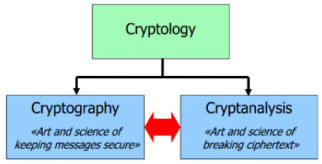
IT Security

Jil Zerndt
FS 2025

Overview of IT Security

Key IT Security Goals Fundamental CIA triad:
- **Confidentiality** - Ensuring data is only accessible to authorized users
- **Integrity** - Ensuring data is not modified in an unauthorized way
- **Availability** - Ensuring systems and data are accessible when needed



Business Continuity Management and Disaster Recovery
- **Recovery Plan** - Detailed procedures for recovering from incidents
- **Recovery Tests** - Regular testing of recovery procedures
- **Redundancy** - Duplicate systems, power supplies, and network connections
- **Offline backups** - Protection against ransomware and other threats

Security Control Frameworks provide structured approaches to implementing security controls:
- **CIS Controls** - Prioritized set of actions to protect organizations
- Controls are typically organized in implementation groups based on difficulty and impact
- Focus on preventing the most common attack vectors first

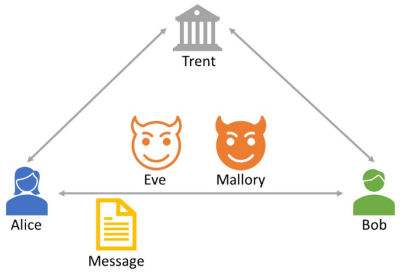
Types of Security Measures Security measures can be categorized based on their focus:
- **Preventive** - Block threats before they occur (firewalls, access controls)
- **Detective** - Identify when a breach has occurred (IDS, audit logs)
- **Corrective** - Mitigate damage after an incident (backups, incident response)

Introduction to Security and Cryptology

Cryptography Goals Cryptography aims to achieve four fundamental security properties:
- **Confidentiality and Integrity**
- **Authenticity** - Origin of information can be verified
- **Freshness/Non-repudiation** - Information is current and sender cannot deny transmission

These goals align with the broader CIA triad of information security but add authentication and non-repudiation.

Kommunikationsmodel



- **Alice** sendet eine Nachricht an Bob. Dabei sollen die oben genannten Ziele, confidentiality, integrity, authenticity, non-repudiation und freshness erfüllt werden.
- **Bob** empfängt die Nachrichten von Allice. Er will überprüfen können, dass die Ziele eingehalten wurden.
- **Eve** ist ein Attacker. Sie kann Nachrichten mitlesen, aber sie nicht verändern.
- **Mallory** ist ein anderer Attacker. Er kann Daten sowohl mitlesen als auch verändern. Er kann auch Nachrichten abfangen und später weitersenden, oder ganz verwerfen. Er kann auch neue Nachrichten generieren.
- **Trent** ist eine Drittperson/Instanz, welcher sowohl Alice und Bob vertrauen. Trent unterstützt Alice und Bob bei der sicheren Kommunikation.

Attacktypen auf Kryptosysteme Auf was hat Attacker Zugriff?
- **Ciphertext-only attack** - Zugriff auf Ciphertext, aber nicht auf Plaintext oder Schlüssel.
- **Chosen-ciphertext attack** - kann Ciphertexte generieren und diese entschlüsseln lassen. bekommt entweder den zum gewählten Ciphertext gehörenden Plaintext (daraus evtl. Rückschlüsse auf den verwendeten Schlüssel) oder Teilinformationen wie 'Verschlüsselung konnte durchgeführt werden / nicht durchgeführt werden'
- **Known-plaintext attack** - kennt sowohl Teile des Plaintext als auch dazugehörenden Ciphertext.
- **Chosen-plaintext attack** - kann Plaintexte wählen, welche er verschlüsseln lassen will. Erhält dazugehörenden Ciphertext: daraus Rückschlüsse auf andere Plaintexte oder gar Schlüssel
- **Brute-force attack** - Der Attacker probiert alle möglichen Schlüssel aus, bis er den richtigen gefunden hat (Plaintext erscheint sinnvoll)

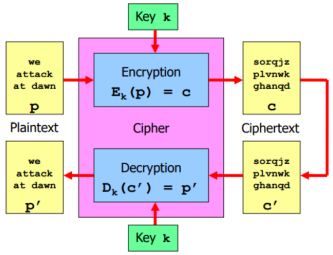
Grundsätzlich können alle Verschlüsselungsalgorithmen mittels brute-force Attacken geknackt werden. Ein wichtiges Evaluationskriterium eines Verschlüsselungsalgorithmus ist also, wie lange eine solche brute-force Attacke im Durchschnitt benötigt. Dieser Anzahl sagt man auch kryptographischer Work Factor.

Model der Verschlüsselung Um eine vertrauliche Kommunikation zu erreichen, werden Nachrichten vor dem Senden verschlüsselt und nach dem Empfangen wieder entschlüsselt.

- **Plaintext (Klartext):** Der Plaintext ist der Text, so wie er geschrieben, respektive gelesen werden kann. Er wird mit dem Buchstaben "päbgekürzt.
- **Ciphertext (Verschlüsselter Text):** Der Ciphertext ist der Text, welcher durch die Verschlüsselung entsteht. Er wird mit dem Buchstaben "cäbgekürzt.
- **Encryption (Verschlüsselung):** Die Verschlüsselung macht aus dem Plaintext den dazugehörenden Ciphertext. Dazu wird ein Schlüssel verwendet. Die Verschlüsselung kann wie folgt angegeben werden: $c = E[k](p)$. Der Verschlüsselungsalgorithmus selbst ist öffentlich bekannt und kann von allen analysiert werden, um mögliche Schwachstellen zu finden.
- **Decryption (Entschlüsselung):** Die Entschlüsselung macht aus einem Ciphertext den dazugehörenden Plaintext. Dazu wird ein Schlüssel verwendet. Die Entschlüsselung kann wie folgt angegeben werden p' = D[k](c'). Der Entschlüsselungsmechanismus ist öffentlich bekannt und kann von allen analysiert werden, um mögliche Schwachstellen zu finden.
- **Key (Schlüssel):** Nur mit dem richtigen Schlüssel, kann eine Nachricht richtig entschlüsselt werden. Je nach Art der Verschlüsselung wird derselbe Key für die Verschlüsselung und die Entschlüsselung verwendet (Secret Key Kryptographie) oder es werden unterschiedliche Schlüssel verwendet (Public Key Kryptographie). Damit die Verschlüsselung sicher ist, muss der Schlüssel, welcher für die Entschlüsselung gebraucht wird, geheim bleiben.

Encryption Encryption is a transformation from P (plaintexts) to C (ciphertexts) under control of some key, chosen from K (key space).
The idea is to have a whole family of transformations, where each key gives a different transformation.
- $c = E(k, p) = E_k(p)$
- $p = D(k, c) = D_k(c)$

Each transformation must be reversible. It follows that $|C|$ can not be smaller than $|P|$. Meistens: $P = C$.



Cryptographic Work Factor

Work Factor The number of times it takes to come upon the correct key is called (cryptographic) work factor.
- Work factor (WF) = average number of keys to try
- Work factor is usually given in bits: $\log_2(n)$, $n = key\ space\ size$

Entropy of Cryptographic Keys
Cryptographic keys are typically created with random generators, so they can be considered as elements of a random variable.
What's the entropy of a key with 128-bit?
- Independent with equal probability: $128 \cdot 1 = 128$ bits
- Dependent with inequal probability: $p(0) = 0.25, p(1) = 0.75 \rightarrow 128 \cdot 0.81 \approx 104$ bits

Entropy Entropy is maximal if all outcomes are equally likely

$$H = \sum_{i=1}^{n} p(i) \cdot \log_2\left(\frac{1}{p(i)}\right)$$

$$H_{binary} = \sum_{i=1}^{n} \frac{1}{n} \log_2(n) = \log_2(n)$$

Relation between entropy and work factor
Work Factor ≈ entropy, key size = max. entropy ≈ max. work factor

Computationally secure Work factor ≈ key entropy

Information-theoretically secure Intercepting a ciphertext tells you nothing about the plain

Secret Key Cryptography

Perfect Secrecy A cryptographic system has perfect secrecy if for all possible plaintexts p and ciphertexts c:

$$P[p|c] = P[p]$$

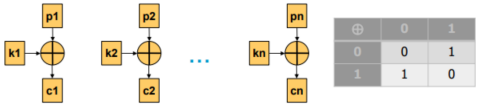
This means that observing the ciphertext provides no information about the plaintext.

Limitations of One-Time Pad

- Key must be as long as the message
- Key must be completely random
- Key must be pre-shared securely
- Key can only be used once

One-Time Pad (Vernam Cipher) The only known encryption scheme with perfect secrecy:

- Key must be completely random and at least as long as the message
- Encryption: $c_j = p_j \oplus k_j$ for all bits j
- Each key can only be used once
- If reused: $P_1 \oplus P_2 = C_1 \oplus C_2$ (major vulnerability)



⊕	0	1
0	0	1
1	1	0

Modern Symmetric Encryption

Modern Cipher Properties Since perfect secrecy is impractical, modern ciphers aim for computational security:

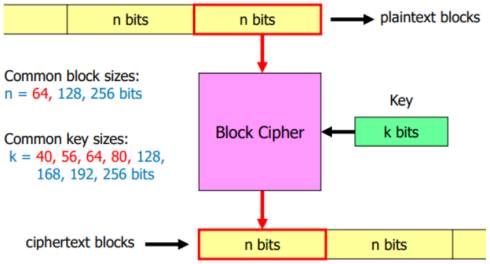
- Publicly known algorithms
- No known practical attacks
- Work factor $> 2^{128}$
- Use standardized implementations only

Overview of AES Modes AES can operate in different modes to handle messages longer than single block (16 bytes):

- **Confidentiality-only modes**: ECB, CBC, CFB, OFB, CTR
- **Authenticated encryption modes**: GCM, CCM, OCB
- Each mode has different security properties and use cases

Block Ciphers encrypt fixed-size blocks of data:

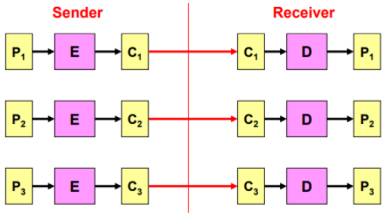
- Require padding for messages not multiple of block size
- Use modes of operation for longer messages
- Examples: DES (insecure), Triple-DES (legacy), AES (current standard)



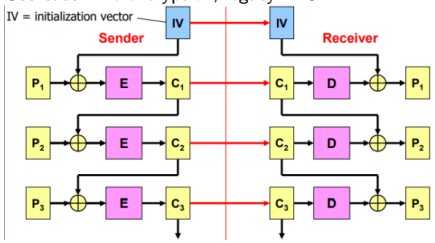
Electronic Code Book (ECB) Mode

simplest mode each block encrypted independently

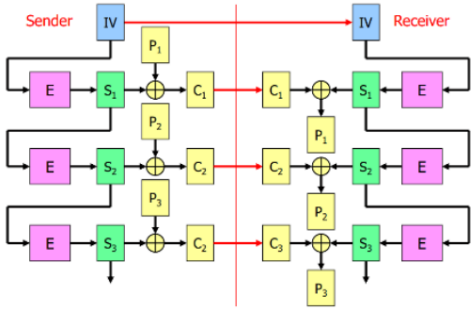
- $C_i = E_K(P_i)$ for each plaintext block P_i
- **Major weakness**: Identical plaintext blocks produce identical ciphertext blocks
- Patterns in plaintext remain visible in ciphertext
- **Use case**: Almost never - only for single blocks or random data



Cipher Block Chaining (CBC) Mode Each plaintext block XORed with previous ciphertext block before encryption:
$C_i = E_K(P_i \oplus C_{i-1})$, $IV = C_0$
Advantages: Identical plaintext blocks produce different ciphertext
Disadvantages: Sequential encryption, padding oracle attacks possible
Use case: File encryption, legacy TLS



Output Feedback (OFB) Mode Creates keystream by repeatedly encrypting the previous output:
$O_i = E_K(O_{i-1})$, $C_i = P_i \oplus O_i$, with $O_0 = IV$
Advantages: No padding needed, self-synchronizing
Disadvantages: Sequential operation, no integrity protection
Use case: Stream-oriented transmission (rarely used today)

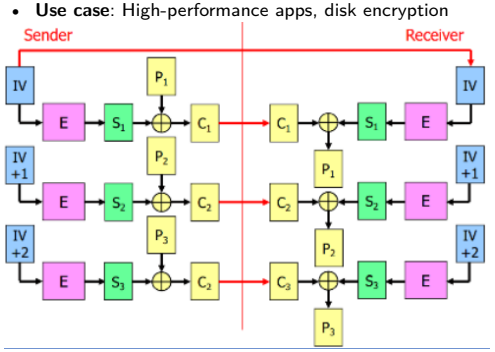


Mode Selection Guidelines

Never use ECB - except for single blocks
Use GCM - when you need both encryption and authentication
Use CBC - for compatibility with legacy systems (with HMAC for authentication)
Use CTR - when you need parallel processing and will add authentication separately
Avoid OFB/CFB - unless specifically required by legacy protocols

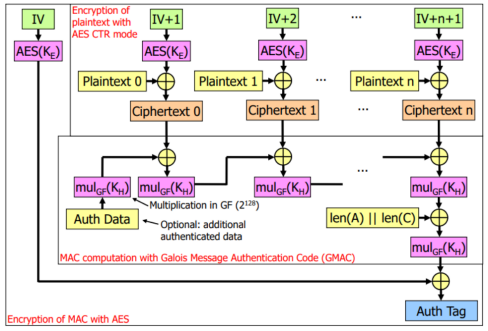
Counter (CTR) Mode Converts block cipher into stream cipher by encrypting a counter:

- $C_i = P_i \oplus E_K(\text{IV} + i)$
- **Advantages**: Parallel encryption/decryption, no padding needed
- **Disadvantages**: CNT must never repeat with same key
- **Use case**: High-performance apps, disk encryption



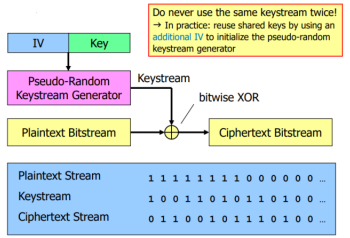
Galois/Counter Mode (GCM) Authenticated encryption mode combining CTR mode with Galois field multiplication:

- Provides both confidentiality and authenticity
- $C_i = P_i \oplus E_K(\text{IV} + i)$, plus authentication tag
- **Advantages**: Parallel operation, built-in authentication, high performance
- **Use case**: Modern TLS, IPsec, wireless protocols



Stream Ciphers Stream ciphers encrypt data bit-by-bit or byte-by-byte:

- No need to wait for complete blocks
- Generally faster than block ciphers
- Most historical stream ciphers are broken (RC4, A5/1, A5/2)
- ChaCha20 is the current standard for stream ciphers



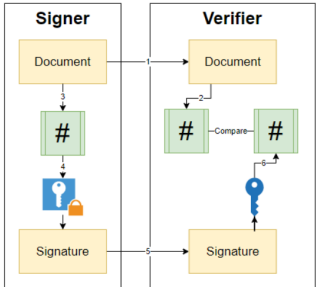
Hash Functions and Message Authentication

Cryptographic Hash Function A function that takes arbitrary-length input and produces fixed-length output:

- **Efficient computation** - Fast to compute
- **Preimage resistance** - Hard to find input for given output
- **Collision resistance** - Hard to find two inputs with same output
- **Pseudo-random output** - Small input changes cause large output changes

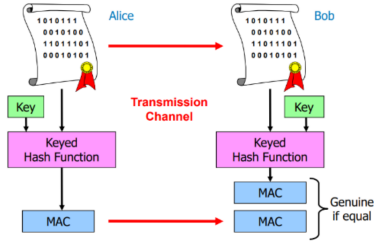
Digital Signatures vs. MACs

- **MACs**: Fast, require shared secret, provide authentication
- **Digital Signatures**: Slower, use public keys, provide non-repudiation
- Signatures enable verification by anyone with the public key
- MACs only verifiable by parties with the shared secret



Message Authentication Code (MAC) Combines hash functions & secret keys for message integrity:

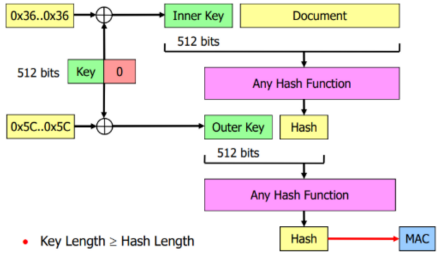
- Detects unauthorized message modifications
- Provides authentication of message origin
- Requires shared secret key between sender and receiver
- HMAC is the most common MAC construction



HMAC Construction

Hash-based Message Authentication Code:
$\text{HMAC}(K, M) = H((K \oplus \text{opad})||H((K \oplus \text{ipad})||M))$

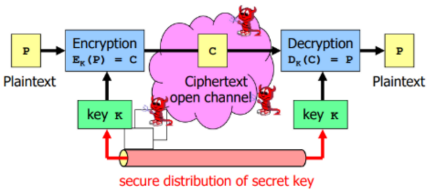
- Uses nested hash function calls with key padding
- Secure (if with cryptographic hash functions)
- Resistant to length extension attacks



Public Key Cryptography

Key Distribution Challenge Secret key cryptography faces significant scalability issues:

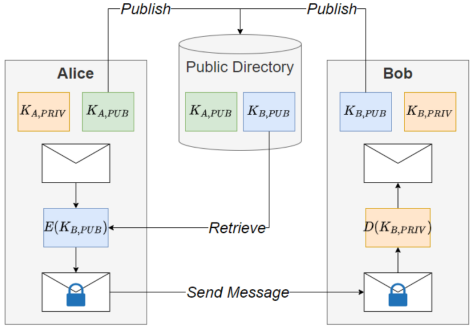
- Keys must be exchanged securely before communication
- Difficult to establish keys with unknown parties
- n users require $\frac{n(n-1)}{2}$ key pairs for complete connectivity
- Key management complexity grows quadratically



Public Key Principles

Public Key Cryptography Uses mathematically related key pairs for encryption and decryption:

- **Public Key** - Known to everyone, used for encryption
- **Private Key** - Known only to owner, used for decryption
- Keys mathematically related: $D(E(M)) = M$
- Private key cannot be feasibly computed from public key



Diffie-Hellman Protocol

Setup

Alice and Bob agree on public parameters:

- Large prime p
- Generator g of the multiplicative group $\mathbb{Z}_p^*$

Key Generation

- Alice chooses private a, computes $A = g^a \bmod p$
- Bob chooses private b, computes $B = g^b \bmod p$
- They exchange A and B publicly
- Shared secret: $S = B^a = A^b = g^{ab} \bmod p$

Alice	Public	Bob
a	g, p	b
$A = g^a \bmod p$	$A \quad B$	$B = g^b \bmod p$
$SA = (g^b)^a \bmod p$		$SB = (g^a)^b \bmod p$

RSA Cryptosystem Based on the computational difficulty of factoring large numbers:

- Public Key: (n, e) where $n = p \cdot q$ (product of large primes)
- Private Key: (n, d) where $ed \equiv 1 \pmod{\varphi(n)}$
- Encryption: $c = m^e \bmod n$
- Decryption: $m = c^d \bmod n$

Digital Certificates

Digital Certificates A digital certificate is a signed statement by a trusted third party (Certificate Authority) that a certain public key belongs to a certain entity (person, organization, or system). Certificates are used to authenticate public keys, not the entities themselves.

Certificate Revocation and Transparency

Certificate Revocation process of invalidating a certificate before its expiration date, typically due to:

- Compromise of the associated private key
- Change in the certificate owner's status
- Cessation of operation by the certificate owner

Certificate Revocation Mechanisms

- **Certificate Revocation Lists (CRLs)** - Lists of revoked certificates published by CAs
- **Online Certificate Status Protocol (OCSP)** - Protocol for real-time certificate status checking
- **OCSP Stapling** - Server includes pre-fetched OCSP response during TLS handshake
- **OCSP Must-Staple** - Requires OCSP stapling for a certificate to be considered valid
- **Browser-Summarized CRLs** - Browser vendors compile and compress CRLs for distribution to browser instances

Revocation Checking Methods

- **CRLs**: Download revocation list from CA and check certificate serial number
- **OCSP**: Real-time query to CA for specific certificate status
- **OCSP Stapling**: Server pre-fetches OCSP response and includes it in TLS handshake

Each method has trade-offs between performance, privacy, and reliability.

Certificate Transparency Certificate Transparency (CT) is a framework designed to detect and prevent the fraudulent issuance of certificates:

- Publicly auditable logs of all issued certificates
- Monitors check logs for suspicious certificates
- Browsers require evidence that certificates are logged
- Helps detect malicious or mistakenly issued certificates

Authentication of Public Keys

The fundamental problem certificates solve is how to reliably distribute public keys in an insecure network:

- Without certificates, it's difficult to know if a public key truly belongs to the claimed entity
- Certificates bind identities to public keys through the endorsement of a trusted third party
- The binding is authenticated through cryptographic signatures

Certificate Types

TLS Certificates:
Domain Validation (DV) - Validates domain ownership only
Organization Validation (OV) - Validates domain ownership and organization details
Extended Validation (EV) - Provides the highest level of validation
Code signing certificates - Authenticate software and files
Client certificates - Identify individual users or machines

Public Key Infrastructure (PKI) PKI is the framework that manages digital certificates:

- **Certificate Authorities (CAs)** - Issue and manage certificates
- **Registration Authorities (RAs)** - Verify identities before certificate issuance
- **Certificate stores** - Store trusted root certificates
- **Revocation systems** - Handle certificate invalidation

X.509 Standard X.509 certificates are described in ASN.1 notation and typically encoded using DER (Distinguished Encoding Rules).

X.509 Certificate Fields

Version - Certificate format version (v1, v2, or v3)
Serial Number - Unique identifier assigned by issuing CA
Signature Algorithm ID - Algorithm to sign the certificate
Issuer Name - X.500 Distinguished Name of issuing CA
Validity Period - Start and end dates for certificate validity
Subject Name - X.500 Distinguished Name certif. owner
Subject Public Key Info - public key& algorithm identifier
Extensions - Additional fields (e.g., key usage)
Certificate Signature - The digital signature created by CA

Certificate Chains Certificates form chains of trust:

- **End entity certificates** - Used by servers, clients, or devices
- **Intermediate CA certificates** - Issued by root CAs to intermediate CAs
- **Root CA certificates** - Self-signed certificates from trusted root authorities

To verify a certificate, each certificate in the chain must be validated up to a trusted root.

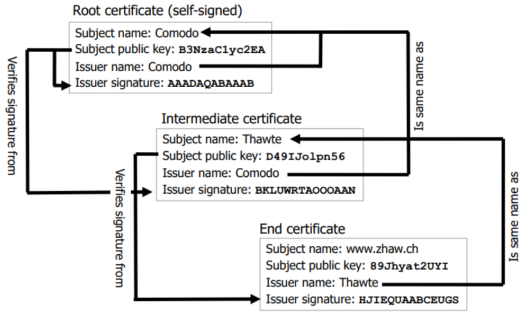
Certificate Validation Process Certificate validation follows a simple chain of trust:

- Each certificate's issuer must match the next certificate's subject
- Each certificate's signature must be verifiable by the next certificate's public key
- The chain must end with a trusted root certificate
- Certificate must be within its validity period and not revoked

Root Certificate Trust

Root certificates are the foundation of trust in PKI:

- Root certificates are pre-installed in operating systems and browsers
- They're the ultimate trust anchors that enable the validation of all other certificates
- The security of PKI depends on the protection of root CA private keys



Transport Layer Security (TLS)

Transport Layer Security (TLS) TLS is a cryptographic protocol designed to provide secure communication over a computer network. It provides:

- **Confidentiality** - Protection against eavesdropping
- **Integrity** - Detection of message tampering
- **Authentication** - Verification of communicating parties
- **Protection against replay attacks** - Prevention of message reuse

TLS is the successor to Secure Sockets Layer (SSL).

TLS Building Blocks TLS 1.3 relies on several cryptographic building blocks:

- Strong block ciphers (AES, ChaCha20)
- Authenticated encryption modes (GCM, CCM, Poly1305)
- Diffie-Hellman key exchange (including elliptic curve variants)
- Public key authentication with certificates
- Cryptographic hash functions for key derivation (HKDF)

TLS Protocol Layers TLS consists of multiple protocol layers:

- **TLS Record Protocol** - The base layer that defines packet format
- **Handshake Protocol** - Negotiates cryptographic parameters and authenticates parties
- **Alert Protocol** - Communicates errors and warnings
- **Application Data Protocol** - Transmits encrypted application data
- **Change Cipher Spec Protocol** - (Deprecated in TLS 1.3, retained for backward compatibility)

TLS Protocol Phases TLS operates in three distinct phases:

- **Handshake** - Establishes cryptographic parameters and authenticates parties
- **Data Exchange** - Transfers protected application data
- **Connection Teardown** - Securely terminates the connection

TLS Protocol Stack

TLS 1.3 Handshake Goals The TLS 1.3 handshake accomplishes several goals:

- Negotiate cryptographic algorithms
- Perform Diffie-Hellman key exchange
- Generate handshake keys for encryption of subsequent messages
- Authenticate the server (and optionally the client)
- Verify message integrity during the handshake
- Generate keys for data encryption

TLS 1.3 Handshake Process

Initial Exchange

- Client sends ClientHello message with:
  - Supported TLS versions
  - Supported cipher suites
  - Supported key exchange groups
  - Client's key shares for preferred groups
- Server responds with ServerHello message:
  - Selected TLS version
  - Selected cipher suite
  - Server's key share for selected group

Key Derivation

- Both parties compute shared secret using Diffie-Hellman
- Handshake keys are derived using HKDF with the shared secret
- All subsequent handshake messages are encrypted

Server Authentication Server sends...

- Certificate message with certificate chain
- CertificateVerify with signature over handshake transcript
- Finished message with MAC over handshake transcript

Client Verification Client...

- validates server certificate
- verifies server's signature in CertificateVerify
- verifies server's Finished message
- sends its own Finished message

Application Data

- Both parties derive application data keys
- Encrypted application data can be exchanged

TLS Record Protection protects app data

- Data is fragmented into manageable chunks
- Each fragment is encrypted and authenticated using AEAD ciphers
- A sequence number is included in the authentication to prevent replay attacks
- TLS records include headers that are not encrypted but are authenticated

TLS Session Resumption Session resumption allows clients to reconnect to servers more efficiently:

- Avoids the computational cost of public key operations
- Reduces the number of round-trips required
- In TLS 1.3, uses pre-shared keys (PSKs) derived from previous connections
- Server provides NewSessionTicket messages containing ticket data
- Client can use this ticket in future connections

TLS Connection Termination

- Uses close_notify alerts to signal completion of data transmission
- Prevents truncation attacks where an attacker prematurely terminates a connection
- Both parties should send close_notify alerts before closing the underlying TCP connection

Datagram Transport Layer Security (DTLS)

DTLS is an adaptation of TLS for datagram transport protocols like UDP:

- Based on TLS with minimal modifications for unreliable transport
- Uses explicit sequence numbers to handle packet reordering
- Implements message loss detection and retransmission for handshake messages
- Provides optional replay detection for application data
- Current version is DTLS 1.3 (RFC 9147)

Layer 2 Security

Security at Different OSI Layers

- **Physical Layer (Layer 1)** - Quantum cryptography, physical isolation
- **Data Link Layer (Layer 2)** - IEEE 802.1X, WLAN security (WEP, WPA, WPA2, WPA3), MACsec
- **Network Layer (Layer 3)** - IPsec
- **Transport Layer (Layer 4)** - TLS, QUIC
- **Application Layer (Layer 7)** - PGP, S/MIME, Signal, WhatsApp

Layer Selection Tradeoffs

Higher Layer Security:

- Easier to deploy (often included in applications)
- Typically provides end-to-end protection
- Less generally applicable (often optimized for specific applications)

Lower Layer Security:

- More difficult to deploy (may require kernel, router, or switch modifications)
- Often provides only hop-to-hop protection at layer 2
- More generally applicable (protects all protocols above it)

Extensible Authentication Protocol (EAP) framework for authentication that supports multiple authentication methods:

- Provides a flexible authentication mechanism
- Designed to be transport-independent (can run over various protocols)
- Allows network access control without knowledge of each individual user
- Supports numerous authentication methods via different EAP types

EAP Authentication Methods EAP supports numerous authentication methods with varying security properties:

- **EAP-MD5** - Simple challenge-response mechanism (insecure, no mutual authentication)
- **EAP-TLS** - Uses TLS protocol with client certificates (strong security)
- **EAP-TTLS** - Tunneled TLS without requiring client certificates
- **PEAP** - Microsoft's Protected EAP, similar to EAP-TTLS
- **EAP-FAST** - Cisco's Flexible Authentication via Secure Tunneling

EAP Packet Format EAP packets have a simple structure:

```
struct eap_packet {
    uint8_t code;       // Request (1), Response (2), Success (3), Failure (4)
    uint8_t id;         // Used to match requests and responses
    uint16_t length;    // Total length of the EAP packet
    uint8_t type;       // When code=1,2: authentication method type
    uint8_t data[];     // Type-specific data
};
```

IEEE 802.1X Port-Based Access Control

IEEE 802.1X port-based network access control:

- Controls access to a network by authenticating devices before allowing network access
- Prevents unauthorized access through publicly accessible network ports
- Uses EAP for authentication
- Provides centralized VLAN assignment based on authentication

802.1X Components

- **Supplicant** - The client device requesting network access
- **Authenticator** - The network device (switch, access point) controlling port access
- **Authentication Server** - Typically a RADIUS server that validates user credentials

Security Limitations of 802.1X While 802.1X provides significant security benefits, it has limitations:

- Man-in-the-middle attacks are possible when an attacker inserts themselves between a legitimate device and the switch
- An attacker could monitor traffic by creating a bridge between the legitimate device and the switch
- Once initial authentication is complete, the ongoing session is not continuously verified
- These limitations are addressed in 802.1X-2010 with the use of MACsec

MACsec IEEE 802.1AE (Media Access Control Security)

- Data confidentiality, integrity, and authenticity on layer 2 (Ethernet)
- Protection for all data including DHCP, ARP, and higher layer protocols
- Security for both physical and virtual links between devices
- Line-rate encryption in hardware implementations

MACsec Operation MACsec operates by:

- Encrypting the payload of Ethernet frames between adjacent devices
- Adding a SecTAG to identify the keys and provide packet numbering
- Integrity Check Value (ICV) based on GCM-AES
- Providing hop-by-hop protection where each device on the path decrypts and re-encrypts the data

WPA/WPA2 Operation

- **Authentication Phase**:
  - WPA-Personal: Uses pre-shared key (PSK)
  - WPA-Enterprise: Uses 802.1X and EAP with a RADIUS server
- **Key Exchange Phase**:
  - Establishes pairwise transient keys for unicast traffic
  - Establishes group keys for broadcast/multicast traffic
  - Implements key rotation to prevent attacks

WLAN Security

WLAN Security Challenges

- No physical cable requirement for packet sniffing
- Signals extend beyond physical boundaries
- Difficult to control who can attempt to connect
- Need for strong encryption and authentication

Wired Equivalent Privacy (WEP) OG security standard for 802.11 networks:

- Shared key between all clients and the access point
- Uses RC4 stream cipher with 40 or 104-bit keys
- Uses a 24-bit Initialization Vector (IV)
- Includes a CRC-32 checksum for integrity checking

WEP Security Flaws

- Small IV space (24 bits) leads to inevitable IV reuse
- Weak key scheduling algorithm in RC4
- CRC-32 is not cryptographically secure for integrity protection
- No protection against bit-flipping attacks
- No replay protection
- No per-user/per-session keys

Wi-Fi Protected Access (WPA) interim solution while waiting for 802.11i (WPA2):

- Introduced the Temporal Key Integrity Protocol (TKIP)
- Provided per-packet key mixing
- Added message integrity code (Michael)
- Implemented a frame counter to prevent replay attacks
- Available in personal (PSK) and enterprise (802.1X) modes

WPA2 (IEEE 802.11i) complete implementation of the IEEE 802.11i standard:

- Introduced Counter Mode with CBC-MAC Protocol (CCMP)
- Based on AES encryption (replacing RC4)
- Provides stronger authentication, integrity, and confidentiality
- Retained TKIP only for backward compatibility
- Available in personal (PSK) and enterprise (802.1X) modes

WPA3 latest Wi-Fi security standard:

- Introduces Simultaneous Authentication of Equals (SAE) to replace WPA2-Personal PSK
- Provides forward secrecy
- Protects against offline dictionary attacks
- Includes enhanced protection for enterprise networks
- Improves encryption for public networks with Opportunistic Wireless Encryption (OWE)

Network Security

Network Segmentation dividing a computer network into smaller, isolated segments:
- Increases operational performance by containing traffic
- Limits damage from cyber attacks to specific segments
- Protects vulnerable devices by isolating them
- Reduces the scope of compliance requirements
- Protects against insider threats

Typical Network Segments
DMZ (Demilitarized Zone) - Hosts public-facing services
Server Network - Hosts internal servers and applications
Client Network - Hosts employee workstations
Guest Network - Hosts visitor devices with limited access
IoT Network - Hosts Internet of Things devices
Industrial Network - Hosts industrial control systems
Admin Network - Hosts administrative systems and tools

Zero Trust Challenges
- Single points of failure in policy enforcement
- Potential for accidental or malicious misconfigurations
- Vulnerability to denial-of-service attacks
- Difficulties with encrypted traffic visibility
- Lack of protection against credential theft

Microsegmentation
- Segments as small as individual workloads or applications
- Implemented through network virtualization or host-based firewalls
- Limits lateral movement within a traditional network segment
- Reduces the attack surface and blast radius
- Balances security with management complexity

Zero Trust Architecture assumes no implicit trust within or outside the network:
Core Principles:
- Verify explicitly - Always authenticate and authorize
- Use least privilege access - Provide minimal necessary access
- Assume breach - Operate as if attackers are already present

Components:
- Policy Enforcement Point (PEP) - Enforces access decisions
- Policy Decision Point (PDP) - Makes access decisions
- Policy Administration Point (PAP) - Manages policies

Firewalls

Packet Filtering Firewalls control traffic flow between network segments:
- Examine packet headers to make filtering decisions
- Apply rules based on source/destination addresses, ports, and protocols
- Operate at the network and transport layers of the OSI model
- Form the foundation of network security architecture

Firewall Rule Management
- **New Request** - Submission, approval, design, testing, deployment, validation
- **Operation** - Monitoring, audits, reports
- **Recertification** - Regular review of existing rules
- **Decommissioning** - Removal of unnecessary rules

Without proper management processes, firewall rule sets tend to grow uncontrolled and become difficult to understand.

Firewall Benefits and Limitations
Benefits: Block unwanted traffic before it enters protected networks, Centralize access control, Hide internal network structure
Limitations: Primarily perimeter protection (assumes threats are external), Cannot prevent internal threats, Basic packet filtering cannot detect application-layer attacks, Cannot protect against allowed protocols misuse

Next Generation Firewalls (NGFW) additional features:
Deep Packet Inspection Examines packet contents
Application Awareness IDs/controls traffic by application
Intrusion Prevention Detects and blocks network attacks
Antivirus/Anti-malware Scans traffic for malicious content
Sandboxing exec suspicious files in isolated environments
Threat Intelligence inc. external information about threats

Host-Based Firewalls additional layer of protection:
- Operate directly on individual devices
- Protect against threats that bypass network firewalls
- Filter traffic based on local policies
- Examples include Windows Defender Firewall, Linux nftables, and macOS firewall

Endpoint Detection and Response (EDR) combine several capabilities to secure endpoints:
Monitoring: Anomaly detection, vulnerability scanning, integrity checks
Protection: Host-based firewall, antivirus, app control
Investigation and Response: device isolation, user session termination, change rollback, evidence collection

Web Application Firewalls (WAF) protect web applications from attacks:
- Examine HTTP/HTTPS traffic at the application layer
- Protect against OWASP Top 10 vulnerabilities: Cross-Site Scripting (XSS), SQL Injection, Cross-Site Request Forgery (CSRF)
- Require TLS termination to inspect encrypted traffic
- Deploy in front of web servers

Secure Web Gateway (SWG) protect users from web-based threats:
- URL filtering to block malicious sites
- Data Loss Prevention (DLP) to prevent sensitive data exfiltration
- TLS inspection to examine encrypted traffic
- User and application control

Network Detection and Response (NDR) monitor network traffic to detect and respond to threats:
- Establish baseline network behavior
- Detect anomalies that may indicate attacks
- Analyze potential incidents to identify true positives
- Automatically respond to confirmed threats

Cloud Access Security Broker (CASB) provide security controls for cloud services:
Shadow IT discovery ID unauthorized cloud service usage
Cloud usage control Set access rights to cloud services
Data leakage prevention ctrl data sharing in cloud services
Anomaly detection Alert on unusual behavior patterns
Implementation methods:
- API scanning - Directly interfaces with cloud providers
- Forward proxy - Controls outbound access to cloud services
- Reverse proxy - Intermediates between users and cloud services

Security Information and Event Management (SIEM) collect and analyze security events:
- Aggregate logs from multiple sources
- Correlate events to identify attack patterns
- Provide a centralized dashboard for security monitoring
- Generate reports for compliance and security analysis

Security Orchestration Automation and Response (SOAR) extend SIEM capabilities:
- Automate security response with playbooks
- Integrate with multiple security tools
- Orchestrate complex security workflows
- Enhance incident response with automation

Linux Firewall with nftables

nftables Framework nftables is a packet classification and filtering framework in Linux:
- Replaces the legacy iptables framework
- Part of the Linux kernel since version 2.4
- Provides packet filtering, network address translation, and packet mangling
- Configured using the nft command-line tool

nftables Architecture
- **Tables** - Group chains for a specific packet type (address family)
- **Chains** - Group rules and attach to hooks in the network stack
- **Rules** - Define matching criteria and actions for packets
- **Expressions** - Match packet properties (addresses, ports, etc.)
- **Actions** - Determine what happens to matching packets (accept, drop, reject, etc.)

Basic nftables Commands

```
# Create a table
nft add table inet filter
# Create a chain in the table
nft add chain inet filter input { type filter hook input priority 0 \; policy drop \; }
# Add a rule to the chain
nft add rule inet filter input tcp dport 22 accept
# List the ruleset
nft list ruleset
# Delete a rule (using its handle)
nft delete rule inet filter input handle 4
# Flush a table (delete all chains and rules)
nft flush table inet filter
```

Port Scanning discover available services on a network:
Identifies open ports on target systems, Helps determine running services and potential vulnerabilities, Used by both attackers for reconnaissance and administrators for security testing, Most commonly performed using tools like nmap

netfilter/nftables netfilter is a mechanism that allows to access the packets in the network stack to analyze, modify, extract, and delete them.
- Hooks that are called at different points during packet processing

nftables rulesets (applied to packets):
- **Table** Container for specific type of package
- **Chain** Container with rules for a specific hook
- **Hook** Called at different points of processing
- **Priority** Lowest priority first until rule is accepted
- **Policy** Default behaviour
- **Rule** Classification + Action
- **Classification** What packets does a rule apply to
- **Action** What to do with the packet

Basic Port Scanning with nmap

```
# Scan a single host for common ports
nmap 192.168.1.1
# Scan specific ports
nmap -p 22,80,443 192.168.1.1
# Scan an entire subnet
nmap 192.168.1.0/24
```

nftables Rules

```
# Drop all packets going to IPv4 address
    8.8.8.8
ip daddr 8.8.8.8 drop
# Accept all packets coming on interface eth2
iifname eth2 accept
# Accept all IPv6 packets carrying TCP
ip6 nexthdr tcp accept
```

Virtual Private Networks

Virtual Private Network (VPN) private network created within a public network infrastructure:
- **Private** - External parties cannot read or modify transmitted data
- **Virtual** - Privacy is achieved through cryptography, not dedicated links
- Creates an encrypted tunnel between endpoints
- Allows secure access to resources across untrusted networks

VPN Core Concepts
- VPNs connect networks (or a host to a network), not just individual hosts
- VPN endpoints (gateways) establish and maintain the secure tunnel
- Internal hosts typically require no special configuration
- Traffic is encrypted between VPN endpoints
- VPNs often hide internal addressing with NAT

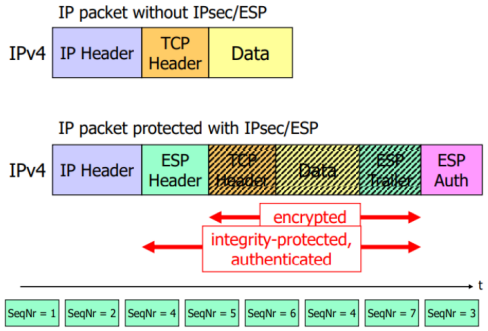
VPN Protocol Types
IPsec - IP Security protocol suite operating at the network layer
OpenVPN - SSL/TLS-based solution running at the application layer
WireGuard - Modern, high-performance VPN protocol
Proprietary protocols - Vendor-specific implementations

Site-to-Site VPN connect entire networks:
branch offices to HQ, partner organizations' networks, implemented using VPN gateways at each location, transparent to end users, typically always-on connections

Remote Access VPN connect individual users to network:
enable employees to work remotely, allow access to internal resources from outside, implemented using VPN client software on user devices, often use dynamic IP addressing with virtual addresses, typically on-demand connections

IPsec VPNs

IPsec Components
- **Authentication Header (AH)** - Provides authentication and integrity (rarely used)
- **Encapsulating Security Payload (ESP)** - Provides encryption, authentication, and integrity
- **Internet Key Exchange (IKE)** - Handles key exchange and security association negotiation
- **Security Associations (SA)** - Parameters for secure communication



IPsec vs. TLS Comparison
- **Layer** - IPsec works at the network layer, TLS at the transport/application layer
- **Protection scope** - IPsec protects all IP traffic between hosts, TLS protects specific application connections
- **Implementation** - IPsec requires kernel integration, TLS runs in user space
- **Configuration** - IPsec typically requires more complex configuration

IPsec Modes
Transport Mode: Protects the payload of the IP packet, OG IP header remains intact, Typically used for host-to-host communications
Tunnel Mode:
- Protects the entire original IP packet
- Encapsulates the original packet in a new IP packet
- Typically used for network-to-network (gateway-to-gateway) VPNs
- Hides internal IP addressing from eavesdroppers

OpenVPN

OpenVPN Architecture
OpenVPN's architecture includes several key components:
- **Virtual network interfaces** (TUN/TAP) to capture and inject network traffic
- **SSL/TLS library** for authentication and key exchange
- **Control channel** for management communication
- **Data channel** for encrypted user traffic
- **Configuration system** for defining connection parameters

OpenVPN Protocol Features
OpenVPN offers several protocol features:
- Can operate over UDP (default, port 1194) or TCP
- Uses SSL/TLS for authentication and key exchange
- Implements reliability mechanisms when using UDP
- Employs OpenSSL for cryptographic operations
- Supports various authentication methods:
  - Pre-shared static keys
  - Certificate-based authentication
  - Username/password authentication
  - Multi-factor authentication

WireGuard

WireGuard Architecture WireGuard's architecture focuses on simplicity:
- Operates at the network layer (Layer 3)
- Implements a "cryptokey routing"concept
- Uses public key authentication
- Always employs perfect forward secrecy
- Maintains a simple, stateless design

WireGuard Features Key features of WireGuard include:
- **Simplified cryptography** - No cipher negotiation, uses fixed set of modern algorithms
- **Fast handshake** - 1-RTT handshake under normal circumstances
- **Clean, minimal codebase** - Easier to audit and more secure
- **Connection-less design** - No persistent connection state
- **Stealth operation** - No response to unauthenticated packets
- **DoS mitigation** - Cookie challenge mechanism
- **Seamless roaming** - Maintains connections across network changes

Protocol Comparison
IPsec:
- Pro: Widely supported, mature standard
- Pro: Hardware acceleration in many devices
- Con: Complex implementation/configuration
- Con: May be blocked by restrictive firewalls

OpenVPN:
- Pro: Cross-platform compatibility
- Pro: Runs in user space
- Pro: Works well through firewalls (can use TCP port 443)
- Con: Slower performance than IPsec and WireGuard

WireGuard:
- Pro: Superior performance and simplicity
- Pro: Modern cryptography by default
- Pro: Small, auditable codebase
- Con: Less mature than alternatives
- Con: Fixed cryptographic algo

User Authentication

Access Control Framework
Identification Claiming an identity (username)
Authentication Proving the claimed identity (password)
Authorization Determining what the authenticated identity may do
Accounting/Auditing - Recording what actions were performed

Authentication Factors
Something you know Knowledge factors (passwords, PINs)
Something you have Possession factors (physical tokens)
Something you are Inherence factors (biometrics)
Somewhere you are - Location factors (GPS)
Something you do - Behavior factors

Password Security Challenges
Human memory limitations - Difficult to remember strong, unique passwords
Password reuse - Users often use the same password across multiple sites
Password sharing - Accounts accessed by multiple people
Password theft - Phishing, keylogging, and data breaches
Brute force attacks - Systematic guessing of passwords

Secure Password Storage

Password Hashing
one-way transformation to securely store passwords:
Converts passwords into fixed-length strings that cannot be reversed
Allows verification without storing the actual password
Must use cryptographic hash functions designed for security
Should be combined with salting and key stretching

Salting Passwords
adding random data to passwords before hashing:
Prevents use of precomputed tables (rainbow tables) for cracking
Forces attackers to crack each password individually
Salt should be unique per user and sufficiently long (64-128 bits)
Salt is stored alongside the hash in the password database

Key Stretching increases the computational work needed to compute password hashes:
- Applies the hash function repeatedly (thousands or millions of times)
- Increases the time needed for brute force attacks
- Can be adjusted as hardware becomes faster
- Implemented in specialized password hashing functions

Password Hashing Functions password storage:
bcrypt - Based on Blowfish cipher, salt and cost factor
PBKDF2 - Password-Based Key Derivation Function, configurable iterations
Argon2 - Winner of the Password Hashing Competition, memory-hard function
scrypt - Memory-hard function designed to be resistant to hardware acceleration
These functions are preferable to general-purpose cryptographic hash functions for password storage.

Authentication Fundamentals verifying that a claimed identity is genuine:
- Establishes trust in the identity claim
- Serves as the foundation for access control
- Must be completed before authorization
- Should be resistant to impersonation attacks

Authentication Protocols

Direct vs. Indirect Authentication
Direct Authentication:
- User authenticates directly to the service
- Service has all information needed to authenticate users
- Credentials are configured on each service
- Example: Local login on a server

Indirect Authentication:
- Authentication delegated to a central authority
- Service trusts authentication decisions of authority
- Credentials are managed centrally
- Examples: RADIUS, Kerberos, SAML, OpenID Connect

Multi-Factor Authentication (MFA)
Requires factors from different categories (know, have, are)
Significantly increases security compared to single-factor authentication
Mitigates risks of compromised passwords
Widely recommended for sensitive systems and accounts

Common MFA Methods
One-Time Passwords (OTP):
- Time-based (TOTP) - Generated from a shared secret and current time
- HMAC-based (HOTP) - Generated from a shared secret and counter
- SMS-based - Codes sent via text message (less secure)

Push Notifications - Authent. req. sent to mobile apps
Hardware Security Keys devices FIDO/U2F standards
Biometric Verification Fingerprint, face, voice recognition

MFA Attack Vectors Despite its strength, MFA can still be vulnerable to certain attacks:
- **Real-time phishing** - Intercepting and forwarding authentication in real-time
- **SIM swapping** - Taking control of a victim's phone number for SMS-based MFA
- **MFA fatigue** - Bombarding users with authentication requests until they approve
- **Malware on endpoint devices** - Capturing authentication data locally
- **Social engineering** - Tricking users into bypassing MFA protections

MFA Fatigue Countermeasures
Number matching - Displaying a number on login screen that must be entered in authenticator
Geographic context - location info of login attempt
Device context - device info of login attempt
Request limiting - Restricting nr of auth. requests
Suspicious activity detection - Identifying unusual patterns

Kerberos

Kerberos Components

Principals - Users, services, or systems that participate
Key Distribution Center (KDC) - Central authentication server consisting of:
- Authentication Service (AS) - Verifies user identities
- Ticket-Granting Service (TGS) - Issues service tickets

Realm - Administrative domain of a KDC
Tickets - Cryptographically protected credentials:
- Ticket-Granting Ticket (TGT) - Used to request service
- Service Ticket - Used to access specific services

Kerberos Security Properties

No password transmission - Passwords never sent over the network
Limited ticket lifetime - Tickets expire after a defined period
Mutual authentication - Both client and server can verify each other
Replay protection - Timestamps prevent reuse of authentication messages
Key distribution - Session keys established for secure communication

Cross-Realm Authentication Kerberos supports authentication across different administrative domains:
- Enables federated identity across organizations
- Requires trust relationship between realms
- Uses a hierarchical trust model or direct cross-realm keys
- Allows users from one realm to access services in another

Kerberos Authentication Process

Initial Authentication

- User logs in and sends username to Authentication Service
- AS checks if user exists in database
- AS sends back data encrypted with user's key (derived from password)
- Client decrypts this using the user's password
- Client extracts Ticket-Granting Ticket (TGT) and session key

Service Ticket Acquisition

- When user needs to access a service, client contacts Ticket-Granting Service
- Client sends TGT and an authenticator encrypted with session key
- TGS verifies TGT and authenticator
- TGS issues service ticket and new session key for the client-service communication

Service Access

- Client presents service ticket to the service
- Client also sends a new authenticator encrypted with client-service session key
- Service decrypts ticket and authenticator
- Service verifies authenticator contents (timestamp within allowed skew)
- Service grants access if verification succeeds

Federated Authentication

Federated Authentication enables identity verification across organization boundaries:
- Allows users to authenticate with their home organization
- Enables access to resources in partner organizations
- Eliminates need for multiple accounts and credentials
- Maintains security boundaries between organizations
- Examples include academic federation (SWITCHaai) and enterprise federations

Shibboleth system for federated identity management:
- Based on SAML
- Separates authentication from authorization
- Preserves privacy by limiting attribute sharing
- Widely used in academic and research communities
- Components include:
  - Identity Provider (IdP) - Authenticates users
  - Service Provider (SP) - Protects resources
  - Discovery Service - Helps users find their home organization

Shibboleth Authentication Flow

Resource Access Request User attempts to access a protected resource, Service Provider determines resource requires authentication, SP redirects user to Discovery Service
Identity Provider Selection and Authentication User selects their home organization from the Discovery Service, User is redirected to their Identity Provider, User authenticates with their home organization credentials
Assertion Generation and Validation IdP generates a SAML assertion containing authentication statement, IdP may include additional user attributes (depending on configuration), Assertion is cryptographically signed and optionally encrypted, User's browser posts the assertion to the Service Provider
Resource Access SP validates the assertion's signature, SP extracts user identity and attributes from the assertion, SP makes an authorization decision based on the attributes, SP grants access to the requested resource if authorized

Passwordless Authentication

Passwordless Authentication

- Based on public key cryptography and digital certificates
- Typically combines possession and inherence factors
- Eliminates password-related vulnerabilities
- Improves user experience by removing password entry

Authentication Protocols messages and procedures for user authentication:
- Standardize authentication across different systems
- Can be direct (authenticating directly to service) or indirect (using a third party)
- Often establish session keys for secure communication
- May support various authentication methods

FIDO2/WebAuthn

- Combines WebAuthn (web authentication standard) and CTAP (client-to-authenticator protocol)
- Uses public key cryptography for authentication
- Supports various authenticator types:
  - Platform authenticators (built into devices)
  - Roaming authenticators (external security keys)
- Protects against phishing through origin binding
- Private keys never leave the authenticator

Single Sign-On (SSO)

Single Sign-On (SSO) enables users to authenticate once and access multiple services:
- Reduces the number of authentication events
- Centralizes credential management
- Improves user experience
- Enhances security by reducing password fatigue
- Can implement consistent security policies across services

OAuth 2.0 authorization framework (not authentication):
Enables third-party applications to access resources without sharing credentials
Uses tokens to grant limited access to resources
Defines various authorization flows for different scenarios
Standardized in RFC 6749

OAuth 2.0 Roles

Resource Owner Entity that grants access (user)
Client Application requesting access to resources
Resource Server Server hosting the protected resources
Authorization Server Server issuing access tokens

OpenID Connect (OIDC) built on top of OAuth 2.0:
- Adds authentication functionality to OAuth 2.0
- Provides user profile information
- Uses JSON Web Tokens (JWT) for identity tokens
- Enables ßocial loginänd federated identity

OpenID Connect Terminology

Relying Party (RP) Application that wants to verify user identity
OpenID Provider (OP) Service that authenticates users
ID Token JWT containing user identity information
UserInfo Endpoint API for retrieving additional user information
Claims Pieces of information about the user

SSO Implementation Approaches

Kerberos - Primarily for on-premises environments (e.g., Active Directory)
SAML - Web-based SSO between different organizations
OpenID Connect - Modern protocol for web and mobile applications
OAuth 2.0 - Authorization framework often used with OpenID Connect

Security Assertion Markup Language (SAML) XML-based framework for exchanging authentication and authorization data:
- Primarily used for web-based SSO
- Enables cross-domain identity and access management
- Separates identity provider from service provider
- Current version is SAML 2.0

SAML Components

- **Assertions** - XML documents containing authentication statements
- **Protocols** - Rules for requesting and receiving assertions
- **Bindings** - Methods for transporting assertions over different protocols
- **Profiles** - Combinations of assertions, protocols, and bindings for specific use cases

SAML vs. OpenID Connect

Format - SAML uses XML, OIDC uses JSON
Platform support - SAML for web only, OIDC for web, mobile, and APIs
Complexity - SAML more complex, OIDC simpler to implement
Age - SAML older and more established, OIDC newer
Token size - SAML assertions larger than OIDC ID tokens

Authorization

Authorization determining whether an authenticated entity is permitted to perform a requested action:
- Follows successful authentication
- Controls access to resources and operations
- Implements security policies
- Enforces principle of least privilege

Fundamental Building Blocks

- **Access Control Model** - Conceptual framework defining how access decisions are made
- **Security Policy** - Rules defining who can access what resources
- **Security Mechanism** - Technical implementation enforcing the security policy

Security Policy Drivers

- **Business Drivers** - Efficiency, usability, operational requirements
- **Security Drivers** - Risk management, threat mitigation, data protection
- **Regulatory Drivers** - Legal compliance, industry standards

Discretionary Access Control (DAC)

Discretionary Access Control resource owner controls access rights:
- Resource owners determine who can access their resources
- Owners can delegate access control to others
- Access rights can be transferred between users
- Most common model in general-purpose operating systems

DAC Implementation Methods

Access Control Lists (ACLs):
- Lists of permissions attached to objects
- Each entry specifies subject and allowed operations
- Efficiently answers "who can access this object?"

Capabilities:
- Unforgeable tokens owned by subjects
- Each token grants specific access rights
- Efficiently answers "what can this subject access?"

DAC Advantages and Disadvantages

Advantages: Flexible and intuitive for users, Simple to understand and administer, Enables fine-grained access control
Disadvantages: Vulnerable to Trojan horse attacks, Cannot enforce system-wide security policies, May lead to uncontrolled information flow, Prone to privilege creep over time

Major Access Control Models

- **Discretionary Access Control (DAC)** - Access decisions made by resource owners
- **Mandatory Access Control (MAC)** - Access decisions enforced by system policy
- **Role-Based Access Control (RBAC)** - Access based on user roles within an organization

Additional models include:

- **Attribute-Based Access Control (ABAC)** - Access based on attributes of users, resources, and environment
- **Rule-Based Access Control** - Access based on predefined rules
- **Relationship-Based Access Control (ReBAC)** - Access based on relationships between entities

Standard UNIX Permissions

UNIX-style file permissions implement a simple form of DAC:

- Each file/directory has an owner and group
- Permissions divided into three categories:
  - Owner (user) permissions
  - Group permissions
  - Others (world) permissions
- Each category can have read (r), write (w), and execute (x) permissions
- Represented as 9 bits, often shown in octal notation (e.g., 755)

UNIX Permission Interpretation

Permission bits have different meanings for files and directories:

For Files:

- Read (r) - View file contents
- Write (w) - Modify file contents
- Execute (x) - Run file as a program

For Directories:

- Read (r) - List directory contents
- Write (w) - Create, delete, or rename files within directory
- Execute (x) - Access files and subdirectories (traverse)

Special Permission Bits

UNIX systems have additional permission bits for special purposes:

- **Setuid bit (4000)** - When set on executable files, the program runs with the permissions of the file owner
- **Setgid bit (2000)**:
  - On executable files: Program runs with the permissions of the file's group
  - On directories: New files inherit the directory's group
- **Sticky bit (1000)** - On directories, files can only be deleted or renamed by their owner (commonly used on /tmp)

POSIX ACLs

POSIX Access Control Lists extend standard UNIX permissions:

- Allow specifying permissions for multiple users and groups
- Maintain backward compatibility with standard permissions
- Include:
  - Minimal ACLs - Equivalent to standard permissions
  - Extended ACLs - Additional users/groups with specific permissions

ACL Components

- **Owner entry** - Permissions for the file owner (user::rwx)
- **Named user entries** - Permissions for specific users (user:alice:rwx)
- **Group owner entry** - Permissions for the file's group (group::rwx)
- **Named group entries** - Permissions for specific groups (group:developers:rwx)
- **Other entry** - Permissions for everyone else (other::rwx)
- **Mask entry** - Maximum permissions for named users/groups and group owner

Mandatory Access Control (MAC)

Mandatory Access Control

system-wide policy determines access rights:

- Access decisions enforced by the system, not resource owners
- Based on security labels assigned to subjects and objects
- Security policy defined by system administrators
- Users cannot override or modify access controls

MAC Implementations

- **SELinux (Security-Enhanced Linux)**:
  - Developed by NSA
  - Label-based access control
  - Fine-grained policy control
  - Used in Red Hat/Fedora distributions
- **AppArmor**:
  - Path-based access control
  - Simpler configuration than SELinux
  - Used in Ubuntu/SUSE distributions
- **Windows Mandatory Integrity Control**:
  - Introduced in Windows Vista
  - Based on integrity levels (Low, Medium, High, System)
  - Prevents lower integrity processes from modifying higher integrity resources

MAC vs. DAC Comparison

Control:

- DAC - Resource owners control access
- MAC - System policy controls access

Policy Management:

- DAC - Distributed management
- MAC - Centralized management

Security Assurance:

- DAC - Lower, vulnerable to user errors
- MAC - Higher, enforces system-wide policy

Complexity:

- DAC - Simpler to implement and understand
- MAC - More complex, requires specialized knowledge

Role-Based Access Control (RBAC)

Role-Based Access Control

based on user roles within an organization:

- Users are assigned to roles based on job responsibilities
- Permissions are assigned to roles, not directly to users
- Users acquire permissions through role membership
- Standardized in INCITS 359-2004

RBAC Components

- **Users** - Entities that need access to resources
- **Roles** - Collections of permissions representing job functions
- **Permissions** - Approved operations on protected resources
- **Sessions** - Mappings between users and their activated roles

RBAC Security Principles

- **Principle of Least Privilege** - Users have only the permissions needed for their job
- **Separation of Duties** - Critical operations require multiple users
- **Data Abstraction** - Permissions defined at higher levels than individual objects

Attribute-Based Access Control (ABAC)

Attribute-Based Access Control

ABAC is a flexible access control model based on attributes:

- Access decisions based on attributes of:
  - Subjects (users, applications)
  - Resources (data, services)
  - Actions (read, write, execute)
  - Environment (time, location, security level)
- Uses policies that combine attributes with logical rules
- More dynamic than traditional RBAC
- Can implement complex, context-aware policies

ABAC vs. RBAC

ABAC extends beyond traditional RBAC:

Flexibility:

- RBAC - Predetermined access based on roles
- ABAC - Dynamic access based on multiple attributes

Granularity:

- RBAC - Role-level permissions
- ABAC - Fine-grained, attribute-based decisions

Context Awareness:

- RBAC - Limited environmental context
- ABAC - Rich environmental attributes (time, location, etc.)

Policy Expression:

- RBAC - Role-permission assignments
- ABAC - Complex boolean logic combining attributes

Zero Trust Security Model

Zero Trust

security model that assumes no implicit trust for any entity:

- Eliminates the concept of trusted internal networks
- Requires continuous verification for all access requests
- Applies least privilege access controls
- Inspects and logs all traffic
- Focuses on protecting resources rather than network segments

Zero Trust Principles

- **Verify explicitly** - Always authenticate and authorize based on all available data points
- **Use least privilege access** - Limit user access with Just-In-Time and Just-Enough-Access
- **Assume breach** - Minimize blast radius and segment access
- **Identity-based security** - Identity becomes the primary security perimeter
- **Continuous monitoring** - Collect and analyze data continuously