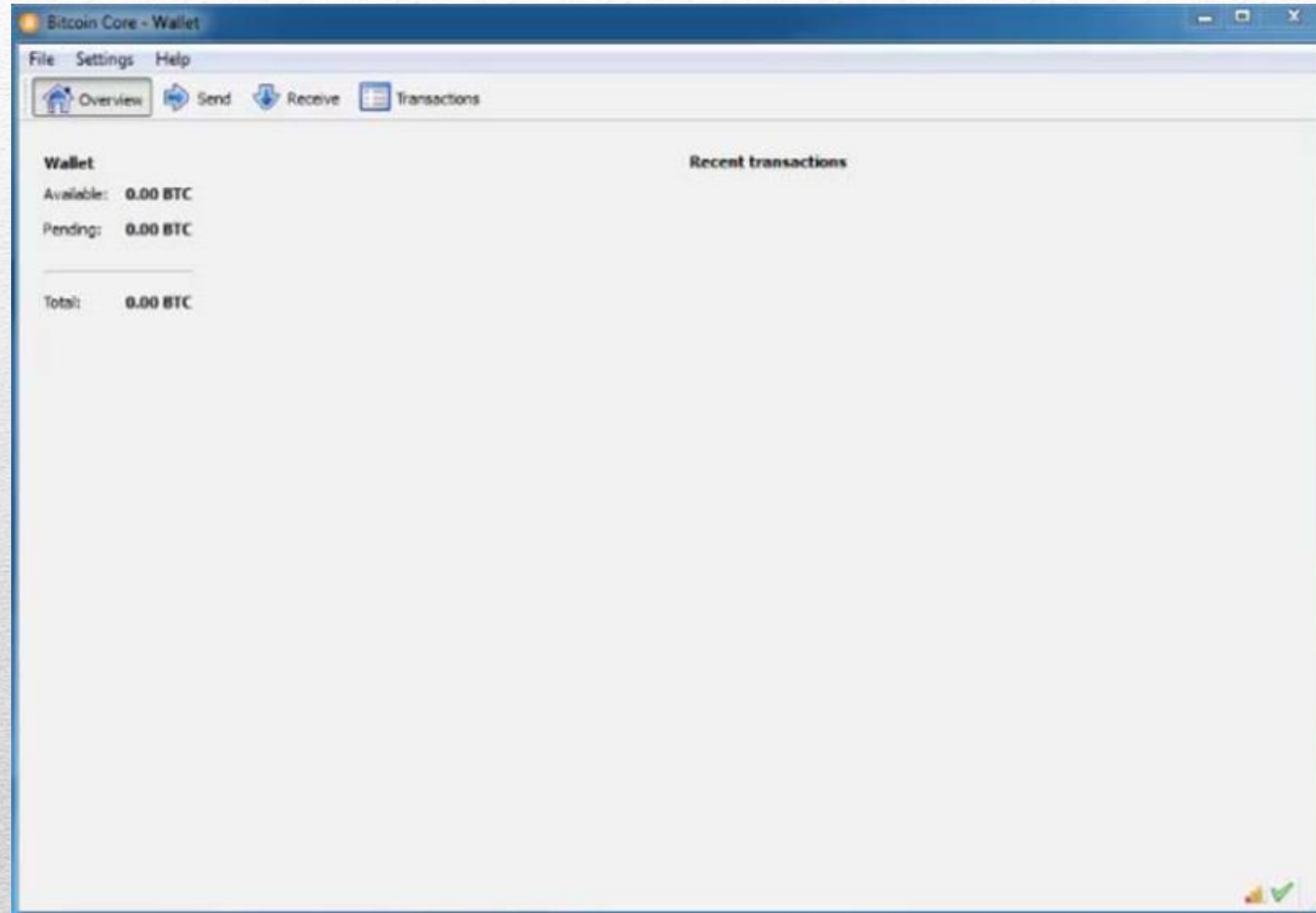
Command	Parameters	Description
getnewaddress	[account]	Returns a new bitcoin address for receiving payments. If [account] is specified payments received with the address will be credited to [account].
dumpprivkey	<bitcoinaddress>	Reveals the private key corresponding to <bitcoinaddress>
importprivkey	<bitcoinprivkey> [label] [rescan=true]	Adds a private key (as returned by dumpprivkey) to your wallet. This may take a while, as a rescan is done, looking for existing transactions. Note: There's no need to import public key, as in ECDSA (unlike RSA) this can be computed from the private key.
getaccountaddress	<account>	Returns the current bitcoin address for receiving payments to this account. If <account> does not exist, it will be created along with an associated new address that will be returned.
getreceivedbyaddress	<bitcoinaddress> [minconf=1]	Returns the amount received by <bitcoinaddress> in transactions with at least [minconf] confirmations. It correctly handles the case where someone has sent to the address in multiple transactions. Keep in mind that addresses are only ever used for receiving transactions. Works only for addresses in the local wallet, external addresses will always show 0.

Managing your wallet (CLI).

Command	Parameters	Description
listtransactions	[account] [count=10] [from=0]	Returns up to [count] most recent transactions skipping the first [from] transactions for account [account]. If [account] not provided it'll return recent transactions from all accounts.
getaddressesbyaccount	<account>	Returns the list of addresses for the given account.
encryptwallet	<passphrase>	Encrypts the wallet with <passphrase>
walletlock		Removes the wallet encryption key from memory, locking the wallet. After calling this method, you will need to call walletpassphrase again before being able to call any methods which require the wallet to be unlocked.
walletpassphrase	<passphrase> <timeout>	Stores the wallet decryption key in memory for <timeout> seconds.
walletpassphrasechange	<oldpassphrase> <newpassphrase>	Changes the wallet passphrase from <oldpassphrase> to <newpassphrase>.

Managing your wallet (CLI). Vanitygen

Video illustration below :

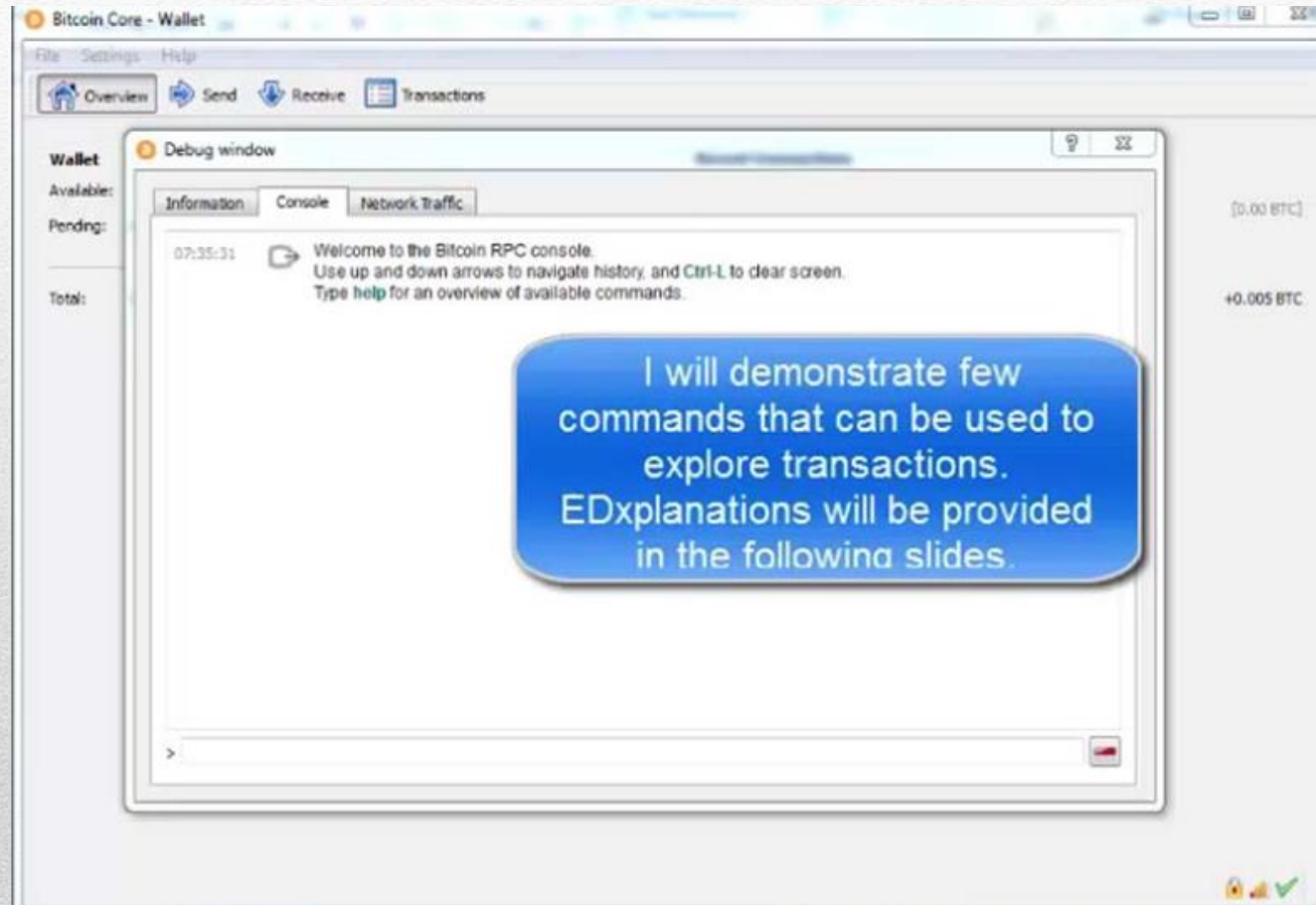


Managing your wallet (CLI).

Command	Parameters	Description
gettransaction	<txid>	<p>Returns an object about the given transaction containing:</p> <ul style="list-style-type: none"> • "amount" : total amount of the transaction • "confirmations" : number of confirmations of the transaction • "txid" : the transaction ID • "time" : time associated with the transaction[1]. • "details" - An array of objects containing: <ul style="list-style-type: none"> • "account" • "address" • "category" • "amount" • "fee"
getrawtransaction	<txid> [verbose=0]	Returns raw transaction representation for given transaction id.
decoderawtransaction	<hex string>	Produces a human-readable JSON object for a raw transaction
getaccountaddress	<account>	<p>Returns the current bitcoin address for receiving payments to this account. If <account> does not exist, it will be created along with an associated new address that will be returned.</p>
getreceivedbyaddress	<bitcoinaddress> [minconf=1]	<p>Returns the amount received by <bitcoinaddress> in transactions with at least [minconf] confirmations. It correctly handles the case where someone has sent to the address in multiple transactions. Keep in mind that addresses are only ever used for receiving transactions. Works only for addresses in the local wallet, external addresses will always show 0.</p>

Managing your wallet (CLI). Vanitygen

Video illustration below :



Exploring Transactions (CLI).

```

gettransaction 0408953d670d0f268b70f449772bd3d1c1b80c690be6f3017e3f9bdaa01c0b6c
{
  "amount" : -0.01080000,
  "fee" : 0.01080000,
  "confirmations" : 0,
  "txid" : "0408953d670d0f268b70f449772bd3d1c1b80c690be6f3017e3f9bdaa01c0b6c",
  "walletconflicts" : [
  ],
  "time" : 1399871859,
  "timereceived" : 1399871859,
  "details" : [
  {
    "account" : "",
    "address" : "1Dima5vfScYn342c7SfcX2pFYSu3rqhtKz",
    "category" : "send",
    "amount" : -0.00500000,
    "fee" : 0.01080000
  },
  {
    "account" : "",
    "address" : "18Z6bDrD4Fce8hRW5DkQ68f23xBCFodyxv",
    "category" : "send",
    "amount" : -0.01080000,
    "fee" : 0.01080000
  },
  {
    "account" : "Dima",
    "address" : "1Dima5vfScYn342c7SfcX2pFYSu3rqhtKz",
    "category" : "receive",
    "amount" : 0.00500000
  }
], .....
}

```

Transaction IDs (txid) are not authoritative until a transaction has been confirmed. Absence of a transaction hash in the blockchain does not mean the transaction was not processed.

This is known as “*transaction malleability*” , as transaction hashes can be modified prior to confirmation in a block.

After confirmation, the txid is immutable and authoritative.

Exploring Transactions (CLI).

```
getrawtransaction
0408953d670d0f268b70f449772bd3d1c1b80c690be6f3017e3f9bdaa01c0b6c
01000000029181c1d7b6b4fc7e2f1f1ee43dfb778468292a7b49a3e26c9c70b2
68fbc9ade000000006c493046022100cc2a0d920c154014d4e7f93878307b1b5a
eab8bba25288e1f00573fe05cf8aac022100b8269506e5c431d55bbe4c6850e98
3db8b9d9016cdece8dd7555a4ebf22816ba012102c8515f4e0512378032d44d5e
d3888bcd50be103ee26e0279f52a1fb935bb8f71fffffffff0e4d456390086dd62
2ce8be50672de7943d2a1d0ee78593a6b4e5c7a9cb6c9c3000000008a47304402
200f9e6e9bacd1f0d44525265455e92014faba5931a0ee8517664777d38c090d9
5022000905089d5bfcf8509589984f9b79182ea1bcbf6e6ae16f765efd8390f3c
8352014104681901c41fe94cab8e809ca1f830fd6bc953d88254337db8ab1db9
448ecd8bb2fec05f74f38abb05f4fd5d7040f9c011365967c24672514c2a40f20
dde07094fffffffff0220a10700000000001976a9148b87c4f4c177a46de7d50b7
dd9840c16caa4728088acc07a1000000000001976a91452dad8a8948da050406
72a11eacaecd916aa39288ac00000000
```

The `gettransaction` command returns a transaction in a simplified form.

To retrieve the full transaction code we can use two commands: *`getrawtransaction` and `decoderawtransaction`.*

The `getrawtransaction` command uses the transaction ID as a parameter and returns the full transaction as a “raw” hex string, exactly as it is on the Bitcoin network.

Exploring Transactions (CLI).

```
decoderawtransaction 01000000029181c1d7b6b4fc7e2f1f1ee43dfcb...
```

```
{
  "txid" :
  "0408953d670d0f268b70f449772bd3d1c1b80c690be6f3017e3f9bdaa01c0b6c",
  "version" : 1,
  "locktime" : 0,
  "vin" : [
    {
      "txid" :
      "de9abc8f260bc7c9263e9ab4a792824678b7fe3de41e1f2f7efcb4b6d7c18191",
      "vout" : 0,
      "scriptSig" : {
        "asm" :
        "3046022100cc2a0d920c154014d4e7f93878307b1b5aeab8bba25288e1f00573fe05cf8a
ac022100b8269506e5c431d55bbe4c6850e983db8b9d9016cdece8dd7555a4ebf22816ba0
1 02c8515f4e0512378032d44d5ed3888bcd50be103ee26e0279f52a1fb935bb8f71",
        "hex" :
        "493046022100cc2a0d920c154014d4e7f93878307b1b5aeab8bba25288e1f00573fe05cf
8aac022100b8269506e5c431d55bbe4c6850e983db8b9d9016cdece8dd7555a4ebf22816b
a012102c8515f4e0512378032d44d5ed3888bcd50be103ee26e0279f52a1fb935bb8f71"
      },
      "sequence" : 4294967295
    },
    {
      "txid" :
      "c3c9b69c7a5c4e6b3a5978eed0a1d24379de7206e58bce22d66d089063454d0e",
      "vout" : 0,
      "scriptSig" : {
        "asm" :
        "304402200f9e6e9bacd1f0d44525265455e92014faba5931a0ee8517664777d38c0
```

The *decoderawtransaction* command shows all the parts of this transaction, including the transaction inputs and outputs.

Exploring Transactions (CLI).

```

"vin" : [
{
  "txid" : "de9abc8f260bc7c9263e9ab4a792824678b7fe3de41e1f2f7efcb4b6d7c18191",
  "vout" : 0,
  "scriptSig" : {
    "asm" :
    "3046022100cc2a0d920c154014d4e7f93878307b1b5aeab8bba25288e1f00573fe05cf8aac0
    22100b8269506e5c431d55bbe4c6850e983db8b9d9016cdece8dd7555a4ebf22816ba01
    02c8515f4e0512378032d44d5ed3888bcd50be103ee26e0279f52a1fb935bb8f71",
    "hex" : "49304..."
  },
  "sequence" : 4294967295
},
{
  "txid" : "c3c9b69c7a5c4e6b3a5978eed0a1d24379de7206e58bce22d66d089063454d0e",
  "vout" : 0,
  "scriptSig" : {
    "asm" :
    "304402200f9e6e9bacd1f0d44525265455e92014faba5931a0ee8517664777d38c090d95022
    000905089d5bfcf8509589984f9b79182ealbcbf6e6ae16f765efd8390f3c835201
    04681901c41fe94cab8e809ca1f830fd6bc953d88254337db8ab1db9448ecd8bb2fec05f74f
    38abb05f4fd5d7040f9c011365967c24672514c2a40f20dde07094",
    "hex" : "4730..."
  },
  "sequence" : 4294967295
}
],

```

This transaction has two inputs as outputs of previously confirmed transactions with IDs starting with *de9a* and *c3c9* respectively

Exploring Transactions (CLI).

```
"vout" : [  
  {  
    "value" : 0.00500000,  
    "n" : 0,  
    "scriptPubKey" : {  
      "asm" : "OP_DUP OP_HASH160 8b87c4f4c177a46de7d50b7dd9840c16caa47280  
OP_EQUALVERIFY OP_CHECKSIG",  
      "hex" : "76a9148b87c4f4c177a46de7d50b7dd9840c16caa4728088ac",  
      "reqSigs" : 1,  
      "type" : "pubkeyhash",  
      "addresses" : [  
        "1Dima5vfScYn342c7SfcX2pFYsu3rqhtKz"  
      ]  
    }  
  }  
]
```

and one output of 5 mBits to our new *1Dima...* address.

Exploring Transactions (CLI).

```
gettransaction
0408953d670d0f268b70f449772bd3d1c1b80c690be6f3017e3f9bdaa01c0b6c
{
  "amount" : -0.01080000,
  "fee" : 0.01080000,
  "confirmations" : 2,
  "blockhash" :
  "00000000000000002320499cc4e60f5a515a03b088925f78b728bdf79ed5ac86",
  "blockindex" : 189,
  "blocktime" : 1399872120,
  "txid" :
  "0408953d670d0f268b70f449772bd3d1c1b80c690be6f3017e3f9bdaa01c0b6c",
  "walletconflicts" : [
  ],
  "time" : 1399871859,
  "timereceived" : 1399871859,
  "details" : [
  {
  ...
```

Once the transaction is confirmed, by inclusion in the block, the *gettransaction* command returns additional information, showing the block hash (identifier) and block index for the block in which the transaction was included.

Multi-signature transactions

Bitcoin has a feature called “*multi-signature*”, in which a transaction must have multiple independent approvals before the funds can be spent. **Multi-signature (or “*multisig*”)** transactions prevent thieves from stealing the contents of a wallet by simply gaining access to a single key-pair.

The most common scheme for multi-signature transactions is to employ an “*M-of-N scheme*”, for instance, **2-of-3**, in which case, at least 2 people (i.e. signers) must approve a transaction.

The way this works is that 3 *public keys* are listed as potential signers of a transaction, and at least 2 of those must be used to create a transaction’s signature in order to spend it.

There is currently a limitation of 15 public keys that can be used to sign a transaction, and any combination of the “*M-of-N scheme*” may be used. However, currently the “*2-of-3 scheme*” is the most practical.

Some popular multisig wallets are: [BitGo](#), [CoPay](#) (by BitPay), and [GreenAddress](#).

Multi-signature transactions

The general form of a multisig (“*M-of-N*”) transaction script looks like:

“M <Public Key 1> <Public Key 2> ... <Public Key N> N OP_CHECKMULTISIG”

In the case of a “2-of-3” multisig transaction the script looks like:

“2 <Public Key A> <Public Key B> <Public Key C> 3 OP_CHECKMULTISIG”

The above forms a “locking script”, which can only be unlocked by an equivalent “unlocking script”, which contains 2 or more signatures computed from the signer’s private keys and corresponding to the listed public keys, as follows:

“OP_0 <Signature B> <Signature C>”

The above “locking” and “unlocking” script, together form a “validation script” which serves to enable multisig wallets:

**“OP_0 <Signature B> <Signature C>
2 <Public Key A> <Public Key B> <Public Key C> 3 OP_CHECKMULTISIG”**
(Validation script)

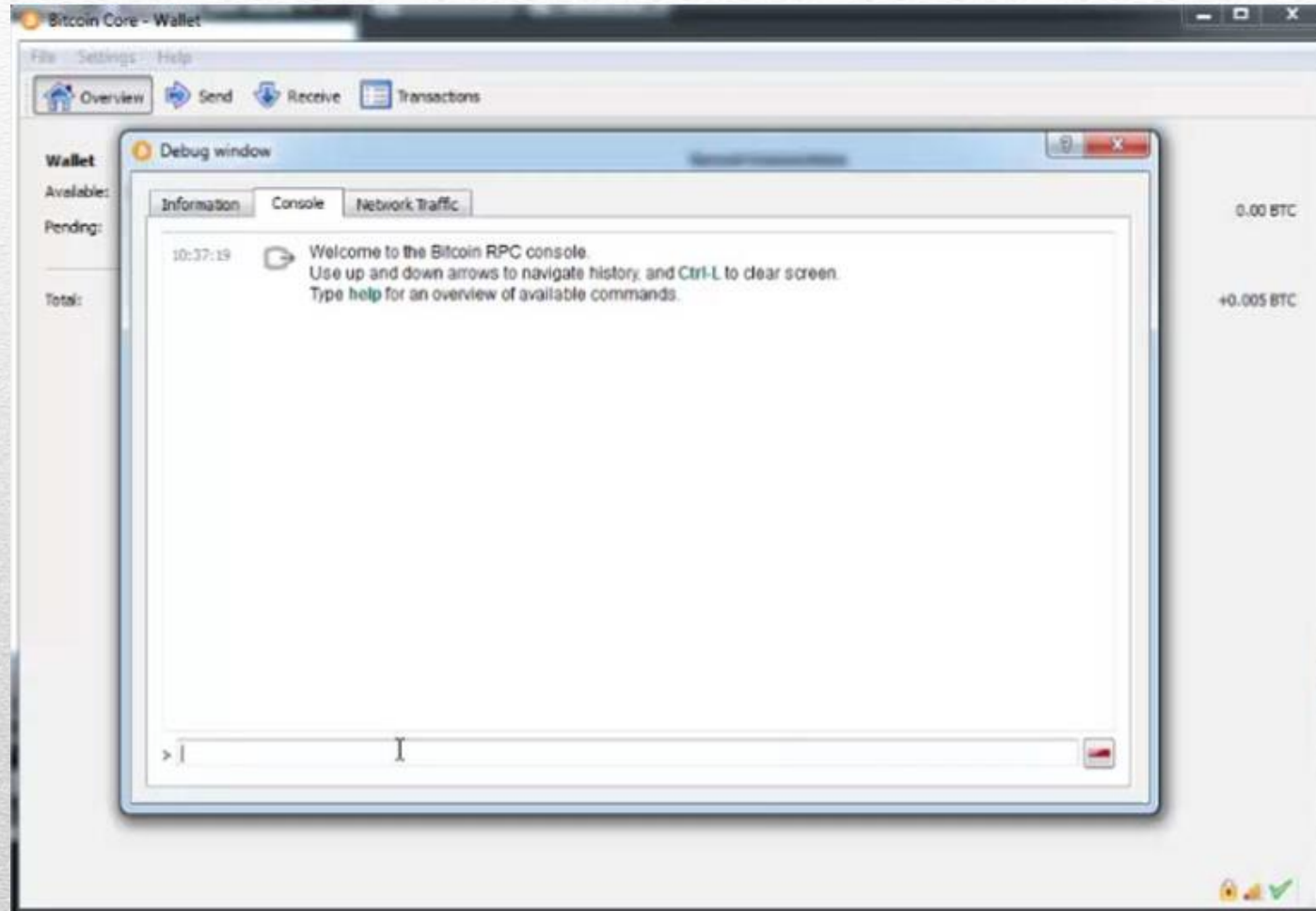
Exploring Blocks (CLI).

Commands to use:

Command	Parameters	Description
getblock	<hash>	Returns information about the block with the given hash.
getblockhash	<index>	Returns hash of block in best-block-chain at <index>; index 0 is the genesis block

Managing your wallet (CLI). Vanitygen

Video illustration below :



Exploring Blocks (CLI).

```
gettransaction
0408953d670d0f268b70f449772bd3d1c1b80c690be6f3017e3f9bdaa01c0b6c
{
  "amount" : -0.01080000,
  "fee" : 0.01080000,
  "confirmations" : 2,
  "blockhash" :
  "0000000000000002320499cc4e60f5a515a03b088925f78b728bdf79ed5ac86",
  "blockindex" : 189,
  "blocktime" : 1399872120,
  "txid" :
  "0408953d670d0f268b70f449772bd3d1c1b80c690be6f3017e3f9bdaa01c0b6c",
  "walletconflicts" : [
  ],
  "time" : 1399871859,
  "timereceived" : 1399871859,
  "details" : [
  {
  ...
```

We are now going to analyze the block we obtained earlier.

Exploring Blocks (CLI).

```

getblock
000000000000000002320499cc4e60f5a515a03b088925f78b728bdf79ed5ac86
{
  "hash" :
  "000000000000000002320499cc4e60f5a515a03b088925f78b728bdf79ed5ac86",
  "confirmations" : 20,
  "size" : 144825,
  "height" : 300323,
  "version" : 2,
  "merkleroot" :
  "5c782440831d895dbe2851999d403f08a59768a633de027288403efa472081c8",
  "tx" : [
    "5a127914b627a7657759c0a09df974d11d6712bd707731e9bb6b7b675d326aeb",
    "050347e8a6babdd74fc60809c29f16d1bc23f0f5dd2ac329499b57166e197e18",
    "999972c1afe4c311e46e475a661dc83397cd3896c79f7cda4374d0372433de9a",
    ...
    "161cb709e9b9e15617d0827af6282fee53484fe9ca4a575258989d8809256fd8"
  ],
  "time" : 1399872120,
  "nonce" : 1602350785,
  "bits" : "1900896c",
  "difficulty" : 8000872135.96816350,
  "chainwork" :
  "0000000000000000000000000000000000000000000000000000000000000005cd4f1cdf66447a1f6c4",
  "previousblockhash" :
  "000000000000000044340d7a81d3165439ddbabc94521754f00daaaaa0aae09b",
  "nextblockhash" :
  "000000000000000001ad17972576160667dd6f246045ff03e50716242d68faf7"
}

```

As you can see, this block contains many transactions.

Exploring Blocks (CLI).

```
"998342493e6deb9efda2250878a86b42b74fbdd99365e5074d9d7517f6f92e50",  
"1f74548a6b92149b2dd523928ebfc830aa80f9f01bca94f281a934e647696981",  
"4fa2eb6c33189190fc515066c0a38da9573583f3b795fe2f0b75147278b3c037",  
"15766c8dee6053f2d36bba568497850ca0d39d782d239816261d7bef9b3d25b3",  
"0408953d670d0f268b70f449772bd3d1c1b80c690be6f3017e3f9bdaa01c0b6c",  
"8062654f3f2de78537da07784a199e6724a5ea43a281b7f7536d9d2ad2a468a6",  
"b90e6279b0e4bf697d71a0129bbc76e993e72c397f411b796935c56f5e58d12b",  
"22246aaaa62afa3d191df40226dfcb3a64fe3e426b67525d4eb1d906a6f9ef87",  
"33d20b9bbbf7ea35f070aabc8a34db51ba0fa172dcaac114fb00b28f31c8cbf3",  
"0cf5e8f7fbc9dc0d853c8699abd5ce8f1ff497da8b1acc819d07a7d2ae54dc30",
```

Our transaction can also be found in this block.

Exploring Blocks (CLI).

[illegible]

We can retrieve block by index (“block height”), where “0” is the height of the *Genesis* block.

The Genesis block only contains one transaction (the *coinbase* transaction). Prior to the first Bitcoin transaction (found on block 170), all blocks only contained the coinbase transaction.

This is how bitcoins are being generated.

Exploring transactions

New bitcoins are generated by *miners*.

Miners also confirm our transactions.

The basics of mining were previously discussed in Session 3.

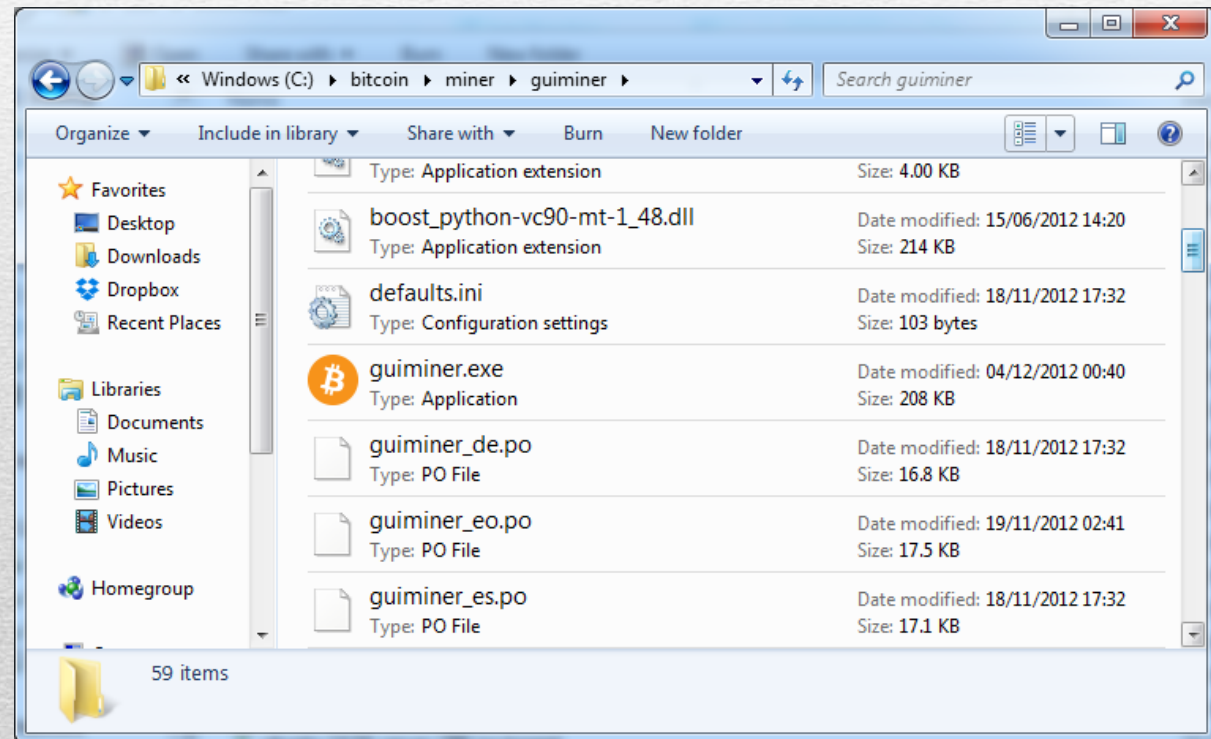
Over the next pages, we are going to discuss practical issues related to mining.

3. Mining and mining pools

What is mining?

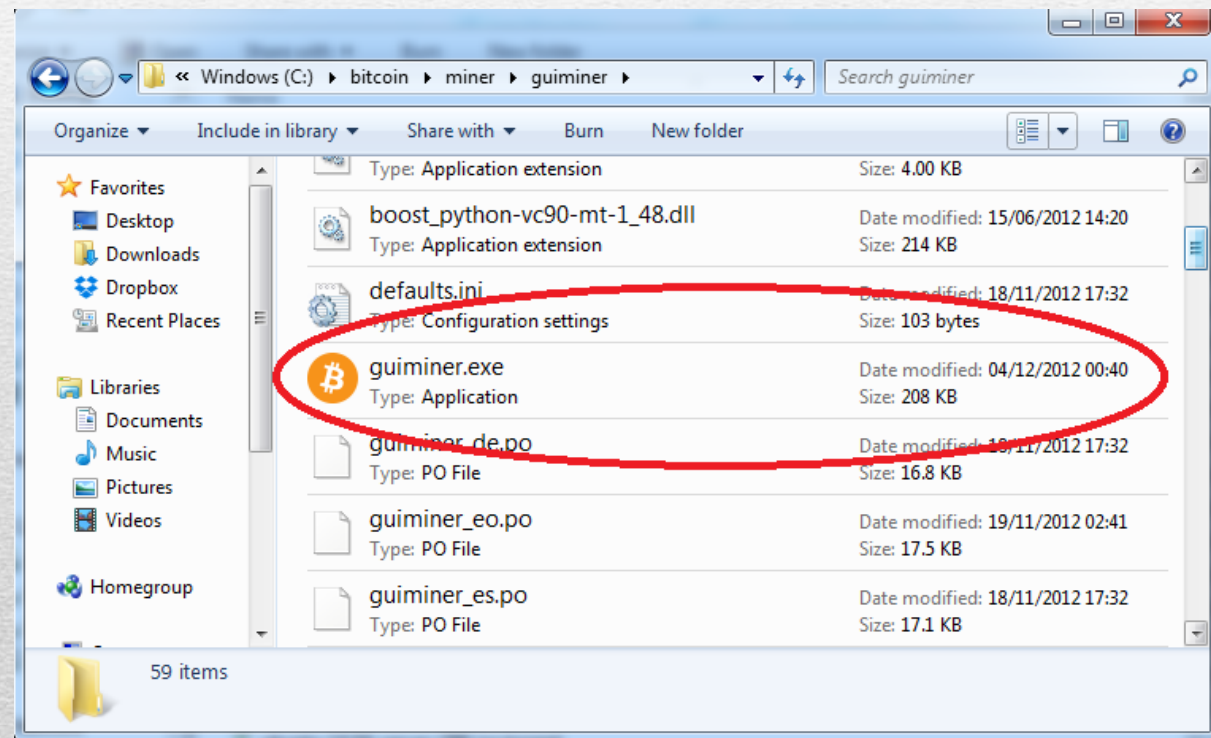
The easiest way to start mining is to download GUI-miner from <http://guiminer.org/>. You will download a self-extracting archive. When you run it you have to specify the desired location, for example, c:\bitcoin\miner\.

After extracting you will have a guiminer folder with the following content:



What is mining?

You can create a shortcut for guiminer.exe



What is mining? Solo Mining

The difference between *solo mining* and *pool mining* was discussed earlier in Session 3. In order to implement *solo mining* you need to have Bitcoin Core installed and synchronized.

```
getwork

{
  "midstate" :
  "1611c59331eb762961e60e685469b57b74baf85fc57009d78fcc2cc972667ac6",
  "data" :
  "00000002795e3874256797b9bde90d71d59697a8e3c5ec62231a1c8200000000000000
  0084352e29c58a57195bf7eb405b92ec97df557144032ab3ede1f23fee444dae0a5370b
  b321900896c000000000000000800000000000000000000000000000000000000000
  000000000000000000000000000000000000000000000000000000000000000000000000",
  "hash1" :
  "00000000000000000000000000000000000000000000000000000000000000000000
  80000000000000000000000000000000000000000000000000000000000000000000000010000",
  "target" :
  "0000000000000000000000000000000000000000000000000000000000000000000000006c890000000000000000"
}
```

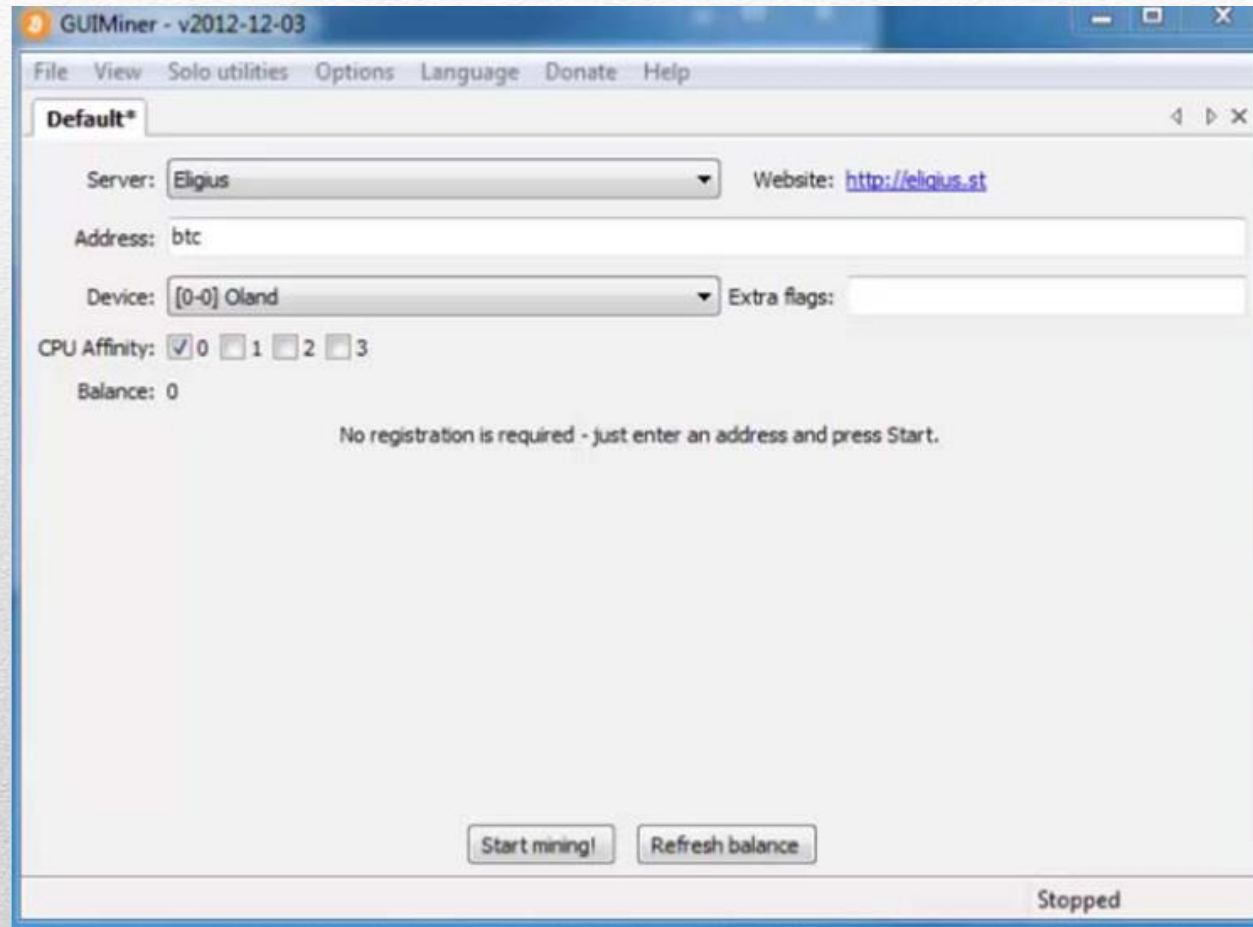
When running in server mode core accepts RPC calls from other programs, including the miner. The miner uses either the *getwork* or *getblocktemplate* RPC calls.

getwork responds with data required to run the mining algorithm. *getblocktemplate* returns a whole template for the next block to be generated.

When a solution is found the miner submits it to the network.

What is mining? Solo Mining

Video illustration below :



Mining - Pool mining

Pool mining doesn't require having a full client. However you have to create an account with one of the mining pools. We have discussed pool selection earlier in Session 3. As an example, we created an account with bitcoin.cz. The registration procedure depends on the chosen pool.

Video illustration :



Mining - ASIC Mining

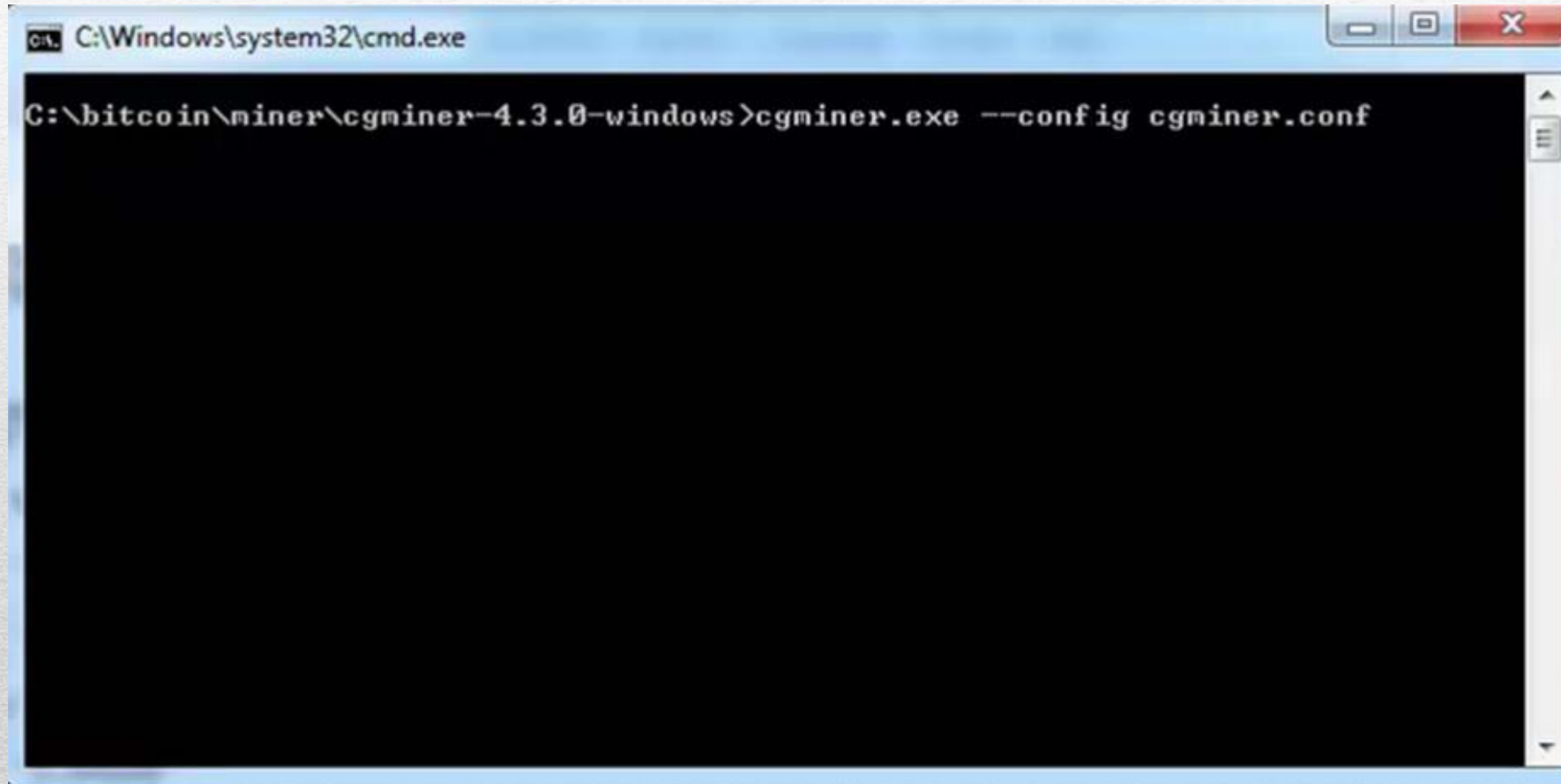
So far we used our CPUs in order to do the mining calculations. This was the case in the earlier days when the mining difficulty was low. Today CPU mining cannot provide any level of meaningful performance. Far more calculations can be performed with specially designed ASICs. ASIC stands for **Application Specific Integrated Circuit**.

ASIC miners can be a standalone devices that are connected to the network. Such miners are relatively expensive and contain dozens of ASIC chips.

GUI-miner usually does not work with USB-based ASIC miners, however one can install the cgminer program, which is a more flexible command line-based miner. Despite this, cgminer is more difficult to install and configure.

Mining - ASIC Mining

In the video example below, the miner reports work progress and performance statistics.

A screenshot of a Windows command prompt window. The title bar at the top reads "C:\Windows\system32\cmd.exe". The command prompt shows the directory "C:\bitcoin\miner\cgminer-4.3.0-windows" and the command "cgminer.exe --config cgminer.conf" entered. The rest of the window is black, indicating that the command is running or has completed without displaying any output.

```
C:\Windows\system32\cmd.exe  
C:\bitcoin\miner\cgminer-4.3.0-windows>cgminer.exe --config cgminer.conf
```

Mining - ASIC vs CPU Mining

The following statistics are obtained from the mining pool. They show the performance difference between CPU and ASIC mining, even though those are outdated. The difference is far more pronounced now, as more efficient chips are developed.

Login	Password	Found blocks	Current shares	Score	Last share at	Mhash/s*
btcunic.asic	BtcUnic2014	0	98	160.6058	0 minutes	116.918554169
btcunic.cpu	BtcUnic2014	0	5	2.4253	7 minutes	5.96523235556

Mining - GPU mining

GPU mining is another type of mining where the GPU (Graphics Processing Unit) is used to perform mining calculations. GPU miners are usually based on custom-built PCs that may have up to 6 graphics cards. GPU mining is faster than CPU mining, and consumes a lot of power.

The significantly higher output of ASICs combined with their low energy consumption per hash (by comparison), has made them the method of choice when mining bitcoins. GPUs are hardly used for Bitcoin mining anymore, though they are used to mine other coins.

4. Segregated Witness (segwit)

Segregated Witness

First presented at the Scaling Bitcoin workshops in Hong Kong in early December 2015, “Segregated Witness” or segwit is an exciting optimization proposal that can improve Bitcoin’s performance in a number of ways.

As we’ve seen in previous sessions, simply put, transactions in Bitcoin have two main types of information contained within them :

- One part is what sum is transferred from where to where, in the form of inputs and outputs
- Proof that those transferences are authorized by the respective private key holders and they can be validly performed

This last part is the “Witness” part.

The basic idea behind segwit is that removing this from the transaction, enables more transactions to be recorded to the blockchain and thus a higher transaction throughput. Lets take a deeper look at what this means, and what other benefits or risks it might introduce, in the next slide.

Segregated witness (cont)

The segwit proposal is still a work in progress.

Once the development underway, is part of Bitcoin, we'll take a deeper and more detailed look on the practical aspects of it. So far benefits and risks identified are as follows :

- More transactions per second since witness isn't included in the blockchain
- No more transaction malleability
- Potential for fraud proofs for SPV (light) wallets
- Witness part to be included in "add-on" blocks verified as part of a miners' coinbase transaction
- The whole value chain within Bitcoin, miners, nodes, wallets, etc needs to upgrade to gain maximum benefits
- If every transaction propagated was "segwit enabled" we could see the potential effectively increase in blocksize to ~1.7m (although the witness information will still need to be conveyed between nodes and miners)
- While most agree on it's usefulness, it is arguably a complicated addition to the software and several voices are arguing against a soft fork implementation and towards a hard fork implementation

For more information on Segregated Witness, see Bitcoin Core's [announcement](#) on the subject, [BIP 141](#) or [part 1](#), [part 2](#), [part 3](#), [part 4](#), [part 5](#), [part 6](#), [part 7](#) and [part 8](#) of Bitcoin Magazine's development series.

5. Conclusions

Conclusions

- The Bitcoin Core client (formerly known as bitcoind) synchronizes with the Bitcoin network and the blockchain (i.e. blocks of transactions) and, therefore, is able to validate those transactions
- Bitcoin Core can be controlled through the CLI (Command Line Interface)
- Bitcoin Core can work as a server and provide services (APIs) to other programs like graphical interfaces or miner
- Solo mining requires an installation of Bitcoin Core.
- Pool mining requires an account with one of the mining pools but no local Bitcoin Core.
- CPU mining does not provide any meaningful performance anymore.
- GPU mining is power consuming and largely obsolete unless mining some altcoins.
- ASIC mining provides the best results for Bitcoin mining, but might not be available for other digital currencies.
- The segwit proposal could increase the network's throughput if implemented by all

6. Further reading

Further Reading

- Bitcoin-Qt
<https://en.bitcoin.it/wiki/Bitcoin-Qt>
- Original Bitcoin client/API calls list
[https://en.bitcoin.it/wiki/Original Bitcoin client/API calls list](https://en.bitcoin.it/wiki/Original_Bitcoin_client/API_calls_list)
- GUIMiner
<http://guiminer.org/>
- CGMiner
<https://en.bitcoin.it/wiki/CGMiner>
- CGMiner binaries
<http://ck.kolivas.org/apps/cgminer/>
- Win7 cgminer setup guide
<https://ozcoin.net/content/win7-cgminer-setup-guide>
- Comparison of mining pools
[https://en.bitcoin.it/wiki/Comparison of mining pools](https://en.bitcoin.it/wiki/Comparison_of_mining_pools)
- Blocksize Arguments :
[https://en.bitcoin.it/wiki/Block size limit controversy](https://en.bitcoin.it/wiki/Block_size_limit_controversy)
<http://bitcoin.stackexchange.com/questions/36085/what-are-the-arguments-for-and-against-the-increase-of-the-block-size-limit>

Questions?



Contact Us

Email:

digitalcurrency@unic.ac.cy

Platform support:

dl.it@unic.ac.cy

Twitter:

@MScdigital