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

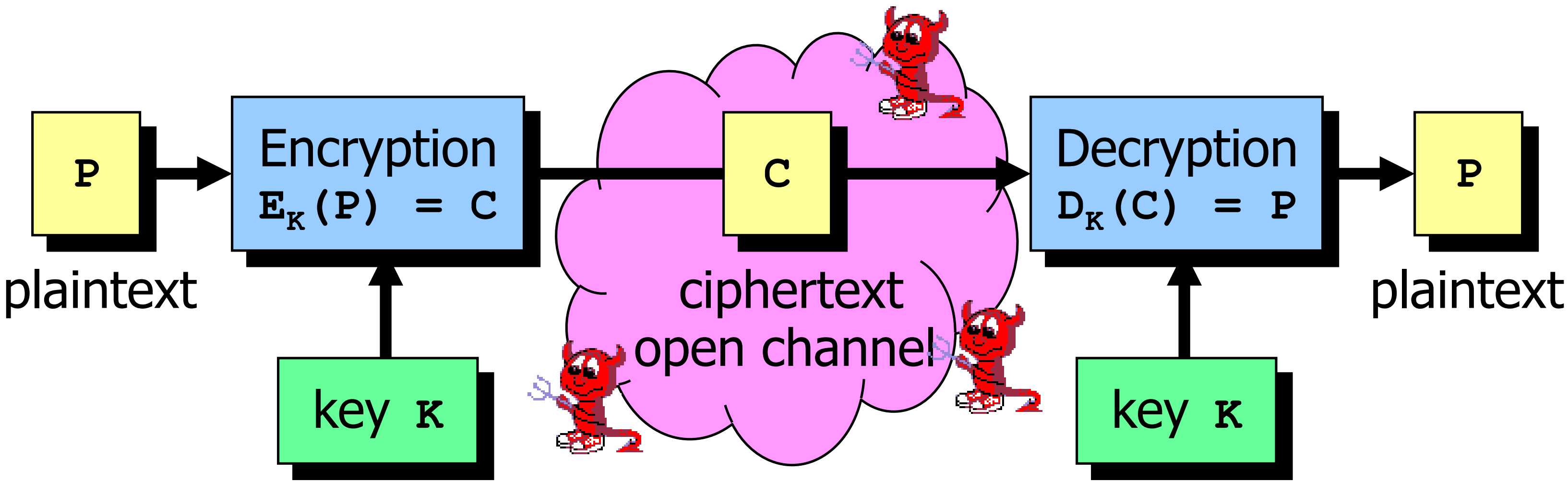
Goals

- 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)

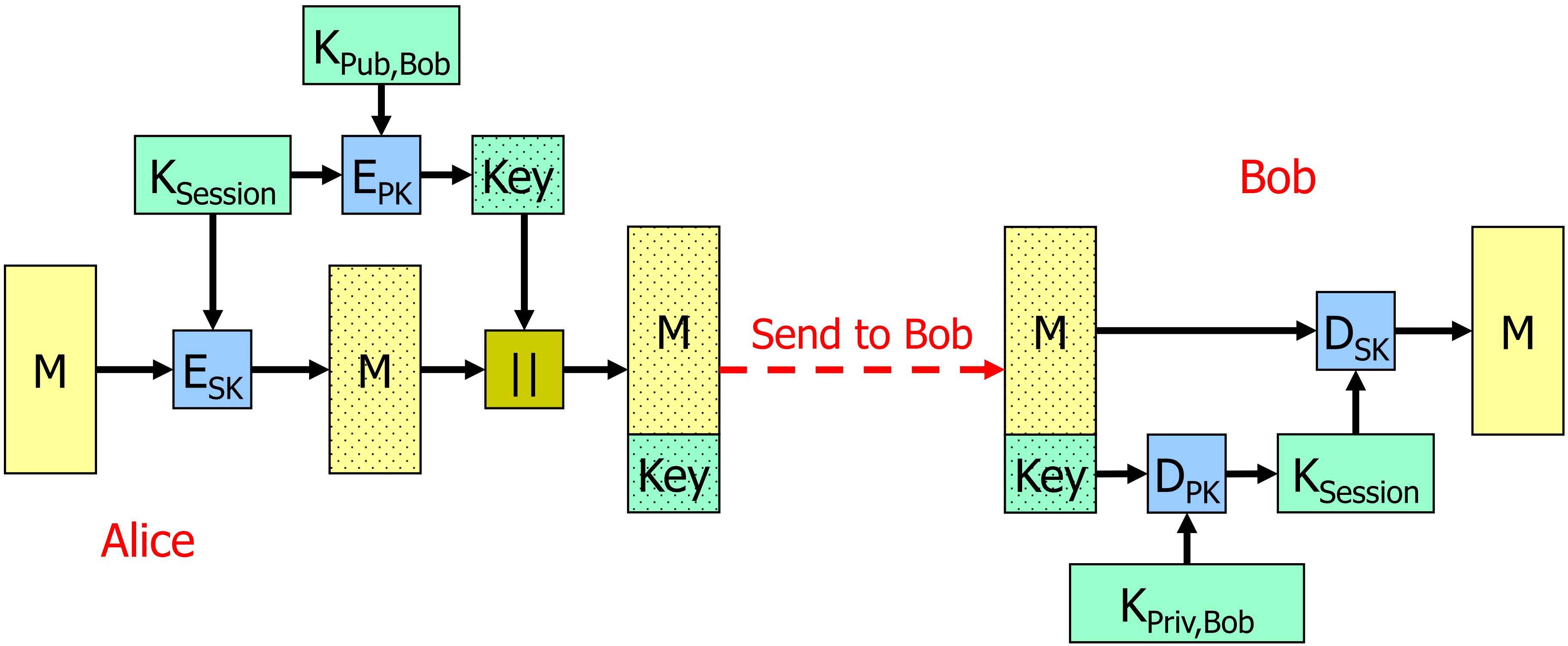
- In contrast to secret key cryptography, two different keys are used for encryption and decryption
  - To encrypt a message, the public key is used
  - To decrypt a message, the private key is used
  - Corresponding public and private key belong together: key pair
- A person Alice can generate a key 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

- Combining secret and public key cryptography: Alice has Bob's public key and wants to send him an encrypted (long) message
  - The message itself is encrypted using a secret key cipher with a randomly selected session key
  - The session key is encrypted with Bob's public key



Cryptographic One-Way Hash Functions (1)

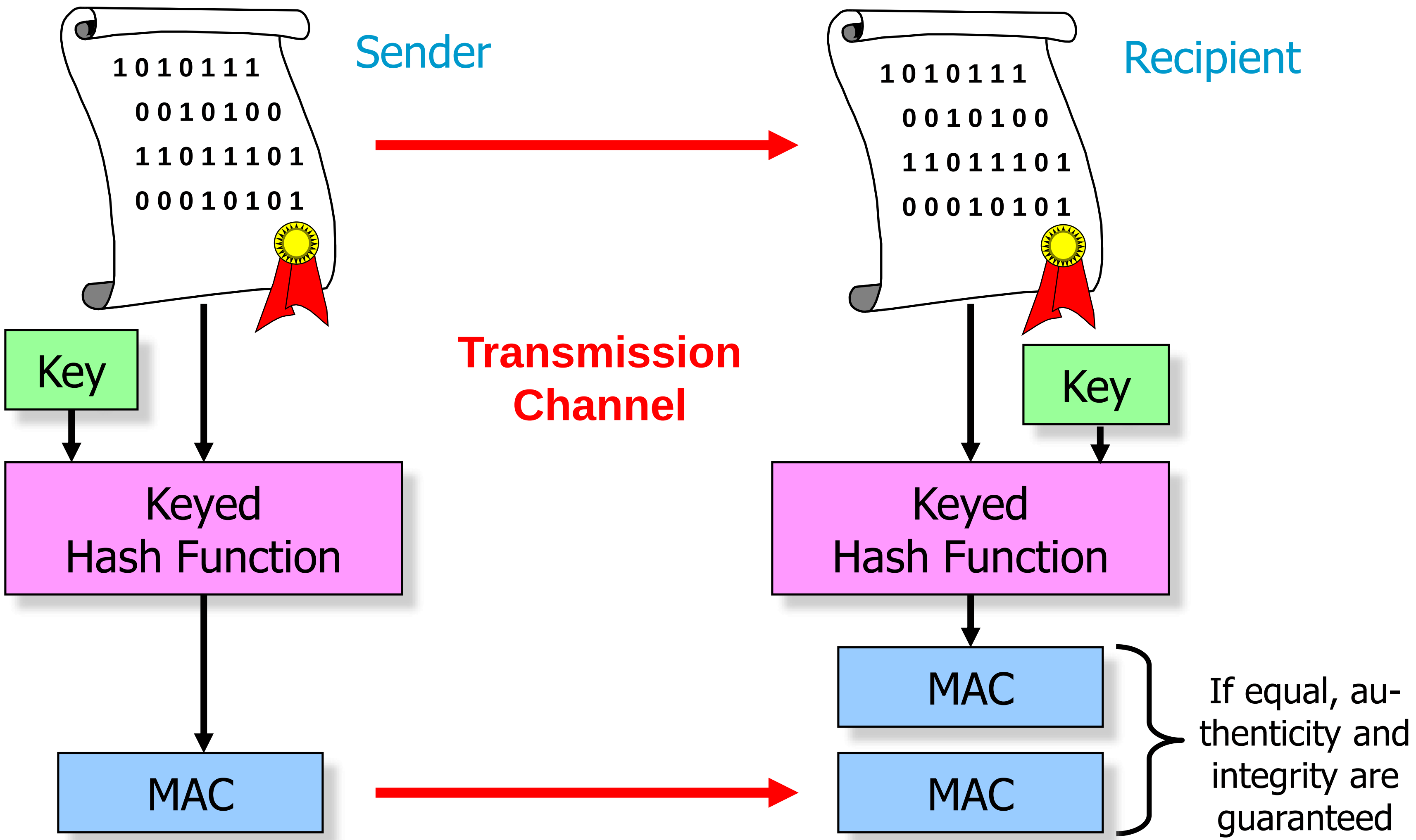
- A cryptographic one-way hash function maps a variable-length input bit string (the message) to a fixed-sized output bit string (the hash or message digest) → many-to-one mapping
- Important properties:
  - Given a hash, it is practically impossible to find a message that produces the hash (one-way)
  - It is practically impossible to find any two messages that map to the same hash (collision-free)
- All properties fulfilled: in practice, a message produces a unique hash and a hash belongs to a particular message → considered as a one-to-one mapping
- Typical hash lengths: 128, 160, 256, 512 bits
- Popular hash functions: MD5, SHA-1, (SHA-2 family), soon SHA-3

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 (MAC) (1)

- Used to protect the authenticity and integrity of a message
  - Prerequisite: sender and recipient share a secret key

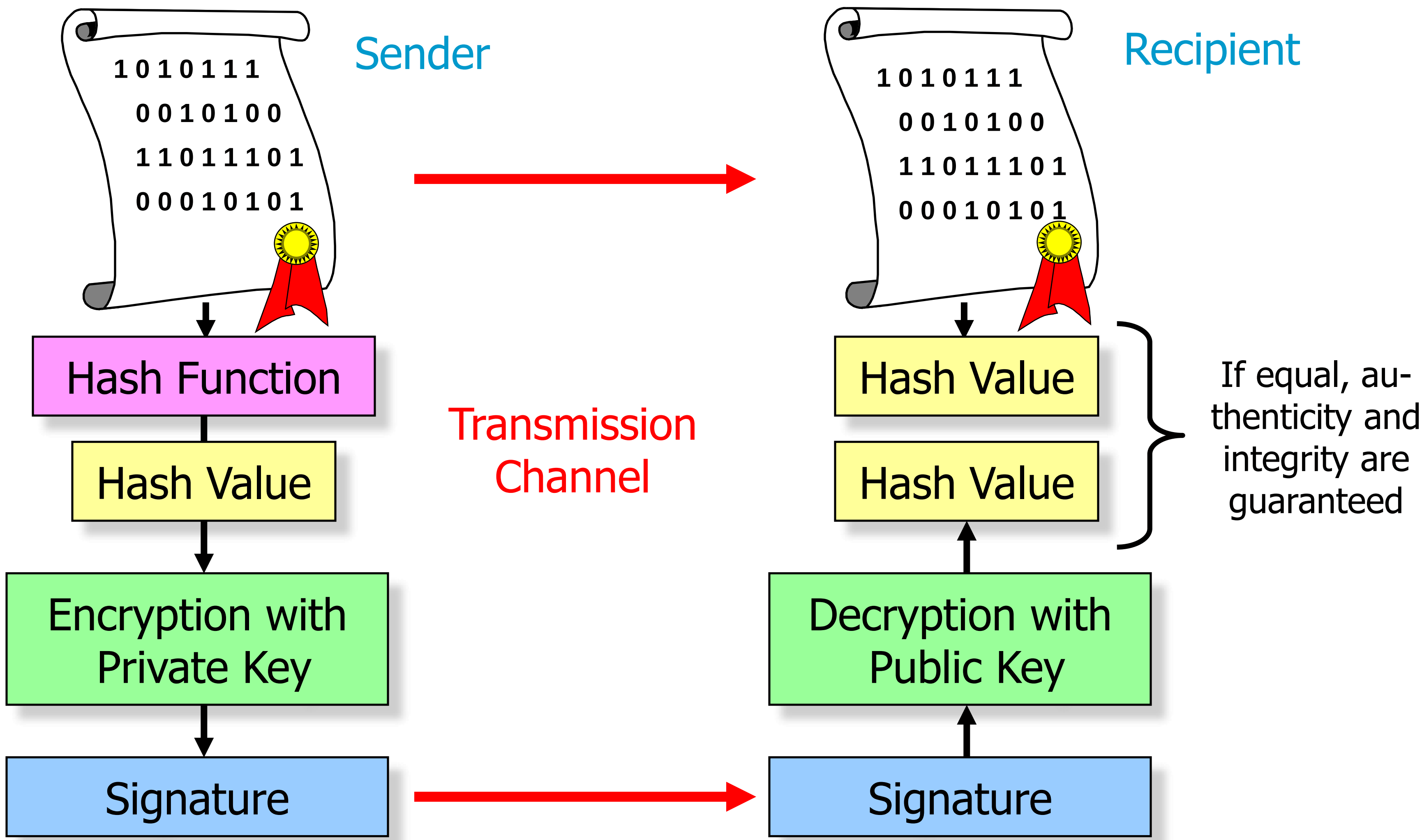


Message Authentication Codes (MAC) (2)

- 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 (1)

- Used to guarantee the authenticity and integrity of a message
  - Prerequisite: sender has a key pair and recipient knows the public key



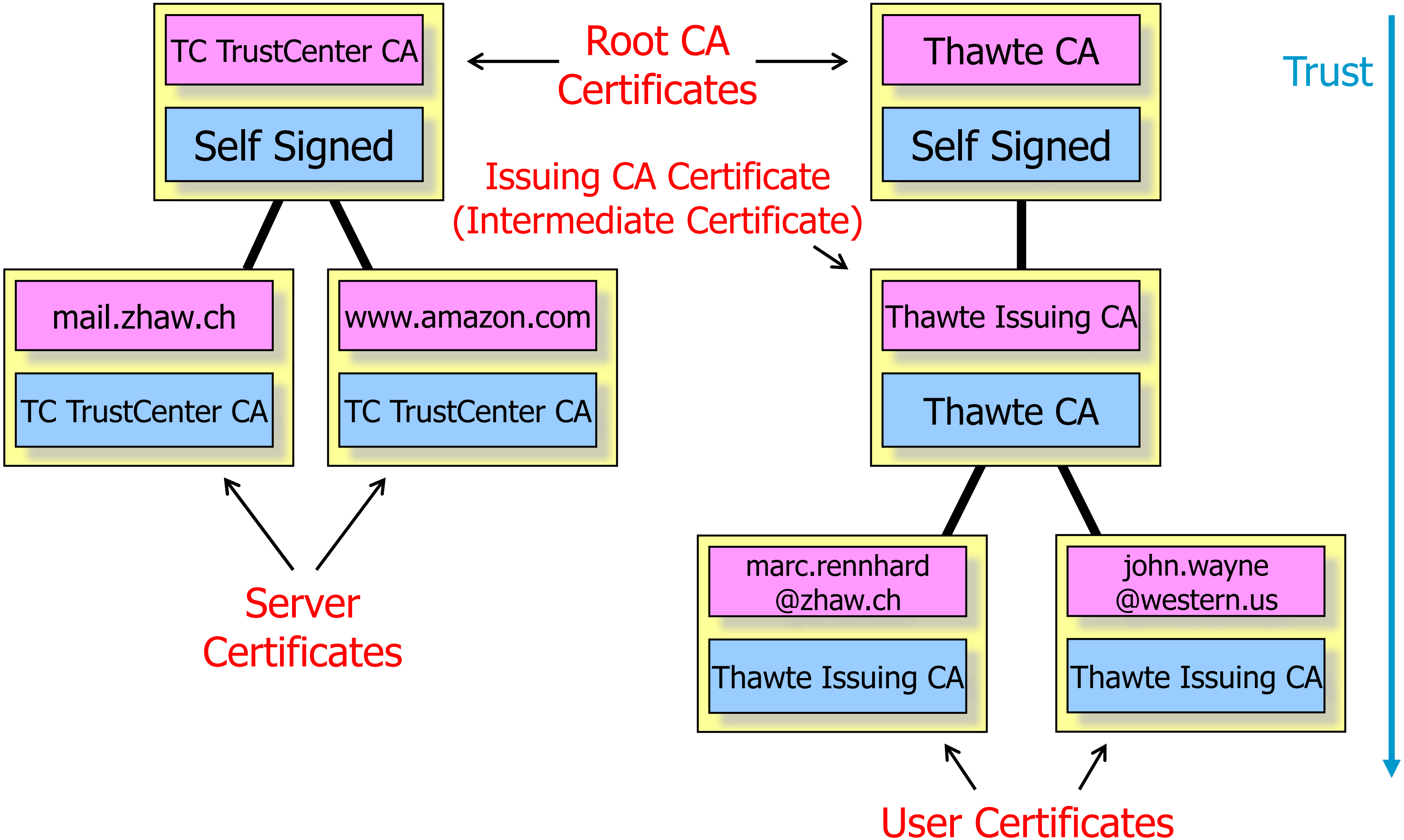
Digital Signatures (2)

- 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)

Digital Certificates (1)

- Digital Certificates bind a public key to an identity
- Most popular standard: X.509 certificates (RFC 2459)
- Certificates are typically issued by Certification Authorities (CA)
  - Various official CAs: Entrust, Thawte, VeriSign etc.
  - The CA checks the identity of the applicant (e.g. is he/she allowed to get a certificate for domain.com?)
  - A certificate is digitally signed by the issuing CA
  - For internal use within e.g. a company, one can also implement an own internal CA, which distributes certificates to internal servers and users
- How to check the validity of a certificate?
  - Requires that the client/server knows the root CA certificate
  - The public key in this root CA certificate is then used to check the signature on the certificate
  - Root CA certificates are preconfigured in web browsers, e-mail clients etc.

Digital Certificates (2) – 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

Authentication is based on...

- What you know (password, PIN, shared secret)
  - Ask for something only the "real" user knows

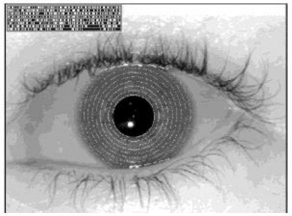
Username:	rennhard
Password:	aznHu4Um

- What you have (Certificate/Private Key, Token, Scratch List, Mobile Phone…)
  - Test for the presence of something only the "real" user has

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

- What you are (biological pattern, e.g. fingerprint)
  - Non-forgeable biological or behavioural characteristics of the user

Fingerprint



Iris/Retina Scanning

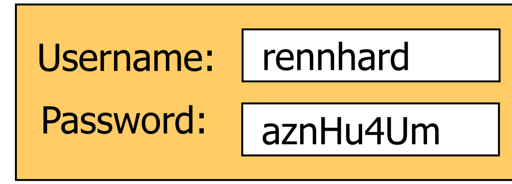
Voice

Face

- Strong authentication: combine (at least) two different factors

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

Passwords (2)

- Applications:
  - User authentication at virtually every service: websites, e-mail servers, local and remote system logins…
- Strengths:
  - Simple to use (in the sense that people know how to use them), cheap to implement/offer, portable
- Limitations:
  - People have problems picking good passwords (too short, only characters, based on words of a natural language)
  - Passwords are often written down, given to others…
  - Passwords are often reused across multiple systems
  - Passwords are susceptible to phishing attacks
  - Passwords are (still) sometimes transmitted across insecure communication links

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

Secure Sockets Layer / Transport Layer Security (1)

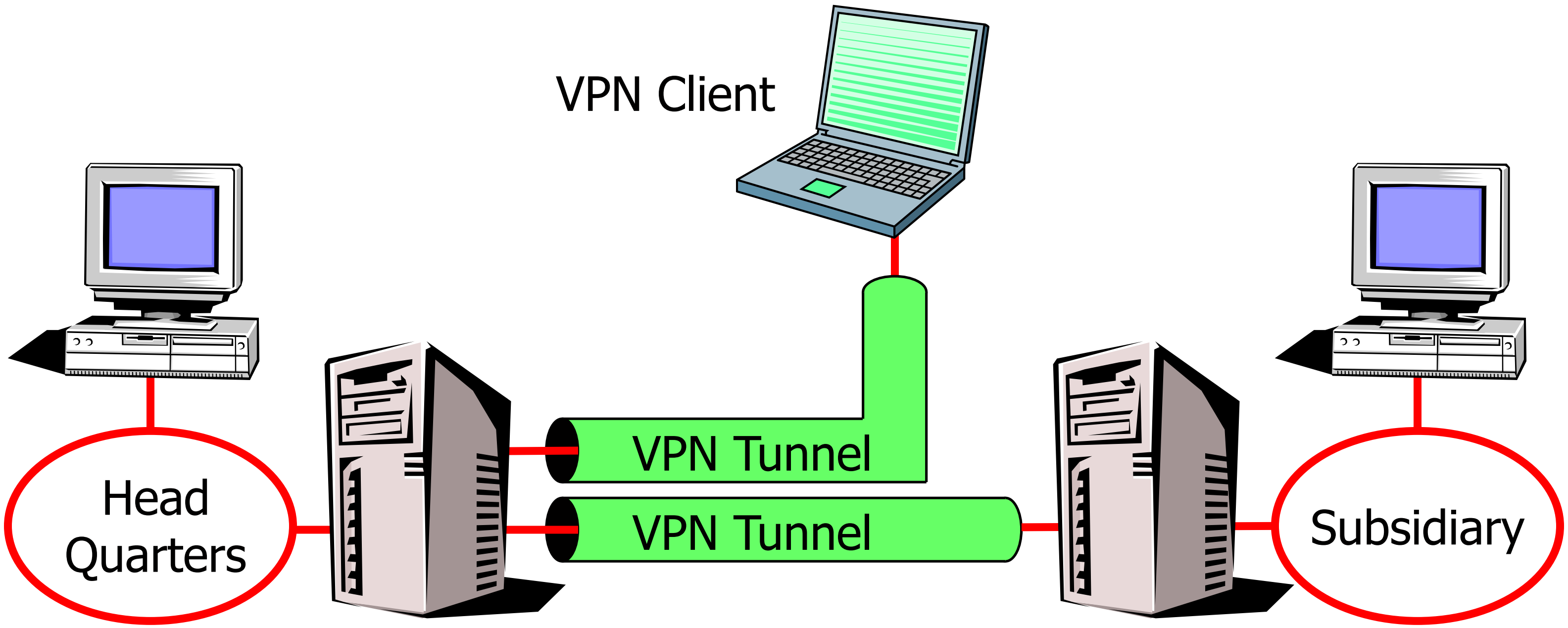
- SSL/TLS provides a secure communication channel above TCP
  - Secure means confidentiality, integrity, authenticity
  - There's also a variant that works on top if UDP (Datagram Transport Layer Security)
- The server typically uses certificates for authentication
  - Pre-Shared secrets (PSK) cipher suites are also supported
- Client authentication can also use certificates, but client-authentication is often not done as part of SSL/TLS
- Several versions:
  - SSL 2.0, Netscape, 1995: contains security flaws, don't use it
  - SSL 3.0, Netscape, 1996: fixes some security issues of 2.0
  - TLS 1.0, IETF, 1999, RFC 2246: similar to SSL 3.0, new MAC computation
  - TLS 1.1, IETF, 2006, RFC 4346: similar to TLS 1.0, some security fixes
  - TLS 1.2, IETF, 2008, RFC 5246, similar to TLS 1.1, some security fixes

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

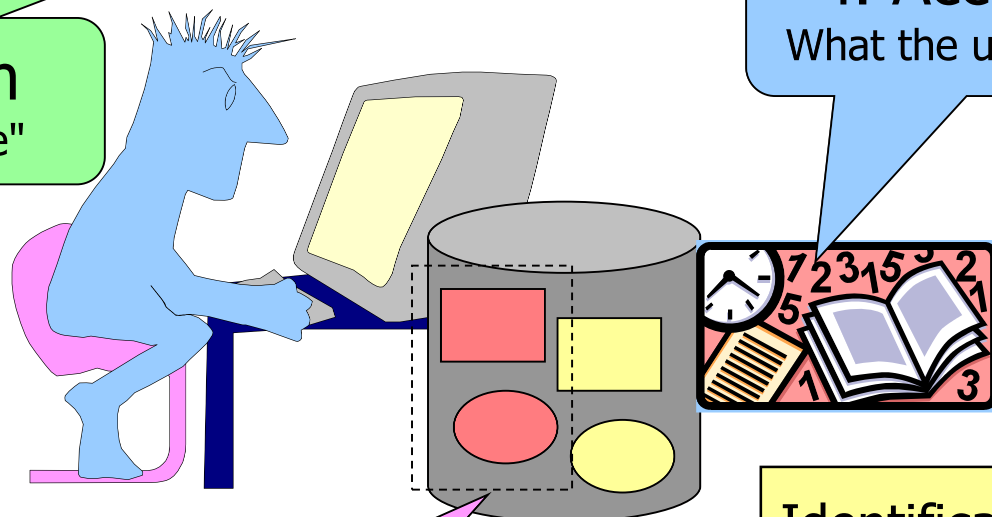
Access Control

1. Identification
I am: "Username"

Username: rennhard
Password: ********

2. Authentication
Prove that I am "Username"

4. Accounting
What the user has done



3. Authorization
What the user is allowed to do

Identification +
Authentication +
Authorization +
Accounting =
Access Control

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)

Authorization (2)

- Discretionary Access Control:
  - Access to an object is controlled by the owner of an object
  - The owner of the resource (e.g. a file) controls which subjects (user, group…) can have access to it and to what degree
  - Dominating in all major operating systems, usually implemented with Access Control Lists (ACLs)

- Mandatory Access Control:
  - Access is controlled (mandated) by the system, a system-wide policy determines who is allowed to do what on the system
    - The policy is configured by a (security) policy administrator
  - In today's operating systems, MAC is sometimes used in addition to DAC to guarantee some access restrictions in any case
    - When an object is accessed, first enforce MAC, then apply DAC
    - Linux: SELinux, AppArmor
    - Since Windows Vista: Windows Mandatory Integrity Control

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)

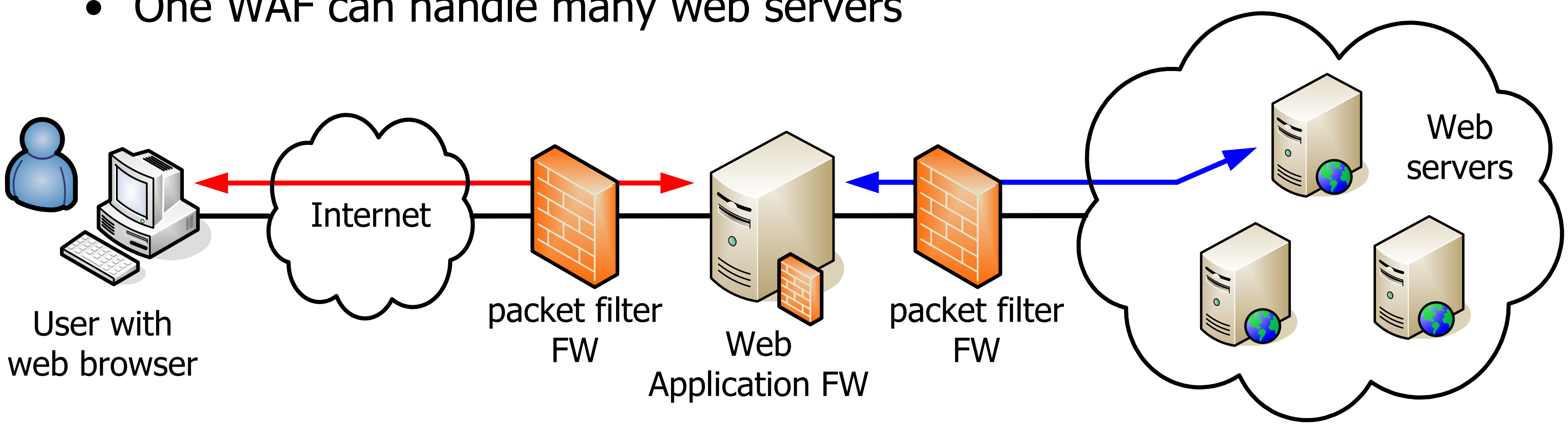
- Applications:
  - To filter traffic between connected networks

- 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)

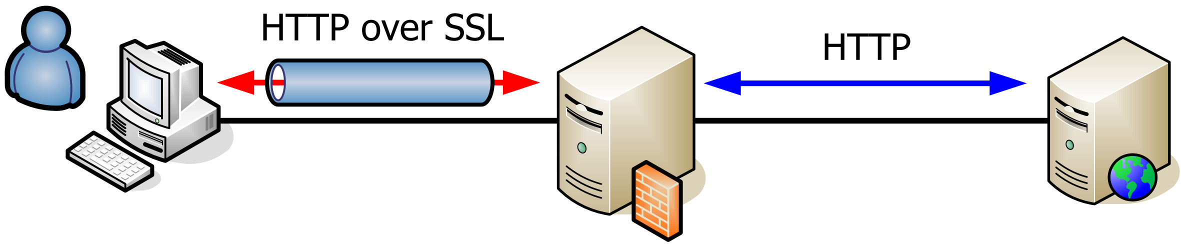
- Web application firewalls (WAF) filter traffic at the application layer and are optimized to protect web applications
- Most frequently, a WAF is operated as a reverse proxy
  - Additional packet filtering firewalls guarantee that web servers cannot be reached directly
  - The web browser connect to 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

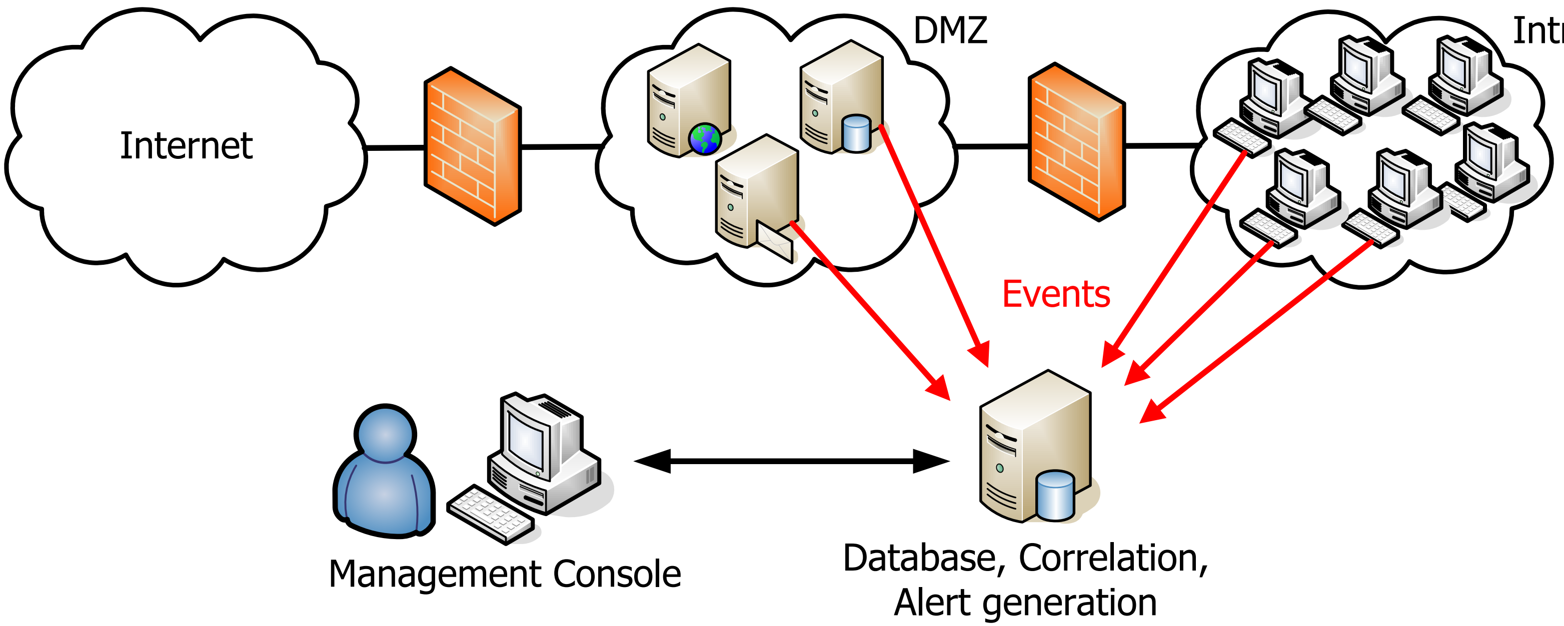
Intrusion Detection / Prevention Systems

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

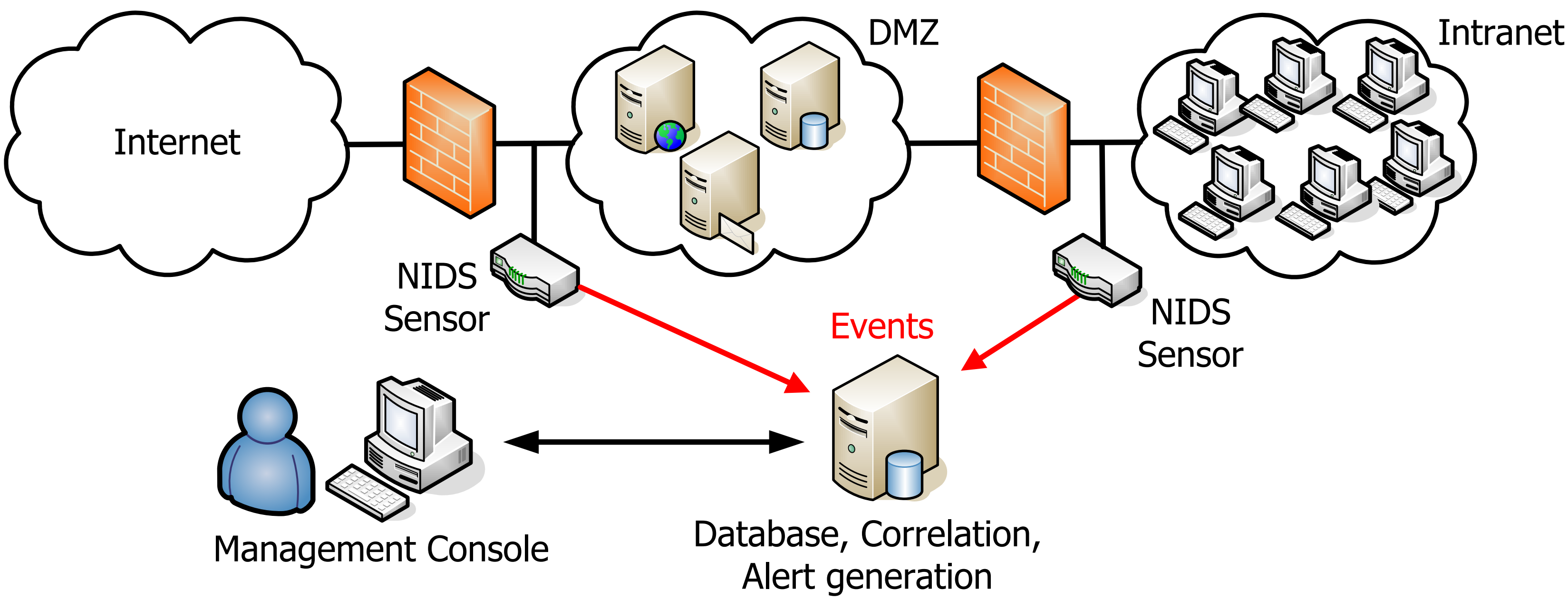
Intrusion Detection Systems (2): Host-based IDS

- 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)



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)

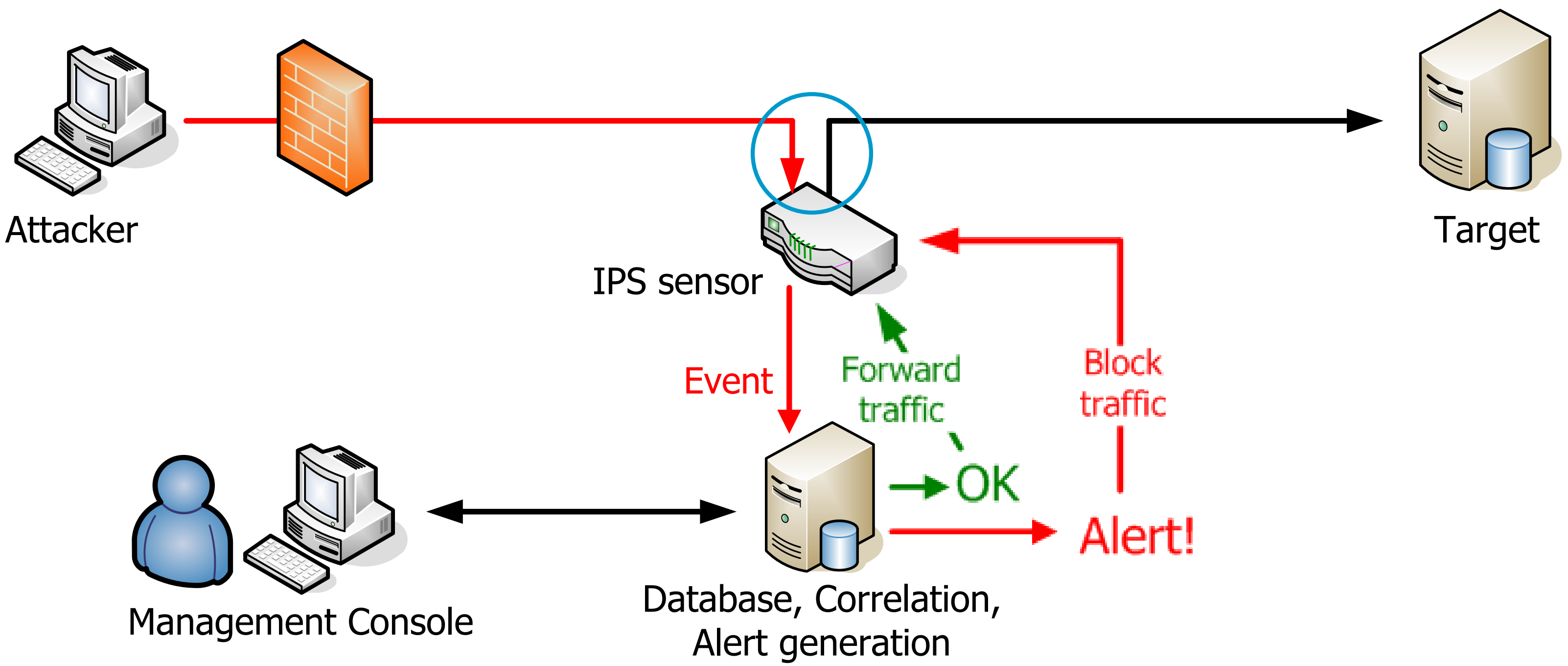


Intrusion Detection Systems (4)

- 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)

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