

Computer Security: Security at Work

B. Jacobs and J. Daemen
Institute for Computing and Information Sciences – Digital Security
Radboud University Nijmegen
Version: fall 2017



Outline

Bitcoins
The ledger

Conclusions
Final remarks



Security issues for financial transactions

- ▶ **Confidentiality** Who should know about your transactions: the receiver, the bank, the authorities?
- ▶ **Integrity**
 - The intended transaction amount and receiver should be the actual amount and receiver
 - You should not be able to create money yourself
- ▶ **Availability** The transaction should be carried out when intended
- ▶ **Authenticity** Only the owner of the amount can transfer it
- ▶ **Non-repudiation** You cannot deny your transactions later on



Electronic money (also known as: e-cash)

Especially for e-cash there are **money-creation** challenges:

- ▶ **minting**: creation of fresh e-coins, out of nothing
- ▶ **double-spending**: using existing e-coins multiple times in different transactions




Reasonable starting point

- ▶ Alice goes to her bank, orders 10€, and gets a unique serial number N in return
- ▶ She then transfers these 10€ to Bob via the signed message:
$$\left[I, \text{ Alice, transfer } 10\text{€ with serial number } N \text{ to Bob} \right]_{d_{\text{Alice}}}$$
- ▶ Bob can check via the bank if the number N has already been “spent”
 - hence the bank can track all e-cash transactions
 - this approach requires a centralised trusted third party (TTP)

Can we do this peer-to-peer, without the bank?

Enter Bitcoin

- ▶ What is ? **Decentralised cryptocurrency!**
 - Bitcoin is the most widely used among such currencies
 - it uses cryptography to secure transactions and control the creation of money
- ▶ Developed by “Satoshi Nakamoto” (only a pseudonym)
 - paper published in 2008
 - open source software in 2009, see github.com/bitcoin
- ▶ Bitcoins can be bought and sold easily, eg. via bitonic.nl; payment in shops possible via eg. bitkassa.nl
- ▶ Bitcoins undermine current financial control
 - used for illicit purchases (recall **Silk Road**)
 - little stability, eg. in bankruptcy of the Mt.Gox exchange (850,000 BTC missing ~ \$450 million)
 - volatile value




Bitcoin value, against US\$ (2012 – dec. 2017)



(source: bitcoincharts.com, dec. 2017)

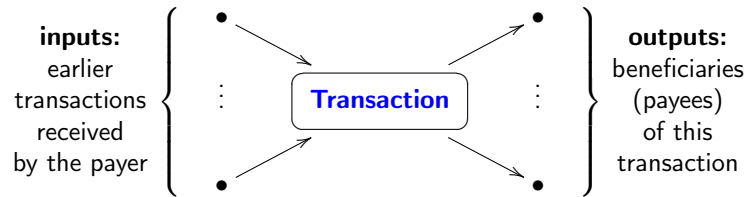
Main points about bitcoin

- (1) Public key cryptography and hashing, as main ingredients of **transactions**
 - hence we can understand it in this introductory course
 - explanation here is **conceptual**, not literally following the code
- (2) **Peer-to-peer** networking: transactions are sent out to the network where all bitcoin nodes can see it within a minute
- (3) The public **ledger** (NL: *groot/kas-boek*): a **blockchain** is maintained as a single list all transactions. Every node on the network has a copy, so that the balance of every address (account) is known — but not necessarily who the owner is.
 - this blockchain is now the real hype

Capitalised Bitcoin is used for the system/protocol, and lower-case bitcoins for the currency units (or BTC, or )



Bitcoin transaction (commonly denoted as: tx)



- ▶ The sum of the bitcoin amounts in the inputs must **exceed** the sum of the amounts in the outputs
- ▶ The difference is the **transaction fee**, which is for the successful "miner" (see later)
 - In practice a non-zero fee is needed to get processed



Bitcoin transaction arithmetic

- ▶ Suppose that Alice wants to pay **5 BTC** to Bob, ...
- ▶ ... and that Alice has been payed herself in two previous transactions, one with **2.5 BTC** and one with **4 BTC**.

How to proceed?

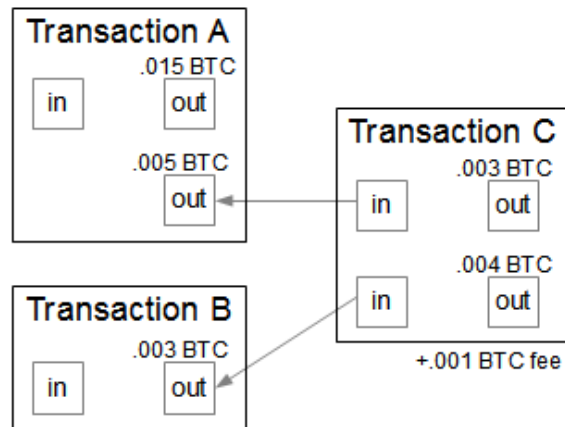
- ▶ For the 5 BTC payment to Bob, Alice can use:
 - **inputs:** both these transactions, of **2.5 BTC** and **4 BTC**
 - **outputs:** **5 BTC** to Bob, and **1,49999 BTC** to herself
 - The transaction fee is thus:

$$(2.5 + 4) - (5 + 1,49999) = 0.0001 \text{ BTC}$$

- ▶ if 1 BTC = 800€, this fee is 8 eurocent.



Transaction inputs, in a diagram



(source: Ken Shirriff's blog, feb. 2014)



Bitcoin addresses and keys

- ▶ A Bitcoin address is a **hash of a public key**
 - Actually, it involves several SHA-256 and RIPEMD-160 operations, but conceptually we treat it has a single hash
 - Notation: $\text{address} = h(\text{pubkey})$
 - The key is a 256 bit ECDSA public key
- ▶ A user may have/generate/use multiple addresses
 - the addresses are all public, but you can hide the link between you and your addresses (eg. via mixers)
 - this provides (some) transaction privacy
 - using multiple addresses gives an additional level of obfuscation

When do you need your public/private key pair?

- ▶ to claim (redeem) an incoming transaction, by revealing your public key, as pre-image of the hash/address
- ▶ to sign an outgoing transaction, using the incoming amount



Bitcoin transaction message structure

Assume:

- ▶ Alice (A) wants to transfer b bitcoins to Bob (B) and c bitcoins to Charly (C); A knows the addresses of B,C
- ▶ this transaction involves only two input transactions tx_1, tx_2 , to addresses $h(e_1), h(e_2)$ of Alice — with private keys d_1, d_2

Bitcoin transaction message structure, continued

The transaction message is a concatenation | of three parts:

$$A \rightarrow \text{Network} : h(m) | m \quad \text{where} \quad m = \text{in} | \text{out}$$

The hash $h(m)$ is used as **identifier** of the transaction, and:

- ▶ $\text{out} = b | \text{address}_B | c | \text{address}_C$
- ▶ $\text{in} = \text{id}_{tx_1} | [\text{id}_{tx_1}, \text{out}]_{d_1} | e_1 | \text{id}_{tx_2} | [\text{id}_{tx_2}, \text{out}]_{d_2} | e_2$

(The signatures $[-]_{d_i}$ are actually more complicated signing scripts)



Verifying a transaction

The verification of transaction involves several aspects:

- (1) checking the identifier **hash** in $h(m) | m$
- (2) checking the **signature** in the input $\dots | [\text{id}_{tx}, \text{out}]_d | e | \dots$
- (3) looking up (in the “block-chain”) the previous transaction tx corresponding to the identifiers id_{tx} in the inputs, and checking that it is “confirmed”
 - what this precisely means follows below
- (4) Checking that the public keys in the current transaction are the **pre-images** of the addresses in these previous transactions
- (5) Checking that the incoming amounts are at least the outgoing ones.



Distributed consensus

- ▶ Transactions must be approved by the “network” or “community”
- ▶ A cheater could try to quickly approve his own transactions
 - in order to prevent this, checking is made really **difficult**
 - more concretely, it requires much computational power
 - this work is called **proof-of-work** or **mining**
- ▶ But then: who would want to do so much work?
 - solution: make mining into a competition
 - the winner is rewarded, . . . , with bitcoins



The block chain, as public ledger

- ▶ The block chain is a **shared public ledger** on which the whole Bitcoin system relies. All confirmed transactions are included permanently in this single block chain.
 - Watch ongoing activity eg. at blockchain.info or blockexplorer.com
- ▶ **Mining** is used to confirm waiting transactions by including them in the block chain. It enforces a chronological order in the block chain.
- ▶ To be confirmed, transactions must be **packed in a block** via a matching hash rule that will be verified by the network. These rules prevent modification of previous blocks.

Adding blocks to the chain

- ▶ Bitcoin transactions are **broadcast** to “the network”, and are received by peers, that may collect them into new blocks
- ▶ These blocks need to contain the solution to a **hash puzzle**; only then can they be added to the block-chain, via a reference to the previous block
 - the peer that solves the puzzle gets all the transaction fees, plus a fixed number of bitcoins (currently 25)
- ▶ The difficulty of the puzzle is regularly adjusted so that new blocks are added roughly every 10 minutes
- ▶ If by chance there are (nearly) simultaneous solutions:
 - the chain may fork, but only temporarily because of the rule: *extend only the longest path*
 - after a fork, work continues on both paths, until one is extended, and work on the other path stops



Proof-of-work: the hash puzzle

- ▶ A peer may decide to collect, say $k = 100$ transactions, check them all, and concatenate them to a string

$$s = \text{last_block_ref} \mid \text{peer_adr} \mid \text{tx}_1 \mid \dots \mid \text{tx}_k$$

- ▶ The **hash puzzle** is now to find a nonce/number N so that:

$$h(s \mid N) \text{ has } t \text{ leading zeros}$$

- ▶ This $t \in \mathbb{N}$ is the “target” that determines the difficulty of the puzzle (an average solution time of 10 min. is intended)
- ▶ Once a peer claims to have found N , it can announce so, and other peers can **easily check** this
 - the block of k transactions is added to the block-chain
- ▶ Only if a transaction is followed by 6 blocks in the chain, it is **confirmed**

Proof-of-work demo, in Python

```
import hashlib, time
h = hashlib.new("sha256")
prefix_length = 6
zeros = "0" * prefix_length
counter = 0
s = "transactions-block"
unfinished = True
while unfinished:
    h.update(s + str(counter))
    prefix = h.hexdigest()[:prefix_length]
    if prefix == zeros:
        print time.clock(), counter, h.hexdigest()
        unfinished = False
    else:
        counter = counter + 1
```



Bitcoin: some final remarks

- ▶ The above explanation glosses over many details and implementation issues (like Merkle trees)
- ▶ Bitcoin fits in internet tradition of: *dump the intermediaries*
 - ie, put intelligence in the end-points, keep the network dumb
 - but: intermediaries can be of value, for quality control
- ▶ Bitcoin is insanely environmentally unfriendly:
 - Bitcoin requires more energy per year than the whole of NL
- ▶ Public authorities have difficulty coping with Bitcoin
 - mixed reactions (banning, tolerating, ignoring)
 - NL attitude (DNB/AFM): “there are risks”
- ▶ Anonymity of bitcoin addresses has advantages and disadvantages . . .
 - grouping transactions is an active research area.
- ▶ Not so much Bitcoin, but the underlying blockchain technology has become a complete **hype**
 - See BJ’s article [Reason yourself out of blockchains](#) (nov’17)

About the exam, part I

- ▶ **Make sure (and check) that you are registered for the exam** (otherwise you simply cannot participate!)
- ▶ Closed book; simple calculator is provided (only +, -, *, /)
- ▶ Questions are in line with exercises from assignments
- ▶ In principle, slides contain all necessary material
 - wikipedia also explains a lot
- ▶ Number theoretic theorems, propositions, lemmas:
 - are needed to understand the theory
 - their *proofs* are not required for the exam (but do help understanding)
 - need *not* be reproducible literally
 - but help you to understand questions



About the exam, part II

What you must surely know:

- (1) Calculation rules (or formulas) must be **known by heart** for RSA & El Gamal en/de-cryption (but not signing), Diffie-Hellman
- (2) Basic protocols for confidentiality, integrity, authentication, non-repudiation
 - both in the symmetric & asymmetric case
- (3) Basic properties of cryptographic primitives: symmetric, hash (birthday), asymmetric (PKI infrastructure)
- (4) block and stream ciphers, and modes of encryption
- (5) Basic number-theoretic constructs:
 - modulo addition, subtraction, multiplication, division, (extended) gcd, square-and-multiply
 - generator, discrete log, order of a group/element — but no abstract group theory

About the exam, part III

- ▶ Questions are formulated in english
 - you may choose to answer in Dutch or English (no other languages!)
- ▶ Give intermediate calculation results
 - just giving the outcome (say: 68) yields **no points** when the answer should be 67
- ▶ Write legibly, and explain what you are doing
 - giving explanations forces **yourself** to think systematically
 - mitigates calculation mistakes
- ▶ Perform checks yourself, whenever possible.



Finally ...

Practice, practice, practice!

(so that you can rely on skills, not on luck)



Final request

- ▶ Fill out the **enquete** form for *Security*
- ▶ This feedback is really used to improve courses!



What is computer security all about?

Original formulation

Regulating access to digital assets

More mature formulation

The protection of information and information systems against unauthorised access or modification of information, whether in storage, processing or transit, and against denial of service to authorised users. Information security includes those measures necessary to detect, document, and counter such threats.

(From: Jones, Kovacich, and Luzwick, Global Information Warfare, 2002)



What this course tried to achieve

- ▶ Insight **both** in:
 - basic computer security mechanisms
 - design & usage issues, in organisations and in society
- ▶ Expected competences on-the-job:
 - **computer** scientists should master technicalities
 - **information** scientists should be able to translate & exploit the relevance of these technicalities for the business/organisation (there is greatest need for people who can do this)
- ▶ But ideally, you should be able to do both!



What you read between the lines, hopefully

- ▶ Information is power
 - informational power leads to societal power
- ▶ Security is about regulating access to information
 - hence it has to deal with these (political) matters
- ▶ Ethical & political issues are part of the field
 - you need a strong moral compass for this field
 - eg. in order not to abuse access (as insider, programmer, hacker)
 - or to make the right design decisions (fair, democratic, ...)

Finally: **enthusiasm** in what you do makes the difference!

- ▶ not only for yourself, but also for the ones you work with
- ▶ We hope that we conveyed some of that enthusiasm. 😊

