5. Security Controls

Prof. Dr. Marc Rennhard
Institut für angewandte Informationstechnologie InIT
ZHAW School of Engineering
rema@zhaw.ch

Content

This chapter provides an overview of security controls that are likely to be relevant when developing secure applications; most of them were discussed in detail in the course Internet-Sicherheit (ISI):

- Cryptographic primitives
- User authentication
- Secure communication protocols
- Authorization
- Firewalls
- Intrusion detection / prevention systems

- For additional information refer to the corresponding ISI course material

Further Security Controls?

Of course, there are even further security controls, but we focus on the most important here.

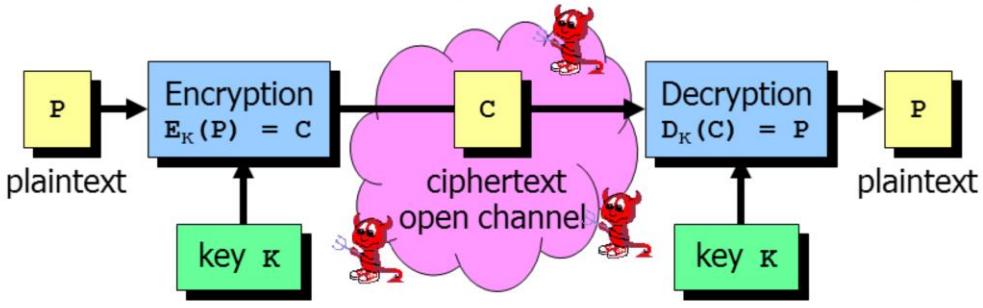
Goals

- You have an overview of a variety of security controls that are relevant when developing secure (distributed) systems
- You can describe the functionality of each security control
- You know the application areas of the different security controls
- You know the strengths and limitations of each security control

Cryptographic Primitives

Secret Key Ciphers (1)

- Secret key ciphers serve to encrypt data to achieve confidentiality



- Popular ciphers: AES (block cipher), RC4 (stream cipher)
- The key must be secretly exchanged beforehand
- The key length should be at least 128 bits
- As always in cryptography, it is extremely important that the key material is picked "as randomly as possible"

Secret Key Ciphers (2)

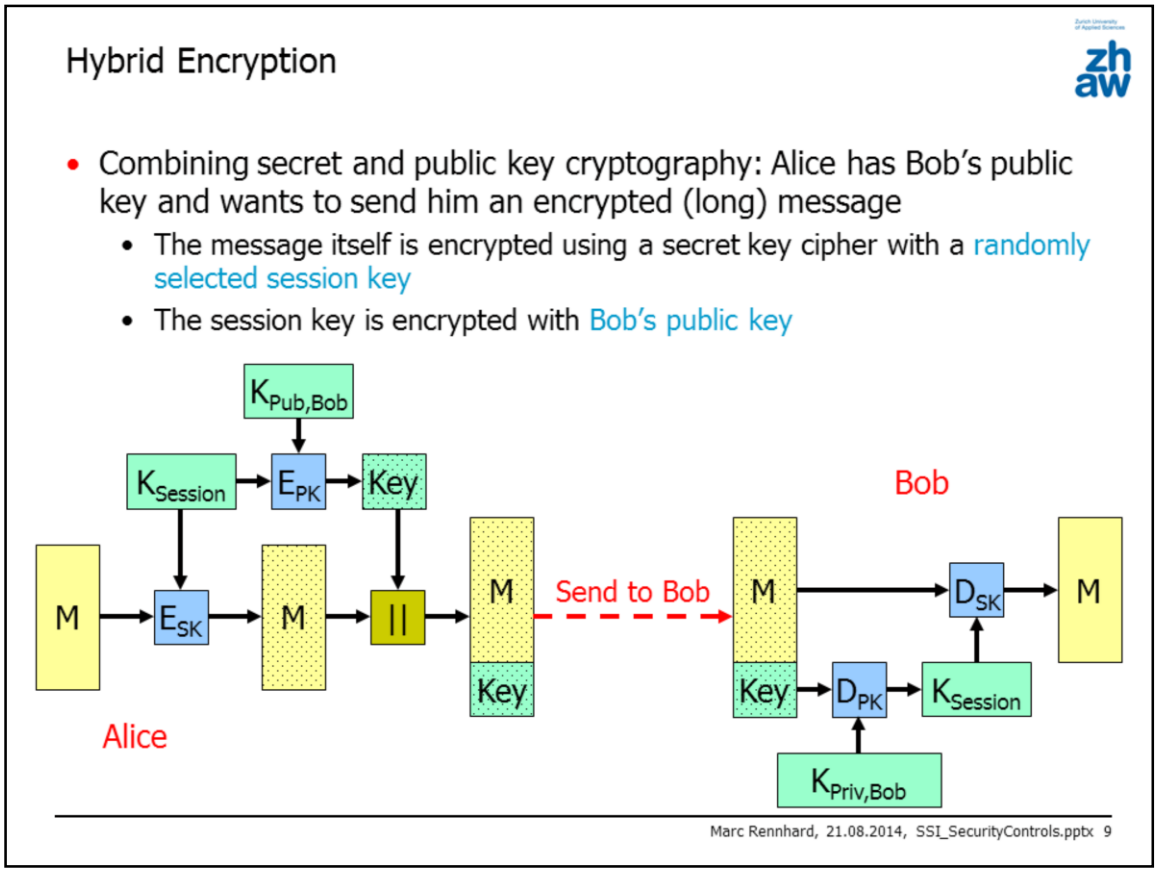
- Applications:
  - As a building block in secure communication protocols to protect "data in motion"
  - To protect "data at rest", e.g. disk encryption
- Strengths:
  - Strong and widely distributed standards (e.g. AES) available
  - Very fast, even on small devices feasible
- Limitations:
  - Only provide encryption but do not solve the key exchange problem
  - When used for "raw encryption", secret key ciphers only provide confidentiality, but attacks on integrity are still possible → combine with appropriate integrity-protection

Public Key Ciphers (1)

- In contrast to secret key cryptography, two different keys are used for encryption and decryption
  - To encrypt a message, the public key is used
  - To decrypt a message, the private key is used
  - Corresponding public and private key belong together: key pair
- A person Alice can generate a key pair and publish her public key
  - Other persons can take Alice's public key and encrypt a message for her
  - Only Alice can decrypt these messages using her own private key
  - Decryption of a message using the public key is not possible
- Popular public key algorithms:
  - RSA (use keys with a modulus size of at least 2'048 bits)
  - Less frequently used: Elliptic Curve Cryptography (ECC)

Public Key Ciphers (2)

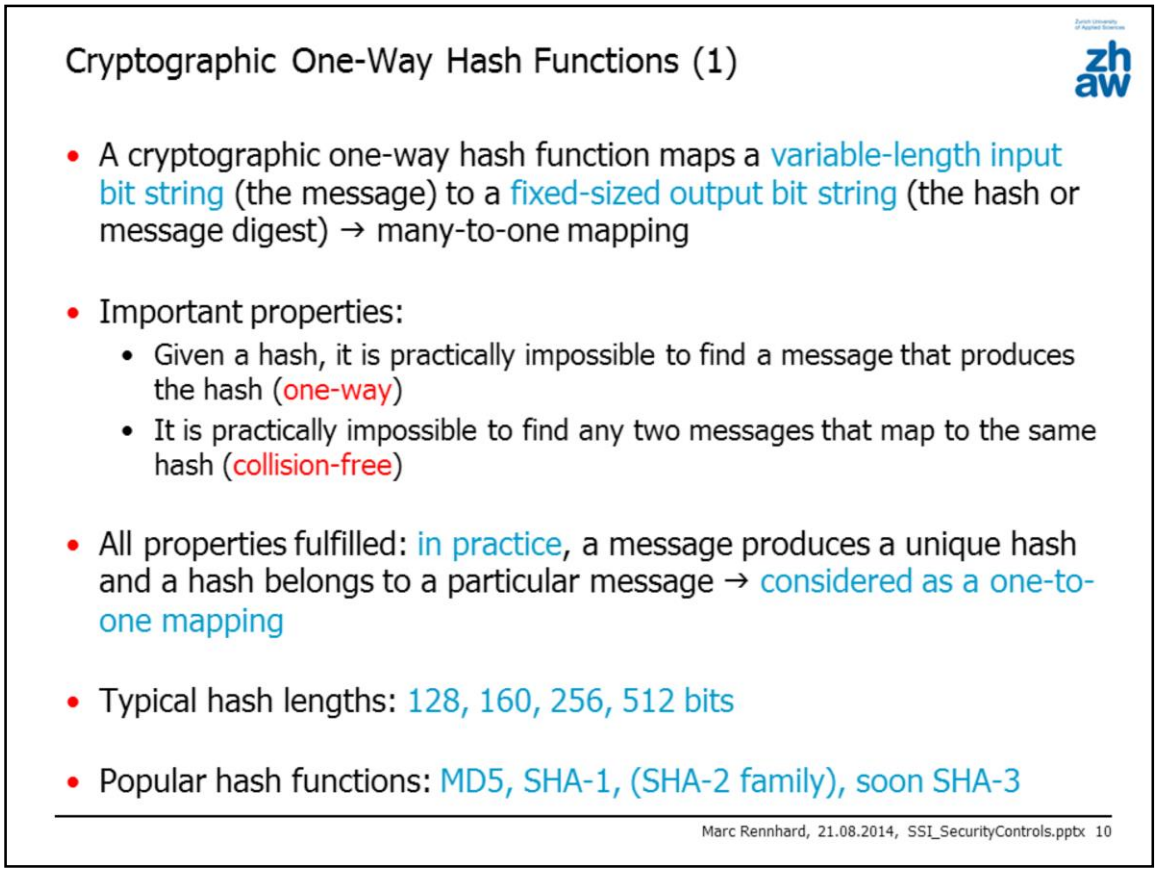
- Applications:
  - As a building block in secure communication protocols to exchange a secret key
- Strengths:
  - Strong and widely distributed standards (e.g. RSA) available
  - Solves the key exchange problem of secret key cryptography (see hybrid encryption), assuming the authenticity of the public key can be verified
- Limitations:
  - Much slower than secret key cryptography, not suited for bulk data encryption



Hybrid Encryption

Hybrid encryption works with every public key cryptosystems that allows public key encryption. The name "session key" is used because the randomly selected secret key is only used to encrypt and transmit a single message, i.e. only for "this session". We will meet the name "session key" again later when discussing secure communication protocols, where session keys will be used to protect a "communication session", e.g. the data exchanged over a TCP connection.

Since secret key operations are much faster than public key operations, hybrid encryption means a significant performance gain compared to pure public key bulk encryption. Only the short (e.g. 128-bits long) symmetric key is encrypted using public key cryptography, which – in the case of RSA – is smaller than the modulus and therefore only requires a single RSA encryption (one modular exponentiation), which is insignificant regarding performance. One popular place where hybrid encryption is used is e-mail security (see chapter 10), where the potentially very long e-mail message is encrypted with a symmetric session key and only the session key is encrypted with the public key(s) of the recipient(s).



Hashes / Message Digests

A hash of a fixed size acts as a **unique fingerprint** for an arbitrary-sized message, document or packed software distribution file. With a common digest size of 128 .. 256 bits, about 10^{38}…10^{77} different fingerprint values can be represented. If on every day of the 21th century 10 billion people wrote 100 letters each, this would amount to $3.65 \cdot 10^{16}$ documents, only. So if each of these letters had its individual fingerprint, only a tiny percentage of all possible values would be used.

One-Way Hash Functions

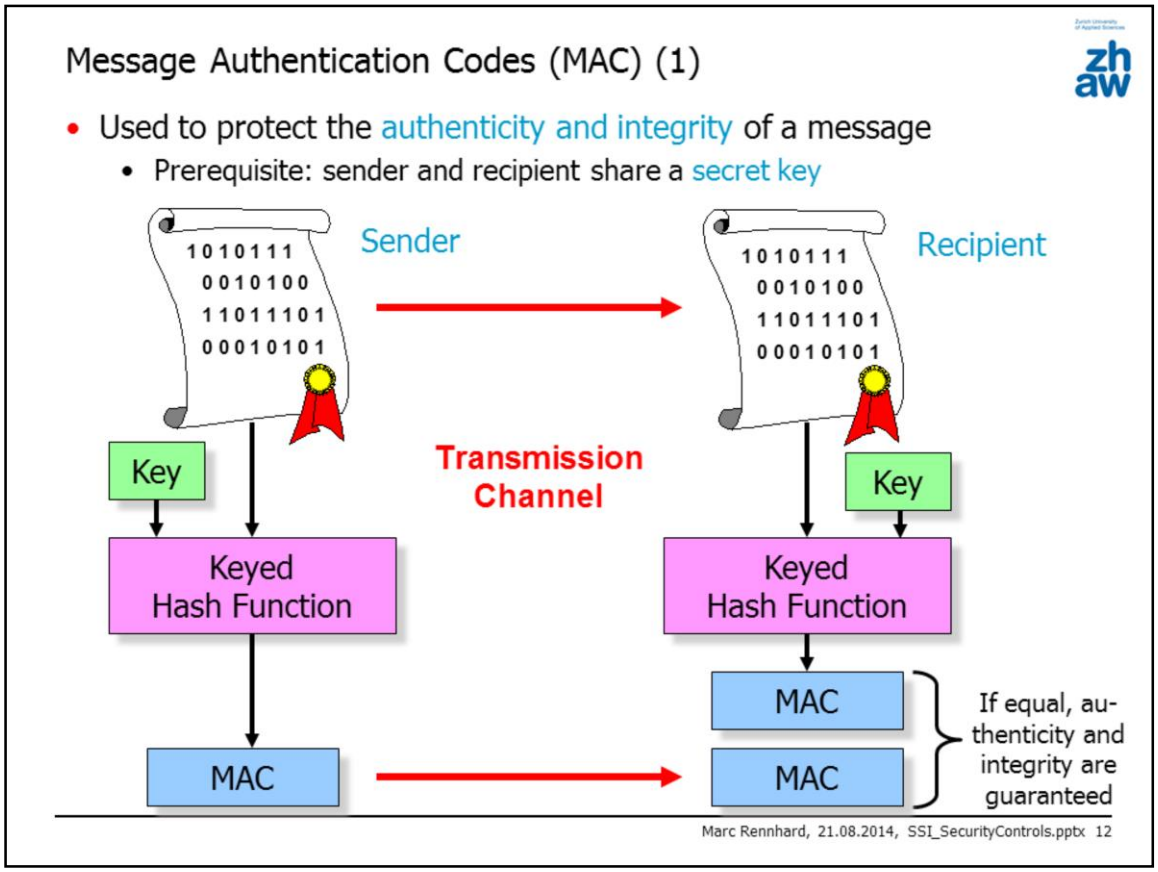
For the computation of hashes special **one-way hash functions** are used. A good hash function should have the following properties:

- The computation of hashes should be fast and efficient, allowing the hashing of messages several gigabytes in size. Since a document is usually much larger than its hash value, the mapping is a many-to-one function. For each specific hash value there potentially exist many documents possessing this fingerprint.
- The message digest value should depend on every bit of the corresponding message. If a single bit of the original message changes its value, or one bit is added or deleted, then about 50% of the digest bits should change their values in a random fashion. A good hash function achieves a pseudo-random message-to-digest mapping, causing two nearly identical messages to have totally different hash values.
- It should be practically infeasible to find a document that produces a given fingerprint. This is why a good hash function is called **one-way**.
- It should be practically infeasible to find any two different documents that produce the same fingerprint. This is called **collision-free**.

If all these properties are fulfilled, it is quite impossible that two distinct messages will ever produce the same digest value. We say a message produces a unique hash because of the collision-freeness property. And we say a hash belongs to a particular message because of the one-way property. So for all of today‘s practical applications we can regard the output of a good hash function as a **quasi-unique fingerprint** of the hashed message.

Cryptographic One-Way Hash Functions (2)

- Applications:
  - As a building block in secure communication protocols to protect the integrity of messages (typically together with a secret key → MAC)
  - As a building block for digital signatures
  - To compute file integrity checks to detect data tampering (hash must be stored offline so it cannot easily be tampered with)
  - To store hashed (and salted) passwords
- Strengths:
  - Very fast, even on small devices feasible
- Limitations:
  - During the recent years, security vulnerabilities were found in MD5 and SHA-1 – which is a problem as no widely accepted alternative is available
  - As a result, a SHA-3 contest (similar to AES) was started by NIST in 2007, the winning algorithm Keccak was announced in October 2012



Message Authentication Codes

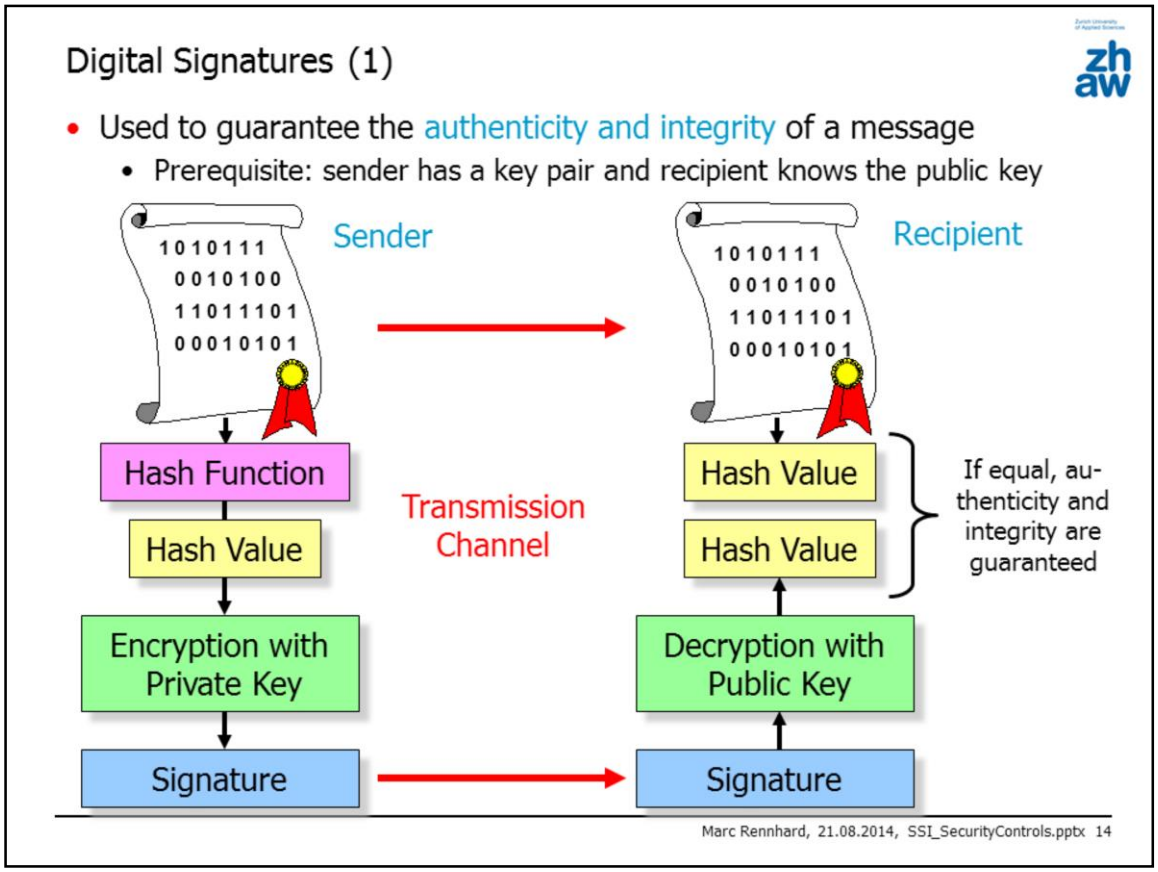
A digital message digest in itself does not offer any protection against unauthorized modifications of a message or document. After any change to a document, a new valid MD5 or SHA-1 hash value could be computed on the new content, since the hash algorithms in use have been published and are well documented.

Only by introducing a secret key into the fingerprint computation a document can be secured against unauthorized modifications. Only the owner(s) of the secret key can produce a valid message digest which is now called a **Message Authentication Code** (**MAC**). Of course the recipient of the secured document must possess the secret key in order to verify the validity of a message by also computing the MAC value and comparing it to the MAC transmitted or stored together with the corresponding document.

In the literature, a MAC is also sometimes called **Message Integrity Check (MIC)**. A MAC and a MIC are the same.

Message Authentication Codes (MAC) (2)

- Standards: HMAC (based on keyed hash function), CBC-MAC protocol (based on secret key cipher)
- Applications:
  - As a building block in secure communication protocols to protect the authenticity and integrity of messages
  - To compute file integrity checks to detect data tampering (key must be stored offline)
- Strengths:
  - Very fast, even on small devices feasible
- Limitations:
  - Often based on MD5 and SHA-1 and therefore relying on partly broken algorithms



Digital Signatures based on Public Key Cryptosystems

The hash value of a message or document is encrypted by the author using his private key, thereby forming a digital signature that is transmitted or stored together with the corresponding document.

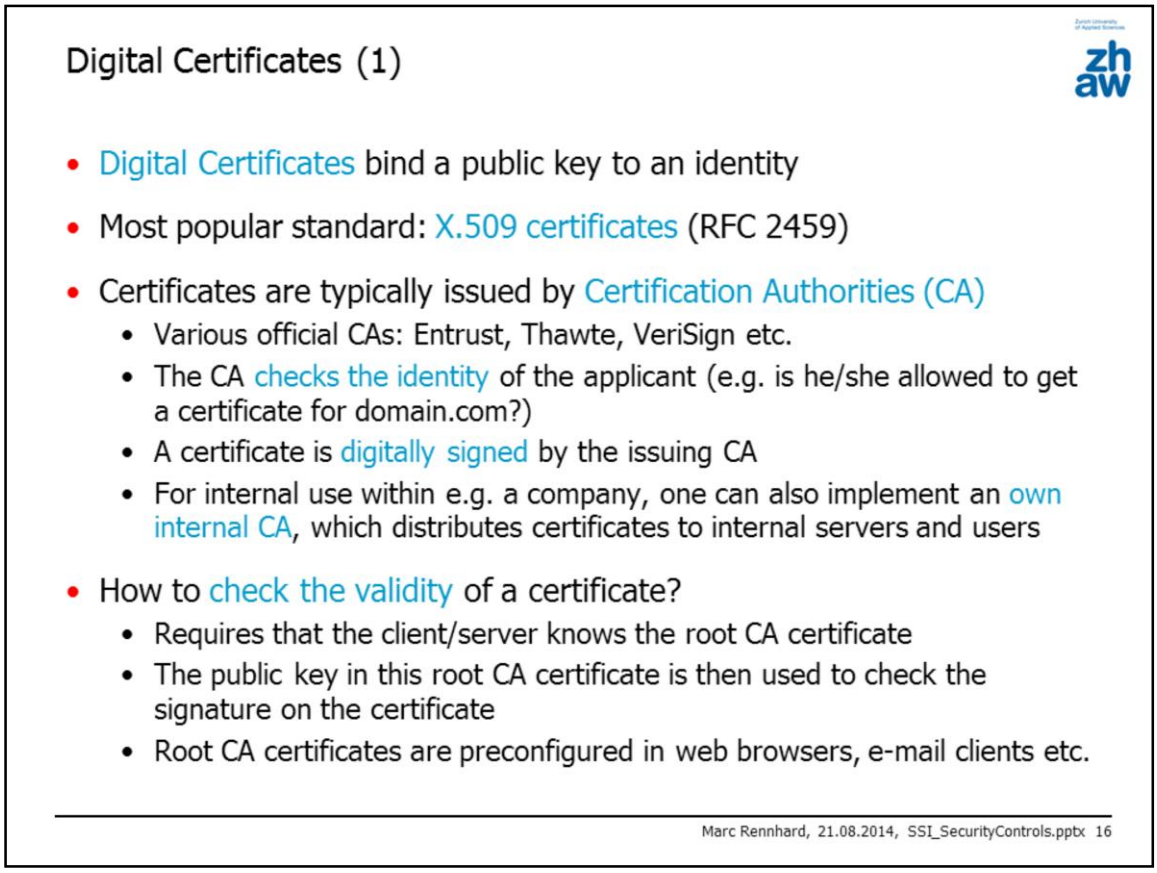
The recipient computes the hash over the received document and decrypts the received signature, using the public key of the document's author. If the decrypted value equals the computed hash value then the document must be authentic, since only the author possesses the correct private key used to encrypt the signature. In addition, this also guarantees the integrity of the document because if anyone had changed the document in transit, the signature would no longer match.

Comparison of MACs and digital signatures:

- **MACs based on Secret Keys**
  - **Pro**: Very fast and efficient, since MAC generation is based on simple hashing functions. Often used for integrity protection and authentication (see later) of bulk data at high data rates (e.g. applied to IPsec datagrams and TLS).
  - **Contra**: Recipient must know the secret key in order to verify the integrity and authenticity of a message. This often leads to a secure key distribution problem.
- **Digital Signatures based on Public/Private Key Pairs**
  - **Pro:** Public keys can be freely and openly distributed, so anyone (not only the communicating partners) can check the authenticity and integrity of a message.
  - **Contra**: Encryption and decryption operations using private and public keys, respectively, are extremely time-consuming. Used for user or host authentication at the beginning of longstanding sessions or for signing individual e-mail messages

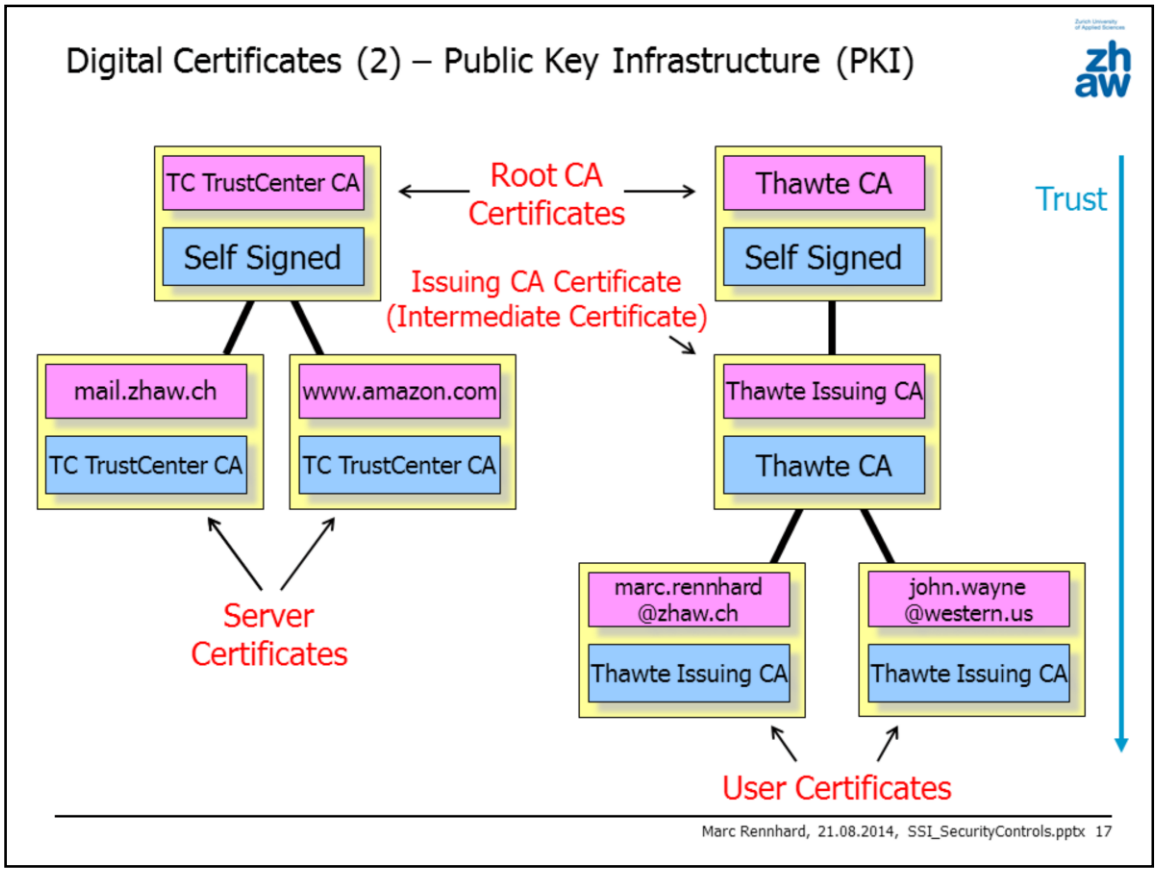
Digital Signatures (2)

- Standards: RSA (dominating), DSA, ECDSA
- Applications:
  - As a building block in secure communication protocols during initial authentication of the endpoints
  - Digitally signing e-mails to protect the authenticity and integrity
  - Digitally signing of documents or contracts (legally binding)
- Strengths:
  - Enables strong authentication
  - Legally binding signatures can significantly increase the efficiency of workflows
- Limitations:
  - Private key must be very well protected as it often has a long lifetime
    - In particular, private key should not be stored in plaintext on user computers
    - Ideally, the private key should be stored on dedicated secure hardware (smartcards) → costs, requires software installation
  - Smartcards can usually not be used on mobile devices (e.g. smartphones)



Certificate Types

The certificates we discuss here are ITU-T X.509 certificates (RFC 2459). They are the primarily used type when dealing with secure and authenticated Internet communications.



Root Certification Authorities - Root CA

At the top of the hierarchical trust chain are a few Root Certification Authorities which are well known and which intrinsically must be trusted. Each Root CA publishes a **Certifcate Practice Statement** (CPS) defining on what policies user or server certificates are issued, how they are managed and how they can be revoked. A Root CA has a key pair; the private key is used to issue certificates for others. Root CAs also issue a certificate for themselves (root certificates), which are just like normal certificates but which are self-signed by the root CA. This means that a root certificate contains the public of the CA and is signed by the own private key of the CA. The root certificate of a CA (especially the enclosed public key) is used to verify the certificates of users, servers and intermediate CAs (see below) that received their certificates from this CA.

CAs build Trust

The task of a CA is a very important one: they build trust in certificates (i.e. that a public key belongs to a particular identity) and they really must make sure that they themselves are operated extremely securely and that they only issue certificates if they are convinced the requestor of the certificate is legitimate to get a certificate for the requested name. If it turns out that CAs often issue certificates to imposters, people will no longer trust the entire concept of CAs issuing certificates. There was one case in 2001 where an imposter managed to obtain two certificates in the name of Microsoft to sign software from VeriSign, but such cases happened very rarely in the past.

Examples of well-known Root CAs are VeriSign, RSA, Baltimore, Entrust, Thawte, TC TrustCenter, Deutsche Telekom etc.

Intermediate Certification Authorities - Intermediate CA

Root CAs can directly issue user or server certificates. Another option is to issue certificates (so called intermediate certificates), that can itself be used to issue further certificates. This results in longer chains between the Root CA and the user/server certificates because a user verifying such a user/server certificate must then be able to build the whole chain up to the Root CA.

Intermediate CAs are often used as Issuing CAs. Like in the Thawte-example above, the Root CA never directly issues user/server certificates, but delegates this task to another internal CA and its certificate. The advantage of this approach is that the very critical private key of the Root CA must be used only relatively rarely (often only to sign Issuing CA certificates and CRLs) and does not have to be accessible by an online system that is connected to the Internet. Only the private key of the Issuing CA must be online on a (hopefully hardened) server to automate the process of issuing user/server certificates. If it happens for any reason that the private key of the Issuing CA is compromised, the private key of the Root CA is still available to, e.g. revoke the Issuing CA certificate and to sign a new Issuing CA certificate. The disadvantage is that more certificates are required to build the entire trust chain up to the Root CA certificate to validate a user/server certificate. In practice, this is solved as follows: software that checks certificates (e.g. a browser or e-mail client) still only needs the pre-installed Root CA certificate (see next silde) and a user/server that presents its certificate to this software not only sends its own certificate, but also all intermediate certificates up to (but without) the Root CA certificate. This allows to verify the software the entire trust chain.

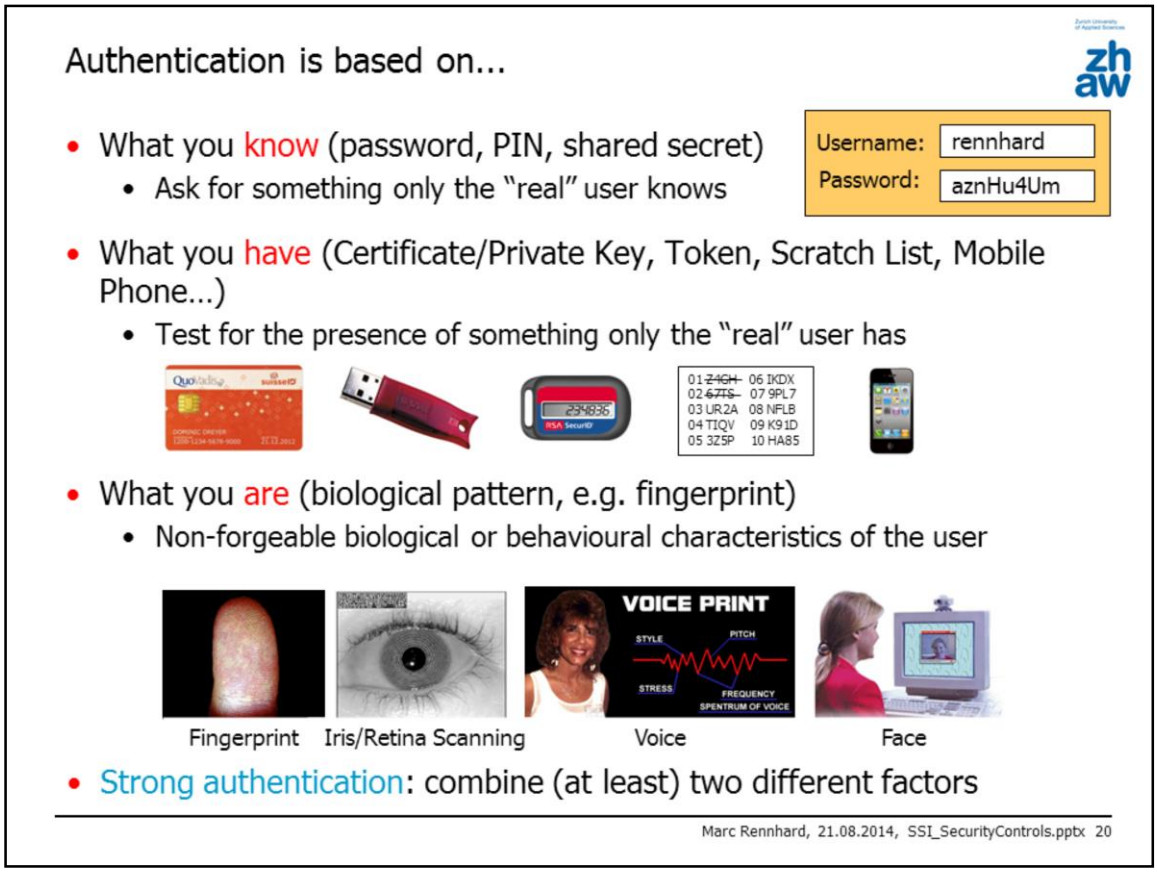
Public Key Infrastructure

This trust hierarchy is also often called Public Key Infrastructure (PKI).

Digital Certificates (3)

- Applications:
  - Basically whenever public key cryptography is used and when the public key must be bound to an identity, e.g. e-mail signatures or server authentication in SSL/TLS
- Strengths:
  - Solves the problem of binding a public key to an identity, which is a fundamental requirement to enable secure communication
  - Certificate-checking is usually transparent for the user, as root CA certificates are preconfigured in certificate-aware software
- Limitations:
  - Correct usage requires checking for revoked certificates via OCSP or CRLs, which are sometimes disabled per default
  - The security of many security standards depends on certificates, which makes them attractive targets
  - Successful attacks against CAs have happened (compromising CA computers, getting a certificate with a wrong identity...)

User Authentication



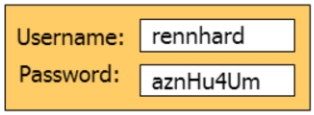
Three Ways to Authenticate the Identity of a User

- The users present something they **know**, such as a password. This approach is known as a **Knowledge factor**. Passwords and PINs (PIN = personal identification number) are the most common method of using confidential knowledge to authenticate users. Easy to administrate and convenient for most users, passwords are also the least expensive method of user authentication. Unfortunately, passwords have some drawbacks. Often, user-selected passwords are very short and simple, which makes them easy to guess.
- The user presents something (physical) they **have** in their possession, such as a key or a card. This approach is known as a **Possession factor**. To authenticate users digitally people provide them with tokens that contain a digital code. Tokens are available as both hardware and software. They may generate a different code within regular time intervals or upon request (e.g. upon reception of a „challenge“). These tokens may also be smart cards, similar in size to a standard credit card which is inserted into a card reader as part of the authentication process. They may contain a digital certificate and they are usually presented in combination with a password or PIN.
- The user presents a personal physical attribute (what they **are**), such as a fingerprint or a retinal scan. This approach is known as a **Being factor**.

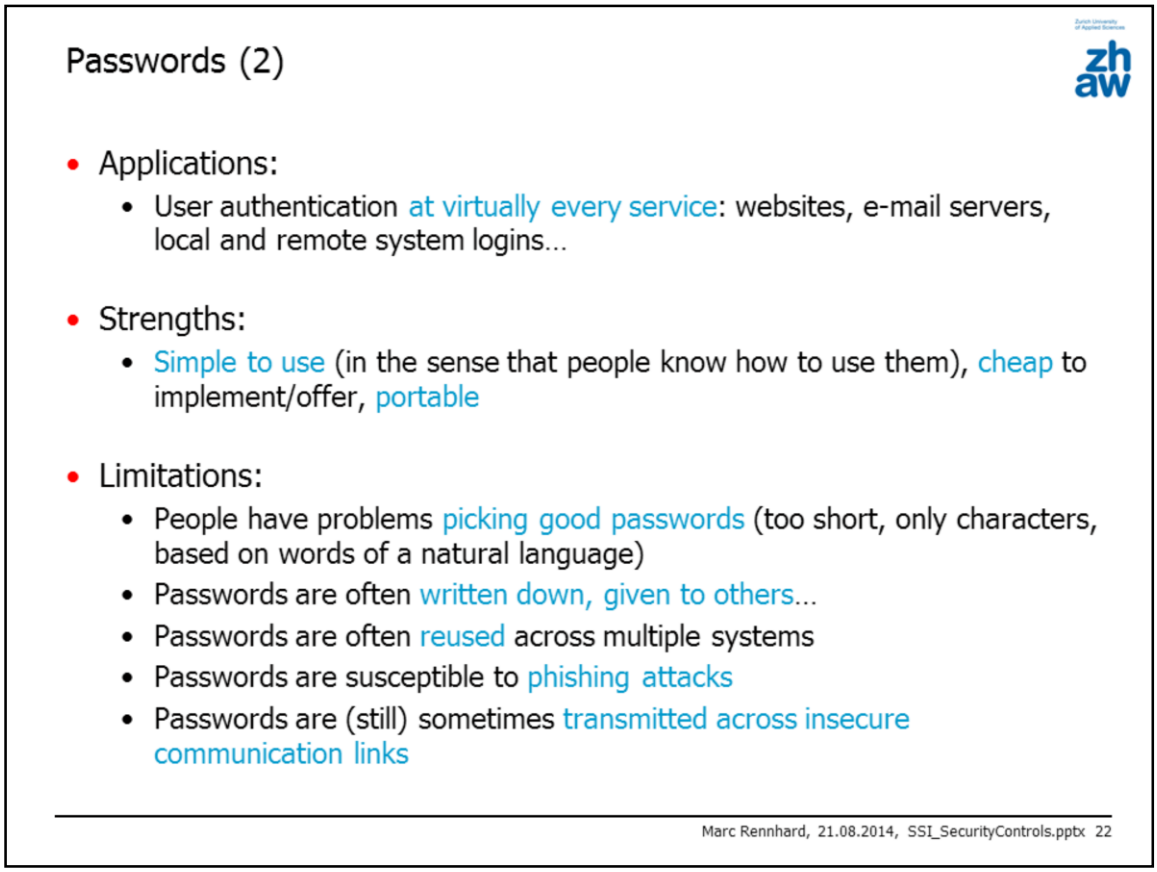
Source: Prof. Dr. P. Heinzmann, Hochschule für Technik Rapperswil

Passwords (1)

- Very popular, easy to implement and use
  - In particular: no additional client-side soft- or hardware is needed, which is one main reason they are still so widely used



- Basic principle:
  - Every user of the system has a unique user id and a password
  - The server stores all user ids and their passwords in a file or a database
  - To authenticate, the user submits user id and password to the server; the server compares the submitted data to the stored data
  - To increase protection from password theft, passwords should not be stored on the server in plaintext
    - Passwords should be salted and hashed



Dictionary Attack

A dictionary attack refers to the general technique of trying to guess some secret by running through a list of likely possibilities, often a list of words from a dictionary. It contrasts to a brute force attack in which all possibilities are tried. The attack works because users often choose easy-to-guess passwords.

One-time Passwords

- Idea:
  - Basically similar to passwords, but a one-time password can only be used once
  - Implementations: scratch lists, electronic tokens (time- or event-based), SMS messages from the service provider

01 ~~Z4GH~~	06 IKDX
02 ~~67TS~~	07 9PL7
03 UR2A	08 NFLB
04 TIQV	09 K91D
05 3Z5P	10 HA85

Datum und Uhrzeit letztes Login: 09.10.2013 20:11
mTAN für neues Login: 147245

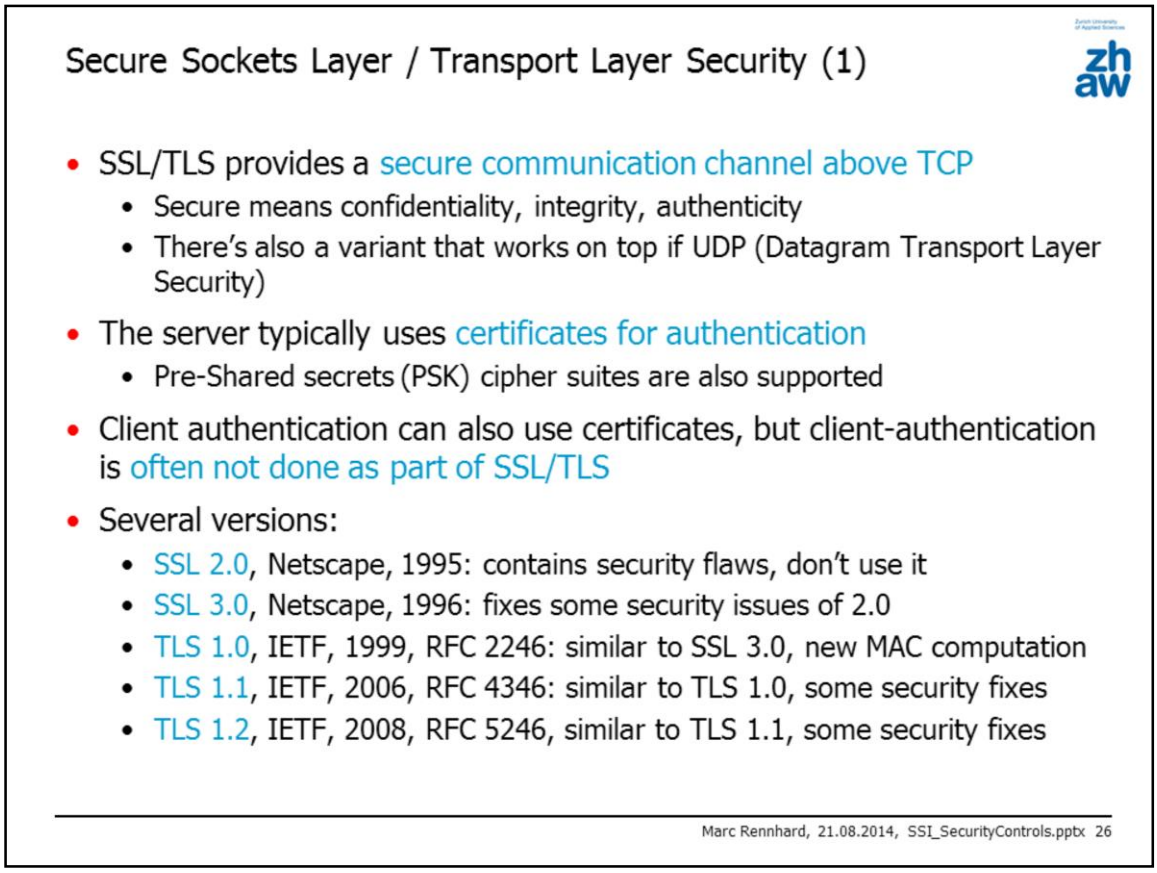
- Applications:
  - Usually in combination with a "normal" password to achieve more secure user authentication (compared to password-only) → two-factor authentication (which is regarded as strong authentication)
- Strengths:
  - Simple to use, "paper-versions" are relatively cheap, portable
- Limitations:
  - Tokens and SMS messages are relatively expensive

User Certificates

- The basic applications / strengths / limitations were already discussed in the section about digital signatures
  - Because when using user certificates for user authentication, a digital signature is usually performed

- Just one additional remark with respect to strong authentication
  - If the private key is stored on a smartcard, then this corresponds to (strong) two-factor authentication solution
  - Because one must both possess the smartcard (something you have)...
  - ...and also know the PIN (something you know) that is usually needed to access the smartcard to perform private key operations

Secure Communication Protocols



Documentation of SSL/TLS

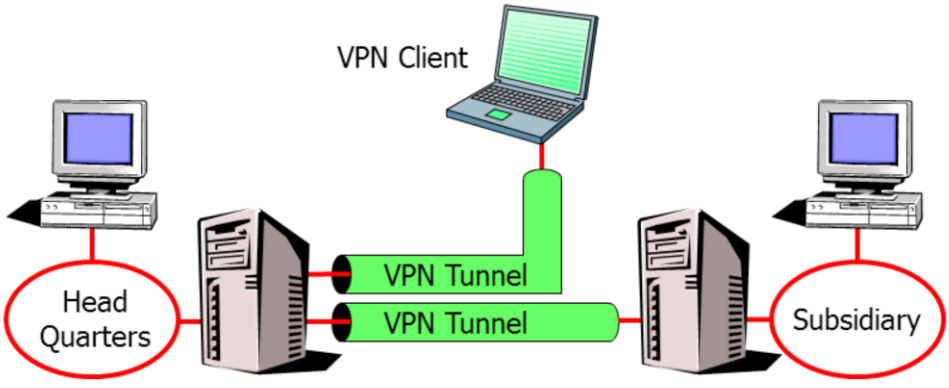
- SSL 2.0: Kipp Hickman. The SSL Protocol. Netscape Communications Corp., Feb 9, 1995.
- SSL 3.0: A. Frier, P. Karlton, and P. Kocher. The SSL 3.0 Protocol, Netscape Communications Corp., Nov 18, 1996.
- TLS 1.0: T. Dierks, C. Allen. The TLS Protocol Version 1.0. RFC 2246, January 1999. (The protocol format and message flow is virtually unchanged compared to SSL 3.0, but there are some major changes regarding "helper functions" such as pseudo random number generators and how the MAC of messages is generated)
- TLS 1.1: T. Dierks, E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1, RFC 4346, April 2006 (TLS 1.1 is only a very small update of TLS 1.0. In the beginning of the RFC, there's the following statement that undermines this: *This document is a revision of the TLS 1.0 protocol, and contains some small security improvements, clarifications, and editorial improvements.*). This version also fixed the BEAST attack.
- TLS 1.2: T. Dierks, E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2, RFC 5246, August 2008

Secure Sockets Layer / Transport Layer Security (2)

- Applications:
  - To secure many TCP-based application protocols: https, pop3s, imaps, smtps…
  - As a basis for OpenVPN to build user-space virtual private networks
- Strengths:
  - Well-established and analyzed
  - Runs in the user space and can therefore be easily integrated in application software (web, e-mail,...)
- Limitations:
  - Some attacks have been published in the past years
    - Renegotiation attack, BEAST, CRIME, BREACH, Lucky 13 (all are difficult to carry out in practice)
  - TLS 1.1/1.2 fix some of them, but servers mostly support only TLS 1.0
    - All modern browser now support TLS 1.2, but sometimes disabled per default

IPsec (1)

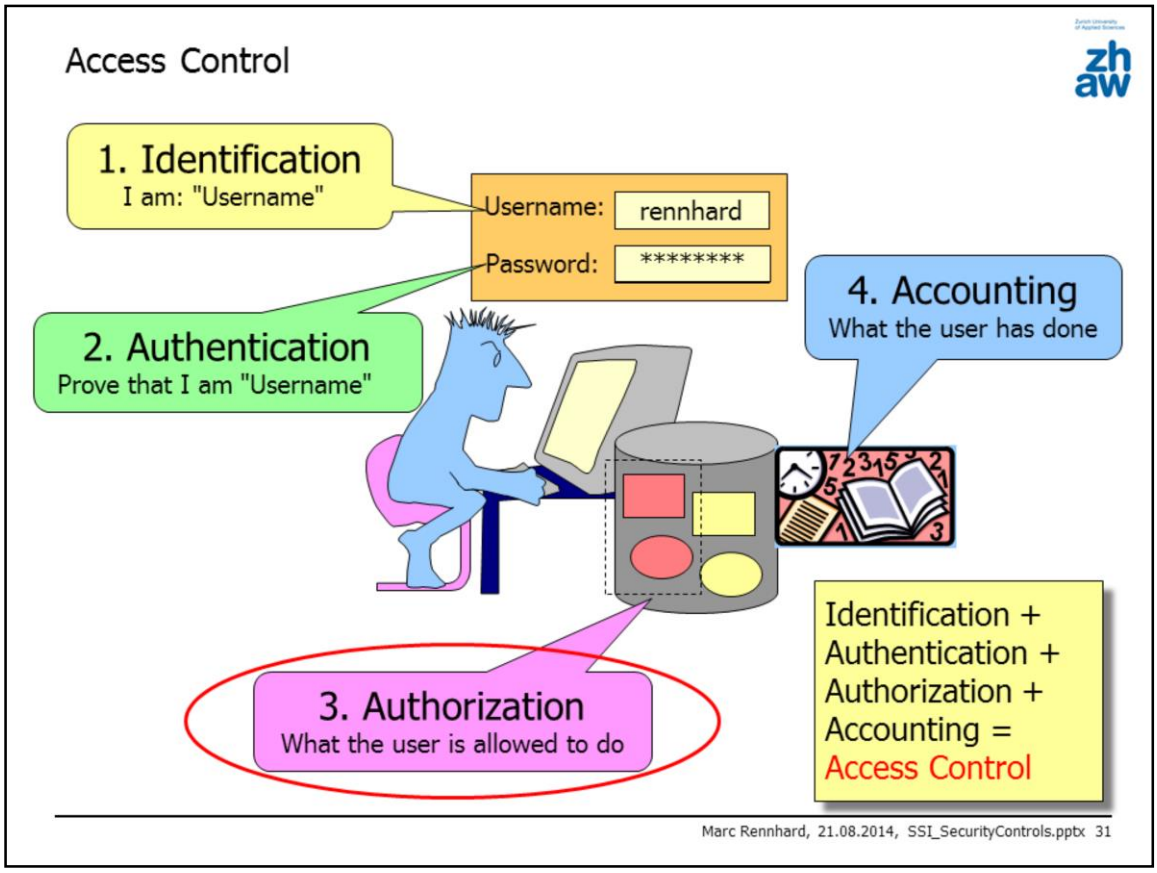
- IPsec provides a secure communication channel above IP
  - Secure means confidentiality, integrity, authenticity
- Several authentication methods are supported
  - In practice, certificates and PSK are dominant
- Can be used end-to-end (transport mode) or for VPNs (tunnel mode), tunnel mode is clearly dominant



IPsec (2)

- Applications:
  - To provide VPNs between remote networks
  - To provide secure access to networks for mobile users
- Strengths:
  - Well-established and analyzed
  - Supported by a wide spectrum of systems, including mobile platforms (smartphones)
  - Protection includes transport layer header and – in tunnel mode – internal IP addresses
- Limitations:
  - Over-engineered protocol, much more complex than it would have to be
  - Runs in the kernel space and therefore require support by the OS
  - OpenVPN and other, proprietary VPN solutions are slowly replacing IPsec

Authorization



Identification

Identification establishes who you claim to be: The user claims an identity, usually by supplying a user ID or a user name.

Authentication

Authentication verifies that you are who you claim to be: The user supplies authentication information, which proves the binding between the user and the identity.

Authorization

Authorization establishes what you‘re allowed to do e.g. which files and applications you may access: The systems authorizes the (authenticated) user to do what he is allowed to do.

Accounting

Accounting performs the bookkeeping about what you do.

Source: Prof. Dr. P. Heinzmann, Hochschule für Technik Rapperswil

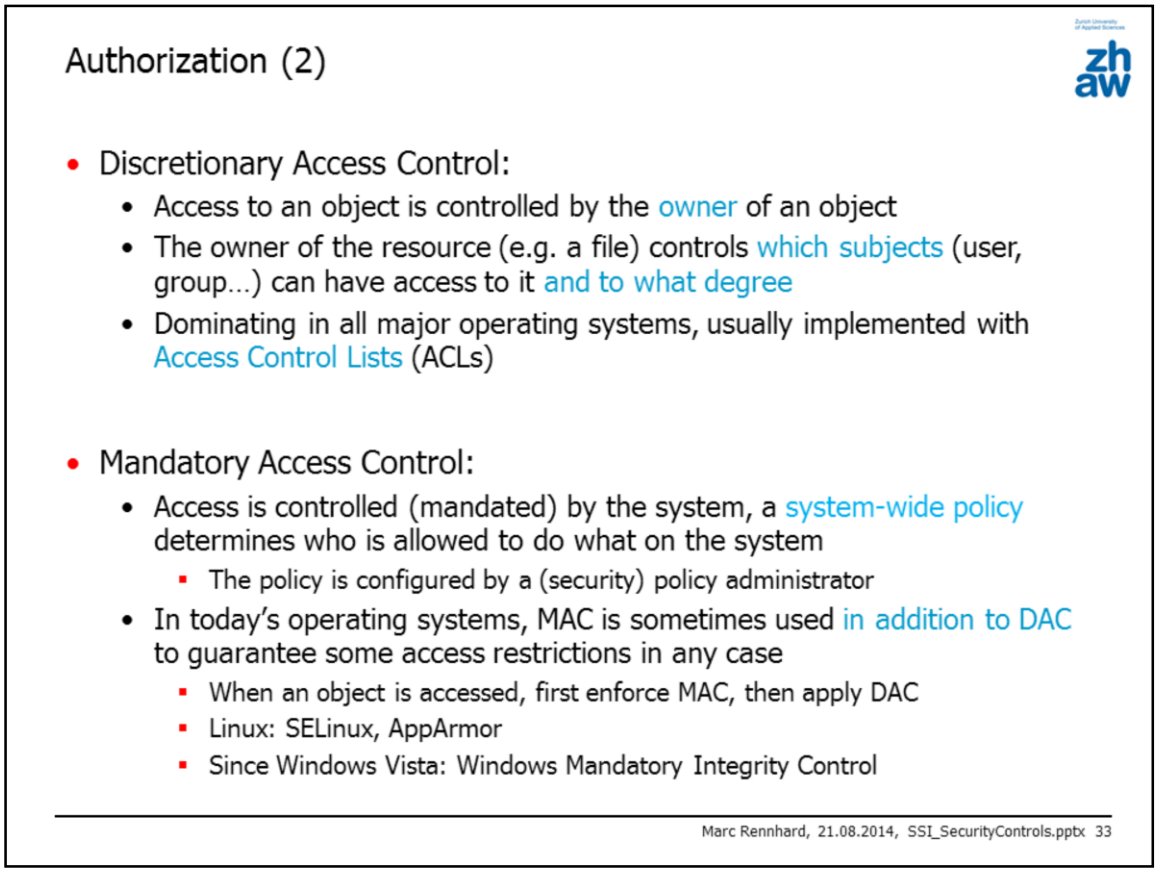
AAA → Access Control

The triple Authentication/Authorization/Accounting is often abbreviated as **AAA**. Together, they provide complete **Access Control**, i.e. controlling and enforcing that only authenticated users can access the resources they have been authorized to use and make sure that relevant activities of these users logged (accounting).

Authorization (1)

- Authorization is primarily used by operating systems and applications to determine what a particular user is allowed to do

- There exist three models for authorization:
  - Discretionary Access Control (DAC)
  - Mandatory Access Control (MAC)
  - Role-Based Access Control (RBAC)



Discretionary Access Control

The term is used because control of access is based *on the discretion* of the owner of an object.

MAC and DAC

When both are used, the MAC settings usually have precedence over the DAC settings.

Authorization (3)

- Limitations of DAC and MAC
  - Both are very technical and focused to protect access to low-level objects and as a result, they are mainly used in operating systems
  - For applications, this is less well suited, as access rights in applications are usually concerned with user groups that are allowed to do transactions
  - User groups and transactions in an e-shop:
    - Anonymous users are allowed to browse goods
    - Registered customers are allowed to buy goods
    - Marketing personnel may define discounts
  - For such scenarios, Role-Base Access Control is usually better suited
- Role-Based Access Control:
  - The rights of a user depends on the roles that are assigned to him
  - Typical approach works as follows:
    - First define suited roles that contain the corresponding rights (transactions)
      - Roles in an e-shop: customer, sales, marketing, administrator...
    - Assign the roles to users, which grants the user the rights of his roles
    - During runtime, enforce that a user has only the rights according to his roles

Firewalls

Packet Filtering Firewalls (1)

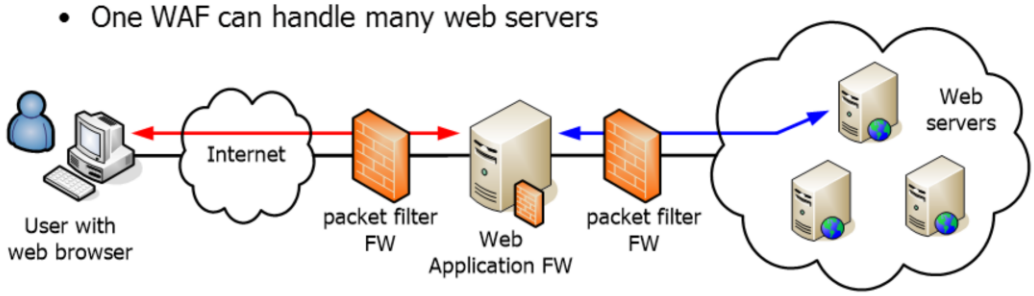
- A packet filtering firewall sits between two or more networks to control the packet flow between them
  - The only possibility for packets to travel from one network to another is through the firewall (which may be replicated, though)
- They usually inspect the data in the network and transport protocol headers
- Depending on configured rules, the traffic is forwarded or blocked
- A typical rule could be as follows:
  - Allow traffic from any host on the network 160.85.37.0/24 to port 80 on host 160.85.215.20, but block access to other ports

Packet Filtering Firewalls (2)

- Applications:
  - To filter traffic between connected networks
- Strengths:
  - Well-understood, easy to configure, usually relatively static rule set
  - Blocks a lot of unwanted traffic before it enters an environment
  - Provides a centralised point to control the allowed packet flows, which is much simpler than controlling this at every individual host
- Limitations:
  - Can only control which systems are allowed to communicate, but not the content of the communication

Web Application Firewalls (1)

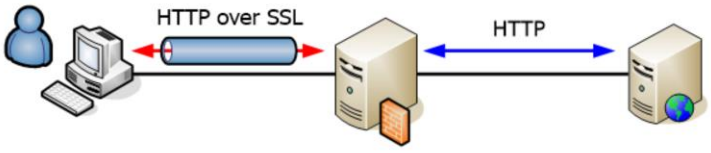
- Web application firewalls (WAF) filter traffic at the application layer and are optimized to protect web applications
- Most frequently, a WAF is operated as a reverse proxy
  - Additional packet filtering firewalls guarantee that web servers cannot be reached directly
  - The web browser connect to the WAF, which relays the traffic to the desired web server
  - Depending on the configured rules, the WAF decides whether to forward traffic or not
  - One WAF can handle many web servers



Web Application Firewalls (2) – Functionality

- Content filtering and adaptation
  - Content filtering: e.g. prevent access if a request contains possible attack data (Javascript code, SQL statements...)
  - Content adaptation: e.g. prevent information leakage by replacing replies that contain detailed error messages with a generic error message

- SSL/TLS termination: the protected SSL/TLS channel is terminated at the WAF
  - This is a prerequisite to allow access to non-encrypted data
  - This enables server certificate management at a centralised place (and not on every individual web server)



Web Application Firewalls (3) – Functionality

- URL encryption
  - The WAF encrypts the paths of all URLs in the HTML page are encrypted with a session-specific key before sending it to the browser
    - So http://www.mybank.com/actions/paybill may get to...
    - http://www.mybank.com/i8RbsO25nwo7+yZqoUjdEW4ljIRQmdo9Rp2Kit
    - When receiving an encrypted URL from the browser, the WAF decrypts it and forwards it in plain text to the web server
  - This means an attacker cannot "produce" valid URLs beyond the ones he receives in the web page – because he does not know the key
    - This protects from forceful browsing and CSRF
- Form protection
  - The WAF remembers values of hidden fields sent to the user and compares them with the received values when the form is submitted later
  - This prevents attacks where an attacker manipulates hidden fields
- Centralized authentication instead by each web application individually (allows e.g. Single Sign On)

Web Application Firewalls (4)

- Applications:
  - To further increase protection for critical web applications (defense in depth, e.g. with e-banking)
  - To provide protection for insecure web application that you cannot (or don't want to) fix
  - To provide a baseline security for several web applications
  - To provide external patching (or just-in-time-patching) to overcome the time between security flaw discovery and application patching
- Strengths:
  - Powerful security device if configured correctly
- Limitations:
  - Must be configured and fine-tuned to work well in a specific environment (to get low rates of both false positives and negatives)
  - Rule set must be regularly updated to detect new attacks

Intrusion Detection / Prevention Systems

Intrusion Detection Systems (1)

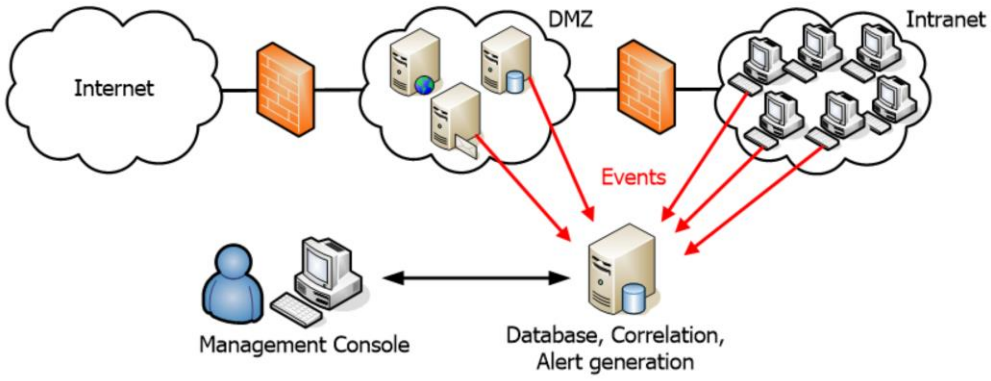
- Intrusion Detection Systems (IDS) monitor network and/or system activity to detect attacks, attack attempts and general abuse

- Types of IDS:
  - Host-based IDS (HIDS) monitoring individual hosts
  - Network IDS (NIDS) monitoring entire (or parts of) networks
  - Often used in combination: Hybrid IDS

- In general, IDS consist of multiple components:
  - Host-based and network sensors, centralised database, correlation engine, management console
  - In the case of a small IDS (e.g. to protect just a single host), these components can all be located on the same system

Intrusion Detection Systems (2): Host-based IDS

- Monitor individual hosts
- Sensors (software component) are directly installed on the hosts that are monitored
- Analyses system calls, application logs, file modifications; but also network data from/to this host
- Examples: OSSEC, Sentry, Tripwire, Snort (when used on a host)



Marc Rennhard, 21.08.2014, SSI_SecurityControls.pptx 44

Monitoring Network Data

A HIDS can – just like a network IDS – monitor network data, but the view is limited to a single host. This is why Snort – despite usually being classified as a network IDS – is listed here as well.

OSSEC (http://www.ossec.net)

OSSEC is an Open Source Host-based Intrusion Detection System. It performs log analysis, integrity checking, Windows registry monitoring, rootkit detection, real-time alerting and active response. It runs on most operating systems, including Linux, OpenBSD, FreeBSD, MacOS, Solaris and Windows.

Sentry (http://sourceforge.net/projects/sentrytools)

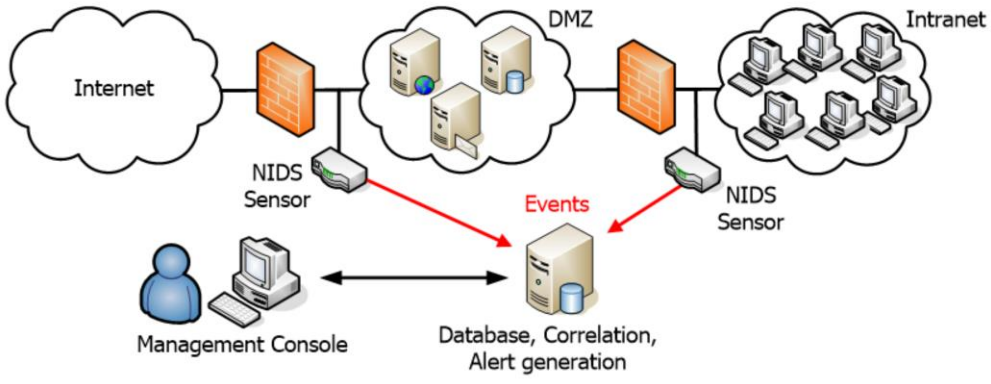
The Sentry tools (Open Source) provide host-level security services for the Unix platform. PortSentry, Logcheck/LogSentry, and HostSentry protect against portscans, automate log file auditing, and detect suspicious login activity on a continuous basis.

Tripwire (http://sourceforge.net/projects/tripwire)

Open Source Tripwire® software is a security and data integrity tool useful for monitoring and alerting on specific file change(s) on a range of systems. The project is based on code originally contributed by Tripwire, Inc. in 2000.

Intrusion Detection Systems (3): Network IDS

- Monitor network segments
- Sensors are dedicated devices in the network that see the desired traffic (often attached to the monitor port of a switch)
- Analyses network traffic for malicious activity such as scanning traffic or attack attempts
- Examples: HP TippingPoint, ISS RealSecure, Snort (dedicated sensor)



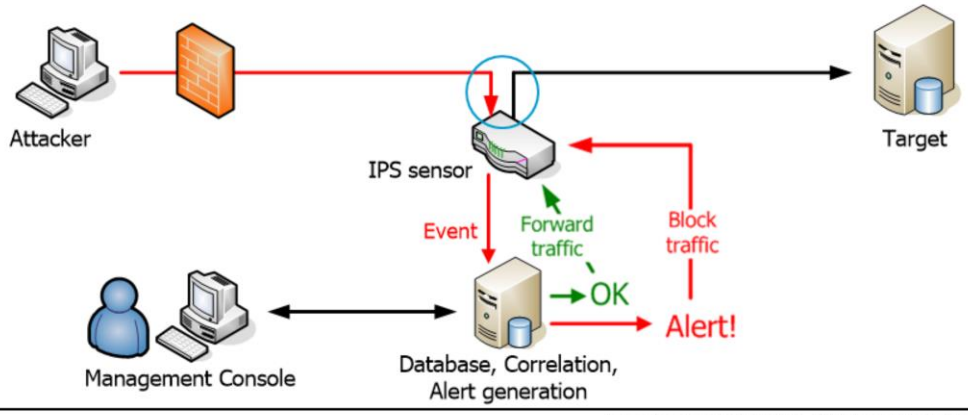
Intrusion Detection Systems (4)

- Applications:
  - To monitor network or system activity to detect attacks and intrusion attempts
- Strengths:
  - Provides you with information whether and in what way you are under attack (log file analysis can also provide this to a certain degree)
  - Can help detecting attacks that are very difficult to detect otherwise (e.g. distributed scans, malware propagation, anomalous system behavior because a system compromise happened…)
- Limitations:
  - Must be configured and fine-tuned to work well in a specific environment (low rates of false positives and negatives)
  - Network IDS cannot see through encrypted traffic → can be overcome by installing network sensors on hosts
  - An IDS does not stop an attack, it merely detects and logs it → human intervention needed (unless the IDS is only used for forensics)

- Malware propagation: Can e.g. be detected if more scanning traffic than usually is observed (more short packets in the traffic mix), which is based on the assumption that infected targets look for further targets to compromise.
- Anomalous system behaviour: Can e.g. be detected if a system starts communicating with an internal server it never or only rarely has used before (e.g. to get interesting internal documentation) or if it starts communicating with a controlling host (e.g. the botnet controller) using a protocol that was never used before to receive instructions.

Intrusion Prevention System (IPS) (1)

- Idea: Unlike with IDS, the traffic does not flow past a sensor (passive monitoring), but through a sensor (active interception)
  - If a sensor generates an event, the traffic is blocked temporarily
  - If no alert is generated, the traffic is forwarded
  - If the event results in an alert (e.g. by correlating it with other events from other sensors), the traffic is blocked
    - Blocked traffic is logged and discarded



Intrusion Prevention Systems (2)

- Advantages compared to IDS:
  - If operated correctly, attacks are not only detected but stopped
  - Because suspicious packets are blocked and only forwarded if no alert is generated

- Additional challenges / limitations compared to IDS:
  - Packets can only be blocked for a very short time to avoid that legitimate communication relationship are significantly delayed
    - This implies that IPS are resource intensive because traffic must be analyzed in near-real time
    - This limits correlation-possibilities with other events from the same or other sensors
  - False positives lock out legitimate user (with IDS, the event/alert is just "registered")

Summary

- There exist several security controls that are helpful when developing secure systems
- This includes:
  - Cryptographic primitives
  - User authentication
  - Secure communication protocols
  - Authorization
  - Firewalls
  - Intrusion detection / prevention systems
  - and more…
- All security controls have their strengths and limitations
- When picking a security control, carefully consider different requirements (security, usability, costs…) to make good decisions