

Computer Security: Security at Work

B. Jacobs and J. Daemen

Institute for Computing and Information Sciences – Digital Security

Radboud University Nijmegen

Version: fall 2016



Outline

Bitcoins

- The ledger

Authentication and Identity Management

- Authentication

- Identity management

- Kerberos, and derivatives

- Attributes instead of identities

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:


$$\left[\text{I, Alice, transfer 10€ with serial number } N \text{ to Bob} \right]_{d_{\text{Alice}}}$$

- ▶ 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. 2016)




(source: bitcoincharts.com, dec. 2016)



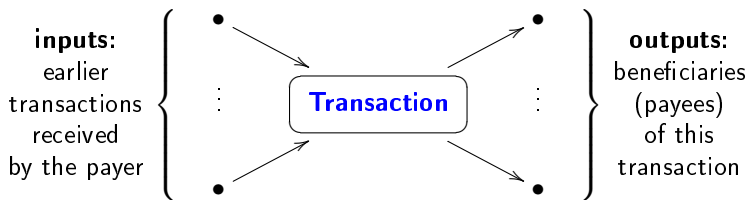
Main points about bitcoin

- (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.

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?

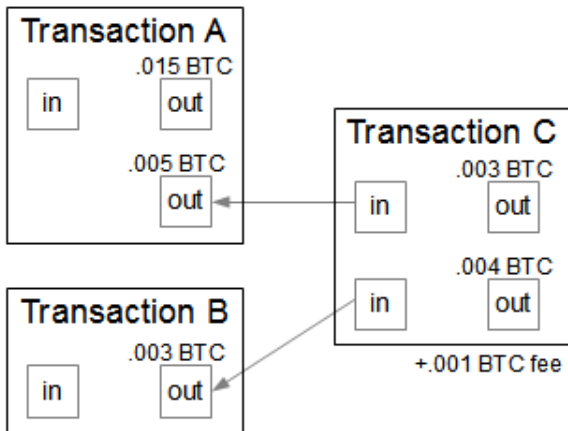
- ▶ For the 5 BTC payment to Bob, Alice can use:
 - **inputs:** both these transactions, of **2.5 BTC** and **4 BTC**
 - **outputs:** **5 BTC** to Bob, and **1,49999 BTC** to herself
 - The transaction fee is thus:

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

- ▶ if $1 \text{ BTC} = 800\text{€}$, 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: 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

(Pay-to-PubkeyHash version)

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:

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

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

- ▶ $\text{out} = b | \text{address}_B | c | \text{address}_C$
- ▶ $\text{in} = \text{id}_{\text{tx}_1} | [\text{id}_{\text{tx}_1}, \text{out}]_{d_1} | e_1 | \text{id}_{\text{tx}_2} | [\text{id}_{\text{tx}_2}, \text{out}]_{d_2} | e_2$

(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 $h(m) \mid m$
- (2) checking the **signature** in the input $\dots \mid [id_{tx}, out]_d \mid e \mid \dots$
- (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 not “green”
 - Forbes’13 claim: \$15M per day in electricity for mining
- ▶ 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**



Real-world and virtual-world authentication

- ▶ In daily life we rely on **context** for many forms of (implicit) authentication
 - uniforms / places / behaviour / etc
- ▶ In the **online world** such contexts are either lacking, or easy to manipulate (fake e-banking site)



"On the internet nobody knows you're a dog"

(Peter Steiner, New Yorker, 1993)

Correction

In the age of profiling this anonymity suggestion is completely outdated!

NOISE TO SIGNAL
RobCottingham.ca/cartoon



How the hell does Facebook know I'm a dog?

Human to computer authentication

Recall: **identification** = saying who you are; **authentication** = proving who you are.

The three basic human-to-computer authentication mechanisms are based on:

- (1) **something you have**, like a (physical) key, or card
Risk? theft, copying
- (2) **something you know**, like a password or PIN
Risk? eavesdropping (shoulder-surfing), brute-force trials, forgetting (how secure is the recovery procedure?), social engineering, multiple use, fake login screens (use wrong password first!)
- (3) **something you are**, ie. biometrics, like fingerprints or iris
Risk? imitation (non-replaceability), multiple use



More about passwords

It is common wisdom that at least a 64 bit string is needed to be secure against password guessing. These 64 bit amount to:

- ▶ 11 characters, randomly chosen
- ▶ 16 characters, computer generated but pronounceable
- ▶ 32 characters, user-chosen

With modern brute force and rule-based techniques, passwords can be broken easily. A well-known system to do so is **Crack**

Heuristics

Reasonably good passwords come from longer phrases, eg. as first letters of the words in a sentence: they are relatively easy to remember, and reasonably arbitrary (with much entropy).

It is then still wise to filter on bad passwords.

An alternative is to use one-time passwords, distributed via an independent channel (eg. via a generator, via GSM or TAN-lists).



Password change policies

Does it make sense to force users to change their passwords periodically (say every 3 months)?

- ▶ **Pro:** compromised passwords are usable for only a relatively short amount of time
- ▶ **Against:** lot's of things:
 - the cause of a password compromise (if any) is ignored, and may be re-exploited
 - users get annoyed, and use escape techniques:
 - ▶ insecure variations: *passwd1*, *passwd-2010* etc.
 - ▶ writing passwords down
(so that they become 'something you have')
 - more helpdesk calls, because people immediately forget their latest version
 - sometimes requests to change passwords are sent by mail, with **login link!!** Every heard of fishing? (This happened @RU)



Password recovery

What to do when a user forgets his/her password? This happens frequently. Hence recovery procedures should not be too complicated (or expensive). What to do?

Some options:

- ▶ **self service password reset**, by supplying answers to previously set security questions, like “where was your mother born?” “what’s your first pet’s name?” etc.
Often, answers can be obtained by social engineering, phishing or simple research (recall the Sarah Palin mailbox incident in 2008)
- ▶ Provide a new password via a **different channel**
 - face-to-face transfer is best, but not always practical
 - *ING bank* provides new password via SMS
(recall: GSM (esp. SMS) is now broken)
- ▶ force re-registration (like *DigiD* does in NL)



Biometrics: intro

Biometrics refers to the use of physical characteristics or deeply ingrained behaviour or skills to identify a person.

- ▶ **Physical characteristics**: facial features, fingerprints, iris, voice, DNA, and the shape of hands or even ears.
- ▶ **Behaviour or skill**: handwritten signature, but also someone's gait, or the rhythm in which someone types on a keyboard.

Different types of biometrics have **important differences** in:

- ▶ accuracy (percentage of false matches/non-matches)
- ▶ how easy they are to fake
- ▶ which population groups they discriminate against
- ▶ how much information they reveal about us, and how sensitive this information is (eg. your DNA may reveal health risks of interest to insurance companies)



Biometrics: intentional or unintentional

Important difference between types of biometrics:

- ▶ necessarily **intentional** and conscious production, like with signature (except under extreme coercion)
- ▶ possibly **unintentional** production: people leave copies of their fingerprints and samples of their DNA wherever they go.
 - With the increased use of surveillance cameras we also leave our facial image and gait in many places. This is what enables such biometrics to be used in **law enforcement**
 - It also makes fingerprint information more valuable to the owner, and to potential attackers, as fake fingerprints could be planted at a crime scene.



Biometric systems in operation

A biometric system works in several steps

- (1) its **sensors** capture a presented biometric
- (2) this input signal is then processed to **extract features** from it
- (3) these features are **compared** to previously recorded and stored biometric information
- (4) it is decided if there is a **match** or not

Ideally, not the raw biometric information is stored, but a **template** with crucial info about features extracted from the raw data

Fingerprint example

- ▶ **raw information**: image of the fingerprint (stored eg. in e-passport)
- ▶ **template**: so-called minutiae, bifurcations and endpoints of ridges,

which most fingerprint recognition systems use
Storing such templates goes some way towards preventing abuse, assuming that fingerprints cannot be reconstructed from the templates.



Biometrics for verification or identification

Biometrics can be used in two completely separate ways:

- ▶ **Verification**: a person is matched with one particular stored biometric (template), eg. the fingerprint on his e-passport, to check that someone has a certain claimed identity
- ▶ **Identification**: a person is matched with a large collection of stored biometrics, for example to see if he occurs in a database of known criminals, or has not already applied for a passport under a different name

(Clearly, this is more error-prone than one-to-one matches, since in one-to-many matches errors accumulate)



e-Passport example in NL

- ▶ originally proposed for **verification only** (against look-alike fraud)
- ▶ **function creep** happened in the form of **central storage** of all biometrics: now usable for identification and law enforcement
- ▶ in 2011 these central storage plans were **abandoned** again
 - official reason: technique not ready
 - opposition in parliament: privacy concerns, fear of data loss



DNA example in NL

- ▶ In 2012 similar plans emerged to store DNA of **all citizens** in NL in order to find criminals more easily
 - in the wake of the solving of the Marianne Vaatstra murder
 - most likely this is not allowed by law (ECHR): authorities are only allowed to collect data on suspects
- ▶ What else is the **problem** with this?
 - Remember that in a state of law there should be a balance of power between citizens and the authorities!
 - Also remember the historical experience that authorities may become unfriendly
 - And imagine the privacy-disaster if such a DNA database gets compromised, by hacking or mismanagement



Biometric systems are not perfect

- ▶ **False match:** the system reports a match when in fact the stored biometric comes from someone else
Example: innocent person barred from boarding a plane
- ▶ **False non-match:** the system reports that the two don't match, even though both are from the same person
Example: Bin Laden gets on board

Note on terminology

False matches are often called **false accepts**, and false non-matches **false rejects**. This can be confusing: if a database of biometrics is used to check that known terrorists do not enter the country, then a false non-match leads to a false accept (into the country), not a false reject



Biometrics performance

- ▶ Exact rates of false (non-)matches depend on the type of biometric used and the particulars of the system (eg. verification or identification).
- ▶ There is a **trade-off** between the false match and non-match rates: by turning up the precision required for a match, the false non-match rate of a system can be decreased at the expense of a higher false match rate.

Tuning the system for a good balance

- ▶ what is the **purpose**: do you prefer a higher false non-match rate or a higher false match rate?
- ▶ **who controls** the tuning: guards with a no-entry list hate false matches because of the hassle (angry customers). Hence they minimise false matches, leading possibly to a greater risk of false non-matches (terrorist entering the building)



Biometrics performance studies

NL passport fingerprint study (2005, 15.000 participants)

- ▶ At enrollment phase, 3.2% of fingerprints could not be recorded
 - 1.9% impossible to record two fingerprints
 - 1.3% only possible to record one
- ▶ In verification phase, in 4.3% one finger could not be verified; in 2.9% neither finger

US-VISIT study (2004, 6.000.000 in database)

- ▶ false match rate of 0.31% (1 in 300 hassle for innocent travellers)
- ▶ changing operational parameters:
 - false match rate reduced to 0.08%
 - false non-match rate rise to 4% to 5%

In NL Ton van der Putte is famous for breaking almost all commercially available fingerprint sensors



Biometrics usage

For identification Useful, with error margins

- ▶ basis for usage in surveillance systems

For authentication Problematic, since it assumes that:

- ▶ only you are the source of fresh biometric measurements
- ▶ freshness of such measurements can be recognised
- ▶ you provide input to these fresh measurements intentionally and consciously

For non-repudiation Unsuitable: same spoofing problems

- ▶ biometrics not suitable as signatures in payment systems

How about biometrics for access to secure facilities

- ▶ only rarely used type of biometrics, like hand-palm or iris
- ▶ spoofing/transfer is more difficult



Privacy issues in biometrics

- (1) biometric measurements may contain **much more information** than is strictly needed for identification
 - eg. DNA contains your genetic build up (and of subsequent generations)
 - also claimed for eyes, by irisscopists 😊
- (2) when **improperly stored** (as original measurements and not as abstract templates) and protected, biometrics may actually increase the risk of identity fraud
- (3) biometric information may be used for **tracing** people, either openly, for instance via public security cameras, or covertly



Biometrics, conclusions

- ▶ biometrics are often proposed as solution to the security problems associated with passwords
- ▶ however, they are problematic themselves (highly overrated)
 - always the same, in every application
 - not replaceable (after compromise)
- ▶ entangled error rates associated with false (non-)matches
 - errors accumulate in one-to-many comparisons
- ▶ really useful only for identification, and not for authentication (or non-repudiation)



What is Identity Management (IdM)?

Allowing many services via a limited number of access / authentication checks. It is a collection of mechanisms for

- ▶ identity synchronisation
- ▶ single-sign-on
- ▶ access management

So-called **federated** IdM is IdM between different organisations.

Possible functions of IdM

- ▶ **Authentication**, esp. via single-sign-on
- ▶ **Autorisation**, via access control lists (ACLs) at objects, or based on capabilities/roles at subjects, supported by credentials
- ▶ **Personalisation**, service adjustment to individual preferences
- ▶ **Provisioning**, *i.e.* automatic propagation of changes in identity data



Advantages & disadvantages of IdM

Advantages of IdM

- ▶ centralisation of control, administration and policy
- ▶ ease for users
- ▶ structuring of roles and responsibilities within organisations
- ▶ cost reduction

Disadvantages of IdM

- ▶ possible reliability reduction, via single point of failure;
- ▶ increased linking of activities, harming privacy.



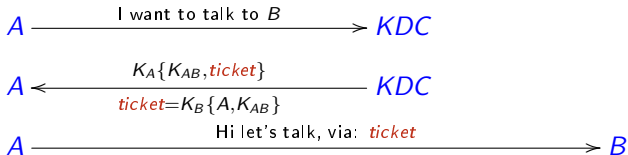
Examples of IdM systems

- ▶ Kerberos
- ▶ OpenId
- ▶ DigiD
- ▶ Eduroam
- ▶ Facebook / Google+ login
- ▶ ...
- ▶ **IRMA**, attribute-based authentication, under development at Nijmegen



Key Distribution Center (KDC)

- ▶ A KDC shares a secret key K_X with each participant X
- ▶ **Naive usage**: let all communication, say between A and B , go via the KDC who decrypts and re-encrypts in the middle
- ▶ **More efficiently**: let the KDC provide a session key, to be used by A and B directly, like in:



- ▶ These first steps must be followed by a standard mutual authentication between A and B , using the session key K_{AB} .
- ▶ The KDC does not send the ticket itself to B , but lets A do this, in order to limit its load.



KDC issues

Disadvantages of a KDC

- ▶ It is a single point of failure because it must always be online
- ▶ The KDC can read all traffic (since it knows the keys K_{AB})
- ▶ The KDC can impersonate everyone
- ▶ The KDC may be a performance bottleneck

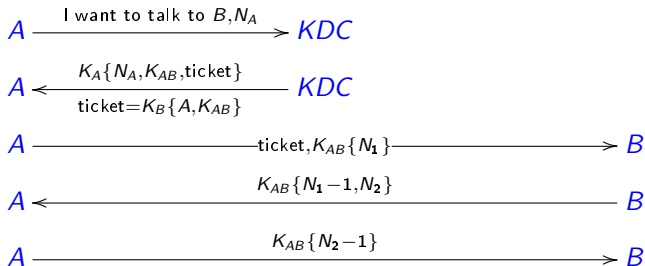
So far, there is no identification of runs

- ▶ not for A , in the link between the initial request and answer from the KDC
- ▶ not for B , in the link between the ticket and the request of A : an old ticket might be re-used.

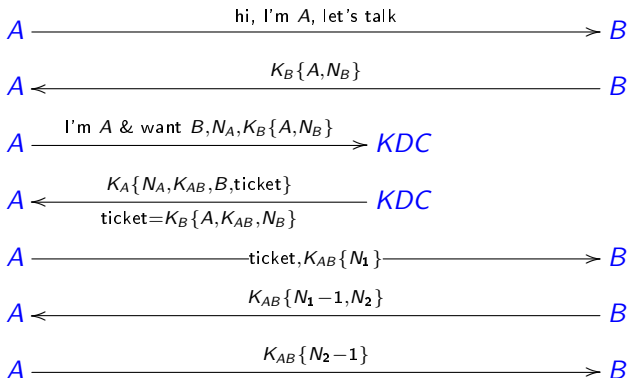


Using tickets via a Key Distribution Center (KDC)

Basis for Kerberos comes from Needham-Schroeder (1978):



Better include nonce as session-binder in a ticket

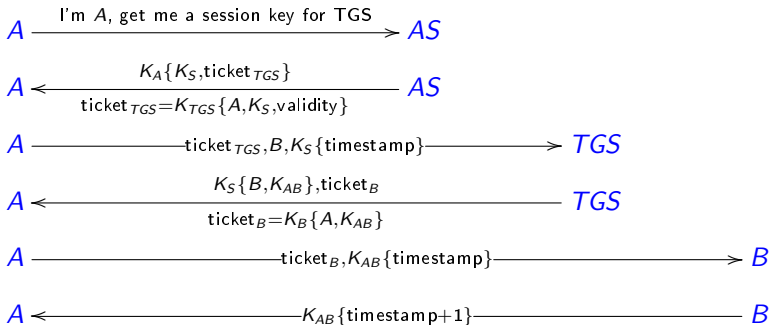


Kerberos intro

- ▶ Kerberos is a secret key based authentication service in a network
 - developed at MIT in 1980s
 - now used in Windows & Linux (and elsewhere)
- ▶ Kerberos splits Key Distribution Center (KDC) into two roles:
 - **Authentication Server (AS)**
Each user X (including the TGS) shares a key K_X with the AS.
 - **Ticket-Granting Server (TGS)**.
- ▶ **Kerberos' aim**: let Alice access servers after she has **authenticated herself once**:
 - by decrypting a secret from the AS
 - at her own workstation
 - by only locally using her password K_ASubsequently, Alice uses a session key K_S .



Kerberos 4, protocol



- ▶ A and B can communicate under cover of K_{AB} ; B trusts that anyone knowing K_{AB} is acting on behalf of A
- ▶ A can use ticket_{TGS} at multiple service providers (for some time)





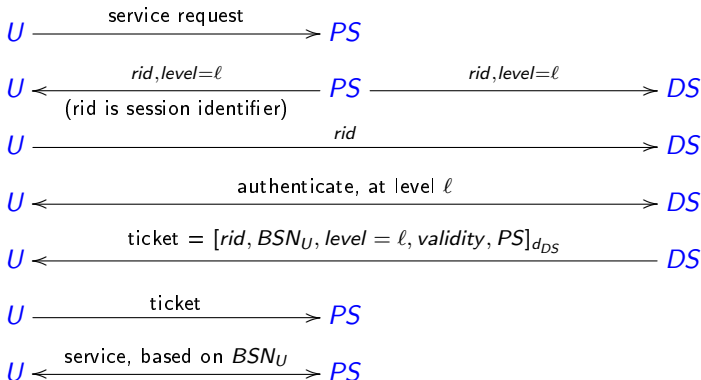
DigiD intro

- ▶ DigiD is central authentication service for government services
 - tax, local authorities, social benefits, etc
 - operational since 2005
- ▶ Citizen identification based on BSN (*Burger Service Nummer*)
 - BSN can be used by all government services & health care
 - use in commercial sector not allowed (except in special mandatory circumstances)
- ▶ DigiD has three levels/strengths of authentication
 - login + password
 - one-time password via SMS
 - smart card based (currently under development, as eID)
- ▶ DigiD is based on A-select, which is based on Kerberos



DigiD protocol essentials

Let U = User, PS = Public Service, DS = DigiD Server in the following messages (protected eg. via SSL)



OpenId

- ▶ Open (standard) framework for Single-Sign On (SSO), used eg. by MicroSoft, Google, Yahoo
- ▶ Main parties involved:
 - **Relaying Party** (RP), eg. website where authentication is required
 - **User** (U), who wishes to use some online service from a RP
 - **Identity Provider** (IP), providing authentication, for multiple RPs.
 - ▶ **In practice**, RP = IDP, since no RP trusts other IdP

- ▶ Basic mechanisms via redirects:

$$U \longrightarrow RP \longrightarrow U \longrightarrow IP \longrightarrow U \longrightarrow RP$$

- ▶ Focus on usability, not security (eg. ssl is not mandatory)
- ▶ OpenID is not widely used; new initiatives, like **FIDO Alliance**, keep coming up (FIDO = Fast Identity Online, based on PKI)



Who are you? Identities and attributes

- ▶ If you wish to buy a bottle of whiskey, you have to show that you are over 18 — fair enough
- ▶ In practice (offline) you wave an identity card in front of the shopkeeper
- ▶ But what if the shopkeeper would make a photocopy, or read your identity document electronically?
 - online this becomes even more problematic
- ▶ The transaction only requires the **attribute** “over 18”
 - and not your **identity** (whatever that is)
 - any additional information, besides “over 18”, can be **abused** (identity fraud, profiling)
 - attribute-usage fits in data minimalisation requirements
- ▶ What we need is **proportional authentication**



Identities & attributes II

- ▶ Some attributes are **identifying**
 - like your social, security number or bank account, or OV-chipcard number
 - they are different for different people
- ▶ Other attributes are **non-identifying** (anonymous)
 - like your gender, whether you're over 18, your home-town
 - whether you have a valid ticket to travel by bus
 - whether you are a nurse or a doctor
- ▶ Sometimes your identity is understood as a (small) set of identifying attributes, like on your passport
- ▶ When going digital, attributes are often replaced by identities, like in public transport
 - why do I have to tell who I am when I get on the bus
 - more unnecessary surveillance / profiling / fraud risk

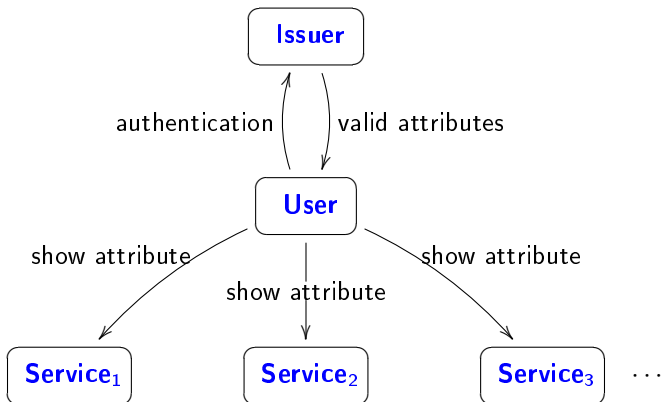


Attribute-based authentication & authorisation

- ▶ Many transactions can be performed with **non-identifying attributes**
 - a cheaper hair-cut for a student,
 - participation in local referendum for locals
 - buying games online (over 16, or over 18)
 - viewer restrictions for missed TV-program website
- ▶ **Attribute-based** extends **role-based** access control
 - the captain of the ship can turn the ship's wheel
 - very relevant in the medical sector (access to files)
 - or in the military, or in any other organisation with different authorisations for different hierarchies/roles
- ▶ Typical transactions involve a **combination of attributes**
 - address, possibly with bank account, for pizza delivery
 - age + bank account for online gambling / XXX / ...
 - “doctor” status + medical registration number for write-access to medical record



Attribute issuance-usage model



One may also have **multiple issuers** (government, banks, ISPs, ...)



Requirements for attribute-based systems

- ▶ **Non-transferability**: my little nephew should not be able to get my “over 18” attribute (and go to XXX sites)
 - realised via binding to my secret key
 - card is PIN-protected
- ▶ **Issuer-unlinkability**: the issuers should not be able to track where I use which attribute
 - typically realised via blind signature
- ▶ **Multi-show unlinkability**: service providers should not be able to connect usage (at different relying parties)
 - realised via zero-knowledge proofs, or via “self-blindable” credentials
- ▶ **Revocation**: rogue attributes (via stolen/lost cards) should be blockable.
 - difficult, but doable, for non-identifying attributes



IRMA project @Nijmegen

- ▶ IRMA = “I Reveal My Attributes”
 - also the name of the secretary of the Digital Security group
- ▶ Project for attribute-based authentication
 - **Idemix** from IBM provides cryptographic basis
 - very fast smart card implementation developed at Nijmegen
 - “pushing the technology”
 - open implementations, see <https://github.com/credentials>
- ▶ Small pilots ongoing, eg. run by Thalia
- ▶ Attributes are the next, hot thing in identity management
 - For more info, check out irmacard.org



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, both en/de-encryption & 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
 - 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. 😊

