Computer Security: Security at Work

Bart Jacobs

Institute for Computing and Information Sciences – Digital Security
Radboud University Nijmegen

Version: fall 2011

Outline

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)

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

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 processes 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
- Identication: 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 abanoned again
  - official reason: technique not ready
  - opposition in parliament: privacy concerns, fear of data loss

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%

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 controle 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 responsabilities 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
- ...

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:

$$A \xrightarrow{\text{I want to talk to } B} KDC$$

$$A \xleftarrow[ticket = K_B\{A, K_{AB}\}]{K_A\{K_{AB}, ticket\}} KDC$$

$$A \xrightarrow{\text{Hi let's talk, via: } ticket} B$$

- 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),

$$A \xrightarrow{\text{I want to talk to } B, N_A} KDC$$

$$A \xleftarrow[\text{ticket} = K_B\{A, K_{AB}\}]{K_A\{N_A, K_{AB}, \text{ticket}\}} KDC$$

$$A \xrightarrow{\text{ticket}, K_{AB}\{N_1\}} B$$

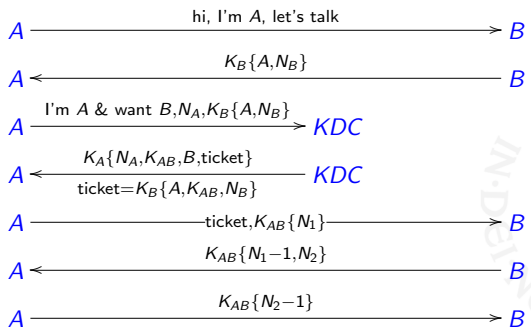
$$A \xleftarrow{K_{AB}\{N_1 - 1, N_2\}} B$$

$$A \xrightarrow{K_{AB}\{N_2 - 1\}} B$$

Note that the ticket may still be reused in the (exceptional) situation when an attacker manages to get hold of either:

- the session key K_{AB}
- the shared key K_A (even if A changes to a new key K'_A)

Better include nonce as session-binder in a ticket

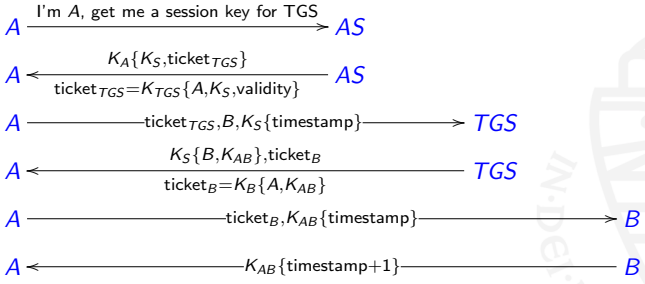


Kerberos intro

- 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_A

  Subsequently, Alice uses a session key K_S.

Kerberos 4, protocol



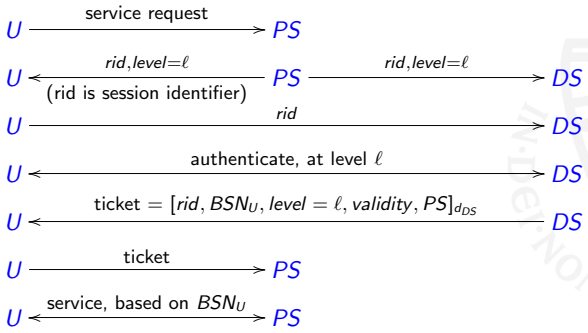
- A and B can communicate under cover of K_{AB}; B trusts that anyone knowing K_{AB} is acting on behalve of A
- A can use ticket_{TGS} at multiple service providers (for some time)

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 not implemented)
- 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)

Access control basics

- Terminology: "subjects" can perform "actions" on "objects"
- Two basic access control approaches:
  1. list per object who can do what, via "access conrol lists"
  2. list per subject (or group, or role) what can be done to which objects, via "capability lists"
- These two can be combined in an access control matrix:

	obj1	obj2	obj3
subj1	read	read, write	read
subj2	write	write	
subj2	exec	read, exec	exec

- The mechanism that checks such permissions is usually called the reference monitor

Slicing things up

If you wish to restrict access, there are two basic forms:

- Horizontal divisions into security levels

top secret
secret
confidential
public

This may be used both for confidentiality and for integrity

- Vertical divisions into security compartments

project 1	project 2	project 3	project 4

"Need to know" or "Chinese wall" approach

These two approaches can also be combined

Operational rules for horizontal slicing

- For confidentiality (most common)
  - No read up: you are not allowed to read material of higher classification than you have (obviously)
  - No write down: this may reveal "higher" material to lower classifications

  This is the essence of the so-called Bell-LaPadula (BLP) model; rules may be enforced via "data diodes"
- For integrity
  - No write up: if your classification says "can write reliably up-to level n", writing higher may spoil integrity there
  - No read down: it may spoil your mind, so that you make mistakes when writing higher up (at your level)

In practice these rules are very restrictive (eg. in multilevel operating systems)

OS security issues

- OS security tasks include: identification, authentication, access control (authorisation), and auditing
- preferably these (sensitive) tasks are concentrated in a small trusted computing base (TCB)
- small security "micro" kernels are a recent development, especially as basis for virtualisation
- Security failures often based on programming errors
  - encryption etc. does not help much against them
- ordinary OSs are highly complex & dynamic; they require constant updating
  - problematic for certification
  - updates may break consistency in larger systems
  - risks from update postponement/neglect and zero-day exploits
  - update process itself may also be vulnerable (via rogue certificates)
  - (alien) driver software in kernel is problematic

Example: Unix/Linux security essentials

- Long history: Unix started as multi-user OS in network, scaled up to servers, and scaled down to PCs (esp. Linux)
- Originally developed for friendly environments (research labs or universities), with weak security mechanisms
  - security controls are add-ons, not part of architecture
- Basic philosophy: security is managed by skilled administrator, not by average user.
- Basic set-up:
  - Users have user and group identities (UID, GID); Users can belong to multiple groups; no security checks for "root" user
  - Objects (resources) are files, directories, devices; permission is octal number (like 654) or `rw-r-xr--`, for owner-group-other
  - Processes has several identities (real, effective, saved)

Network security, in general

- Being connected is a mixed blessing
- Internet protocols have been designed for robustness, not for security (benign environment assumed)
  - sometimes attacks are shockingly simple
- Security focus on access control (including firewalls), intrusion detection, and resilience
- Programming errors main source of problems
  - keeping up-to-date with security warnings is full-time job
  - reporting itself is delicate: responsible disclosure means: informing manufacturer first, and publishing after some delay
  - zero-day exploits dangerous & valuable (eg. stuxnet used 4!)

Basic internet protocols

Layering of protocols:

application (HTTP, FTP, etc)
transport (TCP)
network/routing (IP)
link

- IP = Internet Protocol: routes packets between (multiple) machines, each with 32 bit address (192.168.1.1 notation)
- TCP = Transmission Control Protocol: reliable, delivers a stream of bytes between two machines; IP packets can be lost, duplicated, or delivered out of order, but TCP detects these problems, requests retransmission of lost data, rearranges out-of-order data etc.

Security mechanisms can be implemented at different layers

Other internet protocols

- DNS = Domain Name System: translates domain names meaningful to humans into the IP addresses
  - lookup: `dig porthos.science.ru.nl` yields `131.174.30.39` (among others)
  - reverse lookup: `dig -x 131.174.30.38` yields `poly.science.ru.nl`
- SNMP = Simple Network Management Protocol
  - used for configuring machines in the network
  - originally without security
- BGP = Border Gate Protocol
  - enables groups of routers to share routing information
  - based on local tables, which are updated regularly
- ICMP = Internet Control Message Protocol
  - used for error & status messages
  - includes eg. *ping*

Typical network protocol (attack)

- First, note: on the internet it is easier to send a message with a forged sender address than to intercept a message
  - used for address- (instead of crypto-) based authentication
- Three-way handshake is used for setting up TCP connections, with 'SYN' and 'ACK' as message types
- Simple exchange of 32 bit nonces:
  1. $A \longrightarrow B : A, \mathrm{SYN}(N_A)$
  2. $B \longrightarrow A : \mathrm{ACK}(N_A + 1),\ \mathrm{SYN}(N_B)$
  3. $A \longrightarrow B : \mathrm{ACK}(N_B + 1)$
- easy to forge 1, but difficult to intercept 2, so attacker does not see N_B and does not know how to reply
  - however, in many (old) implementations N_B is predictable
  - eg. by setting up a proper connection with B first
- Once message 3 is accepted, the attacker can request services from B, masqueraded as A: "blind connection forgery"

Another attack (DOS)

DOS attacks are popular these days, eg. Anonymous group using LOIC (Low Orbit Ion Cannon) network stress test tool

- SYN-flooding
  - A sends many $\mathrm{SYN}(N_A)$ packets (message 1), without responding to the second message
  - B reserves some memory for N_B, with each (expected) connection
  - B runs out of memory, at some stage, and falls over
- Crypto-based defence (by Dan Bernstein, see `http://cr.yp.to`)
  - make $N_B = K\{N_A\}$
  - as result, B need not keep state
  - implemented in Linux

More DOS attackes, via amplification

- A broadcast ip address reaches many machines
- it can be abused for a smurf DOS attack:
  - send a ping message to such a broadcase address
  - put the victim's address as (forged) source of this ping
  - this victim will receive ping-replies from all machines reachable via the broadcast address
- Most routers have now been reconfigured to obstruct such attacks
- A fraggle attack is similar, but uses UDP packets

Security at different layers

- Hardening the infrastructure
  - Ideally invisible to the user
  - IPSec (often used for VPNs), DNSsec
- End-to-end security:
  - on top of (insecure) infrastructure
  - can be built into applications, but often requires explicit action
  - SSL/TLS, ssh/scp/sftp, PGP, S/Mime, Tor, etc.

Protocol example: Pretty Good Privacy (PGP)

- PGP provides:
  - confidentiality, via symmetric (IDEA) encryption with session key
  - integrity, authenticity and non-repudiation via (RSA) signature
- As single protocol line, for plaintext m

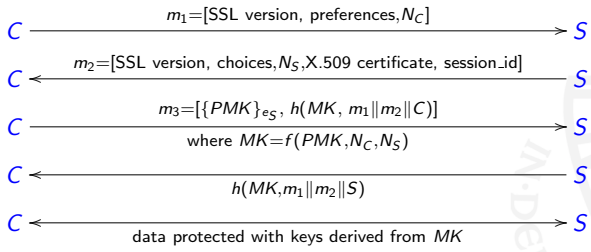
$$A \longrightarrow B : \{K\}_{e_B},\ K\{zip(m, [hash(m)]_{d_A})\}$$

- Recent unsigned data injection vulnerability found by Verheul
  - due to sloppy parsing by PGP Desktop (versions 8-10)
  - see paper on the web for more details
  - demonstrates that critical review of open source software can take a long time ...

Protocol example: Secure Socket Layer (SSL)

- SSL was introduced in the Netscape browser in 1995
  - became an open (slightly different and incompatible) standard "TLS" but is still most frequently used as SSL.
- SSL builds a secure connection between two sockets, including
  1. Parameter negotiation between client and server
  2. One-way (by server) or two-way authentication
  3. confidentiality & integrity
- SSL consists of two subprotocols:
  - establishing a secure connection
  - using it

One-way Client-Server SSL-authentication



- Preferences are of the form 'SSH2-3des-cbc-hmac-sha1'
- PMK is 384 bit PreMaster Key, generated by C; MK is Master Key
- session_id is used for resuming (web) connections

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
- I hope I conveyed some of that enthusiasm. ☺