Einführung

Risiken

Ransom Demand/Ware: Daten werden verschlüsselt und dann ein Lösegeld, meistens in Bitcoin, verlangt. Technik: Ransomware

Fraud: Angreifer bringen ihre Opfer dazu, Geld auf ein Konto ihrer Wahl zu überweisen. Techniken: Phishing-Emails oder Social Engineering

Espionage: Ausspionieren des Konkurrenten. Oft unbemerkbar.

Violation of Regulations: Vor allem im Bereich Datenschutz, DSGVO (europäische Datenschutzverordnung)

Misuse of Computing Resources: Unbemerkter gebrauch von Infrastruktur, z.B. von minen von Bitcoin.

Reputation Loss: Reputationsschaden durch Cyberangriffe.

System Outage: Käufig eine Konsequenz einer Erpessung. Hoher finanzieler Schaden.

Sabotage: Ein System gezielt physisch stören, ohne dass man dies direkt merken würde. Meistens im Bereich der staatlich finanzierten Cyberangriffe.

Brand Misuse: Meistens durch Phishing Kampagne. Technik: Spoofing

Methoden

Malware: Bösartige Software (z.B. Virus) wird installiert.

Ransomware: Daten werden verschlüsselt und Erpressungsfenster angezeigt

Phishing und Social Engineering: Techniken an Benutzerdaten zu kommen

DDoS (Distributed Denial of Service): Ein Botnet schickt viele Anfragen an ein System, das dieses unter der Last zusammenbricht.

Spoofing: Vortäuschen einer Identität

CIS-Framework

Security Framework Provide Guidance (Cyber-Sicherheit Richtlinien)

Levels of security: ■ Mindestanforderung ■ Fortgeschritten ■ Vollumfänglich

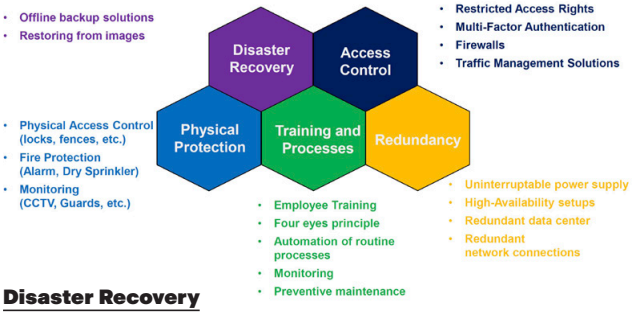
Goals of Information Security

CIA Triad

Ensure data is **confidential (verfügbar)**
Ensure data is **not corrupted (integrity, unverfälscht)**
Ensure data and systems are **available (abhörsicher)**

confidential + integrity = **Cryptology**

Counter-Measures



Disaster Recovery

Ersatz-/Notfalls-Massnahmen

Business Continuity Plan: Ein Plan, was im Falle eines Ausfalles zu tun ist. Abdecken der überlebenswichtigen Prozesse.

Recovery Tests: Testen, ob der Business Continuity Plan funktioniert.

Kryptoanalyse

Die Kryptoanalyse befasst sich damit, wie die Mechanismen der Kryptographie gebrochen werden können.

Begriffe

Cipher/Chiffre: Verschlüsselungs-Algorithmus (z.B. AES).
Block (**feste Längen** 128 (Default), 192, 256 bit) oder **stream** (jedes Bit).
Cipher-Text: Verschlüsselte Daten.
Hash: Ausgabe-/Hashwert (digest) fester Länge einer Hashfunktion (SHA256)
Salt: Zufällig gewählte Zeichenfolge (Verarbeitung z.B. Input Hashfunktion)
Nonce/Initialisierungsvektor (IV): Number used only once.
Zufälliger Wert für jeden Cipher-Block.
MAC (Message Authentication Codes): Kryptografischer Hash. Used to confirm that the message from the sender and has not been changed (integrity).

Kryptographie

Die Kryptographie befasst sich damit, wie Nachrichten sicher gespeichert und übertragen werden können.

Anwendungsfälle

At Rest (speichern): Daten bleiben verschlüsselt beim Anwender (auf der Festplatte).
In Transit (Übertragen): Daten werden vershlüsselt übermittelt.
In Use: (Sehr sensitive) Daten bleiben während dem Bearbeiten verschlüsselt.

Ziele

Confidentiality (Vertraulichkeit): Daten sind nur den berechtigte Personen verfügbar/lesbar. Die Daten können nicht von Dritten gelesen werden. Wird per **Verschlüsselung** sichergestellt. → Per **Secret Key** sichergestellt

Integrity: Daten bleiben **unverfälscht.** Original Daten = Entschlüsselte Daten. Die Daten können nicht durch Dritte unbemerkt verändert werden. → Wird per **Message Authentication Codes (MAC)** sichergestellt.

Authenticity: Daten sind **überprüfbar,** dass sie vom Absender stammen. Der Empfänger weiss, von wem die Daten gesendet wurden. → **MAC**

Non-Repudiation: Es kann bewiesen werden, dass die Daten beim Empfenger angekommen sind. Der Empfänger kann nicht abstreiten, dass er eine Nachricht bekommen hat.

Freshness: Zu einem späteren Zeitpunkt gesendete/erhaltene (korrupmierte) Daten sind erkennbar. Der Empfänger merkt, falls ein Attacker eine alte Nachricht nochmals schicken würde.

Attacktypen

Ciphertext-only: Vom Ciphertext alleine können Rückschlüsse auf den Plaintext oder den verwendeten Schlüssel gezogen werden. → **IV**

Chosen-ciphertext: Attacker generiert Ciphertexte und kann diese vom System entschlüsseln lassen.

Known-plaintext: Es sind Teile des Plaintext und des dazugehörenden Cipher bekannt → Mögliche Rückschlüsse auf andere Plaintexte oder Schlüssel.

Chosen-plaintext: Attacker kann Plaintexte vom System verschlüsseln lassen Mögliche Rückschlüsse durch den Cypher auf andere Plaintexte/Schlüssel.

Brute-force: Attacker probiert alle möglichen Schlüssel.

Work Factor

Mass der Stärke für eine Verschlüsselung. **Durchschnittliche Anzahl Versuche,** um den richtigen Schlüssel durch Brute Force zu finden. Grundsätzlich gilt: Je länger der Schlüssel, desto höher der Work Faktor.

Work Faktor ist dann **maximal**, wenn alle Schlüssel fixer Länger gleich wahrscheinlich sind (alle möglichen Schlüssel **gleichverteilt** sind).

Genügend grosser Work Faktor, wenn es unrealistisch ist, den richtigen Schlüssel per brute force zu erraten. **Mind. >2 ^{128} keys** bzw. **> 128 bits per key**

$$\mathrm{WF}(X) = \sum_{k=1}^{n} k\, p_k$$ p_k = Wahrscheinlichkeit für Schlüssel k

Perfect Secrecy (Vernam Cipher)

Ein Algorithmus, bei dem man den ursprünglichen Plaintext nicht erkennen kann, auch wenn man alle Schlüssel ausprobiert hat.

One-Time-Pad: Voraussetzung: Schlüssel komplett zufällig (can only be used once) und gleich lang wie die Nachricht.
Verschlüsselung: Der Plaintext wird mit dem Schlüssel bitweise ver-xor-ed

$$c_j = p_j \oplus k_j \forall j \in [1, n]$$

Hash Funktion

1. Aus einem beliebig langen Input wird ein **Output mit fixer Länge** generiert.
2. Es gibt keine Möglichkeit aus dem Output den Input wieder herzuleiten.
3. Unterschiedliche Inputs ergeben unterschiedliche Outputs.

Hash Collision: Es gibt keinen Algorithmus, der Eigenschaften 3 erreicht. Es wird immer verschiedene Inputs geben, welche denselben Output generieren.

Hash Funktionen

Funktion	Hash Länge	Work Factor
MD5	128 bit	64 bit (ungenügend)
SHA-1	160 bit	80 bit (ungenügend)
SHA-2/3	224 - 512 bit	112 - 256 bit

Work Faktor entspricht der Hälfte der Hash Länge (Outputslänge) = N/2.

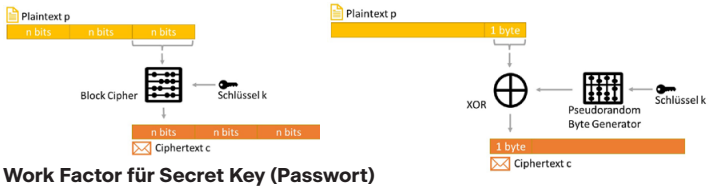
Einsatz:
- In Kombination mit Public Key für die Integritätsüberprüfung.
- Speicherung von Passwörtern, damit nicht der Klartext gespeichert wird.
- Berechnung eines Authentication Tags (symmetrischen Verschlüsselung).

Secret Key Encryption

Derselbe Schlüssel wird für die Verschlüsselung/Entschlüsselung verwendet.
Problem: Schlüssels selbst wird unverschlüsselt ausgetauscht.

Block Cipher: Verschlüsselung erfolgt in Blöcken. Nachricht wird in Blöcke gleicher Länge aufgeteilt **(padding needed)** und jeweils einzeln verschlüsselt.

Stream Cipher: Jedes Bit wird einzeln verschlüsselt. Für jedes Bit wird ein «Schlüsselbit» gebraucht welches mithilfe eines Pseudo-Random-Key-Generator aus dem eigentlichen Schlüssel generiert wird.



Work Factor für Secret Key (Passwort)

Allgemeine Formel: $WorkFactor = (2^n + 1)/2$
Approximierte Formel: $WorkFactor = 2^{n-1}$
n > 128bits: $WorkFactor = 2^n$
n = Schlüssel der Länge n **bits.** Work Faktor in bit: $\log_2(2^n)$ [bit]

Problem: Stellt lediglich die **Vertraulichkeit (Confidentiality)** der Daten sicher. Der Cipheretext ist nur vom Plaintext und vom Schlüssel abhängig, gleicher Plaintext produziert gleicher Cyphertext. Plaintext könnte mithilfe einer **Frequenzanalyse** (Häufigkeit von Zeichen) zurückgewonnen werden.
→ Zufälliger Wert **(Initialisierungsvektor)** für jeden verschlüsselten Block

Initialisierungsvektor (IV)

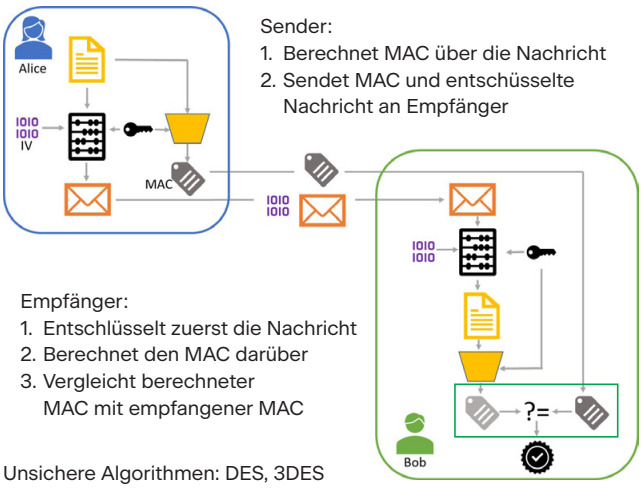
Zufälliger Wert, **Nonce** (number used only once).
Verhindert, dass gleiche Klartexte zu gleichen Ciphertexten führen.

1. Muss für jeden Block (mit demselben Schlüssel) unterschiedlich sein.
2. Muss dem Sender und dem Empfänger bekannt sein.
3. Muss nicht zwingend geheim sein.

Authenticated Encryption (AE)

Verschlüsselung und Message Authentication in einem Algorithmus.
Authenticated Encryption with Associated Data (AEAD): Kann Daten authentisieren, aber nicht verschlüsseln. Standard heute.

Message Authentication Codes (**MAC** = kryptographischer **Hash**) stellt zusätzlich zu **Confidentiality** auch **Authentizität** und **Integrität** sicher.



Sender:
1. Berechnet MAC über die Nachricht
2. Sendet MAC und entschüsselte Nachricht an Empfänger

Empfänger:
1. Entschlüsselt zuerst die Nachricht
2. Berechnet den MAC darüber
3. Vergleicht berechneter MAC mit empfangener MAC

Unsichere Algorithmen: DES, 3DES
Sichere Algorithen sofern **Schlüssel > 128 Bits:** AES, RC5, Blowfish

AES Modes (Counter Modes = Cipher Algorithms)

AES (Advanced Encryption Standard) ist ein **Block Cipher.**
Schlüssellängen in 128 (Default), 192, 256 bit

Mode	Name	Garantien / Beschreibung	Empfehlung	
ECB	Electronic Code Book	keine Es wird kein IV und kein MAC verwendet.	Soll nicht verwendet werden, da kein wirklicher Schutz der Daten erreicht wird.	(stateless)
CBC	Cipher Block Chaining	Confidentiality Es wird nur ein IV aber kein MAC verwendet.	Soll nicht verwendet werden, da kein Schutz gegen Verändern der Daten.	
CTR	Counter Mode	Confidentiality Es wird nur ein IV aber kein MAC verwendet.	Soll nicht verwendet werden, da kein Schutz gegen Verändern der Daten.	
CCM	Counter with Cipher Block Chaining	Confidentiality and integrity (authenticity) Es werden sowohl IV als auch MAC verwendet.	Kann verwendet werden, ist aber langsamer als GCM und bringt sonst keine Vorteile.	
GCM	Galois Counter Mode	Confidentiality and integrity (authenticity) Es werden sowohl IV als auch MAC verwendet.	Dies ist der aktuelle Standard und soll wenn immer möglich verwendet werden.	(statelfull)

Public Key Cryptography (Symmetric Cryptography)

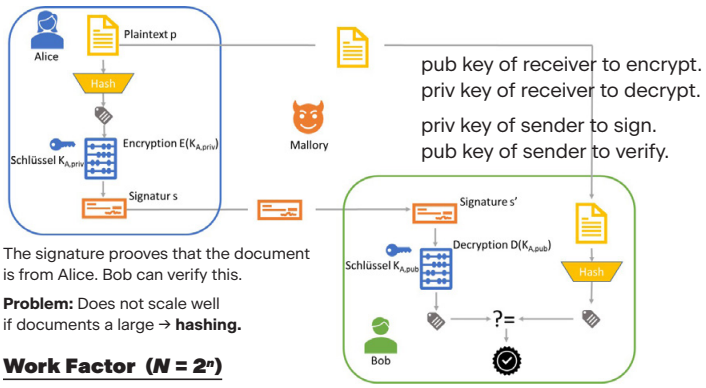
Löst Problem des Schlüsselaustausches in der Secret Key Kryptographie.

Zwei Schlüssel stehen in einer mathematischen Beziehung zueinander:
– K_{pub} verschlüsselt Nachricht, K_{priv} entschlüsselt Nachricht.
– K_{pub} dürfen alle kennen. K_{priv} muss geheim gehalten werden.

Bietet Vertraulichkeit, jedoch nicht Authentizität.

Signaturen mit Public Key

K_{pub} kann (unverschlüsselte) Nachrichten authentisieren
indem der Hash der Nachricht verschlüsslet bzw. signiert wird.



pub key of receiver to encrypt.
priv key of receiver to decrypt.
priv key of sender to sign.
pub key of sender to verify.

The signature prooves that the document is from Alice. Bob can verify this.

Problem: Does not scale well if documents a large → **hashing.**

Work Factor ($N = 2^n$)

- Work Faktor für klassische Algorithmen
  $4096bit \Rightarrow 128\ bit\ work\ faktor$
- Work Faktor für Algorithmen mit elliptischen Kurven
  $256bit \Rightarrow 128\ bit\ work\ faktor$
  $512bit \Rightarrow 256\ bit\ work\ faktor$

Allg. Formel: $(N+1)/2$
n > 128 bits: $N/2$
N = **Anzahl** Schlüssel/ Möglichkeiten (gleichverteilt)

Der WF hängt sowohl von der Schlüssellänge, als auch vom verwendeten Algo ab. WF ist meist kleiner als die Anzahl bits vermuten lassen. Kürzeren Schlüsseln werden bevorzugt (wenn dieselben Sicherheitsgarantien vorliegen).

Session Keys

Session Key basiert auf einem Master Key (Langzeitschlüssel) und wird regelmässig neu ausgehandelt.

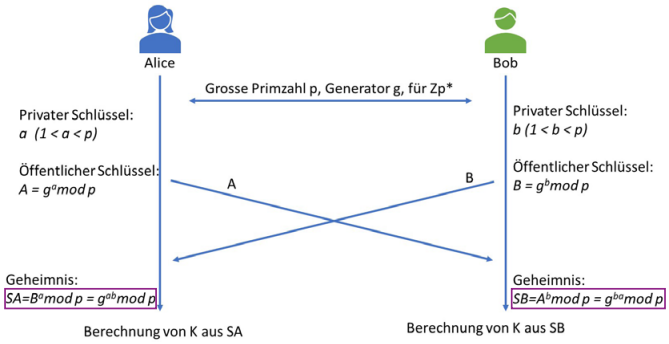
Perfect Forward Secrecy (PFS)

Session Keys können trotz Besitz des Master Key nicht wiederhergestellt werden.

Diffie-Hellman Key Exchange Algorithmus

Allows two parties to securely establish a **shared secret key** over an insecure communication channel. This key can then be used for encrypting messages, ensuring confidentiality in their communication.

Eine **Man in the Middle** Attacke (Angreifer gibt sich als Alice oder Bob aus) ist allerdings möglich. Lösung: **Zertifikate,** öffentliche Schlüssel wird von einer (hoffentlich) vertrauenswürdigen Stelle signiert.



g und p (2000 oder 4000 bits) sind öffentliche Zahlen (**salt**).

RSA (Rivest–Shamir–Adleman)

Implementiert sowohl **Verschlüsselungen** als auch **Signaturen.**
Basiert auf klassischen Feldern (keine Möglichkeit für elliptischen Kurven).

Praxis: RSA wird für das Signieren von Key Exchange Nachrichten verwendet und Diffie-Hellman für den eigentlichen Key exchange. Der ausgetauschte Schlüssel wird dann verwendet, um die Nachricht mittels AES zu verschlüsseln.

e: public key, usually set to 65537
d: private key
n: Modulus, grosse Primzahl. Die Länge von n entspricht der Schlüssellänge

Verschlüsselung: $c = p^e \bmod n$
Entschlüsselung: $p = c^d \bmod n$
Signatur generation: $s = p^d \bmod n$
Signatur verifikation: $p = s^e \bmod n$

Digital Certificates

Problem: Pre-shared keys do not scale (Key Exchange Problem).
Alternative to MAC.
Trusted = Muss vertraut werden. Trust worthy = Kann vertraut werden.

Certificate Types

Zertifikat = Identitätsbeweis. Ein Zertifikat dient primär dazu, einen Public Key an eine Identität (Hostname) zu binden.

TLS Certificates (Server certificates):
Steht jeweils im Zertifikat, welcher Typ.
- **Domain Validation (DV):** Domain name belongs to public key
- **Organisation Validation (OV):** Domain ownership, company name, etc.
- **Extended Validation (EV):** Additional properties such as phone, etc

Code signing certificates: Confirms authenticity of software or files.
Client certificates: Digital ID, identifies individual user, email signatures.

Public Key Infastructure (PKI)

What is **signed** (by CA): That a **public key** belongs to a **server name.**

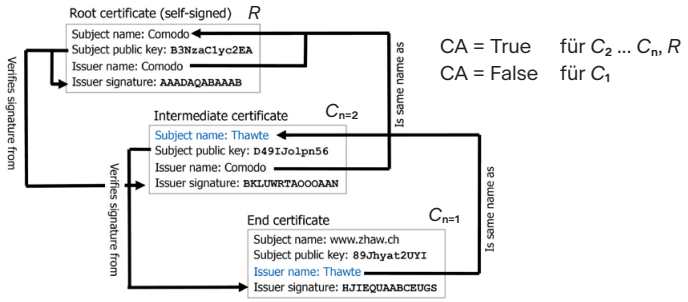
Certification Authorities (CA): Issuer / third parties that sign these certificares. Ein CA-Zertifikat wird nur für die Prüfung eines ausgestellten Zertifikats verwendet (und nicht beim austellen).

Authenticate a communication partner:
1. Authenticate the certificate
2. Partner must prove they have the corresponding private key to the pup key
3. The subject name must match the issuer name in the certificate

Certificate Chain: In order to verify signature, one needs the public key of the CA. The CAs public key is signed by yet another CA.

Root CA and Root / self-signed certificates
Root certificates are the **ultimate anchors of trust.**
Not signed by another CA, but by themselves.
Can be verified by the public key that is in that very certificate.
In practice, root certificates are preinstalled in Browsers.

Certificate Chain



Obtaining a Certificate

In short: 1. Generate key-pair, 2. Certificate Request (CSR), 3. Issue certificate

1. Alice creates a key pair (pub, priv) for www.alice.com
2. Alice sends a **Certificate Signing Request (CSR)** containing (www.alice.com, pub, priv) to her CA of choice.
3. The CA verifies that Alice really owns www.alice.com
4. The CA signs Alice's CSR, giving the signature S
   ➔ **Problem:** An evil CA could impersonate Alice.
5. The CA puts the following into a certificate:
   1. Subject name: www.alice.com
   2. Subject public key
   3. Issuer (CA) name
   4. Issuer (CA) signature S
6. The CA sends the certificate to Alice

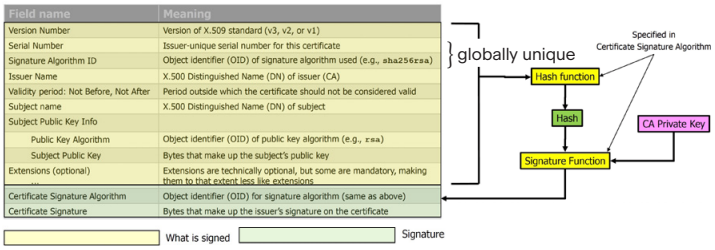
Validating Certificate Chains

Bob connects to www.alice.com
Server sends certificate chain of length *n* to Bob. Chain: C_1 ... C_n, R

1. Check that certificate C_1 name == www.alice.com
2. Verify that C_n validates C_{n+1} :
   1. C_{n+1} subject == C_n issuer
   2. C_{n+1} public key verifies CRL (Certificate Revocation List) signature
   3. Check valid time (NotBefore, NotAfter) of C_n
   4. Check that C_n is not on the CRL (Certificate Revocation List)
   5. C_{n+1} public key verifies C_n signatures
3. Locate Root certifiacte R with C_n Issuer == R subject
4. R public key must verify C_n siganture
5. R public key must verify R siganture

X.509 Standard

Dominating standard for certificates these days. Structure of a certificate:



Revocation

Worst possible incident for a CA.
Trusted root certificate must be removed from all systems.

Problem: Subject can't change their own certificate.
An Attacker can always use the original certificate.

Solution: **Additional check** whether **certificate has not been revoked.**

Two ways of checking:
- **Certificate Revocation List (CRL):**
  List of serial numbers published by the issuer (CA)
- **Online Certificate Status Protocol (OCSP):**
  «Fresher» status response the issuer: *good* or *revoked*

Certificate Revocation List (CRL):
Limitations
- Only updates every couple of days.
- Can't remove certificate even it is expired.
- CRLs only get larger and needs to be parsed → scalability issue.

Online Certificate Status Protocol (OCSP):
Es (user/browser) muss nachgeschaut werden, ob ein Zertifikat noch gültig ist.
Im Zertifikat: Authority Information Access → OCSP Responder: URI
Das Zertifikt wir mittels der **Serial Number** identifiziert.

Advantages: «Fresher» status response and small amounts of data (status)
Disadvantages:
- Privacy issue: CA can collect information about client visiting which website
- Single Point of Failure: Must be always available, subject to DoS attack
  Hard Fail: Client must decide (browser).
  Soft Fail: No response by OCSP → certificate is *good* (not revoked).
  Szenario: Erstverbindung mit einem Webportal mit benötigtem Passwort (z.B. in einem Hotel). Browser hat noch gar keine Internetverbindung um OCSP-Responder abzufragen.

OCSP Stapling:
Server queries (every hour) OCPS-Response itself and caches the response.
Client get OCSP response from server during TLS handshake.

This works because the OCSP-Response is signed by the CA (and therefore can't be forged by the server).

Benefits: Scalability, privacy and less dependent on the availability of OCSP-Responder. **Limitations:** vulnerable time window.

Root Certificate Revocation:
Only solution is to remove it from the system / application.
Root certificates can never be verified. Every other certificate trusted by that root certificate are considered invalid.

Let't Encrypt

Provides free X.509 certificates for **Domain Validation (DV)** (Proof of domain ownership), OV and EV cannot be automated. Most used CA of today.

Let't Encrypt verfügt aktuell über zwei Root Zertifikate. Das eine verwendet RSA und das andere Elliptische Kurven.

1. Domain Validation:
Der Agent (Antragsteller) muss einen vorgegebenen Wert von *Let't Encrypt* auf seiner Domain platzieren, den Wert mit seinem private key *A* signieren und zurückschicken. *Let't Encrypt* kann somit validieren (pub key wird beim Antrag mitgeschickt), dass der Agent im Besitz von *A* ist.

2. Certificate Signing Request (CSR):
Agent (Antragssteller) creates a CSR and signs it with private key *A*. Because *Let't Encrypt* trusts public key *A,* it will create and send the certificate.

Revocation
Agent signs revocation request with authorized private key *A*. *Let't Encrypt* verifies request with trusted public key *A* and published CRL and OCSP.

TLS and DTLS

Transport Layer Security. Aktuelle Version TLS 1.3

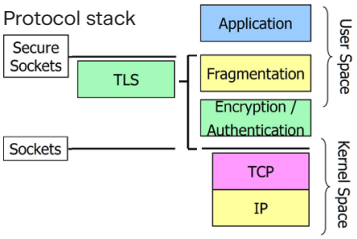
TLS sorgt für eine **sichere Datenübertragung / Kommunikation** im Web zwischen Applikationen (end-to-end). TLS ist eine Pseudoschicht zwischen Transportschicht (4) und Applikationsschicht (7). TLS is implemented in User-Space (and is directly integrated into Applications). Only works on top of **TCP**, but not UDP (because order of blocks are not guaranteed) → DTLS

TLS provides **authenticated, integrity-protected** and **confidential** data exchange. Predecessor: SSL (Secure Sockets Layer).

Securing **web communications**	(HTTP + TLS => HTTPS)
Secure **access to mailboxes**	(POP3/IMAP + TLS => POP3S)
Secure **exchange** between **mail** servers	(SMTP + TLS => SMTPS)
Secure **file up-/download**	(FTP + TLS => FTPS)

Nachteil: Kein Multifaktor-Authentizierung möglich → TLS stirbt aus.

Building Blocks



Authenticated Encryption modes (AES): GCM, CCM, ChaCha20
Diffie-Hellmann Key Agreement Exchange of secret session Keys.
Cryptographic hash function (HKDF) for key generation (AES session keys).
Public key authentication with certificates (auth through signatures).

Cipher Suites (set of cryptographic algorithms):

Example: TLS_AES_128_GCM_SHA256
- AES (Cipher Algorithm) with 128-bit keys for encryption → Confidentiality
- GCM (Galois Counter Mode): AES Mode → Integrity
- SHA256 (Hash-Function) for HMAC-based key derivation → Authenticity

TLS 1.3 Phases:

1. Handshake

Goals: Exchange of keys, agreement on cryptographic algos.
Server authenticeds toward client using certificates.

client –– ClientHello –→ server (Datagram 1)

1. Sends TLS version support and support for cryptographic algorithms: AES, 256-bit, GCM, SHA-256 for HKDF (Cryptographic hash function), etc.
2. Sends support for Diffie-Hellman and **public keys** g^s for supported groups (X25529, P-256, P-384). Uses $P = (g^s)^c$ and HKDF to generate handshake keys

client ←– ServerHello –– server (Datagram 2)

3. Sends TLS version, encryption support and **public key** g^s (e.g. X25529) Uses $P = (g^c)^s$ and HKDF to generate handshake keys.

client ←– ChangeCipherSpec –– server (optional)

Messages are encrypted from now on.

client ←– Certificate –– CertificateVerify –– server (Datagram 2)

4. Sends **certificate** chain (excl. root cert, liegt bereits beim client im Browser):
→ Client validates certificates (sig. is from issuer, validity period, revocation)

Sends **CertificateVerify** (signature over previous handshake). Proofs that server has priv key that corresponds to the pub key in the cert message.
→ Client checks signature with pub key from server certificate.
Prooves that server has private key. **Server is now authenticated.**

client ←– Finished –– server (Datagram 2)

5. Sends hash over all handshake messages.

client –– ChangeCipherSpec –→ server (optional)

client –– Finished –→ server (Datagram 3)

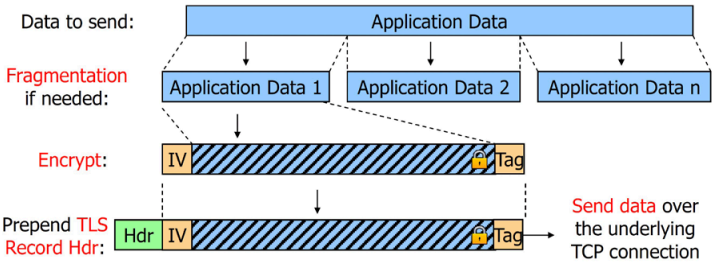
7. Client computes hash over same messages, compares them (with hash from sever) and sends them back if they match. That means the messages are not manipulated:
→ Server compares hashes and accepts communication if they match.

From here on data is encrypted with different key than the handshake: Client/server uses P and HKDF to generate traffic keys.

Session-Resumption

If there is an existing TLS session, additional TLS sessions can be established without public key computations. Client uses the **NewSessionTicket** token. Aus dem token berechnen Client und Server unabhängig ein geheimer Schlüssel für eine weitere parallele Session.
→ Security cost: Server must remember old sessions.

2. Data Exchange



Fragmentation: The data is split into multiple fragments if it is longer than what fits into the payload of an TLS record. The data may have to be **padded** in order to fit the block cipher's block size.

Auth tag: Computed using the data fragment and a sequence number. Sequence numbers: 64 bits, start at 0 and incremented for each TLS record (the underlying TCP connection guarantees ordered delivery).

Receiver: Computes its own **auth tag** and compares to received one.

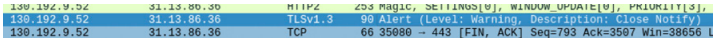
3. Connection Teardown

TLS sessions are terminated when the underlying TCP connection is closed (TCP FIN or RST), otherwise risk of a **Truncation Attack:**

Without a teardown, an attacker could target the **integrity** of the data:

In the middle of the data transfer, attacker blocks TLS record from the server and sends a TCP FIN instead. The client only receives first part of the messages and thinks TCP FIN look legitimate.

Solution: Before a teardown a **close_notify** Alert message is sent, then it sends the TCP FIN or RST. The other endpoint only accepts if close_notify was received before TCP FIN or RST.



DTLS (Datagram Transport Layer Security)

TLS over UDP.
Implicit seq. number problem is solved by using **explicit sequence number.**

TLS Handshake over UDP (Reliability Functions):
- Timeouts and retransmission to handle lost UDP datagrams
- Explicit sequence numbers for reordering

QUIC

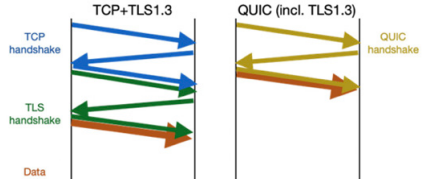
UDP, but with encryption.

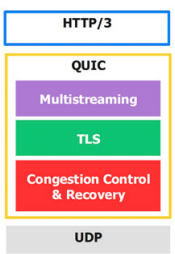
- QUIC uses TLS for the handshake.
- Encryption is baked into the protocol (it's always encrypted)
- Layer 4 (Transport Layer)
- Userspace implementation allows fast update and browser deployment
- Multistreaming, multiplexing

Every packet uses a different key which is based on the previous key.
Initial key is hardcoded (defined in the RFC).

QUIC identifies a connection based on the Server and Client **Connection ID.**
→ Connection stays alive even when the source address or port changes.

Problems with TLS:
- Latency and performance are limited
- Adding new features to TCP is difficult



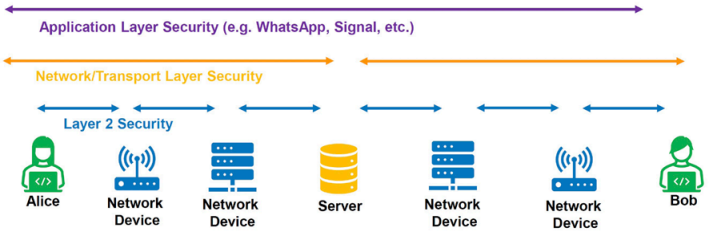


Layer 2 Security

Data Link Layer / Sicherungsschicht

5–7	Application Layer	PGP, S/MIME, whatsApp
4	Transport Layer	TLS, Quic
3	Network Layer	IPSec
2	Data Link Layer	EAP, 802.1X, WEP, WPA, WPA2, WPA3, MacSec
1	Physical Layer	Quantum ~~Cryptography~~

Modulation / Signalisierung



Goals and Threats

Port-based network access control. Protect Network from unautherized access. Originally, the internet was not designed to be secure but available.

Confidentiality: Only the communication endpoints can read the data
- Eavesdropping: Reading data in transit and obtaining credentials

Integrity: Endpoints can detect if data was manipulated in transit
- Manipulating data: Modifying data in transit
- Injecting data: Adding data to an ongoing communication relationship
- Replaying data: Sending previously recorded data again

Authenticity: Masquerading as an endpoint is not possible
- Getting unauthorised access: Using a system without privileges
- Masquerading: Pretending to be somebody else, a person or a system

Non-Repudiation: Endpoint cannot deny having sent/received data
Anonymity: Endpoints cannot identify the other (e.g. democratic election)

(Availability is **not** the responsibility of a secure communicaiton protocol.)

Limits of Secure Communication Protocols

Do not protect from:
- Software vulnerabilities (SQL injection, cross-site scripting etc.)
- Malware (viruses, worms, trojans)
- Distributed Denial of Service attacks (DDoS)

Flaws:
- Protocol design flaws (too short IVs)
- Implementation flaws (bad random number generator)
- Configuration flaws (weak crypto-algorithms allowed)
- Usability flaws (people accepting untrustworthy certificates)

Security Protocols on Higher vs Lower Layers

Higher Layer: Easier to deploy (often included in applications), **less generall** (optimised for an application), typically provides end-to-end protection.

Lower Layer: Difficult to deploy. Layer 2 only provides hop-to-hop protection, **more generall** (protects all layers on top of it).

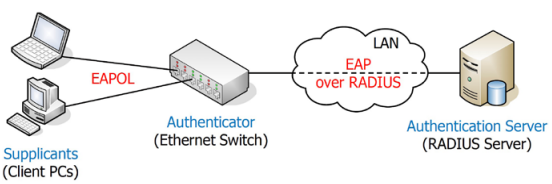
Encryption Use Cases

- **Wired Networks:** Somewhat protected, **physical access required** for packet sniffing.
- **Wireless Networks:** Easy packet sniffing. Always use encryption at layer 2 in wireless networks!

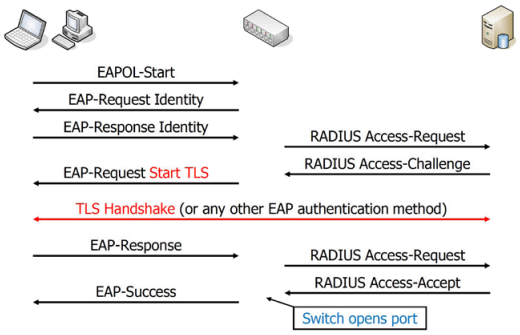
Extensible Authentication Protocol (EAP)

Only allow authenticated users/devices to your LAN network. Current recomendation is the EAP-TLS method. Der Switch forwarded speziell gekennzeichnete Authentifikations-Pakete an einen Authentifikations-Server. Vorteil: Client und Server können Authentifikation durchziehen ihrer Wahl.

Port-based network access control process:
IEEE 802.1X and **Extensible Authentication Protocol (EAP-TLS)**



1. A client connecting to a port of the switch is first blocked. Switch only accepts EAP messages. **2.** Switch forwards authentication packets to an authentication server (RADIUS). **3.** After successfull authentication between client and server, server tells the switch «authentication successful» and the switch opens port, client gets full network access.

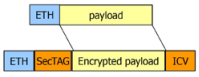


Access-Challenge: Server sends certificate chain and cipher suite.
EAP-Response: Client send certificate chain.

Schwachstelle: Die Überprüfung des Users findet nur bei einer Netzwerkverbindung statt. Ein Angreifer könnte mit physischen Access (Kabel) die bestehende Verbindung missbrauchen (Man in the middle attack).
→ MACsec löst dieses Problem.

MACsec

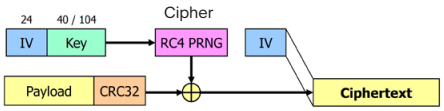
Verschlüsselt alle Daten (DHCP, ARP, etc.) im Netzwerkverkehr zwischen zwei Geräten (zwischen Switches und End device). Cipher: GCM-AES-128/256.



Medium Access Layer Security (IEEE 802.11 WLANs)

Client connect **wirelessly** to an Access Point = Broadcast:
There are **no ports** on a WLAN, everyone can listen and sniff packets.
Kann verschiedene Authentifikationsverfahren verwenden.

Frame Protection in Wired Equivalent Privacy (WEP)
All clients share a preconfigured long-term key (40 or 104 bits, very weak) with the Access Point → Every user can read the traffic of every other user.
WEP is totally insecure.

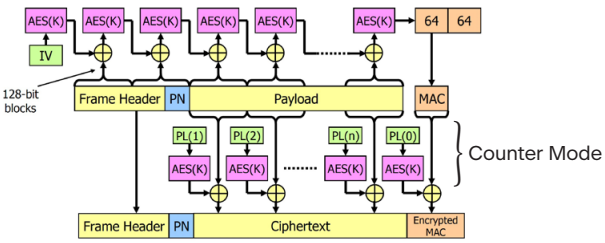


- **CRC** checksum can detect if an ecrypted frame was modified **(integrity).** But integrity can only be achieved with a real MAC (hash + key).
- Concatenating a 24-bit IV and a constant key, the RC4 PRNG can only generates 2^{24} (< 17K) different **keystreams** → **IVs** will be **repeated** and all possible keystreams will be known. Actual key is never uncovered.

WiFi Protected Access (WPA/WPA2)
Successor to WEP (IEEE 802.11i). WEP vs WPA(2): Client first have to authenticate themselves at the access point. Two encryption modes:

Temporal Key Integrity Protocol (TKIP):
(Not concidered save anymore, some weaknesses with RC4)
- Individual encryption key for each frame:
  Used to initialize RC4 (used to encrypt frame)

Counter Mode CBC-MAC Protocol (CCMP):
- Uses Advanced Encryption Standard (AES with 128-bit keys)
- Guarantees confidentiality and integrity/authenticity
- Uses block cipher



- PN (packet number) is an 48-bit value, incremented for each frame
- IV is based on PN → MAC computation is always freshly seeded
- MAC is computed by a block cipher in CBC mode = CBC-MAC
- AES(K): AES used with 128-bit encryption/integrity key
- PL(x): PN with a block counter (x) → encryption key is unique per frame

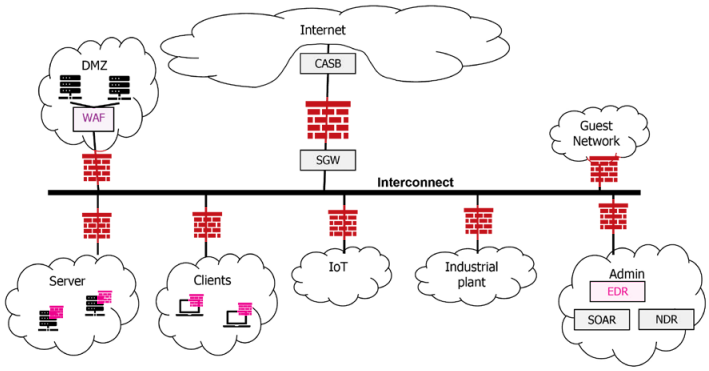
Summary

WEP and WPA/WPA2 with TKIP have major security vulnerabilities.
Today one should use **WPA3** or **WPA2 with CCMP.**

Network Security

Overview

Network Segmentaion: Deals with the variety of devices in the network.
Zero Trust: Protect assets, if an attacker is already in my network.
EDR: Protects individual devices from malicious traffic.
WAF: Protects applications from malicious traffic.
SWG: Avoids users/employees from visiting malicious websites.
CASB: Filters connections between client and cloud services.
NDR: Detect attacks plus automatic remediation (Abhilfe/Behebung).
SOAR (Security Orchestration, Automation and Response): View (dahsboard) of all security devices, generation of reports and automates responses.



DMZ: Alle Services, die vom Internet erreichbar sind. Servers suchs as Webshop, DNS, Email, etc.

Takeaways: Needs an experienced team to manage the tools and follow established processes.

Network Segmentaion

Problem: Vielfalt der Geräte in einem Netz und ihre unterschiedliche Anforderungen → **Network Segmenation:** Geräte mit ähnlichen Anforderungen werden in eigenen Netzwerksegmenten zusammengefasst. Segmente werden durch Firewalls voneinander geschützt.

Firewall = Packet Filtering:
Eine Firewall steht zwischen zwei Netzwerksegmenten (Perimeter) und entscheidet (Firewall Rule) welche Pakete weitergeleitet werden. Typischerweise werden nur headers angeschaut.

Firewalls regeln Sichtbarkeit (by hiding internal network structure):
Wer nicht sichtbar ist, kann nicht angegriffen werden.

Firewall **don't protect** from attacks on the **application layer,** e.g. Software vulnerabilities of a web server open to TCP port 80. Or an infected laptop attached to the internal network might compromise othe machine within the segment. Next Generation Firewall (NGFW) offer additional features.

Zero Trust

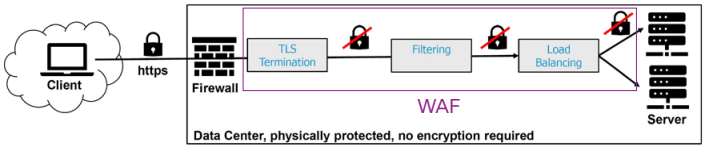
Do not trust anything unless it is authenticated/verified.
Every device needs to be known an given certain access rights.
Problem: Single Point of Failure: Misconfiguration or Denial of Service.

Endpoint Detection and Response (EDR)

Local Firewall
Ease of use: Provides a central dashboard (Firewall Admin) of security operations. Requires agents on all endpoints. **Problem:** Needs resources and attacker can compromise an agent.

Web Application Firewall (WAF)

Application Firewall. Looks into packet content and understands higher protocols (HTTP). Protects from: Cross site scripting (XSS), SQL injection, etc. Requires unencrypted traffic → TLS terminataion: Decrypts data.



Secure Web Gateway (SWG) / Forward Proxy

[LAN: Client –– [SWG] –– Firewall –]→ Internet
- Blocks known malicious URLs.
- Blocks sending sensitive information (requires tagged data).

Cloud Access Security Broker (CASB)

Between a client a cloud-service.
Connections to a cloud service will first go through a CASB:
- Can discover Shadow IT devices (unauthoroused devices not know to IT)
- Sets access rights to cloud services
- Prevents data leaks and sets policies for sharing (e.g. geolocation)
- Detection of anomaly and unusal behaviour

Implementaions:
Forward proxy: [LAN: client → proxy (local server)] → internet → [webserver]
Reverse proxy: [client] → internet → [LAN: webserver → servers]

Network Detection and Response (NDR)

Detect attacks plus automatic remediation (Abhilfe/Behebung) by changing firewall configuration and isolating infected deviced from the network. Benefits: Fast reaction to incidents (attacks). Drawbacks: Needs interfaces to other network devices to trigger response. Wrong classification lead to unnecessary actions. Security Information and Event Managment (SIEM) and SOAR: Dashboard and generation of reports (logs). Also used for compliance.

Port Scanning

Technique to determine the services that run on a host: **nmap** (port scanner)
Example, TCP scan of port 80 of www.zhaw.ch: `nmap -p80 www.zhaw.ch`

TCP: Ping to see if host is available. Establish TCP connection to the ports. If server response with TCP RST, the service is not available (closed port).
UDP: Receiving no ansers can mean (1) service is listening but does not send an answer or (2) a firewall silently drops the request. There is no way to distinguish these cases: Firewalls make port scanning difficult for attackers.

Packet-Filtering Firewalls (Linux)

nfttable rules have: **Classification** says to what packets this rule applies, and **Actions** (what to do with the packet): accept, drop (default), reject or jump (continue processing elsewehere). Reject notifies sender (drop just stops processing). A package can match many chains.

```
table inet my_filter {
 chain my_input_rules {
  type filter hook input priority 0; policy drop; # drop by default
  # allow ICMP echo-request (ping) packets from IP 10.0.1.10 on the interface ens3
  ip saddr 10.0.1.10  meta iifname ens3  icmp type echo-request  accept;
 }
 chain my_output_rules {
  type filter hook output priority 0; policy drop;
  # allow ICMP echo-reply (ping reply) to IP 10.0.1.10 through interface ens3
  meta oifname ens3  ip daddr 10.0.1.10  icmp type echo-reply  accept;
 }
 chain my_forward_rules {
  type filter hook forward priority 0; policy drop;
  ct state established, related accept; # accept established connections
  # allow ssh and ftp from 10.0.3.10 to 10.0.2.10
  ip saddr 10.0.3.10  ip daddr 10.0.2.10  tcp dport {21, 22}  ct state new  accept;
 }
}
```

```
ip saddr 10.0.1.10      Packets with source IP address 10.0.1.10
ip daddr 10.0.1.10      Packets with destination IP address 10.0.1.10
meta iifname ens3       Incoming packets through the interface ens3
meta oifname ens3       Outgoing packets through the interface ens3
icmp type echo-<type>   ICMP packets of type echo-request or echo-reply (ping)
tcp dport {21, 22}      Packets with a TCP destination port of 21 (FTP) or 22 (SSH)
ct state new            Packets that are initiating a new connection
```

Distributed Denial of Service (DDoS)

Angriff gegen die Verfügbarkeit (Availability). Für legitime Nutzer wird es nicht mehr möglich sein auf die Systeme zuzugreifen. A targeted server, service or network is fludded with by overwhelming internet traffic.

Botnets (Malware): Infected and remotely controlled devices that are used for DDoS attacks. **Identify attacks:** Suspicious traffic from a single IP address or IP range, unexplained surge in requests and odd traffic patterns.

Types of attacks:
- **HTTP-flood**
- **Protocol / State-Exhaustion:** Utilizes weaknesses in layer 3 and layer 4. Over-consuming server resources (firewalls or load balancers). Spoofed SYN packets.
- **SYN-flood / Half-Open attack:** TCP handshake-exploit. Server is repeatedly sent many initial connection requests (SYN) on all ports with with spoofed IP addresses. Server responds with SYN/ACK and waits but never receives confirmation. Attacker tries to consume all resources → Port exhaustions (Keine neuen legitimen Verbindungen werden mehr zugelassen).
- **Application Layer:** HTTP GET and HTTP POST
- **DNS Amplification:** Attacker sends request to open DNS resolvers with a spoofed IP address (of the victim). The target then receives a response from the DNS resolvers, overwhelming it with traffic.

Prevention:
- **Blackhole routing:** Funnel traffic into null-route (and then dropped)
- **Rate limiting:** Limiting the number of requests over a time window
- **Web application firewall (WAF):** Reverse proxy. Filtering requests by rules
- **Anycast network diffusion:** Scatters traffic across a network of distributed servers. Traffic is absorbed by the network.

Penetrationstests

Ziel: Tests (Automatische Scans oder manuelle Tests) versuchen mit den Methoden des Angreifers, Sicherheitsschwachstellen zu identifizieren und reduzieren (eliminieren ist fast unmöglich). Testkataloge: Open Worldwide Application Security Project (OWASP).

Testdurchführung

1. Footprinting: Aufspüren der Angriffsoberfläche: Ports, Endpunkte, etc.
2. Fingerprinting: Identifizierung der Technologien
3. Automatische Scans und Manuelle Tests (Source Code Audit, Scripts, etc.)
4. Bestätigung: Ausnutzen der identifizierten Schwachstellen zur Verifizierung

XSS (Cross-Size Scripting) / Script Injection

Bsp: `site-a.com/profile?name=Hans<script>send cookie to page B</scrip>`
`<h2>Hello $name</h2>` ← Beliebiges HTML injizierbar.

Varianten
Persistent/Stored: Z.B. Script in Blogpost injeziert. Wird jedes mal ausgeführt.
Reflected: Wird erst ausgelöst, wenn auf Link mit Injeciton geklick (Bsp. oben)
DOM/Mutational: Inhalte werden verarbeitet, so dass sie wieder zu Scriptcode zusammengebaut werden können.

Massnahmen: Regex. Escaping. Blacklist oder Whitelist of allowed chars

SQL Injection

Bsp: `site-a.com/profile?name=Hans'%20or%20'1'1='1`
Ergebnis: `SELECT name, age FROM profiles WHERE name='Hans' or '1'='1';`
Folge: Union Select aus anderen Tabellen (Passwörter)
Access, Modify and Delete information in a database.

Active command injection: System command made to the server returns the response to the user in the HTML document.
Blind command injection: Does not return the response to the user

Massnahmen: Prepared statements, sanitize the input, list of allowed chars

Insecure Direct Object Reference (IDOR)

Bsp: `dok.com/download?=id=2024100812312345`
IDs sind gut erratbar (Datum, Nutzer-ID und Dokumenten-Nr).

Massnahmen
- Universally unique ID, mit gutem Zufall, erraten ist schwer
- Access Controll / User identification (Login)

Defensive

– Principle of Least Privilege (Rechtereduction); Zero Trust

Virtual Private Networks (VPN)

Establishes a **secure** and **reliable network** connection (not individual hosts, e.g. end-to-end) **over an unsecure** network such as the internet (private/proteced network within a public network). VPN traffic is carried on public network (internet). Protects your internet activity and disguises your identity on the internet.

Virtual means that privacy (=protection) is achieved by **cryptography** and not by dedicated network links.

Use cases

– Secure connection or remote network of a company.
– Allow a partner/user selectively access internal services.

VPN Gateways: Endpoints (usually a router) of the secure channel/tunnel **applies/removes the cryptographic protection.** Secure Tunnel:
[local: IP packets → VPN Gateway] – internet → [remote: VPN Gateway → host]

IPsec Protocol

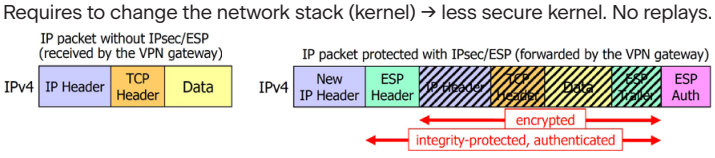
By NSA, 1998, kernel-space, on layer 3 (network layer)

TLS vs IPSec
TLS is on top of a TCP connection. **Protects communication** always between two applications (ports). TLS **keys** are only known to the applications.

IPSec **protects IP packtes** (no matter what application), e.g. communication between two hosts. IPSec **keys** are known between hosts. **Packet loss** is always possible in IPSec and must be handled by higher protocol layers.

IPSec Handshake
Key-exchange mechanism: Internet Key Exchange (IKE) ≈ TLS handshake:
1. select cryptographic algos 2. perform authentication 3. exchange keys

Requires to change the network stack (kernel) → less secure kernel. No replays.

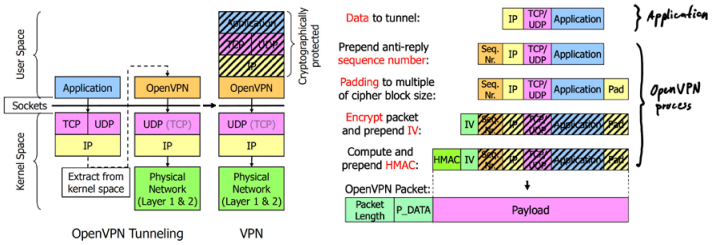


OpenVPN Protocol

Open source, 2002, runs on top of UDP or TCP, user-space
Uses TLS (only) for handshake.

Protects IP packets between VPN endpoints. Usually UDP is used.
No problem because apps can use TCP (handles data loss). But TCP over TCP would results in poor performance, tunnel has **no guarantee for data delivery.**

Protocol Stack: Instead of pushing bits out a wire, the tun-driver pushes them to user space, where the **OpenVPN daemon** receives them and processes them and builds OpenVPN packets to send data across the network to the remote OpenVPN endpoint. Whether or not an IP packet is sent to the normal physical interface (eth0) or the virtual interface for OpenVPN (tun) is determined by the routing table.



WireGuard Protocol

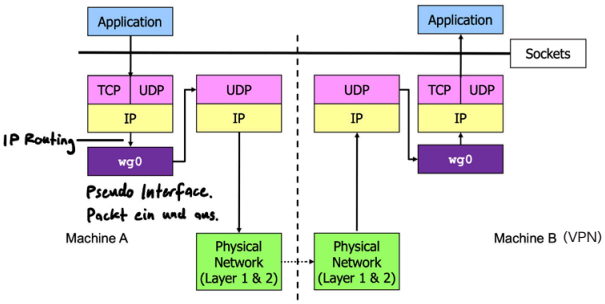
Newest protocol, kernel-space (network stack). On layer 3 (network layer)

1-RTT handshake. Perfect Forward Secrecy (PFS) = vergangener Verkehr kann nicht entschüsselt werden. Uses a Cookie for DoS-mitigation.

Message flow:

[LAN: Packet → *wg0* interface → Router] → ISP → VPN → target IP

1. Packet is routed to pseudo interface *wg0* (handled by wireguard).
2. Wireguard encrypts and wraps packet and send them to ISP.
3. ISP send them to VPN server (instead directly to target IP).
4. VPN acts as ISP, uses DNS lookup, decrypt and unwraps packet and send to target IP.



Wireguard Routing-Table Example:

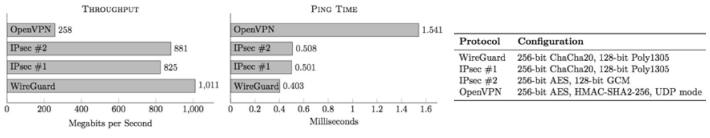
```
# my_wireguard.conf
# Local machine
[Interface]
PrivateKey = yJi5okmlRhlijEuhNoKaWW6mlrjYb1Z8ge685ZrbW2I=
# The interface listens on this port for incoming packets
ListenPort = 51820
# Remote machine
[Peer]
PublicKey = a8sFpmC9fua3pdG8qTmQ0F2VFQkpFuKcQhEzV4kYWXU=
# Remote peer is at IP 10.0.3.10 and listens on port 51820.
# The local machine sends encrypted VPN packets to this IP and port.
# If no endpoint is set, wireguard will use the packets external source address.
Endpoint = 10.0.3.10:51820
# The IP ranges (subnet) can be routed through this peer. Routing table for the VPN
# Traffic destined for 10.0.4.0/24 is securely tunneled through this peer.
AllowedIPs = 10.0.4.0/24
# Exmaple: 'ping 10.0.4.1' will be sent to the remote machine through the tunnel
# Verify Routing: $ ip route -> 10.0.4.0/24 dev wg0

# Then in bash:
# add wg0 interface
ip link add dev wg0 type wireguard
ip address add dev wg0 10.0.4.1/24
# configure routing table
wg setconf wg0 my_wireguard.conf
```

Mobile Use Case

Problem: Mobile user uses a **dynamic IP address** depending on location.
Solution: VPN gateway adds a **virtual IP address** to the mobile user after the VPN tunnel has been established.

Performance



User Authentication

Access Control Components

1. **Identification:** Who am I. Behauptung. E.g. «I am a human» → Capcha
2. **Authentication:** Proof that I am me. Password, Token, FIngeprint, Face, etc.
3. **Authorization:** Berechtigungen. Read file/write, etc.
4. **Accounting:** Audit. What the user has done. Logging accesses

Methoden zur Authentifizierung

1. Something you **know**, e.g. password
2. Something you **possess**, e.g. token, phone
3. Something you **are**, biometrisch, e.g. face, voice, etc.

Passwords

Problems: Not enough for auth. (can be guessed or bought on the dark web). And they are hard to remember because of complex password policies.
A secure system can be completely compromised by a **poorly chosen pw**.

How to store Passwords (as admin)

- Check if newly entered pw of a user is in a pw-list (haveibeenpwned.com).
- Seperate pw for each account. Long passwords only (work factor). Use MFA.
- Rate-limit password tries with waiting time.

Store passwords on a server

1. Hash. `password-file: userID, hash(password)`
Precompiled dictionary attack: Attacker can use lookup table (pw → hash)

2. Salt. Add a salt (rand val, 65–128 bits) to the pw before hashing.
`passwor-file: id, hash(password, salt), salt`
Prevents **precompiled dictionary attack.** Attacker cannot predict the salt.
But still does not prevent **dictionary attack** (offline attack).
With a 48-bit salt, this means 2^{48} hashes per password.

3. Pepper. Secret value only known to server (e.g. in a HW security module)
`passwor-file: id, hash(password, pepper, salt), salt`
Prevents **dictionary attack** (offline attack).

Computational effort to crack a password
Password with 8 characters (lower-upper case, digits and 14 special chars):
$(52 + 14 + 10)^8 = 76^8$ possibilities, Work Factor: $\log_2(76^8) \approx 50$ bits

Im Durchschnitt wird ein Suchwort nach der **Hälfte der Suchzeit gefunden.**

Key streching (iterated hashing)
Apply hash-function multiple times.
n times increaes the computation effort by a factor of *n*.
`hash = ""; for 1 to 100: hash = hashf(hash, password, salt);`

Specialised Password-hashing function: *Argon2d* or *Argon2id*
Is also a key derivation function (password in, key out).

Multi-Factor Authentication

E.g. Password + SMS/Token/Email/etc.
Does not prevent inline attacks (man in the middle through phishing attack).
Problem: MFA Fatigue. User accepts MFA-request when attacks tries to login.
Solution: Rand number at login, user (on mobile) manually enters the number.

Passwordless Authentication

Uses certificates instead of passwords. A cryptographic key is releases by a biometric factor which is then used for the login (e.g. face id, fingerprint).
FIDO2: Key is saved on an external Hardware. Complex ecosystem needed.

User Authentication protocols

Defines the procedures and messages needed to authenticate a user.

Direct User Authentication
Host/server has all information to authenticate the user
Example: local or remote login. Does not scale well.

Indirect User Authentication (= Delegated Authentication)
Host/server does not directly authenticate the user. An additional authentication server is used, e.g. RADIUS Server. Kerberos, SAML, OpenID, EAP allow **Single Sign-On:** Many websites use the same authentication server, e.g. one login for several services/applications. Drawback: centralized.

OAuth 2.0: Framework for authorization (not authentication). Doesn't use a password but an access token to allow authentication via APIs.
OpenID Connect: Does authentication. Based on OAth 2.0.
Goal: Single login for different site. Light weight (JSON). Facebook uses it.
SAML (Secureity Assertion Markup Language): For Single Sign-On in web-browsers (no support for mobile or API). Heavy weight (uses XML).
Shibboleth Protocol: All requests go through users browser (home security domain). User signs all messages for authenticity. Server rejects signed tempered messages. Issues a token to access sevices in another security domain.

Kerberos Protocol: Single Sign-On. Use of ticket. Requires synchronized time among participants (detects replay attacks based on timestamp). Each party shares a common secret with a centralised server.

1. Der Principal (user) schickt eine Anfrage an den Authentication Service (AS).
2. Der Authentication Service (KDC) sendet ein Ticket an den Principal (user).
3. Der Principal (user) schickt eine Anfrage an den Ticket-Granting-Service (TGT).
4. Der Ticket-Granting-Service (TGT) schickt ein Ticket an den Principal (user).
5. Der Principal (user) schickt ein Ticket an einen anderen Principal.

Authorization (Berechtigungen)

Authorizationis a core component of every operating system or can be performed by applications (e.g. admin vs. employee vs. client).

Building blocks: **access control model** (conceptual framework), **security policy** (who is allowed to do what), **security machanism** (actual impl., e.g. is a bit set?)

Security Policy Drivers (Motivation): **Business** (lower costs), **security** (ensure integrity), **regulatory** (comply with regulations, e.g. joiners, leavers, movers).

Discretionary Access Control (DAC)

The owner of a resource determines who is authorised to access the resource. The owner typically created the object. Most modern OS are based on DAC.
To make DAC-systems administrable → **root** user (has always full access).
Discretionary (beliebig) means capabilities can be delegated to others.

```
user, group, others   - No access, r Read(4), w Write(2), x Execute(1)
755 -rwxr-xr-x user group filename      rwx = 421(7)   r-x = 401(5)
File Type: - Regular File, d Directory, l Simbolic link, etc.
--x on a dir:  permission to traverse the dir (cd) but not list (ls).
To ls (list) I need at least r-x permission.
```

To delete or rename a file: The user needs **wx** rights on the directory.

DAC uses **Access Control List (ACL)** for implementation. **Confused Deputy Problem:** User runs a program (deputy) with more privileges, the user gets those privileges as well. Example: user passes secret.txt (designator) to an app to read. **Solution: Capabilities**, grant a program minimal rights.
capability = file descriptor. Another problem: When multiple groups require the same priviliges. (In Linux) only users can be added to groups, but not a group to another group. Instead each user needs to be managed in each group.

Access Control List (ACL)	Capabilities
+ Subjects (users, groups) are often associated with clearly identifiable entities → intuitive + Revocation is straight forward	+ No designation without authority + Principle of least authority (POLA) + Authorization is efficient (check token)
– Confused deputy problem – Authorization is inefficient	– Token can't be revoked – Determining access of objects is difficult
Delegation: By the owner or root (admin) only	Anyone having the token can pass it on

If ACL is active, it **overrides** DAC. `+` in `rw-r--r--+` indicates ACL is active.

Mandatory Access Control (MAC)

The **system** determines who is authorised to access a resource.
Centrally controlled by an admin. Very high degree of control.
Allows to implement organization-wide security policies.
Used if DAC is not enought and confidentiality is critical. Usually in combination with DAC and only used for critical parts (e.g. server processes).

Role-Based Access Conctrol (RBAC)

Authorisations are not given directly to individual users, but rather to **roles.**
Used for enterprise. Users can be assigned and given different roles.
A role defines a **set of transactions (actions)** allowed for its members.

Supports three security principles: Principle of least privilege/authority: minimum set of privileges for a role. Separation of duties: Roles must be mutually exclusive. Data abstraction: Permission is defined at a higher level.

Limitations: Not provided by the operating system. Apps must implement it.
Attribute Based Access Control (ABAC) adds attributes (time, age, location).

Rechtliche Rahmenbedingungen

Speziallisierte Rechte «überschreiben» allgemeine Rechte.
`fedlex.admin.ch` (national), `lexfind.ch` (Kantonal)

Datenschutz (öffentliches Recht)

Wahrung der Persönlichkeitsrechte bei Bearbeitung von Personendaten.
Betrifft nicht Datensicherheit (technische Massnahmen zur Gewährleistung).
Verantwortlicher ist, wer über Mittel und Zweck einer Bearbeitung entscheidet.

Personen müssen informiert werden bei Datenerhebung oder -Änderung.
«Privacy by Default» (e.g. optionale Cookies abgewählt per Default).

CH-Firmen unterliegen dem DSGVO, wenn sie Waren/Dienstleistungen in der EU anbieten. (Nicht aber, wenn Waren nur in der Schweiz geliefert werden.)

Strafrecht (öffentliches Recht)

Ziel: Schutz elementarer Rechtsgüter (Leben, Unversehrtheit, Eigentum, etc.)
Keine Bestrafung, wenn es nicht im StGB steht: «Keine Sanktion ohne Gesetz»

Try Hack Me

Broken Authentication

Cookies: The server can then keep track of users' actions and who is sending what data.

XXE: XML external entity injection (also known as XXE) is a web security vulnerability that allows an attacker to interfere with an application's processing of XML data. It often allows an attacker to view files on the application server filesystem, and to interact with any back-end or external systems that the application itself can access.

XML Prolog: <?xml version="1.0" encoding="UTF-8"?>