crypto251.0 Cryptocurrency and the Smileycoin

Gunnar Stefansson (editor)

29. nóvember 2020

Efnisyfirlit

1 Introduction to cryptocurrencies

1.1 Introduction to the course

These slides and all the tutor-web content can be found under the two links

- https://tutor-web.net/comp/crypto251.0
- https://beta.tutor-web.net/
- Videos in English: https://www.youtube.com/playlist?list=PLzTQcKBiNWB3E7nh5egXI_PaHW1
- Videos in Icelandic: See the Canvas course page

This content forms the basis for an on-line cryptocurrency course as well as a course, “Rafmyntir (STÆ 532M)” at the University of Iceland.
Only the beta version of tutor-web will be used in the actual course. Only work done in the beta version will count towards completion of the course.
The slides and content are **open** so they are freely available and accessible to anyone, anywhere.
To receive **academic credit** for working through the material, a student needs to be **registered** at a university which approves this course. Initially that is only the University of Iceland.
For the UI course, announcements are sent out by email and any additional explanations are stored in the UI Canvas web.

1.1.1 Handout

Handout for Rafmyntir (STÆ 532M) at the University of Iceland
The following describes the setup of the course in fall 2020, the third time it is given, in the year of COVID-19.
There are no in-class lectures!
Attempts will be made to record each lecture (most likely in Icelandic).
Attempts will be made to make very short weekly videos, in English, of the most important topics covered during the week.
The final grade will mainly be based on homework during the semester.
Homework will consist of

- work undertaken only in the tutor-web system (writing/reviewing exercises/drills plus practicing drills) and
- various other tasks listed in Canvas, normally reported in the tutor-web (but check the text of each task).

Note that some (most) of the output from the homework done outside the tutor-web will still be reported within the tutor-web system (the beta version).
Homework inside the tutor-web will range from simple documentation (writing examples) through reporting solutions to exercises to reporting output of tasks/projects using the Smileycoin to test new features of cryptocurrencies. In adddition, students will review and grade each other’s submissions to the tutor-web. Each component will count towards the final grade (submissions, projects, reviews).
The **textbook by Antonopoulos** contains all the important background and main concepts used in the course. The **Smileycoin paper in Ledger** contains all the information needed on the Smileycoin and how it deviates from Bitcoin.
Video describing how to return homework: https://bit.ly/32fTtrb
Other topics will be covered in class or assigned reading, some is available as Steemit articles but other material can be discovered by on-line searching.

1.2 Enrollment, credits and Smileycoin rewards

To obtain credits for the course, a student needs to be registered at a university which provides that kind of accreditation.

However any student, anywhere, is free to take the open tutor-web version of the course, as a self-study course, with or without any association with an instructor or institution.

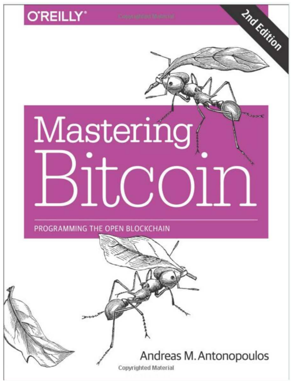
Students should note: If you are formally enrolled in a school anywhere, you should ask your instructor to contact any admin of the tutor-web to make the class a formal class in the tutor-web. This will ensure that the students in the class receive much higher Smileycoin rewards when completing tasks in the system. Students are free to use the system without being enrolled anywhere, but will then receive fewer SMLY.

For further information, see handouts and examples in the PDF version of these tutorial notes.

Video corresponding to this introduction: `https://bit.ly/3aPyIqj`

1.3 Reading material

The book



The SMLY paper (aka the Ledger article)
`http://ledgerjournal.org/ojs/index.php/ledger/article/view/103/84`
The paper Satoshi Nakamoto Bitcoin: A Peer-to-Peer Electronic Cash System
`https://bitcoin.org/bitcoin.pdf`

1.3.1 Handout

The primary text is Antonopoulos' textbook on Bitcoin. This is a fundamental text on cryptocurrencies and any student of cryptocurrency or blockchain should have a copy of this text.

The Bitcoin paper by Satoshi Nakomoto is the foundational paper on Bitcoin.

Since the SmileyCoin is used as an example throughout this course, the SMLY article in Ledger forms a basis to describe the internals of SMLY. Note that this article is from 2017 and the SMLY has been extended considerably since then.

Other articles and papers will be mentioned throughout this document and by your instructor if you are taking this as a real-world course.

1.4 Cryptocurrencies

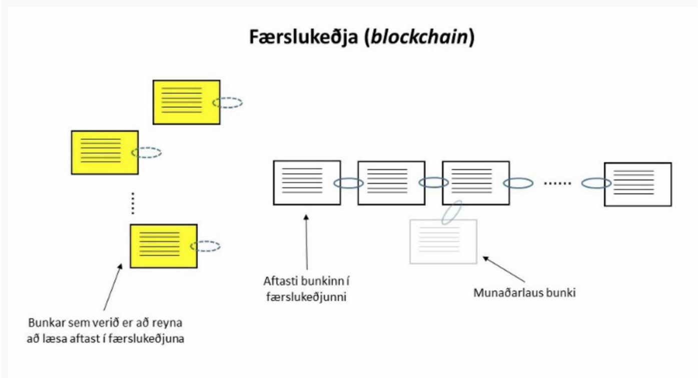
A cryptocurrency is an electronic solution to the task of securely storing and exchanging units of value without any need for trusted intermediaries such as banks or backing by physical objects such as gold, coins or notes.
By taking this course the student will study in detail the technical aspects of cryptocurrencies, including how transfer of value is conducted and how they are made secure.
There are many, many cryptocurrencies:

- Bitcoin
- Litecoin
- Dogecoin
- Etherium
- Auroracoin
- . . .
- Smileycoin (Broskallar) :-)

This course will use the Smileycoin as an example throughout
See `https://coinmarketcap.com/all/views/all/`

1.5 Behind the scenes (in Icelandic)

- Bálkakeðja (Færslukeðja/Bunkakeðja – blockchain)
- Færslur (og grunnhugtakið, UTXO – transactions)
- Námugröftur (– mining)
- Satoshi Nakamoto


Færslukeðja (*blockchain*)
......
Aftasti bunkinn í færslukeðjunni
Munaðarlaus bunki
Bunkar sem verið er að reyna að læsa aftast í færslukeðjuna

(Hjálmtýr Hafsteinsson, Vísindavefurinn)

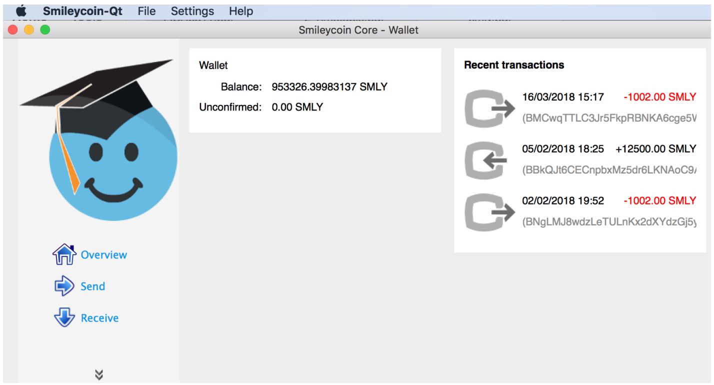
1.6 A useful allegory

- The chain is like an old-fashioned ledger
- Each block is like a page in the ledger
- Each transaction is just like a traditional transaction “Alice pays/lends Bob 10 cents”
- The miner is the accountant:
  - collects transactions
  - records them into a new block - a page in the book
  - gets paid for doing this work

A short video describing the same concepts:
`https://www.youtube.com/watch?v=LcpBlXHOZoc&index=3&list=PLzTQcKBiNWB3E7nh5egXI_PaHW1`

1.7 The user side

- Download a “wallet” (a computer program/app) to a computer (e.g. desktop, laptop, tablet or phone)
- **Receive** cryptocurrency “to the wallet”
- **Send** to others



Wallet
Balance: 953326.39983137 SMLY
Unconfirmed: 0.00 SMLY

Recent transactions
16/03/2018 15:17 -1002.00 SMLY
(BMCwqTTLC3Jr5FkpRBNKA6cge5V
05/02/2018 18:25 +12500.00 SMLY
(BBkQJt6CECnpbxMz5dr6LKNAoC9/
02/02/2018 19:52 -1002.00 SMLY
(BNgLMJ8wdzLeTULnKx2dXYdzGj5y

MEMO Nothing actually gets sent anywhere :-)

1.8 Overview

This section has given a quick overview of the cryptocurrency course and basic concepts. Your instructor will give more detail. At UI more detailed definitions of work are/will be given in Canvas.
This would be a good time to read chapter 1 of Mastering Bitcoin by Andreas Antonopoulos.

1.8.1 Handout

Homework: Add some material to any single subsection of this section.

2 Bitcoin and Smileycoin basics

2.1 Operating a wallet

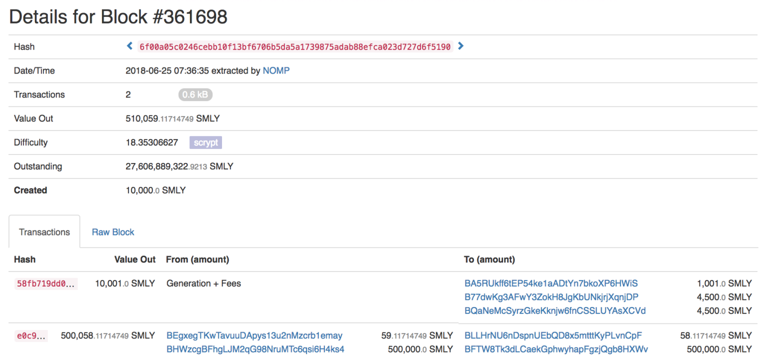
A **wallet** is a computer program (or app) which handles the user's funds.
From the user's side, the basics of operating a wallet are made extremely simple.
With a click of a button the user can

- Request funds
- Send funds

Transmissions of funds are sent out as **transactions**.
If Alice wants to send a transaction to Bob, her wallet needs information on where to send the transaction.
If Alice requests funds from Bob then Alice's wallet will typically show a QR-code which Bob's wallet can scan to set up the required transaction.
A short video introduction is available.

2.2 The block and block explorers

A simple block (SMLY block 361698)



Details for Block #361698

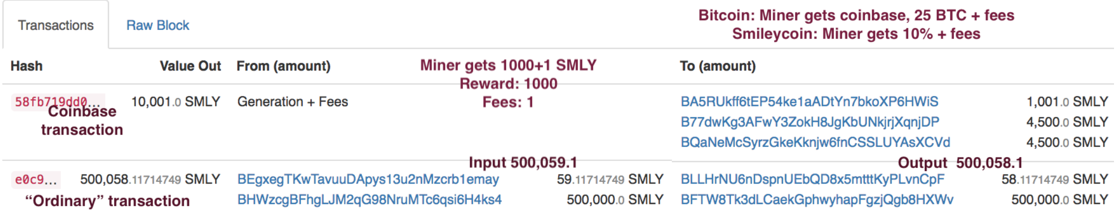
Hash	‹ 6f00a05c0246cebb10f13bf6706b5da5a1739875adab88efca023d727d6f5190 ›
Date/Time	2018-06-25 07:36:35 extracted by NOMP
Transactions	2 0.6 kB
Value Out	510,059.11714749 SMLY
Difficulty	18.35306627 scrypt
Outstanding	27,606,889,322.9213 SMLY
Created	10,000.0 SMLY

Hash	Value Out	From (amount)		To (amount)	
58fb719dd0...	10,001.0 SMLY	Generation + Fees		BA5RUkff6tEP54ke1aADtYn7bkoXP6HWiS B77dwKg3AFwY3ZokH8JgKbUNkjrjXqnjDP BQaNeMcSyrzGkeKknjw6fnCSSLUYAsXCVd	1,001.0 SMLY 4,500.0 SMLY 4,500.0 SMLY
e0c9...	500,058.11714749 SMLY	BEgxegTKwTavuuDApys13u2nMzcrb1emay BHWzcgBFhgLJM2qG98NruMTc6qsi6H4ks4	59.11714749 SMLY 500,000.0 SMLY	BLLHrNU6nDspnUEbQD8x5mtttKyPLvnCpF BFTW8Tk3dLCaekGphwyhapFgzjQgb8HXWv	58.11714749 SMLY 500,000.0 SMLY

Contains two transactions.
See `https://chainz.cryptoid.info/smly/`
and `https://blocks.smileyco.in/`
Video demonstration:
`https://www.youtube.com/watch?v=a7EhHyWCrU4&index=4&list=PLzTQcKBiNWB3E7nh5egXI_PaHW1`

2.3 The transaction

The transactions



Hash	Value Out	From (amount)		To (amount)	
58fb719dd0...	10,001.0 SMLY	Generation + Fees		BA5RUkff6tEP54ke1aADtYn7bkoXP6HWiS B77dwKg3AFwY3ZokH8JgKbUNkjrjXqnjDP BQaNeMcSyrzGkeKknjw6fnCSSLUYAsXCVd	1,001.0 SMLY 4,500.0 SMLY 4,500.0 SMLY
e0c9...	500,058.11714749 SMLY	BEgxegTKwTavuuDApys13u2nMzcrb1emay BHWzcgBFhgLJM2qG98NruMTc6qsi6H4ks4	59.11714749 SMLY 500,000.0 SMLY	BLLHrNU6nDspnUEbQD8x5mtttKyPLvnCpF BFTW8Tk3dLCaekGphwyhapFgzjQgb8HXWv	58.11714749 SMLY 500,000.0 SMLY

Bitcoin: Miner gets coinbase, 25 BTC + fees
Smileycoin: Miner gets 10% + fees

Miner gets 1000+1 SMLY
Reward: 1000
Fees: 1

Coinbase transaction

Input 500,059.1

Output 500,058.1

"Ordinary" transaction

2.4 Where we come from (a): the tutor-web

The tutor-web is an educational resource, mainly developed at the University of Iceland.

The SmileyCoin was originally developed to experiment with rewards in the tutor-web. A short video introduction is available, giving an overview of the tutor-web SmileyCoin and Education in a Suitcase.

2.4.1 Handout

Development of the tutor-web started around the year 2000, but it has been redesigned several times.

The tutor-web is used for research on education and technology. Typically, parameters control the behaviour of the system and these are set to different values to see how to improve learning.

The references at the end of this section give some examples of this research.

2.5 Where we come from (b): Education in a Suitcase

eias

Non-profit organisation, registered in Iceland as **Styrktarfélagið Broskallar**

Purpose: Raise funds to donate educational tech to low-income regions

2.5.1 Handout

Education in a Suitcase (EIAS) is a project led by a non-profit organisation, Styrktarfélagið Broskallar (SB), registered as such in Iceland.

SB applies for grants for EIAS, which is organised in cooperation with several other non-profit entities.

SB has an income in SMLY.

The EIAS project has donated tablets and servers running tutor-web to hundreds of students and schools in several locations in Kenya: Takawiri Primary School, Shivanga Secondary School, Maseno University, Naivasha Maximum Security Prison.
Most of these areas have unstable electricity, no WiFi and poor to no Internet connections. The server setup therefore provides the WiFi and serve the tutor-web to students. In addition, the servers provide the content of the Khan Academy, the entire Wikipedia in English and the Gutenberg Library of ofer 60 thousand titles.
New implementation methods are tested each year. Currently under development is a **library model** whereby a library receives the tablets from EIAS but the students can purchase the tablet once they have earned enough SLMY in the tutor-web system.

2.6 Where we come from (c): SMLY

smly

Primary purpose: Rewarding students in the tutor-web system
Long term goal: Provide $1/day for low-income students

2.6.1 Handout

The SmileyCoin was originally developed to test the effects of cryptocurrency rewards in the tutor-web system.
Since the tutor-web system is completely open, this also gives open access to anyone wanting to earn SMLY for their studies.
A stated long-term goal is to extend the use of tutor-web and SmileyCoin through Education in a Suitcase so students in low-income regions can earn the equivalent of $1 per day through studying in the system.

2.7 Overview

The handout lists homework.
Your instructor will give more detail.

2.7.1 Handout

Homework: Add some material to any single subsection of this section.

2.7.2 References

Some publications on tutor-web, SmileyCoin and Education in a Suitcase.

Constantinescu, D. and Stefansson, G. 2010. On building a web-based university. US-China Educational Review, 7(12), 89-97.

Everson et al. 2013. “Teaching Online on a Budget.”JSM Proceedings, Statistical Education Section, Alexandria, VA: American Statistical Association.

Jonsdottir 2015. Development and testing of an open learning environment to enhance statistics and mathematics education. PhD thesis.

Jonsdottir et al. 2015. Development and use of an adaptive learning environment to research on-line study behaviour. Educ. Tech. & Society, 18 (1), 132–144.

Jonsdottir and Stefansson 2014. Using an Online Learning Environment to Teach an Undergraduate Statistics Course: the tutor-web. EDULEARN13 `http://arxiv.org/abs/1407.0308`

Jonsdottir and Stefansson 2014. From evaluation to learning: Some aspects of designing a cyber-university. Computers&Education 78, 344-351

Jonsdottir et al. 2013. Könnunarpróf nýnema í stærðfræði við Háskóla Íslands. Niðurstöður og forspárgildi. Timarit um menntarannsoknir 10: 11-28.

Jonsdottir and Stefansson 2011. Enhanced Learning with Web-Assisted Education. In JSM Proceedings, pp. 3964 -3795. See Arxiv:1310.4667.

Jonsdottir et al. 2017. Difference in learning among students doing pen-and-paper homework compared to web-based homework. J. Statistics Education.

Jonsdottir et al. 2019. Learning Wherever, Whenever: Education in a Suitcase. EDULEARN19

Lentin et al. 2014. A mobile web for enhancing statistics and mathematics education. Presented at icots9. See `http://arxiv.org/abs/1406.5004`

Njurai et al. 2017. Initial reflections on teaching and learning mathematics using tablet in a prison education centre. SIMC 2017

Olafsdottir, E. I., Hreinsdottir, F., Stefansson, G. and Oskarsdottir, M. 2009. Mathematics electives by gender in a secondary school in Iceland (In Icelandic: Námsval stúlkna með tilliti til stærðfræði í Menntaskólanum við Hamrahlíð). Science Institute technical report RH-12-2009.

Sigurdardottir and Stefansson 2009. Greining á framförum nemenda innan vefkerfis sem býður upp á gagnvirk krossapróf. Timarit um raunvisindi og stærðfræði, 6 (1): 23-32.

Stefansson, G. and Sigurdardottir, A. J. 2009. Interactive quizzes for continuous learning and evaluation. In Joint Statistical Meetings (JSM) proceedings 4577-4591.

Stefansson et al. 2017. Evidence-based technology to enhance mathematics education from Iceland to Kenya. `https://library.iated.org/view/STEFANSSON2017EVI`

Stefansson and Lentin 2017. From smileys to Smileycoins: Using a cryptocurrency for rewards in education. Ledger vol. 2, 38-54 `http://ledgerjournal.org/ojs/index.php/ledger/article`

Stefansson and Jonsdottir 2015. Design and analysis of experiments linking on-line drilling methods to improvements in knowledge. J. of Statist. Sci, and Applic. 3(5-6), 63-73.

Stefansson, G. and Sigurdardottir, A. J. 2011. Web-assisted education: From evaluation to learning. J. Instr. Psych. 38(1): 47-60.

Stefansson and Constantinescu 2009. The tutor-web, a step towards building an active web based university. Proc 5th Internat Sci Conf “eLearning and Software for Education”, pp 273-280.

Stefansson, G. 2004. The tutor-web: An educational system for class-room presentation, evaluation and self-study. Computers&Education, 43 (4): 315-343.

Stefansson, G. 2003. Gagnvirkur kennsluvefur (in Icelandic). Timarit um raunvisindi og stærðfræði, 2: 115-116.

Medium and Steemit publications, and publications in popular media:

Fundi, 2u18: `https://medium.com/@fundimaxwell/technology-hoped-to-improve-academic-excell`

Mbugua, 2018: `https://medium.com/@kamaumbugua/smileycoin-5385a3b046ed`

Stefansson, 2019: `https://medium.com/@gunnarstefansson/a-coin-for-education-and-philanthro`

Stefansson, 2017: `https://steemit.com/kenya/@gstefans/takawiri-island-primary-school-with-`

3 Picking up and using a wallet

3.1 Single-coin vs multi-coin wallets

Most cryptocurrency wallets are designed to handle a **single coin**.
Multi-coin wallets are used to handle **multiple currencies**.
A multi-coin wallet may assist the user in converting from one currency to another.
The usual single-coin wallets store the user's keys and addresses and keep track of the coins associated with each address.
Video explanations are available

3.1.1 Handout

A popular multicoin wallet is the Coinomi wallet.
All coins also have dedicated wallets for desktop/laptop computer.
Some coins have HTML5-wallets. This is code which implements the wallet as a web-page in a browser.
All of the above are implemented for SmileyCoin.

3.2 Smileycoin wallets

The reference (core) wallet for Smileycoin is available for

- Linux (as source code)
- Windows (as binary)
- OS X (as binary, for Mac)

from `https://tutor-web.info/smileycoin/download`
Tablet users can pick either the Coinomi wallet (App/Play store) or the HTML5 wallet at `https://wallet.smileyco.in`
(for detail, see handout)

3.2.1 Handout

The reference (core) wallet for Smileycoin is available for several operating systems.

- Linux: This is only made available as source code. It should be ready to compile and install for Ubuntu. Components may need to be installed for other Linux versions.
- Windows: Only a binary version is made available. This is typically a bit behind the Linux version.
- OSX: A binary version is available. This is typically a bit behind the Linux version.

Directions on where to pick these up can be found at `https://tutor-web.info/smileycoin/download`
An HTML5 wallet is also available at `https://wallet.smileyco.in` and runs directly in the browser. Be specially careful to store the passphrase. This can be used for several experimental testing projects. This wallet is also used for general feature development and testing.
For tablets and phones a Coinomi wallet is available (Android and iOS). Searching for Coinomi in the appropriate store should work. This is a stable wallet and appropriate for

users not wanting any extra features developed specifically on the SMLY chain (such as on-chain services, voting or traceability).
Warning: The core wallet **needs** to be used for most projects in the course which refer to the BASE address. Any of the versions (Linux, Mac, Windows) can in principle be used.

3.3 The configuration file

The **core wallet** (Linux/Mac/Win) has a configuration file (the web wallet does not)
Name of file: `smileycoin.conf`
Needs to contain a user and a password. Setup is automatic for Qt wallets, not so for command-line-only wallets (Linux).
The configuration file must include a rpcpassword - if not, the server terminates with an error.
Just follow the directions given in the error message. It is fine to use the password in the message (not the one given above).

3.3.1 Handout

On a Linux machine the user needs to set up a configuration file in their home directory.
Start by running the smileycoin daemon to get the error message: `smileycoind --server`
It will terminate with an error message which includes lines such as the following:

```
/home/user/.smileycoin/smileycoin.conf
It is recommended you use the following random password:
rpcuser=smileycoinrpc
rpcpassword=EAUbvD7ddK7eiS1izojpb9ZgMdqVsb36dL8KcAjDKyzL
```

Just copy the two important lines (`rpcuser=...` and `rpcpassword=...`) and insert them into the `smileycoin.conf` file:
If you already started the server once, then the folder should already exist:

```
cd
mkdir .smileycoin  # should return an error
ls .smileycoin     # should show some files
cd .smileycoin
cat >> smileycoin.conf  # append to the file .smileycoin.conf
#paste in the two lines from the error message above:
rpcuser=smileycoinrpc
pcpassword=EA...
ctrl-D # hold control while presing D
```

The configuration file can be used to control the behaviour of the server in various ways, connect to specific nodes on the network and so forth. The above two lines are the only ones which are always needed.
For more information see e.g. `https://github.com/bitcoin/bitcoin/blob/master/share/examples/b`

3.4 Overview

Homework: Pick up a wallet to use (WIN, MAC, Linux)
UI: First Sweep and later Import the private key issued as a part of the course
See the handout for the homework task

3.4.1 Handout

The first thing to do is to obtain a wallet and start experimenting with it. The primary wallet should be one for Windows, Mac OSX or Linux. The secondary is the HTML5 wallet. You will need to use both.

Students enrolled in the cryptocurrency course at the University of Iceland will receive an address and a corresponding private key (keys are explained later).

As a part of different exercises this private key should be

- swept into the HTML5 wallet
- imported into the primary wallet.

These functions are not the same. The playlist includes a video on how to sweep and import keys.

Several of the exercises will refer to this private key or corresponding address.

UI students: Exercises are listed in Canvas. Some are also listed individually in handouts such as this one. If there are conflict between descriptions of exercises/homework, Canvas is correct.

Task 0: Solve some exercises somewhere in the tutor-web to earn some SMLY&have them sent to the base address (UI students: see also the Google doc). This task will be listed as **solved** once the base address has received funds from the tutor-web wallet.

4 Compiling the wallet

4.1 The Linux steps

- Get a computer running Linux
- Use git to download the SMLY wallet source
- Run the tools to compile the code
- Install the wallet where you want it

Quick video commenting on these slides

4.2 Get a computer running Linux

Any Linux machine will do.

Some favourites:

- NUC
- Raspberry pi
- Any PC

Caveat: An old computer will do, but if it is **very** small (low on memory/disc space/slow) then it may be difficult to compile or run the wallet.

4.3 Linux: Use git to download the SMLY wallet source

Simple:

```
git clone http://github.com/tutor-web/smileyCoin
```

Gives a new folder, smileyCoin

4.4 Linux: Run the tools to compile the code

- ./autogen.sh
- ./configure
- make

Caveat: May need to modify src/Makefile on a Raspberry pi.

4.5 Linux: Install the wallet where you want it

Typical

- cd
- mkdir bin
- cd smileyCoin/src
- mv smileycoind smileycoin-cli $HOME/bin

4.5.1 Handout

What the commands mean...

missing

4.6 Windows: Compiling the wallet

missing...

4.7 Mac OSX: Compiling the wallet

missing...

4.8 Overview

This section has given an overview of how to pick up a Smileycoin wallet
The Handout subsection describes some homework. Your instructor will give more detail.

4.8.1 Handout

Homework: Add some material to any single subsection of this section.

5 Introduction to the SMLY command line

5.1 The SMLY command line

The core wallets have a **command line** which can be used to access information as well as send and receive coins
Example:

- Linux shell: smileycoin-cli getinfo
- Windows/MAC: open the debug window and type “getinfo”

Example:

- sendtoaddress BEtZyyYqDXqmRJJ45nnL15cuASfiXg9Yik 5

A video explanation is available

5.1.1 Handout

Under Linux the wallet comes in the form of two programs, `smileycoin` and `smileycoin-cli`. The former runs in background and the latter is the **command line interface**, used to give commands from the Linux command line.
Anything which can be done on the command line can also be given in the GUI command window, so the following are equivalent.

- Linux shell: `smileycoin-cli getinfo`
- Windows/MAC: open the debug window and type `getinfo`

5.1.2 Examples

Some useful commands
* “sendtoaddress BEtZyyYqDXqmRJJ45nnL15cuASfiXg9Yik 5“
* “help“
* “getinfo“
* “help getinfo“

6 Basic cryptocurrency economics

6.1 Background

Prices of cryptocurrencies have fluctuated wildly
Prices of several cryptocurrencies have started or risen to very high level before plummeting
What are the driving factors behind these fluctuations?

6.2 The issues

The main issues which drive the price of cryptocurrencies appear to be

- supply and demand
- coinbase (mining)
- difficulty
- pump and dump
- hype and fud
- airdrop
- lack of - or increase in use cases
- speculation vs investment
- change (increase) in public interest

These relate to **supply and demand**. Other economic aspects include

- Donations
- Divident payments
- Universal Basic Income (UBI)

6.3 The coinbase, difficulty and mining strategy

Each coin has a built-in plan for the generation of new blocks and new coins
Bitcoin:

- A new block should be made every 10 minutes
- The coinbase for Bitcoin is currently 12.5 Satoshi
- The coinbase is also the miner's fee
- The miner receives 12.5 BTC for mining a new block

Smileycoin:

- Same principle but 10,000 SMLY and one block per 3 minutes
- Coinbase is halved approximately every 7 years
- Miner receives only 10% of the coinbase

The strategy: The "hash" of the block must decrease (**difficulty** increases) if the blocks are generated too fast.

6.3.1 Handout

The student should do some research into hash functions.

See for example Example: Bitcoin `http://bit.ly/2PXIpYR`.
and Example: Litecoin `http://bit.ly/2oCpdTZ`.

6.4 Mining: The tragedy of the commons

```
Block Hash
5000 000000004d78d2a8a93a1d20a24d721268690bebd2b51f7e80657d57e226eef9
10000 0000000099c744455f58e6c6e98b671e1bf7f37346bfd4cf5d0274ad8ee660cb
25000 00000000ae4b125eb183e689b7231eafa8c992d5b8c952d9f3cd30a79a788ddf
50000 000000001aeae195809d120b5d66a39c83eb48792e068f8ea1fea19d84a4278a
100000 000000000003ba27aa200b1cecaad478d2b00432346c3f1f3986da1afd33e506
200000 000000000000034a7dedef4a161fa058a2d67a173a90155f3a2fe6fc132e0ebf
300000 000000000000000082ccf8f1557c5d40b21edabb18d2d691cfbf87118bac7254
518367 000000000000000000164ac8a0f61d8157b0920d13cb53cb7d47610bde077898
```

This would be called the “tragedy of the commons” in fisheries: The problem is that there is open access to a new resource and the fee for entry (zero) is not high enough.

6.5 Mining development

- Bitcoin: Currently only large companies (“data centres”)
- Originally on desktop computer, then using GPSs, followed by ASICs
- Other coins: Commonly “mining pool” (grafarahópar?), but similar development
- SMLY: Still mostly “solo” mining (coinbase-split is difficult for pools to implement)

Example: `http://prohashing.com/`.

6.6 Basic economics

Supply and demand drive the price of almost anything.
The supply and demand of a cryptocurrency can be influenced by

- mining to generate new, the coinbase
- airdrop
- lack of - or increase in use cases

Increased difficulty will make mining more expensive but will NOT directly affect the price over any period of time: The increased difficulty may mean that some miners will stop mining or technological development will lead to better ASICs being used.
An **airdrop** is used to hand out large amounts of a cryptocurrency to groups of users. **Auroracoin** is such an example.
Examples of **pump and dump** or **hype and fud** abound. These are techniques used by groups and individuals who intend to affect the prices of cryptocurrencies.

6.6.1 Handout

Q: Why do the SMLY have a value?
A: Limited supply and there is some utility
Anything which has some utility and a limited supply will inevitably have some value.

In the case of SmileyCoin the direct utility is obtained by setting up a handful of cases such as smly.is etc
But like any other possible investment, there will be several other factors. Any investor will set up an investment strategy which involves several factors:

- Distribution of risk (portfolio investment)
- Most investors will think about the collapse probability (50% of cryptocurrencies in 2014 were lost by 2018)
- SMLY is a part of several international research projects
- New: Investing in a “good cause” (cf Quote Magazine)

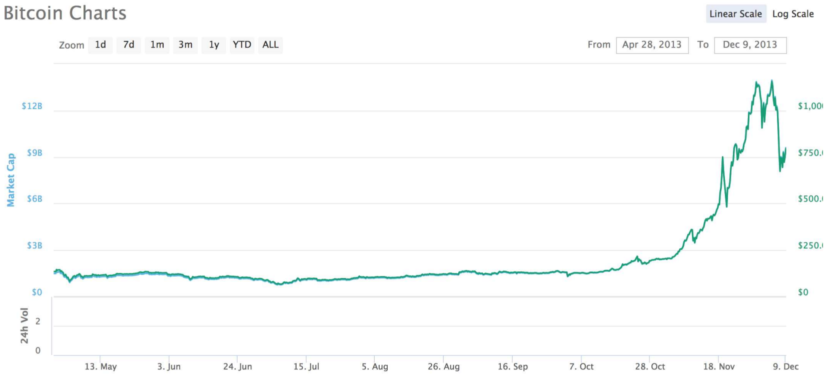
6.7 Investment and speculation

- speculation vs investment
- change (increase) in public interest
- like any asset, cryptocurrencies can be used for investments

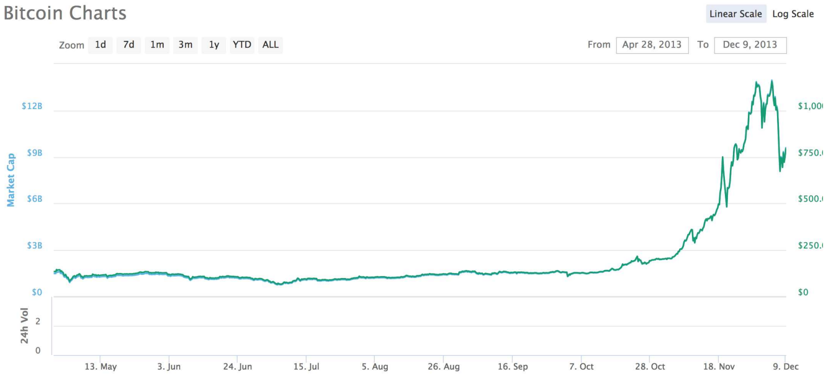
See handout at `https://tutor-web.net/comp/crypto251.0/lec01200/sl01240` for Bitcoin price development
Example of analysis, see `https://hackernoon.com/https-medium-com-zvnowman-building-a-crypto`

6.7.1 Handout

Bólur og svindl
`https://coinmarketcap.com/all/views/all/`
Skrýtin verðþróun, en verð á gulli og demöntum ræðst líka af framboði og eftirspurn (þ.m.t. væntingum og spákaupmennsku)
Bitcoin verðþróun . . .

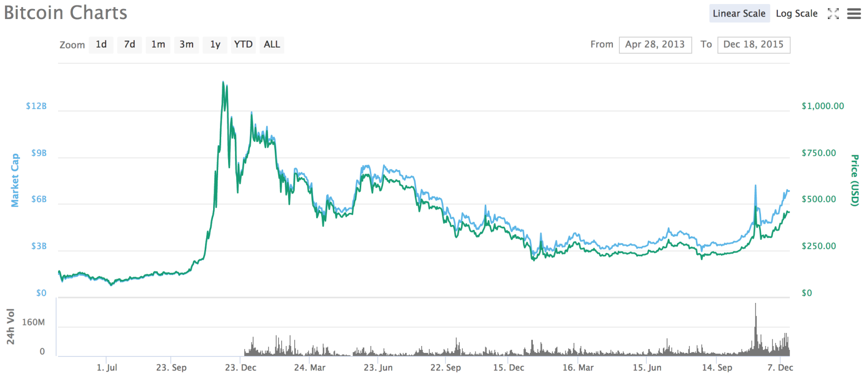


Bóla? Sprungin?
Bitcoin verðþróun út 2013



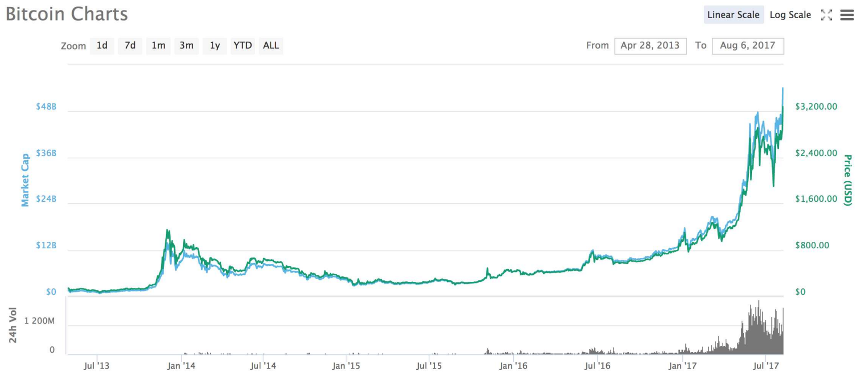
Bóla? Sprungin?

Bitcoin verðþróun til 2015



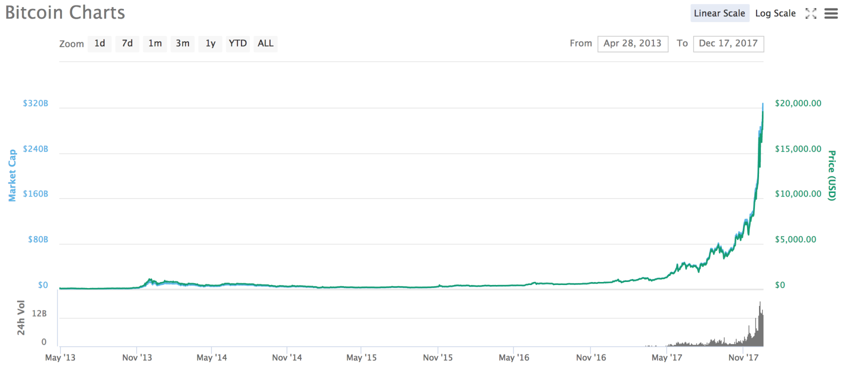
Bóla? Sprungin?

Bitcoin verðþróun inn í 2017



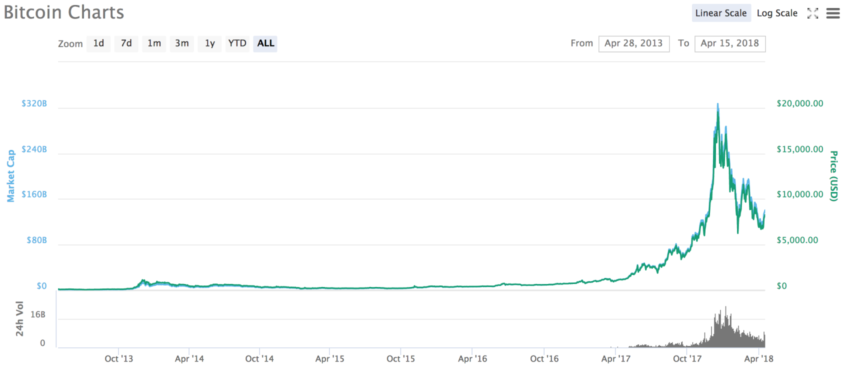
Bóla? Sprungin?

Bitcoin verðþróun út 2017



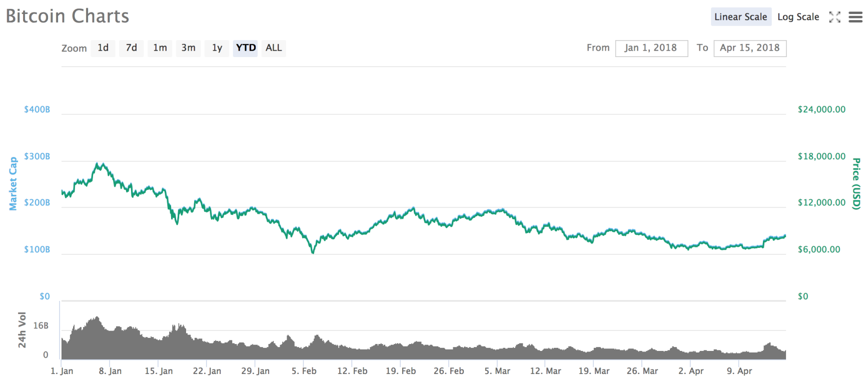
Bóla? Sprungin?

Bitcoin verðþróun út 2018



Bóla? Sprungin?

Bitcoin verðþróun árið 2018



Bóla? Sprungin?
Example of analysis, see `https://hackernoon.com/https-medium-com-zvnowman-building-a-cryptoc`

6.8 The airdrop fallacy

- airdrop

Airdrop: Giving money to everyone
Airdrop without incentive to invest or methods to spend: Useless :-)

6.9 Setting up use cases

The only way for a cryptocurrency to have a value in the long term is for it to have a use case
The use case might simply be investment:

- Bitcoin is a classic case
- Cryptocurrencies appear to be used as investment portfolios

Use cases can also consist of companies accepting a cryptocurrency as a payment (Bitcoin in particular, but also Auroracoin)
Selling donated coupons is a common method used by non-profit organisations. Coupons can easily be sold for crypto (`http://smly.is`)
In the early days of Auroracoin there were too few use cases to support holding or using the coin.
Note: Groups **could** agree that all members of a crypto group should put something up for sale and thus be ready to accept payments in the cryptocurrency.

6.10 Donations

Several cryptocurrencies encourage

- donations in the currency

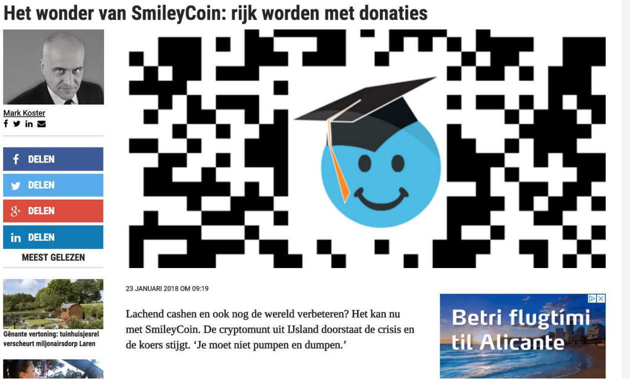
The effect on price would normally be none, unless the coinbase is used as a donation which goes directly into circulation.
An interesting twist on donations is to use

- investment in the currency as a means to support a cause financed by the currency (Education in a Suitcase)

(see handout)

6.10.1 Handout

Quote



Het wonder van SmileyCoin: rijk worden met donaties

Mark Koster

f DELEN

DELEN

DELEN

in DELEN

MEEST GELEZEN

Gênante vertoning: tuinhuisjesrel verscheurt miljonairsdorp Laren

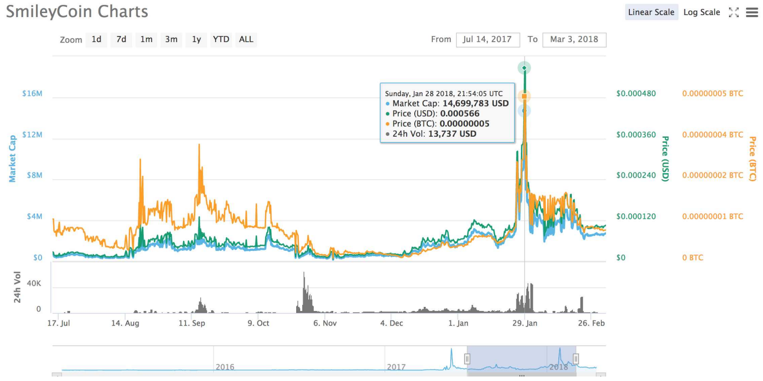
23 JANUARI 2018 OM 09:19

Lachend cashen en ook nog de wereld verbeteren? Het kan nu met SmileyCoin. De cryptomunt uit IJsland doorstaat de crisis en de koers stijgt. ‘Je moet niet pumpen en dumpen.’

(The miracle of SmileyCoin: getting rich with donations)

See `http://www.quotenet.nl/Nieuws/Het-wonder-van-SmileyCoin-rijk-worden-met-donaties-2`

The effect of a single article. . .



6.11 Divident payments

Reasoning:

- Reduce dumping
- Increase investment incentive

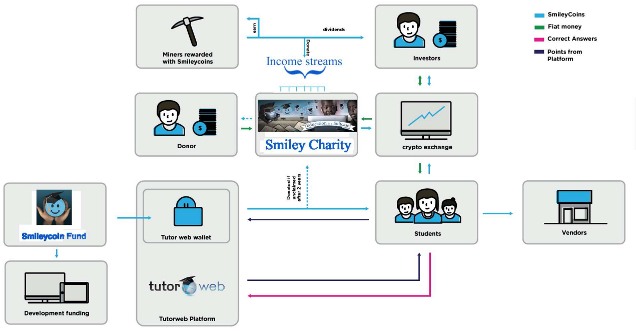
Smileycoin approach:

- Fixed portion, 45%, of coinbase goes to dividends
- Recipient group: Addresses with at least 25 M SMLY
- Method: Oldest untouched address receives entire next payment

Key figures:

- Amount: 4500 SMLY per block
- Frequency: 480 payments per day
- Rich list: `https://chainz.cryptoid.info/smly/#`!rich
- 180 addresses as of Sept 7, 2018

6.12 The SmileyCoin economy



6.12.1 Handout

The SmileyCoin economy: Miners earn a few SMLY from mining but the coinbase is mostly (i) donated in 10 income streams to the Smiley Charity (fully automatic and decentralised) or (ii) paid as dividends to over 200 large supporters (also fully automatic and decentralised). Donors can also donate fiat directly to the Smiley Charity. The SmileyCoin Fund supporting the tutor-web will support other projects. Students earn SMLY while studying in the tutor-web. Any holders of SMLY can sell SmileyCoin on cryptocurrency exchanges. Vendors supporting the projects provide various discount coupons for sale on smly.is where students and other can purchase them for SMLY. Students may redeem or donate their hard-earned SMLY; and non-redeemed SMLY are eventually donated to the Smiley Charity.
Missing from the graphic is how the Smiley Charity pays forward the SmileyCoin income streams to include other charities as recipients. The forward payments are automatic and transparent.

6.13 Cryptocurrencies as a Universal Basic Income

Universal Basic Income (UBI) is a popular term and commonly linked to technological developments which may eventually lead to mass unemployment.
A cryptocurrency could in principle be used as for UBI through a number of means:

- airdrop
- splitting the coinbase
- splitting the transaction fee

but all of these only increase supply, not demand. Hence, none of these will work unless there is simultaneously a setup which provides demand for the coin.
An airdrop could be implemented through a premine, but experience to date suggests that this is not a very good idea (Auroracoin, Smileycoin) and it would be better to use the coinbase+fees for this purpose.
Several cryptocurrency-based UBI projects are listed at `https://bitcointalk.org/index.php?topic=3242065.0`

6.14 Solving UBI implementation issues: delivery and demand

In addition to problems with a premine, the coinbase alone is unlikely to be enough (exercise: test the increase in supply and demand needed to make this work for Iceland with e.g. 100,000 recipients of a UBI equivalent of 100,000 ISK per month).
If a coin is set up such that the UBI recipient are active parts of the community through

- sending the UBI to other addresses (generating a fee)
- putting a service or object up for sale

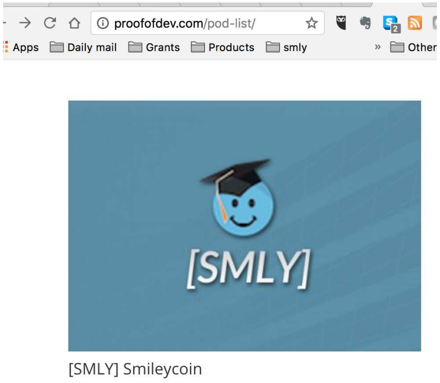
then a UBI might be feasible.
Supply would mostly be through the transaction fees.
Demand would be generated by the users themselves.

6.14.1 Examples

Consider 100,000 recipients of a UBI equivalent of 100,000 ISK per month and suppose this has to come out of the coinbase.
The coinbase of 10,000 new coins are generated in each block, every 3 minutes. With 20 blocks per hour, 480 blocks are generated per day, or 14,400 blocks per month. The total coinbase is therefore 144 million SMLY per month.
For the above UBI, this coinbase of 144 million SMLY has to be worth 10,000 million ISK per month, so each SMLY needs to be worth 10000/144 or 69 ISK.
There are currently almost 30 bn SMLY in circulation (30 10ˆ9) so for the above to work, the market value of all SmileyCoin in circulation needs to be over 2 thousand billion ISK (2 10ˆ12). For comparison, the amount of money in Iceland (as measured by M0) is about 40 bn. Even taking into account money in savings accounts etc (M3), the market value of SMLY needs to be far too high compared to a typical economy, if the UBI is to be generated from the coinbase alone.
Of course in a typical economy, wages are not paid by printing money. Once paid, wages are first used to pay income taxes and then purchase goods, resulting in sales taxes. These taxes are subsequently used to pay wages again. A crypto-based UBI needs to mimic this circular behaviour of wages and taxes.

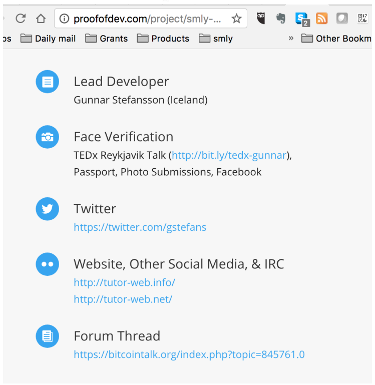
6.15 Keeping or avoiding developer anonymity

Svindl og svínarí? Ýmsar leiðir, t.d. proof of developer . . .



proofofdev.com/pod-list/

[SMLY] Smileycoin



proofofdev.com/project/smly-...

Lead Developer
Gunnar Stefansson (Iceland)

Face Verification
TEDx Reykjavik Talk (http://bit.ly/tedx-gunnar),
Passport, Photo Submissions, Facebook

Twitter
https://twitter.com/gstefans

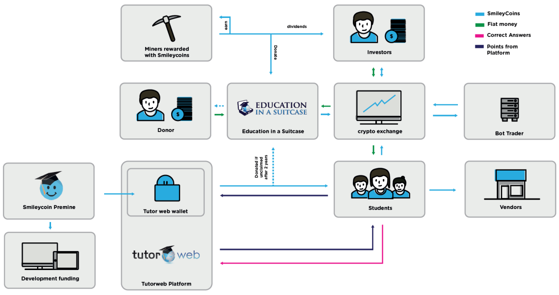
Website, Other Social Media, & IRC
http://tutor-web.info/
http://tutor-web.net/

Forum Thread
https://bitcointalk.org/index.php?topic=845761.0

En Baldur og Satoshi geta líka búið til sína mynt - nafnlaust.
Mörg dæmi um mynt til að hafa fé af fjárfestum. . .
Tilraunaverkefni (rannsóknaverkefni)

- Tengsl umbunar og vinnu/einkunna nemenda - fjölvalsspurningar
- Notkun umbunar fyrir verkefnaskil (t.d. semja texta)
- Gera tutor-web sjálfbært (umbuna fyrir þróun)
- Lengri tíma: Áhrif í Kenýa – t.d. 1 USD/dag?

Flæði Broskalla



Tilraunaverkefni (kennsluverkefni)

- Bestun: Verslun með rafmyntir (þ.m.t. arbitrage)
- Viðskiptavakt
- Veðmál á keðjunni (sendtoaddress BCJW4iZw7PechFHgtqqSdHmymjnFA6LjNJ 10)
- Skilaboð eftir keðjunni
- Sjálfvirk myntskipti
- Frumskipti (atomic swap)
- o.s.frv.

Sjá ýmsar Steemit greinar

Meira

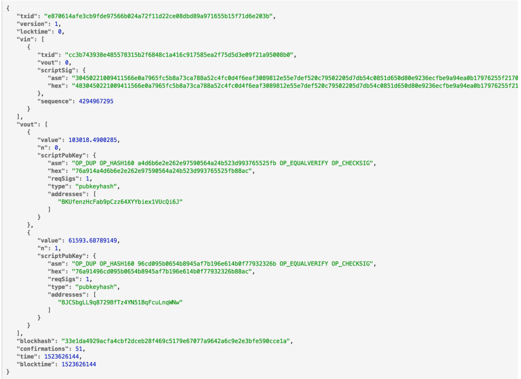
7 The transaction

7.1 Background

The concept of a transaction as a description of transfer of funds is simple but not enough
How does one guarantee that the funds are not sent twice?
How does one ensure that the sender is authorised to spend the funds?
To see how this is done we need to look inside the transactions and study their structure

7.2 A typical transaction

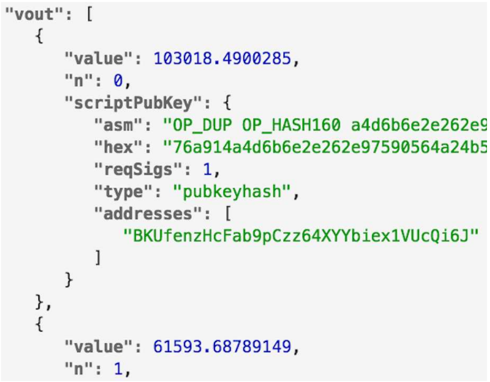
Consider a specific SMLY transaction, eg `e870614afe3cb9fde97566b024a72f11d22ce08dbd89a971655b15`
which can be seen in block 332353, at
`https://chainz.cryptoid.info/smly/block.dws?33e1da4929acfa4cbf2dceb28f469c5179e67077a9`
A summary of the transaction is given at
`https://chainz.cryptoid.info/smly/tx.dws?e870614afe3cb9fde97566b024a72f11d22ce08dbd89a`
but we want to see some of the detail.



```
{
   "txid": "e870614afe3cb9fde97566b024a72f11d22ce08dbd89a971655b15f71d6e203b",
   "version": 1,
   "locktime": 0,
   "vin": [
      {
         "txid": "cc3b743938e485578315b2f6848c1a416c917585ea2f75d5d3e09f21a95008b0",
         "vout": 0,
         "scriptSig": {
            "asm": "30450221009411566e0a7965fc5b8a73ca788a52c4fc0d4f6eaf3089812e55e7def520c79502205d7db54c0851d650d80e9236ecfbe9a94ea0b17976255f2170
            "hex": "4830450221009411566e0a7965fc5b8a73ca788a52c4fc0d4f6eaf3089812e55e7def520c79502205d7db54c0851d650d80e9236ecfbe9a94ea0b17976255f21
         },
         "sequence": 4294967295
      }
   ],
   "vout": [
      {
         "value": 103018.4900285,
         "n": 0,
         "scriptPubKey": {
            "asm": "OP_DUP OP_HASH160 a4d6b6e2e262e97590564a24b523d993765525fb OP_EQUALVERIFY OP_CHECKSIG",
            "hex": "76a914a4d6b6e2e262e97590564a24b523d993765525fb88ac",
            "reqSigs": 1,
            "type": "pubkeyhash",
            "addresses": [
               "BKUfenzHcFab9pCzz64XYYbiex1VUcQi6J"
            ]
         }
      },
      {
         "value": 61593.68789149,
         "n": 1,
         "scriptPubKey": {
            "asm": "OP_DUP OP_HASH160 96cd095b0654b8945af7b196e614b0f77932326b OP_EQUALVERIFY OP_CHECKSIG",
            "hex": "76a91496cd095b0654b8945af7b196e614b0f77932326b88ac",
            "reqSigs": 1,
            "type": "pubkeyhash",
            "addresses": [
               "BJCSbgLL9q8729BfTz4YN51BqFcuLnqWNw"
            ]
         }
      }
   ],
   "blockhash": "33e1da4929acfa4cbf2dceb28f469c5179e67077a9642a6c9e2e3bfe590cce1a",
   "confirmations": 51,
   "time": 1523626144,
   "blocktime": 1523626144
}
```

7.3 Inside the transaction: The output

Consider the outputs from transaction e870614afe3cb9fde97566b024a72f11d22ce08dbd89a971655b15f71d6e

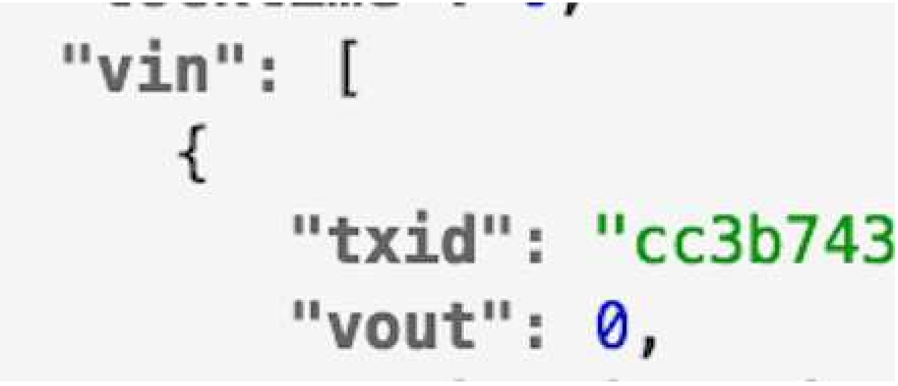


```
"vout": [
   {
      "value": 103018.4900285,
      "n": 0,
      "scriptPubKey": {
         "asm": "OP_DUP OP_HASH160 a4d6b6e2e262e9
         "hex": "76a914a4d6b6e2e262e97590564a24b5
         "reqSigs": 1,
         "type": "pubkeyhash",
         "addresses": [
            "BKUfenzHcFab9pCzz64XYYbiex1VUcQi6J"
         ]
      }
   },
   {
      "value": 61593.68789149,
      "n": 1,
```

The outputs form two UTXOs: “n”=0 og “n”=1
These can later be referenced, e.g. as UTXO n=0 from Tx=e870614afe3cb9fde97566b024a72f11d22ce08dbd89a971655b15f71d6e203b

7.4 Inside the transaction: The input

TxId: e870614afe3cb9fde97566b024a72f11d22ce08dbd89a971655b15f71d6e203b



```
  "vin": [
      {
          "txid": "cc3b743
          "vout": 0,
```

The input is only defined as an older output, which has not been spent, UTXO, as the following components:

- Start of input description: vin
- The input transaction refers to an older transaction: TxId
- “vout” refers to a numbered output (“n”) in that transaction
- NB: The amount is not listed!
- NB: The address is not listed!

So the input to our transaction is output number 0 from transaction cc3b743938e485578315b2f6848c1a416c917585ea2f75d5d3e09f21a95008b0
We can verify by looking up that UTXO.

7.5 The UTXO

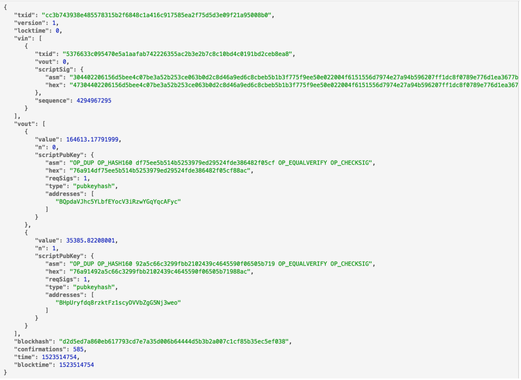
We have seen that

- the input to transaction e870614afe3cb9fde97566b024a72f11d22ce08dbd89a971655b15f71d6e203b is
- the UTXO from transaction cc3b743938e485578315b2f6848c1a416c917585ea2f75d5d3e09f21a95008b

To verify this we can look up that UTXO as seen in the handout.

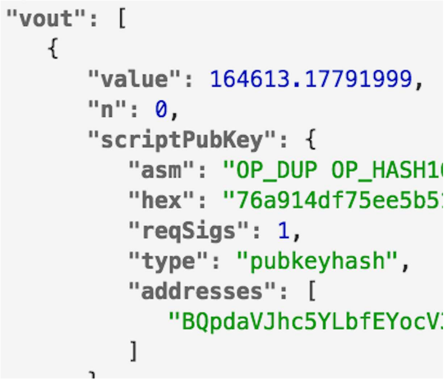
7.5.1 Handout

To verify this we can look up that UTXO and we find



```
{
   "txid": "cc3b743938e485578315b2f6848c1a416c917585ea2f75d5d3e09f21a95008b0",
   "version": 1,
   "locktime": 0,
   "vin": [
      {
         "txid": "5376633c095470e5a1aafab742226355ac2b3e2b7c8c10bd4c0191bd2ceb8ea8",
         "vout": 0,
         "scriptSig": {
            "asm": "304402206156d5bee4c07be3a52b253ce063b0d2c8d46a9ed6c8cbeb5b1b3f775f9ee50e022004f6151556d7974e27a94b596207ff1dc8f0789e776d1ea3677b
            "hex": "47304402206156d5bee4c07be3a52b253ce063b0d2c8d46a9ed6c8cbeb5b1b3f775f9ee50e022004f6151556d7974e27a94b596207ff1dc8f0789e776d1ea367
         },
         "sequence": 4294967295
      }
   ],
   "vout": [
      {
         "value": 164613.17791999,
         "n": 0,
         "scriptPubKey": {
            "asm": "OP_DUP OP_HASH160 df75ee5b514b5253979ed29524fde386482f05cf OP_EQUALVERIFY OP_CHECKSIG",
            "hex": "76a914df75ee5b514b5253979ed29524fde386482f05cf88ac",
            "reqSigs": 1,
            "type": "pubkeyhash",
            "addresses": [
               "BQpdaVJhc5YLbfEYocV3iRzwYGqYqcAFyc"
            ]
         }
      },
      {
         "value": 35385.82208001,
         "n": 1,
         "scriptPubKey": {
            "asm": "OP_DUP OP_HASH160 92a5c66c3299fbb2102439c4645590f06505b719 OP_EQUALVERIFY OP_CHECKSIG",
            "hex": "76a91492a5c66c3299fbb2102439c4645590f06505b71988ac",
            "reqSigs": 1,
            "type": "pubkeyhash",
            "addresses": [
               "BHpUryfdq8rzktFz1scyDVVbZgG5Nj3weo"
            ]
         }
      }
   ],
   "blockhash": "d2d5ed7a860eb617793cd7e7a35d006b64444d5b3b2a007c1cf85b35ec5ef038",
   "confirmations": 585,
   "time": 1523514754,
   "blocktime": 1523514754
}
```

or specifically



```
"vout": [
   {
      "value": 164613.17791999,
      "n": 0,
      "scriptPubKey": {
         "asm": "OP_DUP OP_HASH1
         "hex": "76a914df75ee5b5
         "reqSigs": 1,
         "type": "pubkeyhash",
         "addresses": [
            "BQpdaVJhc5YLbfEYocV
         ]
      }
```

7.6 Keys

Cryptocurrencies use cryptographic keys
For example, ownership is demonstrated using a combination of keys and addresses

- public-private key pairs
- Private key -> public key -> address

This will be explained in more detail later.

- An address can be freely distributed
- The private key is never disclosed
- A transaction can be signed using the private key
- A signature can be **verified** using the public key
- The public key is only disclosed when a transaction is spent

A **spending transaction** publishes the public key and a signature.

7.6.1 Handout

A **private key** is just a string of random numbers.
A **public key** is generated from the private key.
An **address** is generated from the public key.
A good description of the process is available:
`https://en.bitcoin.it/wiki/Technical_background_of_version_1_Bitcoin_addresses`

7.7 Spending the UTXO

The permission to spend the UTXO is determined by the programming code written into the transaction.
Will be described later in the course, but a short code snippet is seen in every transaction.
It is an incomplete snippet, usually with components of the form

- OP_DUP
- OP_HASH160
- a4d6b6e2e262e97590564a24b523d993765525fb
- OP_EQUALVERIFY
- OP_CHECKSIG

To spend this UTXO the spending transaction needs to prepend to this another snippet so the combined code can be run and will return "TRUE" and nothing else.
Completion of this particular snippet is done with

- signature
- public key

7.7.1 Handout

See `https://en.bitcoin.it/wiki/Script` for a description of the codes involved.
Note that **OP_HASH160** involves two operations: The input is hashed twice: first with SHA-256 and then with RIPEMD-160

7.8 The transaction on the command line

Step-by-step example of how to generate, sign, check, announce and inspect a transaction - to be done in detail in class

- `listunspent`
- `createrawtransaction '[{"txid": "fbd60d37acfb30eba7153db741dce7d1ebf71c0ee0ec880`
  `"vout": 1}]' '{"B79tjNk8oZktdd7DLnznKXu9UA67GMWP9g": 2000,`
  `"BHgx5rehx2Wkx4wME2DXwZAHL7KskUjXmK": 2499}'`
- `signrawtransaction 01000000018fba0254869ea2fb0288ece00e1cf7ebd1e7dc41b73d15a7eb30`
- `decoderawtransaction 01000000018fba0254869ea2fb0288ece00e1cf7ebd1e7dc41b73d15a7eb`
- `sendrawtransaction 01000000018fba0254869ea2fb0288ece00e1cf7ebd1e7dc41b73d15a7eb30`
- `getrawtransaction e98b533cf3290fa58c23074aa0b1e273e25e4756321155e7ad165f2d3ed6176`
- `decoderawtransaction 01000000018fba0254869ea2fb0288ece00e1cf7ebd1e7dc41b73d15a7eb`

This is just the **how-to**. The next few lectures will go into what is actually going on!

7.8.1 Handout

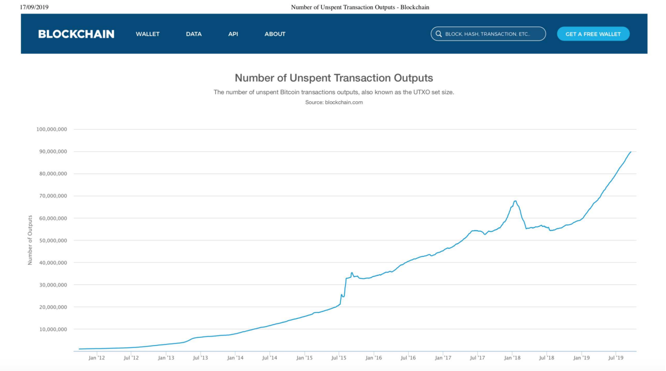
Example of how to generate, sign, check, announce and inspect a transaction

- `listunspent`
- `createrawtransaction '[{"txid": "fbd60d37acfb30eba7153db741dce7d1ebf71c0ee0ec8802`
  `"vout": 1}]' '{"B79tjNk8oZktdd7DLnznKXu9UA67GMWP9g": 2000, "BHgx5rehx2Wkx4wME2DX`
  `2499}'`

- `signrawtransaction 01000000018fba0254869ea2fb0288ece00e1cf7ebd1e7dc41b73d15a7eb30f`
- `decoderawtransaction 01000000018fba0254869ea2fb0288ece00e1cf7ebd1e7dc41b73d15a7eb3`
- `sendrawtransaction 01000000018fba0254869ea2fb0288ece00e1cf7ebd1e7dc41b73d15a7eb30f`
- `getrawtransaction e98b533cf3290fa58c23074aa0b1e273e25e4756321155e7ad165f2d3ed6176C`
- `decoderawtransaction 01000000018fba0254869ea2fb0288ece00e1cf7ebd1e7dc41b73d15a7eb3`

7.9 The UTXO set

The UTXO set has a tendency to increase in size.
For Bitcoin (from `https://www.blockchain.com/charts/utxo-count?timespan=all`):



17/09/2019

Number of Unspent Transaction Outputs - Blockchain

Number of Unspent Transaction Outputs

The number of unspent Bitcoin transactions outputs, also known as the UTXO set size.

Source: blockchain.com

7.9.1 Handout

The UTXO is one of the basic concepts in Bitcoin and other cryptocurrencies. Each unspent transaction output represents a unit which the holder of a private key can spend.
Each transaction results in one or more UTXO and only these **unspent** outputs can be used as inputs in a subsequent transaction.
A full node verifies transactions and every full node therefore needs to keep track of the entire UTXO set.

7.10 The transaction fee

Most transactions include a transaction fee
The fee is simply the difference between the inputs and the outputs
The fee is not explicitly specified

7.10.1 Examples

Most transactions include a transaction fee, but you can explicitly define a transaction with no transaction fee.
The fee is simply the difference between the inputs and the outputs

7.11 Manual transaction example - maintaining a fund

If a wallet is asked to send x SMLY it will just find some unspent transactions and aggregate them as input, send x to the destination and make a new address for the change, after taking some for the transaction fee.
There are many instances when one wants to do things differently. For example one may want to maintain all the funds under a single address for transparency.
This is how the Pineapple Fund worked and this is how the SmileyCoin Fund works.
`https://www.blockchain.com/btc/tx/081f68e146922f23039bf67a5bdaa53365b311b9dba5d80163c`

7.11.1 Examples

Homework: Send 100 SMLY to an address A.
Use createrawtransaction to send 10 SMLY from A to C and 89 back to A in a single transaction, leaving 1 SMLY for the miner.
This is how the Pineapple Fund worked:
`https://www.blockchain.com/btc/tx/081f68e146922f23039bf67a5bdaa53365b311b9dba5d80163c6`

8 The block, the blockchain and the network

8.1 The block and the chain

- Alice and Bob have **wallets**
- A **transaction** is generated by Alice's wallet when Alice sends Bob Smileycoins
- Alice's wallets **broadcast** the new transaction to the network
- The transaction then enters the **mempool**
- Any wallet on the network can examine the transaction
- A **miner** aggregates these transactions into a **block**
- A miner may simply be a wallet set to **mine**
- The block is **linked** to the previous blocks in a **chain**
- The miner broadcasts the block to the network
- A block needs to satisfy certain **difficulty** criteria

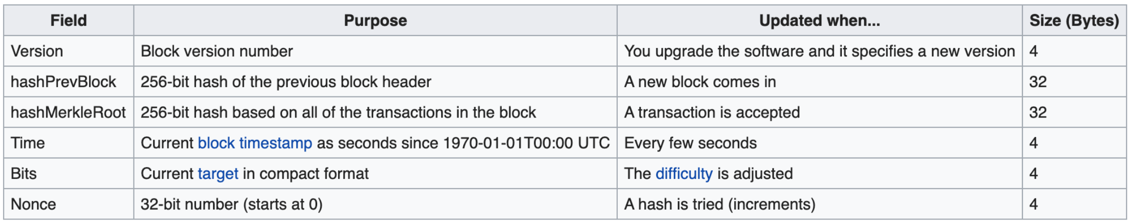
(more later)

8.2 The hash and the nonce

See `https://en.bitcoin.it/wiki/Block_hashing_algorithm` to see the code below and a description of the composition of the header

8.2.1 Handout

The block hashing algorithm produces a sha256d hash of 256 bits (32 bytes) based on the following 640 bit input:



Field	Purpose	Updated when...	Size (Bytes)
Version	Block version number	You upgrade the software and it specifies a new version	4
hashPrevBlock	256-bit hash of the previous block header	A new block comes in	32
hashMerkleRoot	256-bit hash based on all of the transactions in the block	A transaction is accepted	32
Time	Current block timestamp as seconds since 1970-01-01T00:00 UTC	Every few seconds	4
Bits	Current target in compact format	The difficulty is adjusted	4
Nonce	32-bit number (starts at 0)	A hash is tried (increments)	4

(from `https://en.bitcoin.it/wiki/Block_hashing_algorithm`)

8.2.2 Examples

Example python code:

```
>>> import hashlib
>>> header_hex = ("01000000" +
 "81cd02ab7e569e8bcd9317e2fe99f2de44d49ab2b8851ba4a308000000000000" +
 "e320b6c2fffc8d750423db8b1eb942ae710e951ed797f7affc8892b0f1fc122b" +
 "c7f5d74d" +
 "f2b9441a" +
 "42a14695")
>>> header_bin = header_hex.decode('hex')
>>> hash = hashlib.sha256(hashlib.sha256(header_bin).digest()).digest()
>>> hash.encode('hex_codec')
'1dbd981fe6985776b644b173a4d0385ddc1aa2a829688d1e0000000000000000'
>>> hash[::-1].encode('hex_codec')
'00000000000000001e8d6829a8a21adc5d38d0a473b144b6765798e61f98bd1d'
```

(from `https://en.bitcoin.it/wiki/Block_hashing_algorithm`)

8.3 The network

The full (core) **wallets** are really just computer programs which “talk” together across the Internet, forming “points” which are connected using a protocol.
Each such point is called a **node**.
The collection of SmileCoin nodes forms the SmileyCoin network. This network can be studied in several ways and some of the block explorers do so:
`https://chainz.cryptoid.info/smly/#`!network
When a node sees a transaction, this is sent across the network. This collection is called the **mempool**.
A miner picks up transactions in the mempool and puts them into a block. Note that different miners may have seen different transaction so they may no all be mining the same content into a block.
See the handout to look at commands to link to other computers and view the mempool.

8.3.1 Handout

The command `getrawmempool`displays the transactions in the mempool. This command is particularly useful if mining is slow, just to verify that the transaction is being sent across the network.
When the wallet starts up, it has a hard-wired IP address (the dnsseed) which it connects to. That computer gives the wallet the addresses of other computers on the network.
It is possible to enhance connectivity by connecting to more nodes or just to specific nodes.
This is done using the `addnode` command:

```
addnode 191.121.45.21 add
```

Lists of nodes can be obtained from block explorers, e.g. the **node list** at `https://chainz.cryptoid.info/sm`

9 Cryptocurrency mining

9.1 Mining, hashes and the cryptography puzzle

Bitcoin mining uses the hashcash proof of work function; the hashcash algorithm requires the following parameters: a service string, a nonce, and a counter.
See for example Example: Bitcoin `http://bit.ly/2PXIpYR`.
and Example: Litecoin `http://bit.ly/2oCpdTZ`.
(more later)

9.2 Mining from a wallet

Desktop mining is not reasonable for Bitcoin, Litecoin or other heavily mined coins.
It is, however, quite feasible for SmileyCoin (in 2019).
Most coins have gone through phases where mining is initially done using a computer's CPU, then a graphics card followed by specialised hardware. In-between, mining pools are typically set up, where miners cooperate on mining a coin and share block rewards and transaction fees.
Mining outside mining pools is called **solo mining**.
SmileyCoin is typically still mined by individual computers (in 2019).

9.2.1 Handout

Under Linux one can start the SmileyCoin daemon from the command line using

```
smileycoind -algo=qubit -gen -genproclimit=1 --server
```

The `gen` command-lin option sets the coin generation to true and `genproclimit` sets the number of cores to be used.
Alternatively, the options can be put into smileycoin.conf

```
algo=qubit
genproclimit=4
```

With this configuration file, the actual mining must then be turned on after the daemon is started, using either the command line

```
smileycoin-cli setgenerate true 1
```

or a similar command, `setgenerate true 1` from within the wallet command window.
The numeral `1` here refers to the number of cores to be used for mining.
For laptops it is **essential** to set a bound on the number of cores used by the wallet to avoid overheating the computer.
Under Linux the default configuration options are read in from the file

```
.smileycoin/smileycoin.conf
```

under the user's home directory.
On the Mac OSX this file is stored as

```
Library/Application Support/Smileycoin/smileycoin.conf
```

(beware of the space in the directory name).

9.3 GPU mining

Screen displays on desktop computers are handled by graphics chips with considerable computing power.
Graphics cards are dedicated cards, inserted into the computer, to handle complex graphics.
These graphics cards are much more powerful for mining than is the typical central processing units (CPU) of a computer.
Generic software is freely available to mine arbitrary coins using such graphics cards.

9.3.1 Handout

One popular miner is `bfgminer`, available at `http://bfgminer.org//` or, on Ubuntu:

```
apt-get install bfgminer
```

9.4 Mining using specialised hardware (ASIC mining)

SmileyCoin can be mined using e.g. the `bfgminer` with a scrypt ASIC.

9.4.1 Handout

For SMLY mining using a Scrypt ASIC, the following has been tested.
There are 1 or 2 machines involved. The following assumes the (Linux/Win/OSX) wallet runs on one machine and the miner (bfgminer) on another, where the ASIC is hooked up.
First the wallet machine. In the config file, usually

```
~/.smileycoin/smileycoin.conf
```

on a Linux machine, make sure that you have the two lines

```
rpcuser=<your-user-for-RPC>
rpcpassword=<your-password-for-RPC>
```

where normally one just uses the user and password provided when you set up the wallet.
You will need these later, when you connect bfgminer to the wallet.
Next, you'll want lines of the following form:

```
server=1
rpcport=14242
rpcallowip=127.0.0.1
rpcallowip=<the-IP-of-the-mining-computer>
```

so for example, if your (bfg)miner is on the local area network with IP 192.168.1.57 then that is what you insert here so the wallet accepts incoming calls from that machine.
Also, make a note of the name or IP of the wallet computer.
Once you have this set up, make sure the wallet is running. Under Linux it'll be something like:

```
smileycoind --server &
```

Next, the machine where bfgminer runs (where the ASIC is connected). Here it should be enough to just run bfgminer off the command line. The settings for bfgminer are highly dependent on the ASIC you are using. The following are the settings for a particular Scrypt ASIC (entire command should go on one line):

```
bfgminer --scrypt
    -o http://<walletmachine>:14242
    -u <your-user-for-RPC>
    -p <your-password-for-RPC>
    -S ALL
    --set MLD:clock=600
```

where <your-user-for-RPC> is usually set to “smileycoinrpc” by default for the SMLY wallet and <your-password-for-RPC> is usually set to a long string generated at startup. You may have replaced both so make sure to check how the wallet is set up (smileycoin.conf above).
Similarly, <walletmachine> needs to be replaced by whatever you call the computer where you run the wallet.
The port here is 14242. The number is largely irrelevant, but it needs to be the same in the wallet config file as on the bfgminer command line (or the bfgminer config file). It should probably be a high number so that it does not interfere with system ports or priviliges.
This particular setup was tested on a small USB-stick miner, the Futurebit Moonlander 2.0, obtained in 2017 from `https://asicpuppy.com/magentoPuppy/index.php/fbmoonlander.html`
For that hardware you may or may not need a specias version of bfgminer: `https://bitcointalk.org/index`
The above text is slightly updated from `https://bitcointalk.org/index.php?topic=845761.msg3019528`

9.5 Mining using a small ASIC

This is for the Futurebit Moonlander 2

9.5.1 Handout

One may need to install additional Futurebit software, in addition to bfgminer and the SMLY wallet.

9.6 Which hashes and how

9.6.1 Handout

Hash functions are used in several places, from inside the script programming language through solving cryptographic puzzles as proof-of-work to linking the blocks.
The best-known use of hashing is in the cryptographic puzzle which is solved as proof-of-work to generate a valid block.
The transactions in a block are summarised into a single hash using **merkle trees**, combined with a **nonce** and **hashed**.
The most common hash function is **sha256d**, described in detail in `https://csrc.nist.gov/publications/`
Note that it is not essential for the same hash function to be used for proof-of-work as for linking the blocks.
Several other hash functions are used for Bitcoin transactions and even more are used for multi-algo coins such as SmileyCoin and Auroracoin.
more detail needed

9.7 The mining algorithm

The **sha256d** mining algorithm

9.7.1 Handout

From `https://en.bitcoin.it/wiki/Getwork`
calculate:

```
hash = SHA256(SHA256(EndianFlipForEach32Bits(First80BytesOf(data))))
```

If that meets the difficulty, you win (generated a block or share)!
If not, increment the Nonce that is a number stored in portion of the data that starts 608 bits in (bytes 76 to 79), and try again.

9.8 Mining, energy and other uses

As seen elsewhere in this document, mining Bitcoin requires a tremendous amount of computing power.
This generates heat which is commonly dissipated using fans or other methods.
Preferred locations include cool countries where it is easier to get rid of the heat.
A few use cases have taken the excess heat and used it for heating houses or other facilities.

Farmers in Iceland
`https://www.vis`
`https://www.wir`
If the facilities nee
mean that there is r
More recent intern
`https://news.bi`
`https://hotmine`
`https://www.qar`

10 Cryptography and cryptocurrencies

10.1 Cryptography use by cryptocurrencies

Cryptocurrencies use cryptography for several tasks, including:

- signing transactions using a private key
- verifying ownership of an amount to be spent using a public key
- summarising a transaction into a hash
- summaring all transactions in a block into a hash
- summarising a block into a hash
- defining criteria for a block hash to satisfy for a block to be acceptable

11 Hash function introduction

12 Elliptic curves

13 The trilogy: tutor-web, Smileycoin and Education in a Suitcase

13.1 This is just a placeholder!!

WARNING This is just a placeholder at the moment - don’t even bother reading it :-)
This whole section will become a double lecture on tw, EIAS and SMLY

- Bitcoin
- Litecoin
- Etherium
- Auroracoin
- Broskallar :-)

13.2 Where we come from

tutor-web

eias

smly

13.3 The tutor-web system

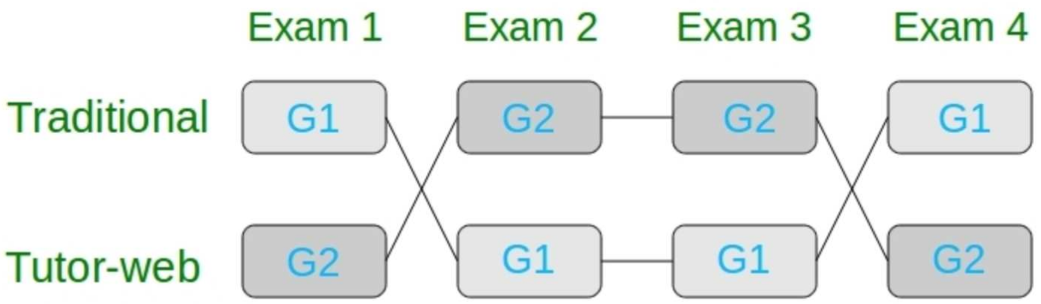
tutor-web kerfið

- tutor-web er kennslukerfi á netinu sem er opið öllum án endurgjalds: `http://tutor-web.net`
- Rannsóknar- og þróunarverkefni hóps sem tengist VoN
- Allur hugbúnaður sem kerfið notar er opinn (open source) og getur hver sem er notað og jafnvel breytt kennsluefninu (Creative Commons License)
- Styrkt af HÍ, Rannís, ESB, ráðuneytum, UNU FTP o.s.frv.

tutor-web

- Í tutor-web eru eru yfir 6000 fjölvalsæfingar í stærðfræði og tölfræði á framhalds- og háskólastigi
- Æfingarnar eru ekki til að prófa kunnáttu nemenda heldur til að þeir læri af því að svara þeim
- Nemendur geta svarað eins mörgum spurningum og þá lystir eins lengi og þeir vilja
- Nemendur og kennarar geta fylgst með hvernig gengur
- Eftir að nemandi svarar fær hann að sjá hvaða svarmöguleiki var sá rétti og útskýringu á rétta svarinu

Árangur rannsakaður



Nemendur stóðu sig betur á stöðumótum efir að hafa notað tutor-web.

13.4 sl03030

Kenya

Sudan
Ethiopia
Lake Turkana
Somalia
Uganda
Kenya
Nairobi
Indian Ocean
Lake Victoria
Mombasa
Tanzania

Education in a suitcase

Í Kenía

- er óalgengt að fólk hafi aðgang að tölvum
- ekki sjálfgefið að komast í netsamband
- getur rafmagn verið óstöðugt
- …

Lausn: Education in a suitcase

Education in a suitcase

EDUCATION IN A SUITCASE

STYRKTARFÉLAGIÐ BROSKALLAR

Dæmigerð Keníaferð

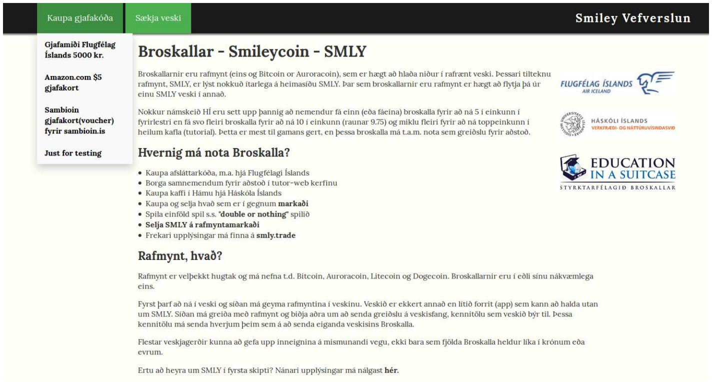
- Fangelsi í Naivasha
- Grunnskóli á Takawirieyju í Victoríuvatni
- Háskóli í Maseno
- Shivanga framhaldsskólinn í Kakamega sýslu

Broskallar - Smileycoin

- Notaðir til að verðlauna fyrir góða frammistöðu í tutor-web
- Mest til skemmtunar en eru rafmynt!
- Rannsóknir á áhrifum þess að greiða nemendum í rafmynt í kennslukerfinu
- Geta keypt kaffi, flugmiða, bíómiða, …

13.5 sl03040

http://smly.is/



Gjafamiði Flugfélag Íslands 5000 kr.

Amazon.com $5 gjafakort

Sambioin gjafakort(voucher) fyrir sambioin.is

Just for testing

Broskallar - Smileycoin - SMLY

Broskallarnir eru rafmynt (eins og Bitcoin or Auroracoin), sem er hægt að hlaða niður í rafrænt veski. Þessari tilteknu rafmynt, SMLY, er lýst nokkuð ítarlega á heimasíðu SMLY. Þar sem broskallarnir eru rafmynt er hægt að flytja þá úr einu SMLY veski í annað.

Nokkur námskeið HÍ eru sett upp þannig að nemendur fá einn (eða fáeina) broskalla fyrir að ná 5 í einkunn í fyrirlestri en fá svo fleiri broskalla fyrir að ná 10 í einkunn (raunar 9.75) og miklu fleiri fyrir að ná toppeinkunn í heilum kafla (tutorial). Þetta er mest til gamans gert, en þessa broskalla má t.a.m. nota sem greiðslu fyrir aðstoð.

Hvernig má nota Broskalla?

- Kaupa afsláttarkóða, m.a. hjá Flugfélagi Íslands
- Borga samnemendum fyrir aðstoð í tutor-web kerfinu
- Kaupa kaffi í Hámu hjá Háskóla Íslands
- Kaupa og selja hvað sem er í gegnum **markaði**
- Spila einföld spil s.s. **"double or nothing"** spilið
- **Selja SMLY á rafmyntamarkaði**
- Frekari upplýsingar má finna á **smly.trade**

Rafmynt, hvað?

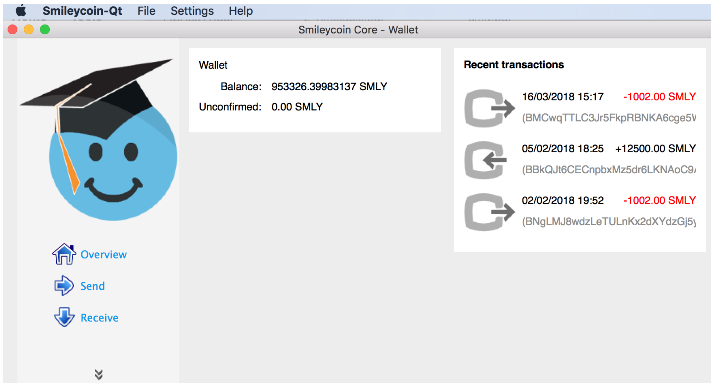
Rafmynt er velþekkt hugtak og má nefna t.d. Bitcoin, Auroracoin, Litecoin og Dogecoin. Broskallarnir eru í eðli sínu nákvæmlega eins.

Fyrst þarf að ná í veski og síðan má geyma rafmyntina í veskinu. Veskið er ekkert annað en lítið forrit (app) sem kann að halda utan um SMLY. Síðan má greiða með rafmynt og biðja aðra um að senda greiðslu á veskisfang, kennitölu sem veskið býr til. Þessa kennitölu má senda hverjum þeim sem á að senda eiganda veskisins Broskalla.

Flestar veskjagerðir kunna að gefa upp inneignina á mismunandi vegu, ekki bara sem fjölda Broskalla heldur líka í krónum eða evrum.

Ertu að heyra um SMLY í fyrsta skipti? Nánari upplýsingar má nálgast **hér**.

Hlið notandans
- Hlaða niður veski (forriti) á tölvu (t.d. spjald eða síma)
- Fá “senda” rafmynt
- “Senda” öðrum rafmynt



Wallet

Balance: 953326.39983137 SMLY
Unconfirmed: 0.00 SMLY

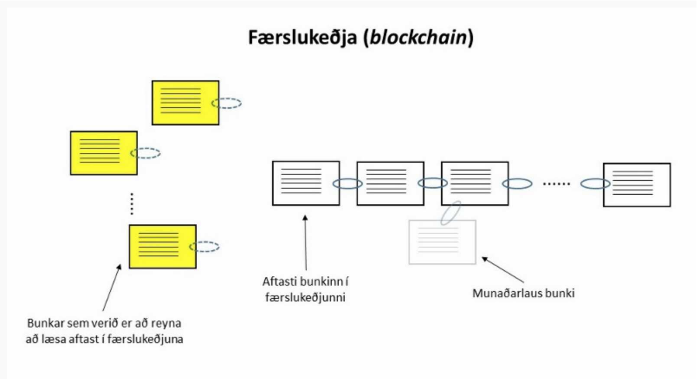
Recent transactions

16/03/2018 15:17 -1002.00 SMLY
(BMCwqTTLC3Jr5FkpRBNKA6cge5V

05/02/2018 18:25 +12500.00 SMLY
(BBkQJt6CECnpbxMz5dr6LKNAoC9/

02/02/2018 19:52 -1002.00 SMLY
(BNgLMJ8wdzLeTULnKx2dXYdzGj5y

Á bakvið töldin
- Færslukeðja/Bunkakeðja (blockchain)
- Færslur
- Grunnhugtakið: UTXO
- Námugröftur
- Satoshi Nakamoto



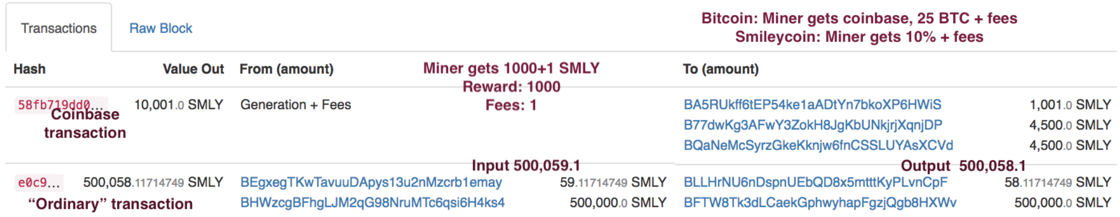
Færslukeðja (*blockchain*)
Bunkar sem verið er að reyna að læsa aftast í færslukeðjuna
Aftasti bunkinn í færslukeðjunni
Munaðarlaus bunki

(Hjálmtýr Hafsteinsson, Vísindavefurinn)

Leikmannaskýring
- Keðjan er eins og færslubók
- Hver blokk er eins og síða í færslubókinni
- Hver færsla er eins og hefðbundin færsla “Jón sendir Gunnu 10 kr”
- Námugrafarinn er bókarinn
- Keðjan er eins og færslubók
- sér um að taka saman færslur
- skráir þær í nýja blokk - síðu í bókinni
- fær umbun fyrir

13.6 sl03050

Færslurnar



Bitcoin: Miner gets coinbase, 25 BTC + fees
Smileycoin: Miner gets 10% + fees

Hash	Value Out	From (amount)		To (amount)	
58fb719dd0...	10,001.0 SMLY	Generation + Fees		BA5RUkff6tEP54ke1aADtYn7bkoXP6HWiS	1,001.0 SMLY
				B77dwKg3AFwY3ZokH8JgKbUNkjrjXqnjDP	4,500.0 SMLY
				BQaNeMcSyrzGkeKknjw6fnCSSLUYAsXCVd	4,500.0 SMLY
e0c9...	500,058.11714749 SMLY	BEgxegTKwTavuuDApys13u2nMzcrb1emay	59.11714749 SMLY	BLLHrNU6nDspnUEbQD8x5mtttKyPLvnCpF	58.11714749 SMLY
		BHWzcgBFhgLJM2qG98NruMTc6qsi6H4ks4	500,000.0 SMLY	BFTW8Tk3dLCaekGphwyhapFgzjQgb8HXWv	500,000.0 SMLY

Coinbase transaction

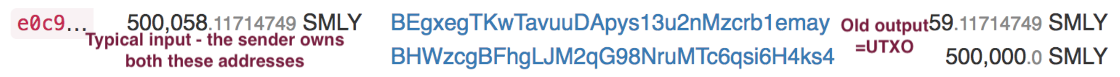
Miner gets 1000+1 SMLY
Reward: 1000
Fees: 1

Input 500,059.1

Output 500,058.1

“Ordinary” transaction

Inntakið

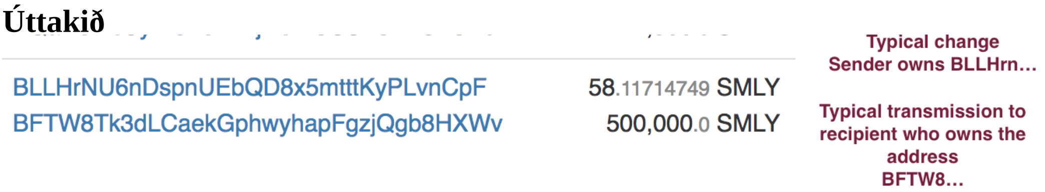


e0c9...	500,058.11714749 SMLY	BEgxegTKwTavuuDApys13u2nMzcrb1emay	59.11714749 SMLY
		BHWzcgBFhgLJM2qG98NruMTc6qsi6H4ks4	500,000.0 SMLY

Typical input - the sender owns both these addresses

Old output =UTXO

Úttakið



BLLHrNU6nDspnUEbQD8x5mtttKyPLvnCpF	58.11714749 SMLY
BFTW8Tk3dLCaekGphwyhapFgzjQgb8HXWv	500,000.0 SMLY

Typical change
Sender owns BLLHrn...

Typical transmission to recipient who owns the address BFTW8...

Leyfið til að eyða UTXO

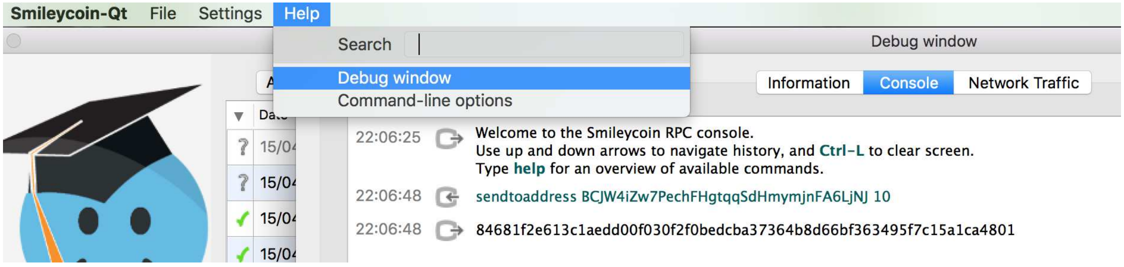
Munum eftir scriptPubKey:

- OP-DUP
- OP-HASH160
- a4d6b6e2e262e97590564a24b523d993765525fb
- OP-EQUALVERIFY
- OP-CHECKSIG

Þegar eyða skal þessu UTXO þarf að bæta framan við forritið stubbi þannig að samsetta forritið skili “TRUE” og engu öðru:

- undirskrift
- dreifilykill

Veskin Veski fyrir Linux, Windows, Mac o.s.frv. leyfa notanda að gefa skipanir, skoða færslur og smíða sérhæfðar færslur.
Veskin geta séð um námugröft.



```
22:06:25  Welcome to the Smileycoin RPC console.
          Use up and down arrows to navigate history, and Ctrl-L to clear screen.
          Type help for an overview of available commands.
22:06:48  sendtoaddress BCJW4iZw7PechFHgtqqSdHmymjnFA6LjNJ 10
22:06:48  84681f2e613c1aedd00f030f2f0bedcba37364b8d66bf363495f7c15a1ca4801
```

13.7 sl03055

x

13.8 sl03060

xx

13.9 sl03070

Myntskoðarar (blockchain explorers)
Hægt að skoða

- Blokkir
- Færslur
- Addressur
- Ríka lista
- o.m.fl.

Dæmi: `http://chainz.cryptoid.info/smly`.
Kauphallir

- Viðskipti með rafmyntir
- Rafmyntir fyrir fiat (og öfugt)

Dæmi: `https://isx.is/`.
Dæmi: `https://tradesatoshi.com/Exchange/?market=SMLY_LTC`.
Dæmigerð kauphöll býður marga markaði með rafmyntir.
Verðmyndun
Í upphafi var Bitcoin verðlaust

- 2 pizzur á 10 000 BTC
- Nú 1 BTC ca 1 M ISK

Í dag: Fleiri notendur að BTC en að ISK?
Grundvallaratriði:
Takmarkað framboð og hefur notagildi => verð > 0

14 The SmileyCoin Fund

14.1 Premining a cryptocurrency

A **premine** is a process where originators of a new coin mine it before the chain is open for general mining

A premine is generally not a good idea

14.1.1 Handout

A **premine** is a process where the originators of a new coin mine it before the chain is open for general mining. This approach has been used for a number of coins, and for different reasons. For example, Auroracoin was premined and the premine was mostly distributed in an **airdrop** to Icelanders. In some cases this is easily justified.

However, a premine is generally not a good idea as it can be used to hide spending and abnormally reward coin developers. Thus, even with the best intentions, coin developers will need to go to extreme lengths to explain why their coin has a premine.

If a premine is to be used, it needs to be implemented as openly as possible.

14.2 The SmileyCoin premine

The SmileyCoin was originally premined

The purpose of the coin was to reward students in the tutor-web system

The premine was mostly used for this (but also for development and grants)

Other methods **could** have been used instead of the premine (see chapter **splitting the coinbase**)

14.2.1 Handout

The SmileyCoin was originally premined: Of the 48 bn SMLY to be mined, 50% were premined and kept for use in the tutor-web.

Planned and actual use of the SmileyCoin premine was discussed in a public forum and the use was subsequently described, also in a public forum, as well as described on Twitter

In spite of openness, a premine will always face considerable criticism.

Better approaches are needed.

14.3 Setting up a cryptocurrency fund: The SmileyCoin Fund

The SmileyCoin premine has been changed to a formal cryptocurrency fund: **The SmileyCoin Fund**

The SmileyCoin Fund has a **Board** which accept applications for funding

The process of spending has moved to be open and transparent

This is explained in more detail in a later section

14.3.1 Handout

The remainder of the SmileyCoin premine was moved to a formal cryptocurrency fund: **The SmileyCoin Fund**.

The SmileyCoin Fund has a **Board** which accept applications for funding. The Board has members from four different organisations, including the Rector's office of the University of Iceland, as described in a public announcement.

The Board has a formal mandate, and announcements of spending are sent out on Twitter.

The entire SmileyCoin Fund is stored in **one multisig address** which corresponds to four private keys. Two of these keys are needed to sign any transfer from the fund. Each Board member holds exactly one of these private keys. All of the corresponding addresses are publicly known and transfers can therefore be verified by anyone with Internet access.

The process of spending has thus moved to be open and transparent.

The details of the methods are given in a later chapter.

15 Splitting the coinbase: No longer just a miner's fee

15.1 Alternatives to premines and funds

A premine can be used to fund development or special projects
A better approach is to set up a **formal fund** for the same purpose
A still better approach is to formally program the mining process to donate to the projects
This uses the **coinbase** for more than just the block reward for the miner

15.1.1 Handout

Recall from the previous section that a premine can be used to fund development or special projects. The premine is what the developers of a coin decide to mine before opening the coin to general mining.

A better approach is to set up a **formal fund** for the same purpose as has been done in the case of the SmileyCoin Fund. In the case of the SmileyCoin, the remainder of the premine was moved to the Fund, but it could have been done at the outset.

A still better approach is to formally program the mining process to donate to the projects. This implies using the **coinbase** for more than just a reward to the miner for finding the block.

15.2 Splitting the coinbase: Why?

The **coinbase** is a prespecified number of coins which the miners can generate when they mine a new block
Usually miners can send the coinbase to an address of their own choosing
In this case the **coinbase** is the same as the (miner's) **block reward**
A community can also decide to do something else with the coinbase
If the miner's reward is too high then a large number of miners will start to mine the coin
If a large pool starts to mine a small coin then the difficulty shoots up until the pool stops mining

15.2.1 Handout

The **coinbase** is a prespecified number of coins which the miners can generate when they mine a new block. For SmileyCoin this is initially set to 10 thousand SMLY per block.

Usually miners can send the coinbase to an address of their own choosing. Thus a miner will normally keep the coinbase and it becomes the (miner's) **block reward**.

But a community can also decide to do something else with the coinbase: If all the wallets, including the miners' wallets, are set to only accept blocks where the coinbase is used for donations, then this use has been hardwired into the coin.

There can be many different reasons for choosing this path.

- If the miner's reward is too high then a large number of miners will start to mine the coin
- If a large pool starts to mine a small coin then the difficulty shoots up until the pool stops mining

15.3 The SmileyCoin coinbase split

- 10% Miner's reward
- 45% Donations
- 40% Dividends

15.3.1 Handout

Since 2017, the SmileyCoin coinbase has been split three ways

- 10% Miner's reward
- 45% Donations
- 40% Dividends

15.4 Effects of the coinbase split

- No large pools
- 1bn SMLY in donations over 1-2 years
- over 250 dividend-seekers

15.4.1 Handout

After the SmileyCoin coinbase was been split three ways, several changes were seen in the behaviour of the SMLY blockchain.

- No large pools
- 1bn SMLY in donations over 1-2 years
- over 250 dividend-seekers

16 Staking and proof-of-stake

16.1 Staking

Staking refers to having a **stake** in a venture
Staking in a cryptocurrency context implies owning some coins in the currency

16.2 Proof of stake

Proof-of-stake (PoS) is an alternative method to Proof-of-Work to maintain a blockchain.
In a PoS network the holders of coins may take turns in generating the next block.
This replaces the competition for mining by a method where only allowing those who have demonstrated a stake to participate.
An obvious advantage is the reduction in mining costs.
An obvious disadvantage is the reduction in competition and possibility of monopoly.
The implementations may vary, ranging from a simple weighted lottery for who gets the next block to setting a minimum stake to enter the pool of miners (or individuals permitted to generate blocks).

17 The tutor-web as a faucet

17.1 Cryptocurrency faucets

Faucets are...
Examples of faucets:
* x * y

18 The command line from a Linux script

18.1 The Linux shell

The bash shell accepts commands such as

- ls
- cd

Output can be redirected into other programs or into a file

- ls | sort
- ls > delete.me

Most commands are just programs. Commands may take command-line options.

- smileycoin-cli getinfo
- smileycoin-cli listunspent

18.2 Startup files

Many programs use a startup file if it exists somewhere

- .smileycoin/smileycoin.conf

The Smileycoin/Bitcoin/Litecoin config files change the behaviour of the wallets
Example:

- walletnotify=/home/user/bin/readIncoming %s

specifies a command to be run every time an incoming transaction is observer
The script `readIncoming` must exist and be executable. It should assume that there is one argument, the transaction id. Commonly this is a `shell script`, which is just a collection of shell commands.

18.3 Betzy

Betzyyy is an example of a Linux script which is called to handle incoming transactions. See
`https://steemit.com/blockchain/@gstefans/double-or-nothing-on-the-blockchain`
– though `BEtZyyYqDXqmRJJ45nnL15cuASfiXg9Yik` is more commonly used as the recipient address

18.3.1 Handout

Check the script `ATMDoubleOrNothing` to see exactly how this works, based on just adding the command

```
walletnotify=/home/gstefans/atm/ATMDoubleOrNothing %s
```

to `smileycoin.conf`. Note that the script is available on github.
Note also that there is a difference between the notification commands

```
walletnotify=/home/.../script1 %s
blocknotify=/home/.../script1 %s
```

18.4 The command script

Upon startup, a typical `readIncoming script` will call the wallet to inspect the incoming transaction:

- txId=$1
- smileycoin-cli gettransaction $txId
- tx=‘smileycoin-cli getrawtransaction $txId‘
- stuff=‘smileycoin-cli decoderawtransaction $tx‘

and then inspect the elements of `stuff` to extract whatever data is needed.

19 Building slightly more complex transactions on the command line

19.1 A simple transaction

Recall how to create a simple transaction, with just one input and one output Use `smileycoin-cli listunspent`to find an UTXO
Suppose this includes the output lines:

```
"txid": "cf808bcc1f38fdaa4930cb0bdf0ad71f970cd253994d4c571ad2fd08d3cd793d",
"vout": 0,
"amount": 13493.00000000,
...
},
{
...
"address": "B69QTo216bcaA3SD2Da7Q9arThMy7Z8ayJ",
...
```

Then the following one-line Linux command will aggregate the two addresses:

```
smileycoin-cli createrawtransaction '[{"txid":"cf808bcc1f38fdaa4930cb0bdf0ad71f970cd2
'{"B69QTo216bcaA3SD2Da7Q9arThMy7Z8ayJ":13492.00000000}'
```

19.1.1 Handout

Create it
“smileycoin-cli createrawtransaction ’["txid":"cf808bcc1f38fdaa4930cb0bdf0ad71f970cd253994d4c571ad2fd0
’"B69QTo216bcaA3SD2Da7Q9arThMy7Z8ayJ":13492.00000000’“‘
“01000000013d79cdd308fdd21a574c4d9953d20c971fd70adf0bcb3049aafd381fcc8b80cf0000000000ffffffff01
Compare with the “listunspent“ output – we did account for the transaction fee.
Sign it
“smileycoin-cli signrawtransaction 01000000013d79cdd308fdd21a574c4d9953d20c971fd70adf0bcb3049aafd
““ “ "hex":“ “"01000000013d79cdd308fdd21a574c4d9953d20c971fd70adf0bcb3049aafd381fcc8b80cf000000
“ "complete": true“ ““
Then just send it:
“smileycoin-cli sendrawtransaction“ “01000000013d79cdd308fdd21a574c4d9953d20c971fd70adf0bcb3049aa
“5cc83b9728ec3eead163ce8640b7d65076ad43534734da47706d008f8db862ee“

19.2 Maintaining a single address

It is often useful to maintain a single main address.
Example: Anonymous user ‘Pine’ used `3P3QsMVK89JBNqZQv5zMAKG8FK3kJM4rjt` as the Bitcoin address for a fund of 5104 Bitcoin, as described at `https://pineapplefund.org/`
A typical transaction is Bitcoin transaction `f065cc0bbede00d3fb56d1dd704fb8e85e706f7d22cee5ec5541` where Pineapplefund transfers 300 BTC to a destination address and sends the entire remainder of the fund back to the original address, `3P3QsMVK89JBNqZQv5zMAKG8FK3kJM4rjt`.
This transaction is seen in `https://www.blockchain.com/btc/tx/f065cc0bbede00d3fb56d1dd704fb8e`
Such transactions are very easy to generate on the command line.
Keeping the entire fund at a single address makes it extremely easy to publicly verify the development of the fund as grants are dispensed to recipients.

19.2.1 Example

Consider the following output from a `listunspent` command

```
    "txid": "faadd4f329cc234a9b22368fe36131252002ff295ab466b9fdf4b2d1eb13d38c",
    "vout": 0,
    "address": "B69QTo216bcaA3SD2Da7Q9arThMy7Z8ayJ",
    "account": ,
    "scriptPubKey": "76a91412987f0ac5ac71d66bd672d6be6f227a0ec9895888ac",
    "amount": 1975796.00000000,
```

One can then send just 1000 SMLY to a destination address and keep the entire rest in the original address using

```
smileycoin-cli createrawtransaction '[{"txid":"faadd4f329cc234a9b22368fe36131252002ff2
'{"BEtZyyYqDXqmRJJ45nnL15cuASfiXg9Yik":1000,"B69QTo216bcaA3SD2Da7Q9arThMy7Z8ayJ":19747
01000000018cd313ebd1b2f4fdb966b45a29ff0220253161e38f36229b4a23cc29f3d4adfa0000000000ff
smileycoin-cli signrawtransaction 01000000018cd313ebd1b2f4fdb966b45a29ff0220253161e38f
{
"hex": "01000000018cd313ebd1b2f4fdb966b45a29ff0220253161e38f36229b4a23cc29f3d4adfa000
"complete": true
}
smileycoin-cli sendrawtransaction 01000000018cd313ebd1b2f4fdb966b45a29ff0220253161e38f
1fa1ebcb69a361b56eeb283fb3adb87c0031ed88ca8b9e539fc3b33fcd225a38
```

The output from the last command was the TxId and as always the transaction can be viewed in any block explorer, e.g. `https://chainz.cryptoid.info/smly/tx.dws?1fa1ebcb69a361b56ee`

19.3 Making a non standard transaction using P2SH

19.3.1 Handout

by
Magnea Haraldsdóttir
The instructions that I followed are:

- Standard transaction: `https://medium.com/@darosior/bitcoin-raw-transactions-the-hard-wa`
- P2SH transaction: `https://medium.com/@darosior/bitcoin-raw-transactions-part-2-p2sh-9`

Using functions from these instructions I modified the code and functions and ended up with a small python program I am using for this.
The functions that I am using are `hash160()` which is the `ripemd160(sha256())` frequently used in Smileycoin just like in Bitcoin
The next function is just a sizeof() which gets the size in bytes of an integer
Then we have the class Script() which represents a Smileycoin script and a function parse() that takes in the opcode names as strings and returns them as the hex value of that opcode.
The last function is serialize() which takes in all elements of a transaction and makes the hex needed for the signrawtransaction in the smileycoin-cli.
The main steps are taken after we have got all of these functions working:

1. We need to get the previous hash which is the txid of the smly that we want to spend, called prev_hash

```
prev_hash = binascii.unhexlify('7b1e1d86d0ce8f614cfad93f1ab592a1973b097aba0fd357aa3acd6
```

The next step is done when wanting to make a standard transaction, described in the first link linked above 2A. We need the public key of the address we want to send to, a new address can be obtained with `getnewaddress` in the smileycoin command line and the public key of that address is found by doing `validateaddress <address>`. After that we get the address not encoded with base58

```
pubkey = b'02e0ec45655eb4f1b7cd76ea116f9cd80c4b1df060c2f2500ff7fd7e87528f8121'
address = hash160(binascii.unhexlify(pubkey))
```

Then we need to make the scriptPubKey:

```
script = 'OP_DUP OP_HASH160 ' + address + ' OP_EQUALVERIFY OP_CHECKSIG
scriptPubKey = Script(script)
scriptPubKey= scriptPubKey.serialized
```

After this we go to step 3
Step 2B is done when wanting to make a non standard transaction using P2SH 2B.
Here we don't need an address because the address will be the script we want to lock the transaction with.
The script here is:

```
script = 'OP_2 OP_ADD OP_4 OP_NUMEQUALVERIFY'
scriptPubKey = Script(script)
```

We then need to get the hash160 of the scriptPubKey as that is how P2SH works, then I construct the lockingscript.

```
scriptPubKeyHash = hash160(scriptPubKey.serialized)
lockscript = 'OP_HASH160 ' + spkhash + ' OP_EQUAL'
lockingScript = Script(lockscript)
lockingScript = lockingScript.serialized
```

3. This step is then the same for a standard transaction and a non standard transaction

```
#The pubkey of the previous output can be found in the vout['scriptpubkey']['hex'] entr
scriptsig = binascii.unhexlify('76a914ccd7179ee7e6fa0039f6c8a279cf1d7cad35741f88ac')
#Amount to send (0.97)
value = int(97000000).to_bytes(8, 'little')
#The vout
index = b'\x00\x00\x00\x00'
```

Then the last step is to print what is needed for signing the transaction:

```
print("to sign:")
print(serialize(prev_hash, index, scriptsig, value, spk2))
```

The example can be seen here:

```
https://chainz.cryptoid.info/smly/tx.dws?322ec02926d74f2baac2beca804b43a2aaf7a32f7e435
```

20 Cryptocurrency exchanges

20.1 Smileycoin exchanges

Smileycoin can be bought and sold at cryptocurrency exchanges, but through time exchanges may close down temporarily or permanently.
Exchanges supporting SMLY include

- southxchange
- tradesatoshi

The steps involved are

- buy Bitcoin for fiat money (e.g.at `https://isx.is/`)
- transfer Bitcoin to one of the above exchanges
- if appropriate, convert Bitcoin to Litecoin or Dogecoin
- convert to SmileyCoin

20.2 The honeypot problem

A cryptocurrency exchange typically holds a large number of coins in multiple wallets
This attracts illegal activity: hackers break in or employees run away with the stash
It is generally not a good idea to store large amounts for a long time on an exchange

20.3 Tracking stolen goods

An example of chasing down thieves:

`https://www.youtube.com/watch?v=BDAiSeRgi6E&list=PLzTQcKBiNWB3E7mh5egXI_PaHW1MLnX`

20.4 An inside job

Sometime exchanges get hacked from the inside:
When CoinLim shut down, wallets were emptied and the admins closed down all communication routes to the outside
This looks very much like an inside job...

`https://twitter.com/SmileycoinNews/status/1329227229180682242?s=20`

21 API access to exchanges

21.1 Automating access to cryptocurrency exchanges

Most cryptocurrency exchanges allow some sort of programmed access.
This is usually called an application programming interface or API and is normally done through a browser-style access (URL or URI).
For the user this implies that it is possible to write programs to monitor prices or even automate buying and selling in different markets.
The programs can in principle be written in any programming language, but language support for HTML varies quite a bit.

22 Automation on the blockchain (stores, ATM, gambling etc)

22.1 Doing stuff on the blockchain

The blockchain can be used for a lot more than transactions:

- Sending data, using the data fields
- Coding data into the amounts
- Monitoring transactions
- Executing code as a response to transactions

22.2 So how do you do stuff?

Method 1: the config file...
option to monitor incoming transactions:
`walletnotify=/home/user/bin/command %s`
code everything into the transactions
Benefits: No changes to the wallets
Method 2: Change the wallet, add wallet commands
Benefits: No changes to the protocol
Method 3: Change the protocol

22.3 Gambling on the blockchain

basics...
`sendtoaddress BCJW4iZw7PechFHgtqqSdHmymjnFA6LjNJ 10`
etc
vanity address: Betzy...
`sendtoaddress BEtZyyYqDXqmRJJ45nnL15cuASfiXg9Yik 10`
... needs to be written up
See `https://steemit.com/blockchain/@gstefans/double-or-nothing-on-the-blockchain`

22.4 Messages on the blockchain

See github thread sendwithmessage for Smileycoin
Can send messages by coding it into the blockchain
`smileycoin-cli sendwithmessage BEtZyyYqDXqmRJJ45nnL15cuASfiXg9Yik`
`1000 "Hello, Betzyy"`
See transaction `daf75d1ae31877b51856b4dee931600a5a5db819f52a5d98627f8a070a72b723`:
`https://chainz.cryptoid.info/smly/search.dws?q=daf75d1ae31877b51856b4dee931600a5a5db8`
Note how the coding uses ASCII characters 32-128 (0x20-0x7F), requiring 2 bytes per character, see `http://www.asciitable.com/` (subtracting 32 dec from each ASCII code)

22.5 A very simple ATM on the blockchain

An Automatic Teller Machine, ATM, dispenses money when you put in a credit card.
An ATM could also dispense Euro when you put in USD.
A blockchain ATM could dispense SMLY when you put in LTC.
For this you need to use two chains, the LTC and SMLY chains.

- To “put in LTC” means to send LTC to an address.
- To “dispense SMLY” means to send SMLY to a SMLY address.

For this to work (1) the ATM needs to have a LTC address and (2) the ATM needs to be told about a SMLY address.
This can be done by sending a SMLY address encoded in a LTC transaction.
For more info see
`https://steemit.com/blockchain/@gstefans/more-messing-around-with-the-blockchain-an-a`

22.6 A more elaborate ATM on the blockchain

for more info see
`https://steemit.com/crypto/@gstefans/on-line-atm-looking-for-testers`

22.7 Traditional data

Normal use is through the op return
usually limited to 80 characters, but see
`https://bitcoin.stackexchange.com/questions/78572/op-return-max-bytes-clarification`
and example below

22.7.1 Examples

Example of the use of the data field in a SmileyCoin transaction:

```
createrawtransaction
  '[{"txid":"b99f638e3a2763f55dba9515382ba6a9f23f6789989660bb893047430c335105",
     "vout":0}]'
  '{"BQLegZNT2hid77hNXgyFK1vZJ1BU2AvcdV":9,
    "data":"31313131"}'
```

The reader should try to generate several such transactions to see exactly what changes in the hex code by including strings of different lengths.

22.8 API access to the blockchain

Some block explorers give API access
This means that arbitrary programs can access data from the blockchain
Example:
`https://blocks.smileyco.in/api/tx/3105284208c04db1675c128089a7a47ae5f829dac64b82e3d8ca`
gives a JSON string for a SMLY transaction

22.8.1 Handout

Block explorers give information about the blockchain. This typically includes information about individual blocks, transactions or addresses.
This means that arbitrary programs, written in almost any programming language, can access data from the blockchain.
The website `https://blocks.smileyco.in` is a block explorer for SMLY and this explorer includes an API.
Some use case examples include:

Get a JSON string for a transaction `https://blocks.smileyco.in/api/tx/3105284208c04db1675c12`

Get a full block `https://blocks.smileyco.in/api/block/eece533e1b264db015d0a3bee5053236a197`

Get the balance behind an address `https://blocks.smileyco.in/api/addr/BS5SyUXeCnJ5ERZEyHLi4`

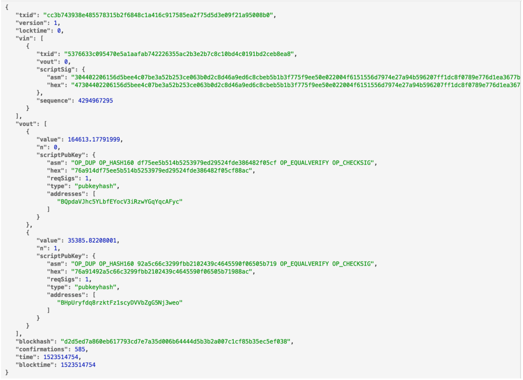
In the Linux shell, curl can be used to extract information and in R, read_json can be used:

> read_json(“`https://blocks.smileyco.in/api/block/eece533e1b264db015d0a3bee5053236a197`

23 The Bitcoin programming language

23.1 From input to output

Tx cc3b743938e485578315b2f6848c1a416c917585ea2f75d5d3e09f21a95008b0 output 0



```
{
   "txid": "cc3b743938e485578315b2f6848c1a416c917585ea2f75d5d3e09f21a95008b0",
   "version": 1,
   "locktime": 0,
   "vin": [
      {
         "txid": "5376633c095470e5a1aafab742226355ac2b3e2b7c8c10bd4c0191bd2ceb8ea8",
         "vout": 0,
         "scriptSig": {
            "asm": "304402206156d5bee4c07be3a52b253ce063b0d2c8d46a9ed6c8cbeb5b1b3f775f9ee50e022004f6151556d7974e27a94b596207ff1dc8f0789e776d1ea3677b
            "hex": "47304402206156d5bee4c07be3a52b253ce063b0d2c8d46a9ed6c8cbeb5b1b3f775f9ee50e022004f6151556d7974e27a94b596207ff1dc8f0789e776d1ea367
         },
         "sequence": 4294967295
      }
   ],
   "vout": [
      {
         "value": 164613.17791999,
         "n": 0,
         "scriptPubKey": {
            "asm": "OP_DUP OP_HASH160 df75ee5b514b5253979ed29524fde386482f05cf OP_EQUALVERIFY OP_CHECKSIG",
            "hex": "76a914df75ee5b514b5253979ed29524fde386482f05cf88ac",
            "reqSigs": 1,
            "type": "pubkeyhash",
            "addresses": [
               "BQpdaVJhc5YLbfEYocV3iRzwYGqYqcAFyc"
            ]
         }
      },
      {
         "value": 35385.82208001,
         "n": 1,
         "scriptPubKey": {
            "asm": "OP_DUP OP_HASH160 92a5c66c3299fbb2102439c4645590f06505b719 OP_EQUALVERIFY OP_CHECKSIG",
            "hex": "76a91492a5c66c3299fbb2102439c4645590f06505b71988ac",
            "reqSigs": 1,
            "type": "pubkeyhash",
            "addresses": [
               "BHpUryfdq8rzktFz1scyDVVbZgG5Nj3weo"
            ]
         }
      }
   ],
   "blockhash": "d2d5ed7a860eb617793cd7e7a35d006b64444d5b3b2a007c1cf85b35ec5ef038",
   "confirmations": 585,
   "time": 1523514754,
   "blocktime": 1523514754
}
```

"asm:" "OP_DUP OP_HASH160 **df75ee5b514b5253979ed29524fde386482f05cf** OP_EQUALVERIFY OP_CHECKSIG",
"hex:" "76a914df75ee5b514b5253979ed29524fde386482f05cf88ac",
i.e.
hex: **76 a9** 14 **df75ee5b514b5253979ed29524fde386482f05cf 88 ac**

23.2 The assembler

where we see (from `https://en.bitcoin.it/wiki/Script`) the meaning of the sequence
`OP_DUP OP_HASH160 OP_EQUALVERY OP_CHECKSIG OP_EQUAL OP_VERIFY`
in the table in the handout.

23.2.1 Handout

code	dec	hex	Input	Output	Description
OP_DUP	118	0x76			Duplicates the top stack item.
OP_HASH160	169	0xa9	in	hash	The input is hashed twice: first with SHA-256 and then with RIPEMD-160.
OP_EQUALVERIFY	136	0x88	x1 x2	Nothing/fail	Same as OP_EQUAL, but runs OP_VERIFY afterward.

code	dec	hex	Input	Output	Description
OP_CHECKSIG	172	0xac	sig pubkey	True / false	The entire transaction outputs, inputs, and script (from the most recently-executed OP_CODESEPARATOR to the end) are hashed. The signature used by OP_CHECKSIG must be a valid signature for this hash and public key. If it is, 1 is returned, 0 otherwise.
OP_EQUAL	135	0x87	x1 x2	True/false	Returns 1 if the inputs are exactly equal, 0 otherwise.
OP_VERIFY	105	0x69	True / false	Nothing/ / fail	Marks transaction as invalid if top stack value is not true. The top stack value is removed.
	20	0x14			push 20 bytes onto the stack the following 160 bit hash

23.3 Simple example

```
2 7 OP_ADD 3 OP_SUB 1 OP_ADD 7 OP_EQUAL
```

23.3.1 Example

Consider how the following code would be executed
`2 7 OP_ADD 3 OP_SUB 1 OP_ADD 7 OP_EQUAL`

2			2 goes on stack
7	2		2 on stack; 7 goes on stack
	7	2	7 and 2 on stack
OP_ADD	7	2	+ operator adds from stack
	9		sum gets put on stack
3	9		3 goes on stack
	3	9	3 and 9 on stack
OP_SUB	3	9	- operator
	6		difference is put on stack
1	6		1 goes on stack
	1	6	1 and 6 on stack
OP_ADD	1	6	+ operator adds from stack

	7		sum gets put on stack
7	7		new 7 to be put on stack
	7	7	
OP_EQUAL	7	7	= operator compares
	TRUE		result is TRUE

23.4 spending

Then the value gets used in
Tx e870614afe3cb9fde97566b024a72f11d22ce08dbd89a971655b15f71d6e203b
(see handout)

23.4.1 Handout

with
asm:

```
30450221009411566e0a7965fc5b8a73ca788a52c4fc0d4f6eaf3089812e55 \
e7def520c79502205d7db54c0851d650d80e9236ecfbe9a94ea0b17976255f \
2170259b652524d0b301
```

hex:

```
4830450221009411566e0a7965fc5b8a73ca788a52c4fc0d4f6eaf3089812e55 \
e7def520c79502205d7db54c0851d650d80e9236ecfbe9a94ea0b17976255f \
2170259b652524d0b3012102df1da8c39a9823016dbe56d11b11b1369b4567 \
4115cbbd9a2a531e365a532bc7‘‘
```

23.5 A more detailed look inside the spending transaction

Looking at SMLY transactions
dee018b9be101c519ad326c581807581a4e46711f242b4878afa1d98c5f521e1
and
08a5430a667a700e7b913d1895908c56d035559cd32a999a5c8cd56f409f4d15
The former’s vout:0 gets spent in the latter.

23.5.1 Example

Consider SMLY transactions dee018b9be101c519ad326c581807581a4e46711f242b4878afa1d98c5f521e1
and 08a5430a667a700e7b913d1895908c56d035559cd32a999a5c8cd56f409f4d15
The former’s vout:0 gets spent in the latter.
The output of dee0. . .

```
"vout": [
   {
      "value": 61,
      "n": 0,
      "scriptPubKey": {
         "asm": "OP_DUP OP_HASH160 52a7c9d832b31c044faa63782ef252e7a4a34291 \
                  OP_EQUALVERIFY OP_CHECKSIG",
         "hex": "76a91452a7c9d832b31c044faa63782ef252e7a4a3429188ac",
         "reqSigs": 1,
         "type": "pubkeyhash",
         "addresses": [
            "BBz82bHDLKC3CGMHhJtNPbwTAYFGrRT1o6"
         ]
      }
   },
```

and the spending part in 08a5…

```
"vin": [
   {
      "txid": "dee018b9be101c519ad326c581807581a4e46711f242b4878afa1d98c5f521e1",
      "vout": 0,
      "scriptSig": {
         "asm": "3044022050587d300d060b912526164f24b1e24897ed9822b1d23fce9a7c103e1c6b3c
                 ac26495c3a3f5c34add4c1aa2fecede1c50ce5f22fd06739920102188c91a675464d2c
                 1bc8cd60fe418d99142ed1ce5f8ce45a997080d5",
         "hex": "473044022050587d300d060b912526164f24b1e24897ed9822b1d23fce9a7c103e1c6b
                 582ac26495c3a3f5c34add4c1aa2fecede1c50ce5f22fd0673992012102188c91a6754
                 1bc8cd60fe418d99142ed1ce5f8ce45a997080d5"
      },
      "sequence": 4294967295
   }
],
```

We want to understand the entire process.
First, to spend the UTXO you need to be the owner of the addres, so verify that:

```
validateaddress BBz82bHDLKC3CGMHhJtNPbwTAYFGrRT1o6
{
"isvalid" : true,
"address" : "BBz82bHDLKC3CGMHhJtNPbwTAYFGrRT1o6",
"ismine" : true,
"isscript" : false,
"pubkey" : "02188c91a675464d2dc475d4f01bc8cd60fe418d99142ed1ce5f8ce45a997080d5",
"iscompressed" : true
}
```

Note how the validateaddress command also shows the public key and this is the second part in the “asm” string in the spending transaction. You can also verify that this address does indeed correspond to this public key: `https://en.bitcoin.it/wiki/Base58Check_encoding#Creating_a`
Next, validate the public key to hash transformation, ripemd160(sha256(publickey)) using python:

```
>>> import hashlib
>>> publickey = '02188c91a675464d2dc475d4f01bc8cd60fe418d99142ed1ce5f8ce45a997080d5' \
                .decode('hex')
>>> s = hashlib.new('sha256',    publickey).digest()
>>> r = hashlib.new('ripemd160', s        ).digest()
>>> s.encode('hex')
'7c8edaf5fe99c4729e8903271fb5f0f34fdf3e461c4db1890b298b3807964505'
>>> r.encode('hex')
'52a7c9d832b31c044faa63782ef252e7a4a34291'
>>>
```

or in R, not so subtle:

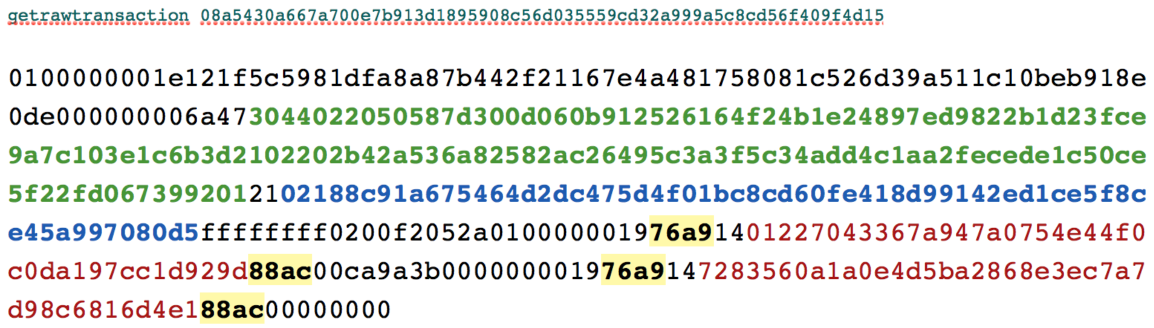
```
> public<-"02188c91a675464d2dc475d4f01bc8cd60fe418d99142ed1ce5f8ce45a997080d5"
> as.character(ripemd160(sha256(as.raw(strtoi(sapply(seq(1, nchar(public), by=2), \
                 function(x) substr(public, x, x+1)), 16L)))))
[1] "52a7c9d832b31c044faa63782ef252e7a4a34291"
```

The final action is to validate the signature, which was the first part of the "asm" string in the spending transaction, i.e.

```
3044022050587d300d060b912526164f24b1e24897ed9822b1d23fce9a7c103e1c6b3d2102202b42a536a8
2582ac26495c3a3f5c34add4c1aa2fecede1c50ce5f22fd067399201
```

This is validated using `OP_CHECKSIG` which is the last operation in the locking script. This opcode takes the two values left on the stack, namely the signature and the public key, and validates the signature. The process is fairly involved but can be found here: `https://en.bitcoin.it/wiki/OP_CHECKSIG`
These various parts of the spending transaction can be reasonably easily found in the hex output from `getrawtransaction` as follows:

```
getrawtransaction 08a5430a667a700e7b913d1895908c56d035559cd32a999a5c8cd56f409f4d15
0100000001e121f5c5981dfa8a87b442f21167e4a481758081c526d39a511c10beb918e0de \
000000006a473044022050587d300d060b912526164f24b1e24897ed9822b1d23fce9a7c10 \
3e1c6b3d2102202b42a536a82582ac26495c3a3f5c34add4c1aa2fecede1c50ce5f22fd067 \
3992012102188c91a675464d2dc475d4f01bc8cd60fe418d99142ed1ce5f8ce45a997080d5 \
ffffffff0200f2052a010000001976a91401227043367a947a0754e44f0c0da197cc1d929d \
88ac00ca9a3b000000001976a9147283560a1a0e4d5ba2868e3ec7a7d98c6816d4e188ac00000000
```



```
getrawtransaction 08a5430a667a700e7b913d1895908c56d035559cd32a999a5c8cd56f409f4d15

0100000001e121f5c5981dfa8a87b442f21167e4a481758081c526d39a511c10beb918e
0de000000006a473044022050587d300d060b912526164f24b1e24897ed9822b1d23fce
9a7c103e1c6b3d2102202b42a536a82582ac26495c3a3f5c34add4c1aa2fecede1c50ce
5f22fd0673992012102188c91a675464d2dc475d4f01bc8cd60fe418d99142ed1ce5f8c
e45a997080d5ffffffff0200f2052a010000001976a91401227043367a947a0754e44f0
c0da197cc1d929d88ac00ca9a3b000000001976a9147283560a1a0e4d5ba2868e3ec7a7
d98c6816d4e188ac00000000
```

The signature part is seen in green and the public key is given in blue.
As this is the spending transaction it will send outputs to new addresses. The corresponding opcodes are highlighted in yellow: 76=OP_DUP, a9=OP_HASH160, 88=OP_EQUALVERIFY and ac=OP_CHECKSIG. All of these can be found at `https://en.bitcoin.it/wiki/Script`.
Right after each 0xa9, one can see 0x14 (=20 decimal) followed by 20 bytes of data, these being the hashed public keys for the new recipients.
Finally, look carefully at the hex code again and you will find the string 00f2052a01 which contains the byte representation of the amount, in **reverse byte order**, so to evaluate the amount, take the reversed hex notation, 0x012a05f200, and convert this to decimal (in R) :

```
0x012a05f200
[1] 5e+09
```

These amounts are given in units of 10^{-8} SMLY (Smile-oshi?), so these are 50 SMLY, as can be seen by looking at `https://chainz.cryptoid.info/smly/tx.dws?08a5430a667a700e7b913d1895`
This can also be computed directly using the algorithm to go from base 16 to base 10:

```
0    1    2       a      0      5      f      2      0      0
```

> ((((((((0*16+1)_16+2)_16+10)_16+0)_16+5)_16+15)_16+2)_16+0)_16+0 [1] 5e+09

The other amount is coded as the string `00ca9a3b00000000` which amounts to `0x3b9aca00` or 10 SMLY.

23.6 A more detailed look at P2SH

A case study in P2SH is given below in the form of using a **multisig address**
Several tutorials are available on this topic
`https://www.soroushjp.com/2014/12/20/bitcoin-multisig-the-hard-way-understanding-raw-`

23.6.1 Handout

A detailed explanation of using a multisig address, from setting up through generating the address to deposits and spending.

23.6.2 Example

The addresses are generated on three Linux computers using the command
`smileycoin-cli getnewaddress`
on each computer separately.
In this case study the following 3 addresses were obtained, belonging to wallets on 3 different computers :

```
BTU58m57Jo61jU3WeWujPj2aZ9LEYLnpYd
BP6AsFWQHPggnXNYTLssykopsx6r3y2Qnh
B66UXukGkPCgasKp9nVTwb93K7XGMzvjTX
```

By running `validateaddress` on each computer, the corresponding public key is also shown.
For example, the `validateaddress` command on computer 1 resulted in the following output: :

```
smileycoind validateaddress BTU58m57Jo61jU3WeWujPj2aZ9LEYLnpYd
{
   "isvalid" : true,
   "address" : "BTU58m57Jo61jU3WeWujPj2aZ9LEYLnpYd",
   "ismine" : true,
   "isscript" : false,
   "pubkey" : "03971fbd962c5b61432efecf07dbf69f93653ea8a14cc104dc71700e7874c63646",
   "iscompressed" : true,
   "account" : ""
}
```

Note that requesting this exact `validateaddress` on another computer will not give the public key, which will be required later. Thus the `validateaddress` command needs to be executed on each of the three computers.
In our example we obtain:

```
Address                            - Public key
BTU58m57Jo61jU3WeWujPj2aZ9LEYLnpYd - 03971fbd962c5b61432efecf07dbf69f93653ea8a14cc104dc
BP6AsFWQHPggnXNYTLssykopsx6r3y2Qnh - 02b416988c8f209b4ccddb96132685882932405641bac69ba3
B66UXukGkPCgasKp9nVTwb93K7XGMzvjTX - 0380304a74398b04af944831a4e51c41cc0c9f29d4b25f6530
```

Normally the process would involve three different individuals as it is usually of importance that the three signatures be independent. Thus no one person should know all private keys nor have access to the three wallets.
At this stage we have all the information to set up a `multisig address` where two of the three signatures are required for spending.
We have two methods for generating the address. The first and simpler is `addmultisigaddress`:

```
smileyCoin/src/smileycoin-cli addmultisigaddress 2 \
'["0380304a74398b04af944831a4e51c41cc0c9f29d4b25f6530976d46db3fee65df",\
"02b416988c8f209b4ccddb96132685882932405641bac69ba3cae38dd21f619983",\
"03971fbd962c5b61432efecf07dbf69f93653ea8a14cc104dc71700e7874c63646"]'
**3KZ98MnX4Uzv7hcQNyA5QfMaGCqLvbF3Sp**
```

The output from the command is the “multisig address” or “script hash”, **3KZ98MnX4Uzv7hcQNyA5QfMaGC**
Note that the backslashes are not a part of the command, which should be all on one line.
Note also that all three public keys are used when generating the address, which is not a traditional address but a hash which can receive payments just like a regular address.
The second method is to use the `createmultisig` command. This is very similar:

```
smileycoin-cli createmultisig 2
'["0380304a74398b04af944831a4e51c41cc0c9f29d4b25f6530976d46db3fee65df",\
  "02b416988c8f209b4ccddb96132685882932405641bac69ba3cae38dd21f619983",\
  "03971fbd962c5b61432efecf07dbf69f93653ea8a14cc104dc71700e7874c63646"]'
```

but provides more output, which will be used later:

```
{
   "address" : "3KZ98MnX4Uzv7hcQNyA5QfMaGCqLvbF3Sp",
   "redeemScript" : "52210380304a74398b04af944831a4e51c41cc0c9f29d4b25f6530976d46db\
                     3fee65df2102b416988c8f209b4ccddb96132685882932405641bac69ba3ca\
                     e38dd21f6199832103971fbd962c5b61432efecf07dbf69f93653ea8a14cc1\
                     04dc71700e7874c6364653ae"
}
```

Note that the `redeemScript` is on one line but is merely printed here on multiple lines (as indicated by the backslash, the continuation symbol).
The redeemScript will be used later, when we will find a way to spend from the address.
Also note that the multisig address is the same as before: If you use the same three public keys and request the same number of signatures, then the same address is generated.
Now, note that the address is valid on all machines, but it is only “mine” on machines which know how it was generated, as seen in this session on another one of the three:

```
smileycoin-cli validateaddress 3KZ98MnX4Uzv7hcQNyA5QfMaGCqLvbF3Sp
{
"isvalid" : true,
"address" : "3KZ98MnX4Uzv7hcQNyA5QfMaGCqLvbF3Sp",
"ismine" : false
}
```

Now we can send to this “address”:: smileyCoin/src/smileycoind sendtoaddress 3KZ98MnX4Uzv7hcQNyA
1000 f05d1c98d761f1b53727436c2b168bcb4f4e17e92779d065a8ca928f17089e60

The transaction Id can be viewed by using `getrawtransaction` followed by `decoderawtransaction`
:

```
smileyCoin/src/smileycoind  decoderawtransaction \
010000000113adb392c276310901597edfdae180cdaa60f71bcbe1148419e27b\
dc6718bf20000000006a47304402204d1ffe1c1d6d03d03912e6ecc495ff1172\
cc61530e17402d436069a6afa33b0a0220036436af1c0d60af4973b07d5e54c8\
ad8830ecb4686354e6cd71a8e766975338012103a75b127dcb90966b99ae8951\
ac83b6fb57752f80138125042d14cafeac8e0cd1ffffffff0200e87648170000\
0017a914c3f4f3243886b8ed77a9d4d464d27be55ae7000b8700ff0f270b0000\
001976a91462745ef11b10e42f05799d1ea6e1289537d52cf388ac00000000
{
   "txid" : "f05d1c98d761f1b53727436c2b168bcb4f4e17e92779d065a8ca928f17089e60",
   "version" : 1,
   "locktime" : 0,
   "vin" : [
       {
           "txid" : "20bf1867dc7be2198414e1cb1bf760aacd80e1dadf7e5901093176c292b3ad13"
           "vout" : 0,
           "scriptSig" : {
               "asm" : "304402204d1ffe1c1d6d03d03912e6ecc495ff1172cc61530e17402d436069
                        afa33b0a0220036436af1c0d60af4973b07d5e54c8ad8830ecb4686354e6cd
                        03a75b127dcb90966b99ae8951ac83b6fb57752f80138125042d14cafeac8e
               "hex" : "47304402204d1ffe1c1d6d03d03912e6ecc495ff1172cc61530e17402d43606
                        a6afa33b0a0220036436af1c0d60af4973b07d5e54c8ad8830ecb4686354e6
                        71a8e766975338012103a75b127dcb90966b99ae8951ac83b6fb57752f8013
                        25042d14cafeac8e0cd1"
           },
           "sequence" : 4294967295
       }
   ],
   "vout" : [
       {
           "value" : 1000.00000000,
           "n" : 0,
           "scriptPubKey" : {
               "asm" : "OP_HASH160 c3f4f3243886b8ed77a9d4d464d27be55ae7000b OP_EQUAL",
               "hex" : "a914c3f4f3243886b8ed77a9d4d464d27be55ae7000b87",
               "reqSigs" : 1,
               "type" : "scripthash",
               "addresses" : [
                    "3KZ98MnX4Uzv7hcQNyA5QfMaGCqLvbF3Sp"
                ]
            }
        },
        {
            "value" : 479.00000000,
            "n" : 1,
            "scriptPubKey" : {
                "asm" : "OP_DUP OP_HASH160 62745ef11b10e42f05799d1ea6e1289537d52cf3 OP_
                "hex" : "76a91462745ef11b10e42f05799d1ea6e1289537d52cf388ac",
                "reqSigs" : 1,
                "type" : "pubkeyhash",
                "addresses" : [
                    "BDRfF71aUNzBWdgw3f7W8Ai1pwF7DVPiC8"
                ]
            }
        }
    ]
}
```

Check the amount in the P2SH address: `https://chainz.cryptoid.info/smly/address.dws?3KZ98MnX4U`
Creating the raw transaction

Recall that the UTXO is:: vout: 0 txid: f05d1c98d761f1b53727436c2b168bcb4f4e17e92779d065a8ca928f17

and this needs to be specified in the transaction.
Now we are all set to try to spend the funds:

```
createrawtransaction ’[{"txid":"f05d1c98d761f1b53727436c2b168bcb4f4e17e92779d065a8ca928
                     ’{"BNVZ3mJ2jadZtEfT8wyw6ttHVuZFos9Vw3":100,"3KZ98MnX4Uzv7hcQNyA5Q

0100000001609e08178f92caa865d07927e9174e4fcb8b162b6c432737b5f161d7981c5df00000000000\
ffffffff0200e40b54020000001976a914c5e9feb90ddcd06bb538ec87ab13643121b5131388ac002375\
ee1400000017a914c3f4f3243886b8ed77a9d4d464d27be55ae7000b8700000000
```

Alternatively we may be on a machine which does not know about the redeemScript and then we need to provide it, aka Gavin Andersen:

```
smileycoin-cli createrawtransaction \
’[{"txid":"f05d1c98d761f1b53727436c2b168bcb4f4e17e92779d065a8ca928f17089e60",\
   "vout":0,\
   "scriptPubKey":"a914c3f4f3243886b8ed77a9d4d464d27be55ae7000b87",\
   "redeemScript":"52210380304a74398b04af944831a4e51c41cc0c9f29d4b\
                   25f6530976d46db3fee65df2102b416988c8f209b4ccddb\
                   961326858829324056 41bac69ba3cae38dd21f619983210\
                   3971fbd962c5b61432efecf07dbf69f93653ea8a14cc104\
                   dc71700e7874c6364653ae"}]’ \
 ’{"BNVZ3mJ2jadZtEfT8wyw6ttHVuZFos9Vw3":100,"3KZ98MnX4Uzv7hcQNyA5QfMaGCqLvbF3Sp":899}’
```

and this gives a fully constructed transaction as before:

```
0100000001609e08178f92caa865d07927e9174e4fcb8b162b6c432737b5f161d\
7981c5df00000000000ffffffff0200e40b54020000001976a914c5e9feb90ddc\
d06bb538ec87ab13643121b5131388ac002375ee1400000017a914c3f4f324388\
6b8ed77a9d4d464d27be55ae7000b8700000000
```

This transaction needs to be signed by at least two of the three address-holders.
User 1:

```
pi@raspberrypi ~ $ smileyCoin/src/smileycoind signrawtransaction
0100000001609e08178f92caa865d07927e9174e4fcb8b162b6c432737b5f161d7981c5df00000000000ff
{
   "hex" : "0100000001609e08178f92caa865d07927e9174e4fcb8b162b6c432737b5f161d7981c5df0
   "complete" : false
}
```

User 2 :

```
signrawtransaction
0100000001609e08178f92caa865d07927e9174e4fcb8b162b6c432737b5f161d7981c5df000000000b500

{
"hex" : "0100000001609e08178f92caa865d07927e9174e4fcb8b162b6c432737b5f161d7981c5df0000
"complete" : true
}
```

Note how the second signature generates a valid transaction (complete=true).
This transaction is seen as
`https://chainz.cryptoid.info/smly/tx.dws?b781767195aec4ecf0f0aa0dca550c5ecd8fd90f6617b`

24 Fun and games with Bitcoin and SmileyCoin

24.1 Puzzles, poetry, bounties etc etc

The Bitcoin blockchain has been used for many things other than transferring funds.
Some examples of puzzles in the blockchain:
`https://en.bitcoin.it/wiki/Script#Transaction_puzzle`
Poetry and other text has also been inserted into the chain
The blockchain also contains bounties: They are automatically paid out to anyone who can solve a specified task. Similar to generic puzzles, a bounty can also serve a purpose. See below for bounties to find sha collisions.

24.2 Sticking data into the blockchain: the data field

The OP_RETURN operator can be used to insert data into a transaction
This is done using the data field of the createrawtransaction command

24.2.1 Examples

Consider the text

:: Betzyy is the double or nothing game BEtZyyYqDXqmRJJ45nnL15cuASfiXg9Yik abcdefgh

which is 80 single-byte characters.
When encoded using 80 hexadecimal ASCII codes, this string becomes

```
4265747a7979206973207468652064f75626c65206f \
72206e6f7468696e672067616d65204245745a797959 \
714458716d524a4a34356e6e4c313563754153666958 \
673959696b206162636465666768
```

(all on one line, no spaces).

```
smileycoin-cli  createrawtransaction
  '[{"txid":"b9cd1f1b74ff1445cc1dd6bfb1540d7d4f06c080538edf9933c5bc1a7ad4073a","vout":
  '{"BEtZyyYqDXqmRJJ45nnL15cuASfiXg9Yik":6,
  "data":"4265747a7979206973207468652064f75626c65206f
          72206e6f7468696e672067616d65204245745a797959
          714458716d524a4a34356e6e4c313563754153666958
          673959696b206162636465666768"}'
```

The resulting hex string can then be signed and broadcast to the network.
In this case, the resulting transaction is

```
c629a1d1e4680d26c44726c672d33022737d637f8c6ca2a441b221c7feba174e
```

which can be seen in a blockchain explorer or viewed using a data-enabled wallet.
:
01000000013a07d47a1abcc53399df8e5380c0064f7d0d54b1bfd61dcc4514ff741b1fcdb90100000000ffffffff020

24.3 Blockchain elections

A cryptocurrency can be used for elections
A voter receives a special *colored* coin as a ballot
The voter sends the coin to a preferred address corresponding to voting for that address
Special attention needs to be given to anonymity

24.3.1 Handout

A mechanism for blockchain-based voting has been set up for SmileyCoin.

For several simple examples, see `http://explore.colorvote.org/`

The central idea is for the voters to receive special coins, **colored coins**, which are marked and only sent to voters. A candidate or survey option is associated with an **address**. Each voter can then send to their address of choice.
The tricky part is not in the voting but in the **voter registration** where anonymity needs to be ensured. This is done by splitting the registration into two main components:

- The voter obtains an Anonymous Id from an **Authentication Server** (AS)
- The voter uses the Anonymous Id to register an address in a **Voting Registry** (VR)

The AS and the VR need to be different entities. In SmileyCoin case studies in 2020 the Authentication Server was based on a Canvas server at the University of Iceland, which provides an official identification to a tutor-web server which randomly **generates and signs** the Anonymous Id. The signature uses a private key in a SmileyCoin wallet owned by the tutor-web. The tutor-web does not store the Id but merely returns it to the user through Canvas.
The signature serves as an anonymous proof of identity and is submitted through a URL to the Voting Registry, which validates the signature and stores the address. A user may submit multiple addresses but only the last one is stored.
Once the addresses have been registered, the **Voting Authority** obtains the address list and sends out a colored coin to each address. Since the address list is anonymized it can be stored on a public web-page.

The software for the process is available at `https://github.com/Ingimarsson/colorvote`

For a description of colored coins, see `https://en.bitcoin.it/wiki/Colored_Coins`

24.4 Bounties: Reporting hash collisions

Taken from `https://en.bitcoin.it/wiki/Script#Incentivized_finding_of_hash_collisions`

24.4.1 Example

In 2013 Peter Todd created scripts that result in true if a hash collision is found. Bitcoin addresses resulting from these scripts can have money sent to them. If someone finds a hash collision they can spend the bitcoins on that address, so this setup acts as an incentive for somebody to do so.
For example the SHA1 script:

- scriptPubKey: `OP_2DUP OP_EQUAL OP_NOT OP_VERIFY OP_SHA1 OP_SWAP OP_SHA1 OP_EQUAL`

- scriptSig: <preimage1> <preimage2>

What this means:

```
OP_2DUP    110 0x6e    x1 x2   x1 x2 x1 x2 Duplicates the top two stack items.
OP_EQUAL   135 0x87    x1 x2   True / false    Returns 1 if the inputs are exactly equa
OP_NOT 145 0x91    in  out If the input is 0 or 1, it is flipped. Otherwise the output
OP_VERIFY  105 0x69    True / false    Nothing / fail  Marks transaction as invalid if
OP_SHA1    167 0xa7    in  hash    The input is hashed using SHA-1.
OP_SWAP    124 0x7c    x1 x2   x2 x1   The top two items on the stack are swapped.
OP_SHA1    167 0xa7    in  hash    The input is hashed using SHA-1.
OP_EQUAL   135 0x87    x1 x2   True / false    Returns 1 if the inputs are exactly equa
```

25 The SmileyCoin Fund revisited

25.1 Background

The SmileyCoin Fund has been briefly explained earlier

25.2 Purpose of the Fund

The SmileyCoin Fund is set up to support education, educational technologies and projects which enhance the use of tutor-web and SmileyCoin

25.3 The Board of the SmileyCoin Fund

A formal Fund needs to have a process to handle applications
The SmileyCoin Fund has 4 Board members, nominated by 4 different organisations

25.3.1 Handout

The SmileyCoin Fund has a **Board** which accept applications for funding.
The Board has members from four different organisations, including the Rector's office of the University of Iceland, as described in a public announcement.

25.4 The Mandate

A formal Mandate has been written and signed by all parties to the Board of the SmileyCoin Fund.
The signed Mandate is publicly available.

25.4.1 Handout

The Board has a formal mandate.

25.5 The multisig address for the Fund

The entire SmileyCoin Fund is stored in
one **multisig** address is available.
A technical document has been written to describe the details of how to operate the address

25.5.1 Handout

The entire SmileyCoin Fund is stored in one **multisig** address

```
3JT9LAzuMChCifVoQQK18BQV9z4BzpbQVH
```

This address can be viewed in a SmileyCoin block explorer .
A technical document has been written to describe the details of how to operate the address.

25.6 Creating, signing and broadcasting a multisig transaction

Some care is needed when sending from a multisig address

25.7 Signing the Mandate electronically

In addition to signing a piece of paper, a corresponding PDF file can be signed electronically.

25.8 Storing the signatures in public

Once a document has been signed, the signatures can be made public
The SMLY blockchain is the obvious place to store SmileyCoin-related signatures

25.8.1 Handout

Step 1: Get the final document in shape, including official signatures and addresses and scan it back in as PDF. This is in file mandate_signed.pdf
Step 2: Get the hash of the file

```
sha256sum < mandate_signed.pdf
09d3b7814390b0badfdf9550d848396f9ee7be202f8c61f6c678d71169ba0f9a  -
```

Step 3: each party signs the message

```
UI:

signmessage BPbwDW2AWsE9KmFDRi1K6QrUdrHvkfbxfn
  09d3b7814390b0badfdf9550d848396f9ee7be202f8c61f6c678d71169ba0f9a
IOYu+G3MZibkoVJigY3VaveGWvFqkbWliiqkp5Q/AYC01u8Rffj3QypV6Pyb6yVLdTdlqIp5+H8y/pm/0dVMTN

EIAS:

signmessage BSZNAqFuQCH3hZTqwmrqv8LDYPJuEYWfyv
  09d3b7814390b0badfdf9550d848396f9ee7be202f8c61f6c678d71169ba0f9a
H5Wr/hJYWTgfZp2fPHAzh5wU7VFuARysMCXekIgOq7rwK9kArEURn9Zy9g430yFC4UyMwamwOVIu1HYPV9nxpq

STL:

signmessage BLE92S2zXshaczZ8GrojAXp8yD54UGRHDk
  09d3b7814390b0badfdf9550d848396f9ee7be202f8c61f6c678d71169ba0f9a
IJ2pW06+guacTtmW6MdzWxcafjviD6MUvRM0Wssfm3Hqtesap6gRFQ6U2VT85/aRs1AvUnTeQRuQQ+e1HhGbxz

AMI:
signmessage BMv1CU9d9ghzB5HdtahWYz9N6NGpFVpSVB
  09d3b7814390b0badfdf9550d848396f9ee7be202f8c61f6c678d71169ba0f9a
IMB3gDqc/al4h9GsaEz7UtypHbCrD7daQ2qIi0s1SJhguYT0J0FVgLk4HrMU2Q6mCdbfVOvUlOt0WGh6cgMw67
```

Step 4: Verify all signatures
UI

```
verifymessage BPbwDW2AWsE9KmFDRi1K6QrUdrHvkfbxfn
  'IOYu+G3MZibkoVJigY3VaveGWvFqkbWliiqkp5Q/AYC01u8Rffj3QypV6Pyb6yVLdTdlqIp5+H8y/pm/0dV
  09d3b7814390b0badfdf9550d848396f9ee7be202f8c61f6c678d71169ba0f9a
```

EIAS

```
verifymessage BSZNAqFuQCH3hZTqwmrqv8LDYPJuEYWfyv
  'H5Wr/hJYWTgfZp2fPHAzh5wU7VFuARysMCXekIgOq7rwK9kArEURn9Zy9g430yFC4UyMwamwOVIu1HYPV9nx
  09d3b7814390b0badfdf9550d848396f9ee7be202f8c61f6c678d71169ba0f9a
```

STL

```
verifymessage BLE92S2zXshaczZ8GrojAXp8yD54UGRHDk
  'IJ2pW06+guacTtmW6MdzWxcafjviD6MUvRM0Wssfm3Hqtesap6gRFQ6U2VT85/aRs1AvUnTeQRuQQ+e1HhG
  09d3b7814390b0badfdf9550d848396f9ee7be202f8c61f6c678d71169ba0f9a
```

AMI

```
verifymessage BMv1CU9d9ghzB5HdtahWYz9N6NGpFVpSVB
  IMB3gDqc/al4h9GsaEz7UtypHbCrD7daQ2qIi0s1SJhguYT0J0FVgLk4HrMU2Q6mCdbfVOvUlOt0WGh6cgMw6
  09d3b7814390b0badfdf9550d848396f9ee7be202f8c61f6c678d71169ba0f9a
```

Step 5: Convert all the signatures to hex

```
python3
```

 >>> import base64
UI

```
base64.decodestring(b'IOYu+G3MZibkoVJigY3VaveGWvFqkbWliiqkp5Q/AYC01u8R
ffj3QypV6Pyb6yVLdTdlqIp5+H8y/pm/0dVMTNk=').hex()
'20e62ef86dcc6626e4a15262818dd56af7865af16a91b5a58a2aa4a7943f0180
b4d6ef117df8f7432a55e8fc9beb254b753765a88a79f87f32fe99bfd1d54c4cd9'
```

EIAS

```
base64.decodestring(b'H5Wr/hJYWTgfZp2fPHAzh5wU7VFuARysMCXekIgOq7rwK9kArE
URn9Zy9g430yFC4UyMwamwOVIu1HYPV9nxpq8=').hex()
'1f95abfe125859381f669d9f3c7033879c14ed516e011cac3025de90880eabbaf02bd
900ac45119fd672f60e37d32142e14c8cc1a9b039522ed4760f57d9f1a6af'
```

STL

```
>>> base64.decodestring(b'IJ2pW06+guacTtmW6MdzWxcafjviD6MUvRM0Wssfm3Hqtesap6gR
   FQ6U2VT85/aRs1AvUnTeQRuQQ+e1HhGbxz4=').hex()
'209da95b4ebe82e69c4ed996e8c7735b171a7e3be20fa314bd13345acb1f9b71eab5eb1aa7a811
150e94d954fce7f691b3502f5274de411b9043e7b51e119bc73e'
```

AMI

```
>>> base64.decodestring(b'IMB3gDqc/al4h9GsaEz7UtypHbCrD7daQ2qIi0s1SJhguYT0J0FVgLk4HrMU
  2Q6mCdbfVOvUlOt0WGh6cgMw67Q=').hex()
 '20c077803a9cfda97887d1ac684cfb52dca91db0ab0fb75a436a888b4b35489860b984f427415580b938
 b314d90ea609d6df54ebd494eb7458687a720330ebb4'
```

Step 6: Stick the signatures into a single block
Next we need to pick any 4 UTXOs and create 4 transactions, which spend those outputs and send the signatures as data in these transactions, one per transaction. EIAS did this by setting up 4 UTXOs to a new address:

```
gstefans@eias_master:~$ smileycoin-cli sendtoaddress B9W6pvnb2WZpPWTA57Z9HZkbe7W3ZNvT1
b867c61387570529e2905ba5c0dfc5912410a4cf9a6d52411e306c0d7128e8e0
gstefans@eias_master:~$ smileycoin-cli sendtoaddress B9W6pvnb2WZpPWTA57Z9HZkbe7W3ZNvT1
2c2fdea0a9c6ecde1680c407dd9c6a821ffc60989dcfb245ce51276403a21f20
gstefans@eias_master:~$ smileycoin-cli sendtoaddress B9W6pvnb2WZpPWTA57Z9HZkbe7W3ZNvT1
8235efa0a79b9fe42de292f51bf3860a25c719e5a7e5119531c8c5a7bdbc97cf
gstefans@eias_master:~$ smileycoin-cli sendtoaddress B9W6pvnb2WZpPWTA57Z9HZkbe7W3ZNvT1
cc15dacac264763ac681cb08ef8e50a53a8e202e638e11c93a3e4f2e4065fac1
```

Check:

```
smileycoin-cli decoderawtransaction ‘smileycoin-cli getrawtransaction
b867c61387570529e2905ba5c0dfc5912410a4cf9a6d52411e306c0d7128e8e0‘
smileycoin-cli decoderawtransaction ‘smileycoin-cli getrawtransaction
2c2fdea0a9c6ecde1680c407dd9c6a821ffc60989dcfb245ce51276403a21f20‘
smileycoin-cli decoderawtransaction ‘smileycoin-cli getrawtransaction
8235efa0a79b9fe42de292f51bf3860a25c719e5a7e5119531c8c5a7bdbc97cf‘
smileycoin-cli decoderawtransaction ‘smileycoin-cli getrawtransaction
cc15dacac264763ac681cb08ef8e50a53a8e202e638e11c93a3e4f2e4065fac1‘
```

It is seen that each has a vout=0 of 11 SMLY to B9W6pvnb2WZpPWTA57Z9HZkbe7W3ZNvT1G.
Next we set up the actual commands. Note that in each case we only transmit 10 SMLY to the destination, leaving 1 SMLY for the transaction fee. We will simply use Betzyy, the EIAS donation address, for the destination.
So the 4 signatures:
:

```
20e62ef86dcc6626e4a15262818dd56af7865af16a91b5a58a2aa4a7943f0180b4d6ef117df8f7432a55e8

1f95abfe125859381f669d9f3c7033879c14ed516e011cac3025de90880eabbaf02bd900ac45119fd672f6

209da95b4ebe82e69c4ed996e8c7735b171a7e3be20fa314bd13345acb1f9b71eab5eb1aa7a811150e94d9

20c077803a9cfda97887d1ac684cfb52dca91db0ab0fb75a436a888b4b35489860b984f427415580b9381e
```

need to be paired with the 4 UTXOs

```
b867c61387570529e2905ba5c0dfc5912410a4cf9a6d52411e306c0d7128e8e0
2c2fdea0a9c6ecde1680c407dd9c6a821ffc60989dcfb245ce51276403a21f20
8235efa0a79b9fe42de292f51bf3860a25c719e5a7e5119531c8c5a7bdbc97cf
cc15dacac264763ac681cb08ef8e50a53a8e202e638e11c93a3e4f2e4065fac1
```

in transactions, using the createrawtransaction command, i.e. using

```
smileycoin-cli createrawtransaction "[{\"txid\":\"TTTTT\",\"vout\":0}]" "
```

{"BEtZyyYqDXqmRJJ45nnL15cuASfiXg9Yik":10,"data":"SSSSS"}"

where TTTTT is the transaction Id for the UTXO and SSSSS is the hex representation of the signature.

```
smileycoin-cli createrawtransaction
 "[{\"txid\":\"b867c61387570529e2905ba5c0dfc5912410a4cf9a6d52411e306c0d7128e8e0\",
 \"vout\":0}]"
 "{\"BEtZyyYqDXqmRJJ45nnL15cuASfiXg9Yik\":10,
\"data\":\"20e62ef86dcc6626e4a15262818dd56af7865af16a91b5a58a2aa4a7943f01
80b4d6ef117df8f7432a55e8fc9beb254b753765a88a79f87f32fe99bfd1d54c4cd9\"}"

smileycoin-cli createrawtransaction
"[{\"txid\":\"2c2fdea0a9c6ecde1680c407dd9c6a821ffc60989dcfb245ce51276403a21f20\",
\"vout\":0}]"
"{\"BEtZyyYqDXqmRJJ45nnL15cuASfiXg9Yik\":10,
\"data\":\"1f95abfe125859381f669d9f3c7033879c14ed516e011cac3025de90880eabbaf02bd900a
  c45119fd672f60e37d32142e14c8cc1a9b039522ed4760f57d9f1a6af\"}"

smileycoin-cli createrawtransaction
 "[{\"txid\":\"8235efa0a79b9fe42de292f51bf3860a25c719e5a7e5119531c8c5a7bdbc97cf\",
 \"vout\":0}]" "{\"BEtZyyYqDXqmRJJ45nnL15cuASfiXg9Yik\":10,
   \"data\":\"209da95b4ebe82e69c4ed996e8c7735b171a7e3be20fa314bd13345acb1f9b
   71eab5eb1aa7a811150e94d954fce7f691b3502f5274de411b9043e7b51e119bc73e\"}"

smileycoin-cli createrawtransaction
  "[{\"txid\":\"cc15dacac264763ac681cb08ef8e50a53a8e202e638e11c93a3e4f2e4065fac1\",
     \"vout\":0}]" "{\"BEtZyyYqDXqmRJJ45nnL15cuASfiXg9Yik\":10,
     \"data\":\"20c077803a9cfda97887d1ac684cfb52dca91db0ab0fb75a436a888b4b35489860b
      984f427415580b9381eb314d90ea609d6df54ebd494eb7458687a720330ebb4\"}"
```

Finally each is signed

```
smileycoin-cli signrawtransaction
```

0100000001e0e828710d6c301e41526d9acfa4102491c5dfc0a55b90e22905578713c667b80000000000fff

smileycoin-cli signrawtransaction

0100000001201fa203642751ce45b2cf9d9860fc1f826a9cdd07c48016deecc6a9a0de2f2c0000000000fffff

smileycoin-cli signrawtransaction

0100000001cf97bcbda7c5c8319511e5a7e519c7250a86f31bf592e22de49f9ba7a0ef35820000000000fff
smileycoin-cli signrawtransaction

0100000001c1fa65402e4f3e3ac9118e632e208e3aa5508eef08cb81c63a7664c2cada15cc0000000000fffff

and broadcast using sendrawtransaction
These transactions were mined into block 538357.

25.9 Validating data from the blockchain

To check data stored in the blockchain, the reverse of the insertion sequence needs to be used.

25.9.1 Handout

It is not enough to be able to store data on the blockchain, as it needs to be possible to verify the data and check any claims made.
First, look at the block. This is block 538357 and can be fetched directly using getblockhash followed by getblock, or viewed using a blockchain explorer:

```
https://chainz.cryptoid.info/smly/search.dws?q=538357
```

Pick one of these transactions, say a40d1b13ffb741d64e6630e0726ef82397d0930f93d639fa3126a519ea49b4af to find the data which goes with the OP_RETURN operator:

```
6a4120e62ef86dcc6626e4a15262818dd56af7865af16a91b5a
58a2aa4a7943f0180b4d6ef117df8f7432a55e8fc9beb254b75
3765a88a79f87f32fe99bfd1d54c4cd9
```

(all on one line with no spaces).
As with other hex codes, each character is codes as a two-digit hexadecimal number. The first two are 6a and 41 and the rest is the actual data string. The 6a is the code for OP_RETURN (see this link).
The actual hex coding is therefore 130 hexadecimal numbers or 65 bytes and since 65 in decimal is 41 hex (or 0x41), this explains the second code, 41, being the length of the data field.
To decode the string, a few python3 commands are useful:

```
codecs.encode(codecs.decode('20c077803a9cfda97887d1ac684cfb52dca91db0ab
0fb75a436a888b4b35489860b984f427415580b9381eb314d90ea609d6df54ebd494eb7458687a720330eb
```

‘base64’).decode() ‘IMB3gDqc/al4h9GsaEz7UtypHbCrD7daQ2qIi0s1SJhguYT0J0FVgLk4HrMU2Q6m

The two newline characters need to be removed from this string, resulting in a signature of

```
'IMB3gDqc/al4h9GsaEz7UtypHbCrD7daQ2qIi0s1SJhguYT0J0FVgLk4HrMU2Q6mCdbfV0vUlOt0WGh6cgMw6
```

Recall that the original hash to be signed was

```
09d3b7814390b0badfdf9550d848396f9ee7be202f8c61f6c678d71169ba0f9a
```

and it was signed by 4 addresses:

```
BPbwDW2AWsE9KmFDRi1K6QrUdrHvkfbxfn
BSZNAqFuQCH3hZTqwmrqv8LDYPJuEYWfyv
BLE92S2zXshaczZ8GrojAXp8yD54UGRHDk
BMv1CU9d9ghzB5HdtahWYz9N6NGpFVpSVB
```

One can now verify this signature using the appropriate address:

```
smileycoin-cli verifymessage BMv1CU9d9ghzB5HdtahWYz9N6NGpFVpSVB
 'IMB3gDqc/al4h9GsaEz7UtypHbCrD7daQ2qIi0s1SJhguYT0J0FVgLk4HrMU2Q6mCdbfV0vUlOt0WGh6cgMw
 '09d3b7814390b0badfdf9550d848396f9ee7be202f8c61f6c678d71169ba0f9a'
```

which returns true as it should.

25.10 Open accounting on the blockchain

Transparency is one part of good governance
It is important for formal funds to demonstrate where grant allocations go
This can be done using a cryptocurrency and announcing exactly what is being done with the funds

25.10.1 Handout

Announcements of spending are sent out on Twitter.

26 Atomic swaps

26.1 Background

- There is a considerable demand for exchanging coins
- This is mostly done on cryptocurrency exchanges
- An exchange is a **honeypot** and hacks are common
- Some exchanges are now **decentralised**

In a truly decentralised exchange the exchange should not hold any user funds: The transaction should be solely between users
The atomic swap is an important concept
Atomic swaps need timeout mechanisms to replace trust

26.2 timeouts

A timeout on a transaction implies that it can not be transmitted before the time limit A timeout on a UTXO implies that it can not be spent before the time limit
CLTV is OP_CHECKLOCKTIMEVERIFY
See Handout and Example for more detail

26.2.1 Examples

Example of use (from here)
Using OP_CHECKLOCKTIMEVERIFY it is possible to make funds provably unspendable until a certain point in the future.

```
scriptPubKey: <expiry time> OP_CHECKLOCKTIMEVERIFY OP_DROP
              OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY
              OP_CHECKSIG

scriptSig: <sig> <pubKey>
```

26.2.2 Handout

There are several timeout features in the Bitcoin protocol, implemented as operators in Bitcoin's scrypt language.
Here we will consider OP_CHECKLOCKTIMEVERIFY, implemented as BIP: 65.
This description of OP_CHECKLOCKTIMEVERIFY, or CLTV is taken mostly from the Bitcoin wiki and the initial CLTV proposal as an improvement to the Bitcoin protocol BIP: 65.
CLTV marks a transaction as invalid if the top stack item is greater than the transaction's nLockTime field, otherwise script evaluation continues as though an OP_NOP was executed. Transaction is also invalid if

1. the stack is empty; or
2. the top stack item is negative; or
3. the top stack item is greater than or equal to 500000000 while the transaction's nLockTime field is less than 500000000, or vice versa; or
4. the input's nSequence field is equal to 0xffffffff. The precise semantics are described in BIP 0065.

26.3 an atomic swap algorithm

by TierNolan
(see Handout)

26.3.1 Handout

more detail to come…

```
A picks a random number x
A creates TX1: "Pay w BTC to <B's public key> if (x for H(x) known and signed by B) or
A creates TX2: "Pay w BTC from TX1 to <A's public key>, locked 48 hours in the future,
A sends TX2 to B
B signs TX2 and returns to A

1) A submits TX1 to the network

B creates TX3: "Pay v alt-coins to <A-public-key> if (x for H(x) known and signed by A
B creates TX4: "Pay v alt-coins from TX3 to <B's public key>, locked 24 hours in the fu
B sends TX4 to A
A signs TX4 and sends back to B

2) B submits TX3 to the network
3) A spends TX3 giving x
4) B spends TX1 using x
```

This is atomic (with timeout). If the process is halted, it can be reversed no matter when it is stopped.

```
Before 1: Nothing public has been broadcast, so nothing happens
Between 1 and 2: A can use refund transaction after 48 hours to get his money back
Between 2 and 3: B can get refund after 24 hours. A has 24 more hours to get his refun
After 3: Transaction can be completed by each of the 2 parties

- A must spend his new coin within 24 hours or B can claim the refund and keep his coi
- B must spend his new coin within 48 hours or A can claim the refund and keep his coi
```

For safety, both should complete the process with lots of time until the deadlines.

26.4 Alternatives

Several decentralised exchanges (DEXs) exist, but the definition of a DEX is not clear
Examples:
Barterdex: https://komodoplatform.com/decentralized-exchange/
Bit Square (bisq): https://bisq.network/
etc
Further reading on atomic swaps etc:
Vitalin Buterik: https://static1.squarespace.com/static/55f73743e4b051cfcc0b02cf/t/5886800
Kyle Samani: https://www.coindesk.com/opportunity-interoperable-chains-chains/
Adrian Mathieu/Viacoin: https://ethereumworldnews.com/viacoin-developers-successfully-com

26.5 The missing link: Information flow

Recall the process:

- A creates TX1: “Pay w BTC to <B’s public key> if (x for H(x) known and signed by B) or (signed by A and B)”
- B creates TX3: “Pay v alt-coins to <A-public-key> if (x for H(x) known and signed by A) or (signed by A and B)”

So **before any exchange is set up**,

- A needs to know that B wants to buy w BTC
- B needs to know that A will sell for v alt-coins

Then, to be able to **start the exchange**

- A needs to know B’s BTC public key
- B needs to know A’s alt-coin public key

This information exchange needs to be done outside the transactions, as an MoU or “announcement(s) of intent”. The info exchange does NOT need to be binding! The info exchange should cost something to avoid spam.
During the exchange the parties need to communicate:

- A sends TX2 to B
- B signs TX2 and returns to A
- B sends TX4 to A
- A signs TX4 and sends back to B

The entire process needs to be without trust and without knowing who the other party is

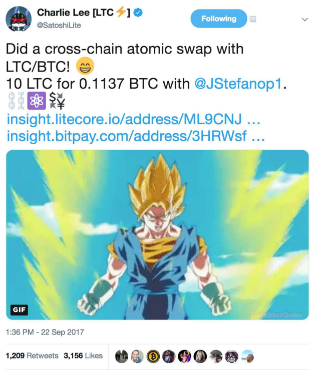
26.6 Announcing the atomic swap

- Use a forum (telegram etc)?
- Use a specialised channel (BarterDex/Bisq)?
- Use an existing coin (mempool)?
- Alice should in principle be able to use the Smileycoin blockchain to announce
  - `SELL 1000 SMLY for 1 LTC`
- and Bob could accept the offer by responding
  - `ACCEPT offer TxId‘`
- etc.

Could be done through modifications of smileycoin-qt
A draft proposal: `https://tutor-web.info/news-1/announcing-intent-cross-chain-atomic-swap`
Dedicated wallets? `https://atomicwallet.io/` (or scam?)

26.7 Atomic swaps between chains: Litecoin and Bitcoin

One of the first ones: https://twitter.com/SatoshiLite/status/911328252928643072



Charlie Lee [LTC ⚡] @SatoshiLite

Did a cross-chain atomic swap with LTC/BTC! 😁
10 LTC for 0.1137 BTC with @JStefanop1.
⛓⚛💱
insight.litecore.io/address/ML9CNJ ...
insight.bitpay.com/address/3HRWsf ...

1:36 PM - 22 Sep 2017

1,209 Retweets 3,156 Likes

10 LTC for 0.1137 BTC
The Litecoin side: https://insight.litecore.io/address/ML9CNJBtSPMABYcCQV58P2t4M7MpPRJK95
The Bitcoin side: https://insight.bitpay.com/address/3HRWsfjpBHiJ7hC3jKJV5nbHMeBgoCPHDq
See also https://github.com/topics/atomic-swap for many, many atomic swap projects.

26.7.1 Handout

Bitcoin 0bb5a53a9c7e84e2c45d6a46a7b72afc2feffb8826b9aeb3848699c6fd856480
(note the locktime)
(takes the one below - 92d9c9. . . - as input)

```
{
  "txid": "0bb5a53a9c7e84e2c45d6a46a7b72afc2feffb8826b9aeb3848699c6fd856480",
  "hash": "0bb5a53a9c7e84e2c45d6a46a7b72afc2feffb8826b9aeb3848699c6fd856480",
  "version": 2,
  "size": 308,
  "vsize": 308,
  "weight": 1232,
  "locktime": 1506182939,
  "vin": [
    {
      "txid": "92d9c9d5d52c618b32484032a22f16dc084841ed29ec1b01c0119425a4e76d24",
      "vout": 1,
      "scriptSig": {
        "asm": "30440220748121e83bee8287a2506ca65256f5bf6b30c5d6948aa334a06c3dd70472a50
        "hex": "4730440220748121e83bee8287a2506ca65256f5bf6b30c5d6948aa334a06c3dd70472a
      },
      "sequence": 4294967295
    }
  ],
  "vout": [
    {
      "value": 0.13336680,
      "n": 0,
      "scriptPubKey": {
        "asm": "OP_DUP OP_HASH160 5d8023cd65e3685726c5df8479206937b64264b9 OP_EQUALVERI
        "hex": "76a9145d8023cd65e3685726c5df8479206937b64264b988ac",
        "reqSigs": 1,
        "type": "pubkeyhash",
        "addresses": [
          "19XPM9tgB2Avj2nF1S5JSM9zJM6oGyH41w"
        ]
      }
    }
  ]
}
```

Bitcoin 92d9c9d5d52c618b32484032a22f16dc084841ed29ec1b01c0119425a4e76d24
(forms input to the one above)

```
{
  "txid": "92d9c9d5d52c618b32484032a22f16dc084841ed29ec1b01c0119425a4e76d24",
  "hash": "92d9c9d5d52c618b32484032a22f16dc084841ed29ec1b01c0119425a4e76d24",
  "version": 2,
  "size": 224,
  "vsize": 224,
  "weight": 896,
  "locktime": 0,
  "vin": [
    {
      "txid": "82ae3ad6c30957a022ef5648ee6bd1883793f34adb1273532fecb842b90d07d7",
      "vout": 1,
      "scriptSig": {
        "asm": "3045022100eabcf4e2d7ca45b54b951624e38caaed7c18598e5091ff3ebf2015db6ace
        "hex": "483045022100eabcf4e2d7ca45b54b951624e38caaed7c18598e5091ff3ebf2015db6a
      },
      "sequence": 4294967294
    }
  ],
  "vout": [
    {
      "value": 0.01629776,
      "n": 0,
      "scriptPubKey": {
        "asm": "OP_DUP OP_HASH160 462f954c6ae2bcf54107191b42d22419f928995e OP_EQUALVER
        "hex": "76a914462f954c6ae2bcf54107191b42d22419f928995e88ac",
        "reqSigs": 1,
        "type": "pubkeyhash",
        "addresses": [
          "17Q7JZsAn4iKotrjpfk7H5WzLnznRVyWSU"
        ]
      }
    },
    {
      "value": 0.13370000,
      "n": 1,
      "scriptPubKey": {
        "asm": "OP_HASH160 ac938614bf4288b3e41385d49fc0531d847551ff OP_EQUAL",
        "hex": "a914ac938614bf4288b3e41385d49fc0531d847551ff87",
        "reqSigs": 1,
        "type": "scripthash",
        "addresses": [
          "3HRWsfjpBHiJ7hC3jKJV5nbHMeBgoCPHDq"
        ]
      }
    }
  ]
}
```

Litecoin 6c497ae07505f6237a810deb4fb366b9d73a2293ce8d8fba21e6203bf93854d2
(note the locktime)
(takes the one below - 75d0ab… - as input)

```
{
  "txid": "6c497ae07505f6237a810deb4fb366b9d73a2293ce8d8fba21e6203bf93854d2",
  "hash": "6c497ae07505f6237a810deb4fb366b9d73a2293ce8d8fba21e6203bf93854d2",
  "size": 308,
  "vsize": 308,
  "version": 2,
  "locktime": 1506204007,
  "vin": [
    {
      "txid": "75d0ab5f6a9da8633c8da91b791a28641c71234ea1bcfbb30ee8eb7f07b70721",
      "vout": 1,
      "scriptSig": {
        "asm": "304402203faac90d00b21bce1079b402a201692e2bc0ebbc22bfb9476dd4a73e8969b2
        "hex": "47304402203faac90d00b21bce1079b402a201692e2bc0ebbc22bfb9476dd4a73e8969
      },
      "sequence": 4294967295
    }
  ],
  "vout": [
    {
      "value": 9.99968600,
      "n": 0,
      "scriptPubKey": {
        "asm": "OP_DUP OP_HASH160 ee3c065dab61a1ed0020eb1c456226600dc44af3 OP_EQUALVER
        "hex": "76a914ee3c065dab61a1ed0020eb1c456226600dc44af388ac",
        "reqSigs": 1,
        "type": "pubkeyhash",
        "addresses": [
          "LgwczUBhCr6XfEWaG4JA22Gi7fW5N38vM1"
        ]
      }
    }
  ]
}
```

Litecoin 75d0ab5f6a9da8633c8da91b791a28641c71234ea1bcfbb30ee8eb7f07b70721

```
{
  "txid": "75d0ab5f6a9da8633c8da91b791a28641c71234ea1bcfbb30ee8eb7f07b70721",
  "hash": "75d0ab5f6a9da8633c8da91b791a28641c71234ea1bcfbb30ee8eb7f07b70721",
  "size": 223,
  "vsize": 223,
  "version": 2,
  "locktime": 0,
  "vin": [
    {
      "txid": "d06f0729fda1564b77480bd38d2a0524b82ae8930a1dec554a26ff82ba146e80",
      "vout": 0,
      "scriptSig": {
        "asm": "304402207325eba06b5a18fb9edadb2c646ee50cffe8062dd64024488419665bf080bd9
        "hex": "47304402207325eba06b5a18fb9edadb2c646ee50cffe8062dd64024488419665bf080b
      },
      "sequence": 4294967294
    }
  ],
  "vout": [
    {
      "value": 0.89955000,
      "n": 0,
      "scriptPubKey": {
        "asm": "OP_DUP OP_HASH160 8b97fa16960b86f69db5d16da02147642aa91494 OP_EQUALVERI
        "hex": "76a9148b97fa16960b86f69db5d16da02147642aa9149488ac",
        "reqSigs": 1,
        "type": "pubkeyhash",
        "addresses": [
          "LXx4FRCeEbZyRB5BNkVR9iNP9oUJzMmNSz"
        ]
      }
    },
    {
      "value": 10.00000000,
      "n": 1,
      "scriptPubKey": {
        "asm": "OP_HASH160 86491d98a99146ab22a066e0d8e6f3a403071af8 OP_EQUAL",
        "hex": "a91486491d98a99146ab22a066e0d8e6f3a403071af887",
        "reqSigs": 1,
        "type": "scripthash",
        "addresses": [
          "ML9CNJBtSPMABYcCQV58P2t4M7MpPRJK95"
        ]
      }
    }
  ]
}
```

27 More on atomic swaps and smart contracts

27.1 The smart contract

Back to Nick Szabo

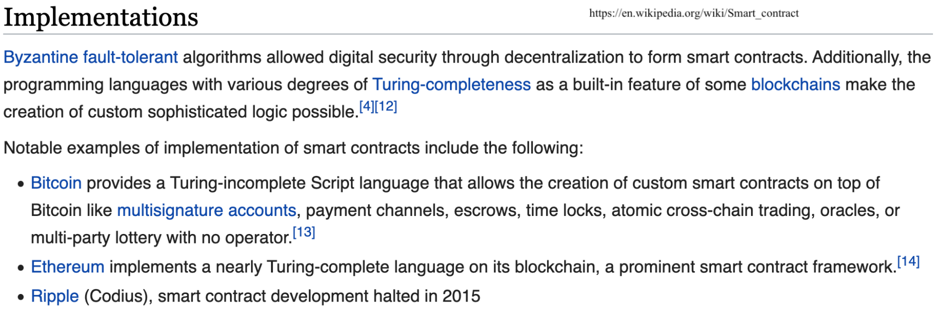
(**Copyright (c) 1994 by Nick Szabo**)

“A smart contract is a computerized transaction protocol that executes the terms of a contract.”

With objectives:

“The general objectives of smart contract design are to satisfy common contractual conditions (such as payment terms, liens, confidentiality, and even enforcement), minimize exceptions both malicious and accidental, and minimize the need for trusted intermediaries. Related economic goals include lowering fraud loss, arbitration and enforcement costs, and other transaction costs[1].”

and from Wikipedia:



Implementations

https://en.wikipedia.org/wiki/Smart_contract

Byzantine fault-tolerant algorithms allowed digital security through decentralization to form smart contracts. Additionally, the programming languages with various degrees of Turing-completeness as a built-in feature of some blockchains make the creation of custom sophisticated logic possible.[4][12]

Notable examples of implementation of smart contracts include the following:

- Bitcoin provides a Turing-incomplete Script language that allows the creation of custom smart contracts on top of Bitcoin like multisignature accounts, payment channels, escrows, time locks, atomic cross-chain trading, oracles, or multi-party lottery with no operator.[13]
- Ethereum implements a nearly Turing-complete language on its blockchain, a prominent smart contract framework.[14]
- Ripple (Codius), smart contract development halted in 2015

27.2 Smart contracts: Misunderstandings

- Example of incorrect statement (more than one error here):
  - ethereum replaces bitcoin’s more restrictive language (a scripting language of a hundred or so scripts) and replaces it with a language that allows developers to write their own programs – `https://www.coindesk.com/information/ethereum-smart-contracts-work/`
- Note that
  - The Bitcoin scripting language **is limited** but so are all programming languages.
  - Developers can write their own programs in the Bitcoin scripting language!!
  - A more flexible language gives more flexibility :-)
  - A more flexible language is often more error-prone and less secure
  - There is **no limit** to the **number of scripts** one can write in the Bitcoin scripting language!!

27.3 Tools for atomic swaps

Examples of tools and discussions

- Very good description with tool-box, Decred: `https://blog.decred.org/2017/09/20/On-Chain-Atomic-Swaps/`

“These tools were built for those who ... have ... transaction script and OP_CLTV support”

- Detailed example based on the Decred tools: `https://hackernoon.com/so-how-do-i-really-do-an-atomic-swap-f797852c7639`

And recall that **“these tools do not address the issue of order book management”**

- for which you need Lightning or other tool for announcements of intent etc

27.4 Which coins are ready?

Nice overview: `https://swapready.net/`

27.5 Lightning

See `https://www.forbes.com/sites/ktorpey/2018/03/15/bitcoins-highly-anticipated-lightn`