Überblick über Vorlesungsinhalte:

- Schwachstellen, Bedrohungen, Ausgewählte Angriffe, Schutzziele, wichtige Maßnahmen
- Kryptographische Grundlagen: Prinzipien symmetrisch, asymmetrisch, Block und Stromchifree, Verschlüsselungsmodi, Beispiele: AES, RSA
- Hashfunktionen und Signaturverfahren
- Schlüsselmanagement: TTP, Diffie-Hellmann, hybrid
- Identitätsmanagement, Authentisierung, PKI, Fallbeispiele: Kerberos, Oauth
- Nezwerksicherheit: Firewalls, TLS, VPN
- Anwendungssicherheit: Main, Messangerdienste
- Zugriffsrechte und Rechtemanagement: Modelle, Konzepte
- Systemsicherheit: Speicherschutz, Dateisystem, Virtualisierung
- Secure Programming: klassische Fehler, Regeln
- Informationssicherheitsmanagement (ISMS), Secure Engineering: Bedrohungs- und Risikoanalyse, Common Criteria (CC), Protection Profile

Kapitel 1: Schwachstelle, Bedrohungen, Angriffe

- Grundlegende Begriffe:
  - Security (Modul → IT-Sicherheit):
    - Verwundbarkeit von zu schützenden Werten systematisch reduzieren
    - Bewahren eine Systems vor Beeinträchtigung und Missbrauch durch Angriffe
    - Angriffe sind Störungen von Außen mit dem Ziel der Daten-Manipulation, des Informationsmissbrauchs oder der Funktionsstörung
      - z.B. Unbefugtes Lesen vertraulicher Information.
  - Safety: Funktions- und Betriebssicherheit
    - Erkennen und Abwehr von Störungen, die die korrekte Funktionalität, die Betriebssicherheit beeinträchtigen
    - Störungen kommen von Innen, durch das technische System, z.B. durch Hardware-Fehler, Programmierfehler.
      - Absturz Ariane 5 Rakete direkt nach dem Start: Fehlen eines Exception Handlings für einen Floating Point Fehler.
  - Schwachstelle (vulnerability):
    - Eine Schwachstelle (u.a. im Code) ermöglicht es, dass die Sicherheitkontrollen des Systems umgangen oder getäuscht werden können
  - Bedrohung (threat):
    - Umstand oder Ereignis mit dem Potenzial, ein System durch unbefugten Zugriff, Zerstörung, Offenlegung, Änderung von Daten und/oder Denial-of-Service durch Ausnutzen einer oder mehrere Schwachstellen zu beeinträchtigen.
  - Angriffsvektor (attack vector):
    - Möglicher Angriffsweg, der eine oder mehrere Schwachstellen ausnutzt, um die Sicherheit eines Systems zu gefährden.
- Angriffsklassen:
  - Ungenügende Eingabevalidierung:
    - Buffer Overflow:
      - Schreiben außerhalb des Speicherbereichs (buffer), der für die Speicherung des Wertes einer Variable vorgesehen ist.
      - Mögliche Effekte bei Erfolg:
        - Ändern von Daten
        - Programm-Crash
        - Zugriffsrechte
    - Code-Injection:

        - Nicht validierte Daten werden von einem Interpreter als Bestandteil einer Anfrage verarbeitet, um z.B. Kommandos auszuführen oder die Semantik zu verändern.
        - SQL-Injection:
            - Query: SELECT * FROM users WHERE id=‘“ + $id + “‘
            - Angriff: 1‘ OR 1=1 --
        - Cross-Site-Scripting (XSS):
            - Angreifer bring JavaScript Code in eine Webseite ein. Ausführung erfolgt im Browser des Opfer-Rechners
            - Ziel: Informationen aus webseite extrahieren, z.B. Cookie, um damit z.B. Session Hijacking durchzuführen
            - Varianten:
                - Persistent: Angreifer-Script auf Webseite gespeichert.
                    - z.B. durch unsichere Kommentarfunktion
                - Non-Persistent: Angreifer-Script über HTTP Anfrage in Webseite eingebracht.
                    - z.B. Phishing-Mail
    - Identitätsdiebstahl:
        - Schwachstellen bei Identitätsprüfung ermöglichen es Angreifern, unter eine fremden Identität zu agieren.
        - Beispiele: ARP-Spoofing, IP-Adress-Spoofing, gespoofte E-Mail-Absenderadressen
    - Man-in-the-Middle Angriff:
        - Angreifer kontrolliert Datenverkehr zwischen zwei Partnern und spooft jeweils den anderen Partner
        - ARP-Cache Poisoning:
            - Angriff in geswitchten Netzen, Ausnutzen fehlender Authentisierung von MAC-Adressen beim ARP Protokoll, ARP-Tabellen mit gefälschten Daten befüllt (posioning)
            - Ablauf:
                - 1. Opfer fragt nach MAC-Adresse des Gateways mit IP 192.168.178.1
                - 2. Angreifer antwortet mit eigener MAC Adresse
                - 3. Opfer übernimmt die Angaben ungeprüft in lokalem ARP-Cache
                - 4. Angreifer ist MitM: Umleiten der Daten vom Opfer zum Gateway 192.168.178.1
    - Angriffe auf die Verfügbarkeit:
        - (Distributet) Denial of Service ((D)DoS) Absichtliche Überlast von Serven oder Routern
    - Faktor Mensch: unwissend, vertrauensselig
        - Social Engineering: täuschen von Menschen (Web-Seite, Fake-Mail, Fake-Telefonate)
    - Web-Application Security: OWASP Top 10
        - Platz 1: Broken-Access-control
        - Platz 2: Fehlerhafte oder fehlende Nutzung von Krypto
- Klassen von Schadcode (Malware):
    - Virus:
        - Nicht selbstständiges Programm, das sich selbst in noch nicht infizierte Dateien kopiert. Der Virus-Code benötigt das Wirtsprogramm, um ausgeführt zu werden.
    - Trojaner:
        - Neben der spezifizierten nützlichen Funktionalität, zusätzlich eine versteckte Funktionalität
        - z.B. Keylogger → Passwort aufzeichnung
    - Ransomware

        - Verschlüsselt Daten auf Opfer-Rechner
        - z.B. WannaCry
- Schutzziele:
    - in Systemen werden idR. Kombinationen von mehreren Schutzzielen gefordert.
    - Basis-Ziele: CIA
        - Informationsvertraulichkeit (confidentiality):
            - Schutz vor unautorisierter Informationsgewinnung
            - z.B. Abhören, Passwort knacken
        - Datenintegrität (integrity):
            - Schutz vor unautorisierter und unbemerkter Modifikation
            - z.B. Buffer-Overflow, SQL-Injection
        - Verfügbarkeit (availability):
            - Schutz vor unbefugter Beeinträchtigung der Funktionalität
            - z.B. Ransomware, Spam
    - Weitere Schutzziele:
        - Authentizität (authenticity):
            - Nachweis der Echtheit und Glaubwürdigkeit der Identität einer handelnden Entität oder eines zu nutzenden Objekts.
            - z.B. Phishing, MitM
            - Schutzkonzepte:
                - Biometrischer Fingerabdruck
                - digitaler Personalausweis
                - Passworte
        - Verbindlichkeit, Zurechenbarkeit (accountability):
            - Schutz vor unzulässigem Abstreiten durchgeführter Handlungen
            - z.B. Identitätsdiebstahl
            - Schutzkonzepte:
                - Digitale Signatur
                - Blockchain
        - Privatheit (privacy):
            - Die Fähigkeit einer natürlichen Person, die Weitergabe und Nutzung seiner personenbeziehbaren Daten zu kontrollieren (informationelle Selbstbestimmung)
            - z.B. Profilbildung, Analytics
            - Schutzkonzepte:
                - Anonymisierung
                - Aggregation
- Security-Policy – IT-Sicherheitsrichtlinien:
    - Regelwerk u.a. in Unternehmen:
        - zur Nutzung eines Produktes
        - für den Zugang zum Unternehmensnetz
    - Festlegen von:
        - Schutzziele
        - Menge von technischen und organisatorischen Regeln und Verhaltensrichtlinien
        - Verantwortlichkeiten
    - Beispiel technische Policy im Browser:
        - Skripte werden im Kontext einer Website ausgeführt
        - Policy legt fest worauf das Skript zugreifen kann
            - Zugriff nur auf Objekte/Inhalte von der gleichen Website (same origin)
            - Gleichheit: Protokoll, Domain und Port
        - SOP nicht ausreichend um XSS zu umgehen.

Kapitel 2: Kryptografische Grundlagen → Vertraulichkeit (confidentiality)

- Grundlagen:
  - Kryptografie: Methoden zur Ver- und Entschlüsselung
  - Kryptoanalyse: Wissenschaft von Methoden zur Entschlüsselungen
- Kryptografisches System (M, C, E, D, K):
  - Menge der Klartexte m, über dem Alphabet M, m \in M*
  - Menge der Kryptotexte c über dem Alphabet C, c \in C*
  - K Menge der Schlüssel, e, d \in K
  - E = {E_e | : M* → C*} Familie von Verschlüsselungsverfahren
  - D = {D_d | : C* → M*} Familie von Entschlüsselungsverfahren
- Symmetrisch v/s. Asymmetrisch
  - Symmetrische Verfahren, Secret-Key Verfahren:
    - Verfahren E, D idR einfach und schnell zu berechnen.
    - e, d sind gleich (symmetrisch) und geheim (Secret Key)
  - Asymmetrische Verfahren, Public-Key Kryptografie
    - Verfahren E, D basieren auf Zahlentheorie und Gruppentheorie
    - Ein Schlüsselpaar pro Entität:
      - öffentlicher Schlüssel → Verschlüsselung
      - privater Schlüssel → Entschlüsselung
    - Einsatz:
      - Privaten Schlüssel müssen vertraulich verwaltet werden
      - Asymmetrische Verschlüsselung ist viel aufwändiger als symmetrische.
        - IdR nur kleine Datenvolumen
    - Anforderungen:
      - Privater Schlüssel muss privat bleiben
      - d kann aus e nicht effizient berechnet werden
    - Theoretische Basis asymmetrischer Verfahren:
      - Einwegfunktion f: X → Y:
        - \forall x \in X gilt: f(x) ist effizient berechenbar
        - \forall y \in Y gilt: $f^{-1}(y) = x$ ist nicht effizient berechenbar
      - Trapdoor-Einwegfunktion:
        - Mit Zusatzinformation sind Urbilder effizient berechenbar
      - Beispiel Einwegfunktion:
        - Faktorisierung:
          - Einwegfunktion: f(p, q) = p*q = n → (einfach)
          - Primfaktorzerlegungsproblem: $f^{-1}(n) = (p, q)$ → (schwer)
        - Diskreter Logarithmus:
          - gegeben Primzahl p, g <= p und y
          - Einwegsfunktion: $f(x) = g^x \bmod p = y$ → (einfach)
          - Diskreter Logarithmus: $x = f^{-1}(y) = \log_g(y) \bmod p$ → (schwer)
    - Beispiel RSA:
      - Basis: Zahlentheorie (Fermat, Euler, …)
      - Berechnung des Schlüsselpaars:
        - Wahl von 2 großen Primzalen p, q
        - Berechnung des RSA-Modul n = pq, n ist öffentlich
        - Berechnung φ(n) = (p-1)(q-1), Eulersche Phi-Funktion effizient zu berechnen, wenn p, q Primzahlen sind
        - Wahl des öffentlichen Exponenten:
          - e \in {0, …, φ(n) – 1}
          - ggT(e, φ(n)) = 1
          - (e, n): öffentlicher Schlüssel

        - Berechnung des privaten Schlüssels d
            - $ed = 1 \mod \varphi(n) \rightarrow$ p, q Trapdoor Informationen
    - Ver- und Entschlüsselung:
        - Verschlüsselung: $m^e \mod n = c$
        - Entschlüsselung: $c^d \mod n = m$
    - Bemerkung:
        - RSA-Modul groß: 2048 – 4096 Bits
        - Textbook RSA Verschlüsselung ist deterministisch und unsicher
        - Lösung: Einbettung von Padding-Mustern
- Anforderungen and kryptografische Verfahren:
    - Kerckhoffs-Prinzip Auguste Kerckhoffs 1883:
        - Stärke des Verfahrens sollte nur von Güte des geheimen Schlüssels abhängen
        - Sicherheit darf nicht von Geheimhaltung der Verfahren abhängig
            - Keine Security by Obscurity
        - Konsequenz:
            - Schlüsselraum muss sehr groß sein → Brute-Force nicht mit praktikablem Aufwand erfolgreich
                - Brute-Force: Durchprobieren möglicher Schlüssel
- Anforderungen an Schlüssellängen:
    - symmetrische Verfahren: Schlüssel >= 128 Bit
        - Beispiele:
            - AES Standard mit 128, 192, 256 Bit Schlüssel
            - ChaCha20: 256 Bit Schlüssel
    - asymmetrische Verfahren:
        - Schlüssel >= 3000 Bit ab 2023 (RSA: 4096 Bit)
        - Schlüssel >= 250 Bit bei ECC (Elliptic Curve Crypto)
        - Beispiele:
            - RSA: 1024, 2048, 4096…
            - ECC (Elliptic Curve Cryptography) u.a 160, 256, 512
- Block- und Stromchiffren:
    - Blockchiffre (symetrische Verfahren)
        - Funktionsprinzip:
            - Klartext m zerlegt in Blöcke m_i fester Länge
            - blockweises Verschlüsseln mit Schlüssel k
            - Gegeben Klartext m, $m = m_1 \| m_2 \| \dots \| m_n$
            - Verschlüsseln: $c_i = E_k (m_i)$
            - Entschlüsseln: $m_i = D_k (c_i)$
        - Padding:
            - Ausfüllen des Klartextes → Länge vielfaches der Blockgröße
            - Beim Verschlüsseln hinzugefügt, beim Entschlüsseln entfernt
            - Oft: PKCS#7 Padding:
                - Wert der Hinzugefügten Bytes entspricht der Anzahl der hinzugefügten Bytes
        - Designprinzip:
            - Diffusion:
                - Jedes Klartextbit beeinflusst jedes Ciphertextbit.
                - Technik: Bitweise Permutation
            - Konfusion:
                - Zusammenhang zwischen Key und Ciphertext verschleiern
                - Technik: nicht lineare Subsitutionen: S(A xor B) != S(A) xor S(B)
            - Diffusion und Konfusion erzeugen Avalanche Effekt

        - Beispiel: AES (Advances Encryption Standard)
            - Blocklänge 128 Bit, Schlüssellängen: 128, 192, 256 Bit
                - 128-Eingaben als 4x4 Matrix
            - Runden: Mehrfaches ausführen der selben Abfolge von Operationen auf dem Klartext
                - Abhängig von der Schlüssellänge hat AES 10, 12, oder 14 Runden
            - Gilt als sicheres Verfahren
            - Starke Diffusion und Konfusion. Jedes Input Bit hängt am Ende von jedem Schlüsselbit ab:
                - Konfusion: durch nicht lineare Substitution
                    - Jedes Eingabebit durchläuft festegelegte S-Box (SubBytes)
                - Diffusion:
                    - ShiftRows: zyklischer Links-Shift der Spalten
                    - MixColumns: Multiplikation jeder spalte der 4x4 Matrix im GF (2^8) mit fester Matrix C. Jedes Byte der Spalte mit jedem anderen Byte verknüpft.
                    - GF(2^8): Endlicher Körper mit 256 Elementen
    - Stromchiffre:
        - Ziel: schnelle Verschlüsselung eines Klartext → xor (ist schnell)
            - Problem: xor keine starke Chiffre!
            - Lösung: für jeden Klartextstrom, individuelle Schlüsselfolge KS als pseudozufällige Folge von Bit
        - Funktionsprinzip
            - Verschlüsselung: m xor KS (bitweise xor bzw. add mod 2)
            - Schlüsselfolge KS gleiche Länge wie Klartext m
            - Kern: „gute“ Schlüsselfolge KS
        - Lösung in der Praxis:
            - kryptografischer Pseudozufallszahlengenerator (CSPRNG) mit Seed-Wert initialisiert, erzeugt Pseudozufallszahlenfolge KS
            - Determinisitischer Generierungsprozess bei gleichem Seed
            - Konsequenz:
                - Seed Wert k ist unabhängig von Nachrichtenlänge und kann wie ein symmetrischer Key k behandelt werden, z.B. 128 Bit großer k
                - Deterministische KS generierung: Mit Kenntnis von k kann KS rekonstruiert werden.
        - 2 Ansätze:
            - Dedizierte Chiffre: z.B. ChaCha20
            - Blockchiffre basiert: CSPRNG durch eine Blockchiffre umgesetzt:
                - Vom BSI Empfohlen: AES im CTR-Modus:
                    - Seed-Wert: geheimer AES-Schlüssel
                    - 128 Bit Schlüsselfolge KS wird pro AES Durchlauf generiert
        - Geforderte Eigenschaft:
            - Unterschiedliche Klartexte m_1, m_2 mit $c_1 = m_1$ xor KS und $c_2 = ,m_2$ xor KS
            - Kennt man c_1, c_2 und m_1 → Man kann m_2 berechnen
            - Konsequenz: Symmetrischer Schlüssel muss sich ändern → Zähler
    - Betriebsmodi von Stromchiffren:
        - ECB: Electronic Code Book Modus:
            - Vorteil:
                - Parallelisierung der Verschlüsselung
                - keine Fehlerausbreitung
            - Nachteil:
                - Gleiche Klartext-Blöcke ergeben gleiche Chiffre-Blöcke
                - Muster in langen Nachrichten bleiben erhalten

        - Konsequenz:
            - Sichere Blockchiffre kann im EBC-Modus angreifbar werden:
                - Statistische Analysen, Muster
    - CBC: Cipher Block Chaining:
        - Klartextblock xor mit vorherigem Chiffretextblock
        - Startvert = Initialisierungsvektor IV nicht geheim
        - Gleiche Klartextblöcke und gleicher Schlüssel k ergeben ungleiche Chiffreblöcke, falls IV unterschiedlich ist
            - IV → zufällig gewählt
        - CBC Sicherer als ECB Modus
        - Problem: Fehlerfortpflanzung durch Verkettung:
            - Nur auf 2 Blöcke beschränkt
            - Wenn c falsch übertragen wird, dann ist nur c und der nächste Block bei der Entschlüsselung fehlerhaft
    - CTR: Counter Mode:
        - Initialisierung eines Zählers ctr mit Zufallszahl Nonce
        - Zählerstand wird pro Block erhöht
        - Beispiel: Blockchiffre AES im CTR-Modus
    - GCM: Galous/Counter Modus:
        - Verschlüsseln von Blöcken m_i, Blocklänge 128 Bit
        - Verschlüsseln durch Blockchiffre im CTR-Modus
        - Authentisieren der verschlüsselten Daten durch Multiplikation im Galoiskörper GF(2^{128})
        - Zusätzlich: Authentizität assoziierter Daten AD (z.B. Header Daten)
        - Implementiert die AEAD-Eigenschaft:
            - (Authenticated Encryption with Associated Data) Modus
                - Authentizität des Ciphertextes (AE)
                - Authentizität assoziierter Daten (AD)
- Elliptische-Kurven-Kryptographie (ECC) (asymmetrisch)
    - Problem:
        - Asymmetrische Verfahren erfordern hohen Berechnungs- und Speicheraufwand aufgrund der großen Schlüssellängen
    - Gesucht:
        - zyklische Gruppen, in denen das Problem des Diskreten Logarithmus (DLP) schwer zu lösen ist und die mit kürzeren Schlüsseln arbeiten können
    - Lösung:
        - Elliptische Kurve: die gesuchte Gruppe wird durch die Menge der Punkte auf der elliptischen Kurve abgebildet
    - Elliptische Kurve (EC):
        - EC ist eine Punktmenge, die eine Polynomial-Gleichung erfüllt.
        - Definition:
            - Elliptische Kurve E über dem Körper Z_p = {0, …, p-1} mod p ist: Menge der Punkte (x, y) in Z_p sodass gilt: $y^2 = x^3 + ax + b$ mod p, sodass $4a^3 + 27b^2 != 0$
            - Parameter a, b definieren die Form der jeweiligen Kurve
            - Abelsche Gruppe (Z_2 \union {θ}, ∔):
                - Operation ∔: P∔Q = Q∔P
                - Imaginärer Punkt θ
    - Kryptographie:
        - Gegeben:
            - Elliptische Kurve E über Primzahlkörper Z_p
            - Generator-Element G auf der Kurve E

        - Generieren eines Schlüsselpaars (T, d):
            - Wähle d und berechne dG = T, d geheim (d Integer-Wert)
            - T ist Punkt auf der Kurve, öffentlicher Schlüssel
        - ECDL-Problem (EC Discrete Log) schwer:
            - Finde Integer Wert d, 1 <= d <= |E|, sodass d = log_G T
            - T = d * G = G + G + … G
- Sicherheitsniveau eines kryptografischen Verfahrens:
    - Niveau von n Bit, wenn:
        - Erfolgreiche Angriffe einen Aufwand erfordern, der der Ausführung von 2^n Verschlüsselungen einer effizienten Blockchiffre entspricht
        - Bei symmetrischen Verfahren entspricht das Bruteforce des Schlüsselraums 2^n

Kapitel 3: Kryptografische Hashfunktion und MAC → Integrität (integrity), Authentizität (authenticity)

- Kryptografische Hashfunktion:
    - Idee: Erstellen eines „digitalen“ Fingerabdrucks h für ein Dokument/ eine Nachricht m, so dass h das Dokument m repräsentiert
    - Vorgehen: Gegeben Hashfunktion H, n Ganzzahlwert
        - Nachricht m mit Beliebiger Länge
        - Hashfunktion H: M* → M^n, z.B. n = 128
        - h = H(m), h nennt man Message Digest, feste Länge n-Bit → kein vertraulicher Wert
    - Ziel: Nutzen von Hashfunktionen für Integritätsüberprüfung
        - Anforderung:
            - Hashwert h charakterisiert Nachricht m eindeutig
            - Modifikation von m ergibt anderen Hashwert
            - Wenn m != m’ → Anderer Hashwert → Um Modifikation zu erkennen
        - Problem: H ist nicht injektiv → Kollisionen möglich
        - Erkenntnis: Funktion H so, dass Kollisionen nicht effizient erzeugbar sind
- Anforderungen an eine kryptografische Hashfunktion:
    - 1. \forall m \in M*: H(m) = h ist einfach zu berechnen
    - 2. Einwegeigenschaft (pre image resistance):
        - Gegeben h = H(m)
        - Bestimmung des Wertes m \in M* mit m = H^{-1}(h) nicht effizient berechenbar
    - 3. Schwache Kollisionsresistenz (second pre image resistance)
        - Gegeben m \in M*
        - Nicht effizient möglich ein m’ \in M*, mit H(m) = H(m’) zu finden
    - 4. Starke Kollisionsresistenz (collision resistance)
        - Nicht effizient möglich Paare m, m’ \in M*, mit H(m) = H(m’) zu finden
    - Wie groß soll Hashwert h sein?
        - Geburtstagsparadoxon
        - Hashfunktion H, n Bit Hashwerte h, 2^n Werte
        - Erforderliche Anzahl der Nachrichten, um mit einer Wahrscheinlichkeit > 0.5 eine Kollision zu erzeugen: sqrt(2^n)
        - Konsequenz:
            - z.B. n = 80
            - Kollisionsangriff 2^{40} Hashwert-Berechnungen → Effizient leistbar → 80 zu klein
            - BSI n >= 256
- 3 Klassen von Hashfunktionen:
    - 1. Basierend auf Block-Chiffren: AES-CBC → letzte Block dient als HashWert
    - 2. Dedizierte Hashfunktionen:
        - SHA-1 (Secure Hash Algorithm), 160 Bit (unsicher!)

        - SHA-2-Familie: SHA-256 Bit, SHA-512 Bit:
            - Basiert auf Merkle-Damgård: length extension
            - SHA-512/256 ist resistent gegen length extension
        - SHA-3:
            - Basiert auf Sponge-Prinzip
            - Länge des Hash: 224-512 Bit
            - Resilient gegen length extension Angriffe
    - 3. Passworthashfunktionen
        - Background:
            - Passworte gehasht abgespeichert
            - Login: Hashen des Passworts und vergleich der Hashes
        - Spezielle Anforderungen an H:
            - Abgleich soll „langsam“ sein: Abwehr von Brute Force Angriffen
            - Abgleich soll viel Speicher und viel CPU-Zeit benötigen
        - Lösung parametrisierbare Hashfunktionen, bei Bedarf „abbremsen“
        - Beispiele: SHA-256 ~ 2900MH/s, bcrypt 1.3 KH/s
- Message-Authentication-Code (MAC) → Authentizität (authenticity)
    - Idee:
        - gemeinsames Geheimnis in der Berechnung
        - Authentizität: Nachweis der Kenntnis des Geheimnisses
        - Geheimnis wird als Schlüssel bezeichnet (Hier nicht zum Verschlüsseln)
    - MAC:
        - Hashfunktion mit Schlüssel H: (M* x EK) → M^n
        - Geheimer (Pre-shared) Schlüssel k_ab zwischen Partnern A, B
        - Berechnung:
            - mac = H(m’) , mit m’ = k_ab | m , m ist Nachricht
            - Empfänger prüft Authentizität und Integrität von m’ mit k_ab
    - Anwendungsbeispiel:
        - Schutzziele: Vertraulichkeit, Integrität, Authentizität
        - AES zur Verschlüsselung, SHA-256 als MAC-Verfahren
        - k_mac ein pre-shared MAC-Key, k_enc ein AES-Key, m Klartext
        - Protokoll – Encrypt-then-mac:
            - (1) Encrypt m
            - (2) mac-Berechnung
            - (3) Daten an B:
            - (4) prüfen
            - (5) Entschlüsseln
    - Sichere MAC-Verfahren:
        - Problem bei Merkel-Damgård: length extension Angriffe
        - Angreifer kann einfach weiterhashen, weil h vollständig in die Berechnung eingeht
        - SHA-1, SHA-2 basieren auf Merkel-Damgård
        - SHA-512/256 ist resistent gegen length extension → Es werden nur 256 von den 512 Bits weiter genutzt
    - Sichere MAC-Konstruktion - HMAC-Verfahren RFC 2104:
        - HMAC Konstruktion kapselt ein existierendes Hash-Verfahren H
        - H wird damit „gehärtet“: HMAC-SHA-1, HMAC-SHA-2, ...
        - HMAC(m, k’) = H (k’ xor opad | (H(k’ xor ipad | m)))
        - 2 Hash-Anwendungen:
            - Innerer und äußerer Hash erhöht die Kollisionsresistenz
            - Length extension nicht möglich
- AEAD (Authenticated Encryption with Associated Data):
    - Bislang: Einzelne Krypto-Primitive für CIA Ziele. Kombinationen in Anwendungen

    - Problem: Falsche Anwendung, führt zu Schwachstellen (Mac-then-encrypt: Padding Oracle)
    - AE und AEAD sind weitere Betriebsmodi einer Blockchiffre:
        - Integration der Schutzkonzepte in einem Mechanismus
        - AEAD: Prüfung auch von nicht verschlüsselten Daten (AD)
            - Assoziierte Daten werden mit Cyphertext verknüpft
            - Header Daten unverschlüsselt aber authentisch
- Elektronische (digitale) Signatur
    - Ziel: Nachweis der nicht-abstreitbaren Urheberschaft eines Dokuments, Nachricht, …
    - Funktionsprinzip:
        - Basis: Public Key Verfahren
        - Alice besitzt Schlüsselpaar (k_sig, k_veri)
            - Privater Signaturschlüssel k_sig (d)
            - Öffentlicher Verifikationsschlüssel k_veri (e)
        - Vorgehen:
            - m Dokument, RSA Verfahren, SHA-256 Hashverfahren für Datenintegrität
        - Ablauf:
            - 1. Hashen: SHA-256(m) = h
            - 2. Signieren: RSA_d(h) = h^d mod n = sig, d privater Schlüssel von A
        - Prüfen der Korrektheit:
            - 1. Rekonstruktion von h: RSA_e(sig) = sig^e mod n = h’
            - 2. Validieren: h == h’
    - Signaturverfahren: analog zu Hashfunktionen:
        - Dedizierte Signaturverfahren: DSA, ECDSA
        - Public Kexy Verschlüsselung: RSA

Kapitel 4: Schlüsselmanagement

- Fokus:
    - Schlüsselerzeugung bei symmetrischen Verfahren
    - Schlüsselaustausch bzw. Schlüsselvereinbarung
- Schlüsselgenerierung:
    - Kandidaten
        - Echte Zufallszahlen TRNG (True Random Number Generator)
            - Entropiequelle basiert auf physikalischen Phänomenen
            - Atmosphärisches Rauschen, Mausbewegung, thermisches Rauschen, Frequenz der Schwingungen eines Oszillograph
        - Pseudozufallszahlen (PRNG)
            - Algorithmus arbeitet deterministisch
            - Deterministisches Verhalten basiert auf Seed
            - Kenntnis des Seeds ermöglicht die Berechnung aller Ausgaben der PRNG-Funktion
            - Wenn ein PRNG zur Generierung von Schlüsselbits genutzt wird, dann muss der Seed-Wert vertraulich sein.
        - Kryptografisch sichere Zufallszahlengenerator (CSPRNG)
            - 1. PRNG dürfen nicht vorhersagbar sein, selbst wenn der Angreifer bereits einen Teil der generierten Zufallsfolge kennt
            - 2. Sie dürfen keinen Hinweis auf vorhergehende Zufallszahlen liefern, dalls der Angreifer eine Zufallszahl errät oder anfängt
            - 3. Sie sollten Zufallsfolgen erzeugen, die statistisch gleichviele Nullen und Einsen haben und nicht komprimierbar sind

            - Beispiel:

        - Hardware: Nichtlinear rückgekoppelte Schieberegister
- Etablierung gemeinsamer Schlüssel:
    - Schlüsselaustausch, -verteilung (Key Distribution):
        - Mit zentraler Schlüsselverteilung:
            - Praxis: Schlüsselaustausch vorab skaliert nicht
            - Lösungsmöglichkeit: Zentrale Schlüsselverteilungs-Server → KDC-Server (Key Distribution Center)
                - Auch als Trusted Third Party (TTP) Bezeichnet
            - KDC besitzt Shared Keys k_a mit jedem Partner A
                - Bei symmetrischer Kryptografie ist k_a ein Secret Key: Maser Key
                - Bei asymmetrischer Kryptografie ist k_a der öffentliche Schlüssel e_a
            - Shared Keys sind pre-shared ausgetauscht und langlebig
            - Austauschwege für pre-shared Keys: QR-Code, SMS, PIN-Brief
            - Nutzung:
                - A möchte mit B kommunizieren:
                - KDC erzeugt einen neuen Key k_ab und verteilt diesen an A, B
            - Erforderlich:
                - Prüfungen der Identität und Authentizität der beteiligten Partner A, KDC, B
                - Prüfungen, dass ausgetauschte Nachrichten nicht modifiziert sind:
                    - Integritätsschutz und Ursprungsnachweis von Nachrichten
                - Prüfungen, dass Schlüssel nicht wieder eingespielt werden
                    - Frischenachweis
                - Nachrichten enthalten alle Informationen die der Empfänger benötigt:
                    - Absender, Empfänger, Gültigkeitsdauer, Adressen, …
        - Mit hybridem Ansatz → Key Encapsulation Mechanism (KEM):
            - Symmetrischer Schlüssel k_ab wird asymmetrisch verschlüsselt
    - PFS (Perfect-forward-secrecy) - „Folgelosigkeit“:
        - Wird ein Schlüssel unsicher (z.B. Master Key, privater oder öffentliche Schlüssel) dürfen andere, zu früheren Zeitpunkten genutzte Schlüssel nicht auch unsicher werden.
        - Kompromittierung eines Schlüssels macht eine sichere Kommunikation nicht im Nachhinein unsicher
        - Neuer Schlüssel darf nicht von alten abhängen
    - Schlüsselvereinbarung (Key Agreement): Diffie-Hellman (DH):
        - Basis:
            - Schwierigkeit des diskreten Logarithmus-Problems (Einwegfunktion)
                - $y = g^x \bmod p$ ist einfach
                - log_g y mod p = x ist schwer zu berechnen
            - Exponentiation ist kommutativ: $k = (a^y)^x \bmod p = (a^x)^y \bmod p$
        - Idee:
            - A und B berechnen k_ab lokal bei sich
            - Es wird keine vertrauliche Information ausgetauscht
        - Phasen:
            - (1) Wähle große Primzahl p (allen Teilnehmern bekannt)
            - (2) Wähle Wert g \in Z_p*, Generator Element (allen Teilnehmern bekannt)
            - (3) Wähle geheime Zahl a, b in {2, …, p – 2} → Private Key von A, B
            - (4) Berechne öffentlichen Schlüssel $A = g^a \bmod p$, $B = g^b \bmod p$
            - (5) Berechne Diffie-Hellman-Secret : $A^b \bmod p = g^{ab} \bmod p = B^a \bmod p$ = DH-k_ab

            - (6) Erzeugung eines symmetrischen Shared Key k_ab der Länge n-Bit:

        - Nutzen einer bekannten Key Derivation Funktion KDF
        - Mit DH-k_ab und weiteren Parametern wie Länge n
        - Jeder Partner berechnet k_ab unabhängig voneinander
    - KDF: Basis sind meist Keyed-Hashfunktionen (MAC), z.B. SHA3
        - Mit Seed bzw. Key-Teil der KDF ist der DH-k_ab
        - KDF erzeugt so viele Bits wie nötig
        - Hashfunktion hierfür mehrfach angewandt um genügend Bits zu generieren
    - Problem: MitM → Lösung signierter Austausch der öffentlichen Schlüssel
        - Basis:
            - Jeder Partner besitzt Signatur-Schlüssel (veri_x, sig_x)
            - Für jede Vereinbarung eines neuen Schlüssels k_ab wird von den Partnern jeweils ein neues DH-Schlüsselpaar (e, d) generiert.
                - DH-Schlüsselpaare nennt man ephemeral keys (flüchtig)
            - Die öffentlichen Schlüssel werden signiert ausgetauscht:
                - e_a: DH-Key von A
                - Signieren des Schlüssels e_a: E_sig_a(e_a) (E z.B. RSA)
                - Empfänger veriziert die Signatur sig mit veri_A: D_veri_A(sig)
        - Praxis:
            - Bei geringer Rechenleistung: 1 Partner (z.B. Server), benutzt ein statisches DH-Schlüsselpaar und der Client generiert immer ein neues DH-Paar für jede neue k_ab Vereinbarung
    - Variante: Diffie-Hellman auf elliptischen Kurven (ECDH) → Analog zum klassischem
        - öffentliche Parameter: Primzahl p, Kurve E über GF(p), Punkt P auf Kurve E
        - (1) Alice und Bob wählen jeweiligen privaten Schlüssel: a, b \in {2, 3, . . . , | E | }
        - (2) Beide berechnen jeweiligen öffentlichen Schlüssel
            - Q_a = aP, Q_b = bP
        - (3) Wechselseitiger Austausch der öffentlichen Schlüssel Q_a, Q_b
        - (4) Beide Partner berechnen gemeinsamen ECDH-Punkt R: R = aQ_b = bQ_a
            - Assoziative Punkt-Addition über EC
        - (5) Partner generieren die Schlüsselbits für gemeinsamen Schlüssel k_ab
            - z.B. durch x Koordinate von R (l Anzahl der Bits des benötigten Schlüssels): k_ab = KDF(R_x, l)
        - => Deutlich kürzere Schlüssellängen als DH, für gleiches Sicherheitsniveau

Kapitel 5: Digitale Identität, Authentisierung

- Begriffe und Einordnung:
    - Authentisierung:
        - Eindeutige Identifikation und Nachweis der Identität
        - Identität: Name, Identifier, charakterisierende Attribute
    - Autorisierung:
        - eine Authentisierung ist idR Basis dafür
        - Vergabe von Zugriffsberechtigungen (z.B. Lese Recht)
    - Personenbeziehbare Daten DSGVO (Datenschutzgrundverordnung)
        - Alle Informationen die sich auf eine identifizierte oder identifizierbare natürliche Person beziehen
    - Pseudonymisierung:
        - Vearbeitung personenbezogener Daten in einer Weise, dass ohne Hinzuziehung zusätzlicher Informationen nicht mehr eine spezifischen Person zugeordnet werden können.
        - Diese zusätzliche Informationen sind gesondert aufbewahrt

        - Bsp. Nickname bei Mail; Matrikelnummer, gehashte Ids, TMSI beim Mobilfunk, Identifier bei CoronaWarnApp
    - Anonymisierung:
        - Verändern personenbezogener Daten, sodass Einzelangaben nicht mehr (oder mit unverhältnismäßig großen Aufwand an Zeit, Kosten und Arbeitskraft) einer bestimmten natürlichen Person zugeordnet werden können.
        - gilt als Verarbeitung von personenbezogenen Daten, unterliegt der DSGVO
        - Anonymisierte, anonyme Daten besitzen keinen Personenbezug, unterliegen der DSGVO nicht
            - Bsp. Ersetzen von Namen durch Zufallszahlen
- 3 Klassen von Basistechniken/Faktoren
    - Wissen (Passwort, PIN, Schlüssel):
        - Passworte:
            - Problem: Passwortdiebstahl
            - Angriffe:
                - Dictionary Angriffe: Durchprobieren
                - Phishing: Abfangen des Passwortes
                - Social Engineering: Nutzer gibt Passwort preis
            - Abwehr: gute Passworte, Mehrfaktor-Authentisierung
        - Was ist ein gutes Passwort?
            - Komplexe Richtlinien → unsichere Praktiken durch Nutzer
            - Nicht regelmäßig Passwort ändern
            - Nicht Erzwingen von mehreren Zeichenklassen
        - Einmal-Passworte (OTP):
            - Software Token:
                - zeitsynchronisiert: Google Authenticator
                    - Basis:
                        - Nutzer besitzt Kennung beim Server
                        - App auf Nutzer Gerät installiert
                        - Server: 80 bit Pre-Shared-Secret s
                        - Übertragung auf Nutzergerät
                        - App speichert s
                        - App und Server generieren unabhängig voneinander alle 30 Sek ein neues OTP = HMAC_SHA1(s|| Unixzeit_counter)
                    - Login:
                        - Kennung, Passwort und OTP
                        - Server validiert OTP
                    - Schwachstellen u.a.
                        - 80Bit zu kurz
                        - s in Klartext auf Smartphone in App gespeichert
                        - Server nicht Authentisiert
                - CR-basiert:
            - Hardware Token:
                - Basis
                    - Token besitzt eindeutige Nummer
                    - Server kennt diese Nummer
                    - Bsp. digitaler Personalausweis (mit Chip), ec-Karte, USB-Dongle

        - zeitsynchronisiert: RSA-SecureID-Token
            - Basis:
                - Token-Nummer und 128-Bit Seed s
                - s auch im RSA-Token gespeichert
                - 1-5 Jahre Tokenlebenszeit, automatisches Abschalten nach Ablauf
                - Erzeugung von OTP auf Server und Token alle 60 sec
                    - OTP = AES(TokenID | s | Zeit), 6-8 Stellige Zahl
            - Login:
                - OTP wird auf Display angezeigt
                - Nutzer gibt OTP ein
                - Validierung durch Server: Es werden die nächsten 3-5 OTP zugelassen
        - CR-basiert
    - Challenge Response (CR): Basisprotokoll zur wissensbasierten Authetizitätsprüfung
        - Allgemeiner Ablauf:
            - A gibt seine Identität an
            - B sendet eine Challenge (z.B. Zufallszahl) an A
            - A erstellt Response (z.B. mittels Verschlüsselung)
            - B prüft Response: falls korrekt, dann hat A ein geheimes Wissen nachgewiesen
        - Zwei Klassen:
            - symmetrisch:
                - Nutzt Pre-Shared key k.
                - Challenge: Zufallszahl
                - Response: Verschlüsselte Zufallszahl
            - assymetrisch:
                - Signatur- und Validierungsschlüssel
                - Challenge: Zufallszahl
                - Response: Signierte Zufallszahl (Zertifikat als Beweis das e_a von Id_a)
- Besitz (Smartcard, Token, SIM Karte)
- Biometrie (Fingerabdruck, Gesicht, Tippverhalten):
    - Biometrisches Merkmal: Verhaltenstypische oder physiologische Eigenschaft eines Menschen, die diesen eindeutig charakterisieren
    - Anforderungen an biometrische Merkmale:
        - Universalität: Jede Person besitzt das Merkmal
        - Eindeutigkeit: Merkmal ist für jede Person verschieden
        - Beständigkeit: Mermkal ist unveränderlich
        - quantitative Erfassbarkeit mittels Sensoren
        - Performance: Genauigkeit und Geschwindigkeit
        - Akzeptanz des Merkmals beim Nutzer
        - Fälschungssicherheit
    - Klassen:
        - physiologische Merkmale (statisch):
            - geringe Möglichkeit zur Auswahl oder Änderung von Referenzdaten
        - Verhaltensmerkmale (dynamisch):
            - nur bei bestimmter Aktion vorhanden
    - Vorgehen
        - Vorbereitung (Einmalig)
            - Messdatenerfassung durch biometrischen Sensor und Digitalisierung
            - Enrollment:
                - Registrierung eines Nutzers

        - Aufnahme, Auswahl und Speicherung der Referenzdaten (5–7 Fingerabdrücke)
    - Bei jeder Authentisierung:
        - Erfassung aktueller Verifikationsdaten
        - Verifikationsdaten digitalisieren
        - Mit gespeichertem Refernzwert (Template) vergleichen
            - Toleranzschwellen sind notwendig
    - Problembereiche:
        - Abweichungen zwischen Referenz und Verifikationsdaten
        - Zwei Fehlertypen:
            - Berechtigter Benutzer wird abgewiesen
                - Akzeptanzproblem (false negative)
            - Unberechtigter Benutzer wird authentifiziert
                - Sicherheitsproblem (false positive)
        - Leistungsmaße zur Bewertung der Güte eines Systems
            - False-Acceptance-Rate (FAR)
            - False-Rejection-Rate (FRR)
            - Equal-Error-Rate (EER): Gleichfehlerrate
        - Enge Kopplung zwischen Merkmal und Person:
            - Bedrohung der informationellen Selbstbestimmung
            - Gefahren durch gewaltsame Angriffe gegen Personen
            - Problem der öffentlichen Daten und rechtliche Aspekte
            - Ethisches Problem: Gefahr der Diskriminierung
- Multi-Faktor Authentisierung:
    - Kombination verschiedener Authentisierungsfaktoren
    - Ziel: Hürde für Angreifer erhöhen
    - z.B: Zwei-Faktor: Kennung und Passwort/PIN && Eingabe dynamisch generierter Einmal-TAN

Kapitel 6: PKI und Single-Sign-on (SSO):

- Fragestellungen:
    - Authentizität von Public Keys: Zertifikate und PKI
    - Authentisierung in verteilter Umgebung: SSO Protokoll
- Zertifikat:
    - Datenstruktur im Standardformat: X.509.v3
    - Bescheinigt Bindung von Public Key an die Identität einer Instanz/Subjekt (Person, Server, Gerät…)
    - Mit digitaler Signatur des Zertifikat-Ausstellers wird die Korrektheit der Daten bestätigt
    - X.509.v3 Zertifikat:
        - Ausgestellt durch eine Certificate Authority (CA)
        - Enthält Angabe zur CA (Issuer)
        - Enthält Public-Key e_a von A und Verfahren (z.B. RSA)
        - Enthält Information zum Ablaufdatum
        - Daten im Zertifikat mit dem privaten Key d_CA der CA signiert
- Public Key Infrastructure (PKI):
    - Komponenten:
        - Registration Authority (RA): bürgt für die Verbindung zw. Öffentlichen Schlüssel und Identitäten/Attributen
        - Certification Authority (CA): Stellt Zertifikate aus
        - Validierungsstelle (VA): Überprüfung von Zertifikaten
        - Verzeichnisdienst: Verzeichnis mit ausgestellten Zertifikaten
        - Personal Security Environment (PSE): Sichere Speicherung des privaten Schlüssels

    - Hierarchie von CAs:
        - Die Wurzel-Instanz: root besitzt ein root-Zertifikat
        - Root-Zertifikate sind selbstsigniert
        - Root CA stellt Zertifikate für untergeordnete CAs aus
        - Validierung eines Zertifikats: über Zertifizierungspfad
    - Zertifikatvalidierung: Prüfen der Gültigkeit der Signatur des Zertifikats
        - Validierungspfad: Jedes Zertifikat auf dem Pfad muss gültig sein
            - Signatur ist gültig
            - Keine veralteten Verfahren (SHA-1)
            - Zertifikat nicht zurückgerufen
            - Zertifikat ist noch nicht abgelaufen
            - Zertifizierungspfad endet bei einer Root-CA
    - Zertifikatsrückruf (Revocation)
        - Problem:
            - Privater Key ist offengelegt (z.B. Diebstahl oder verloren)
            - Privater Key ungültig geworden
        - Lösung:
            - Online Certificate Status Protocol (OCSP)
                - Server holt regelmäßig Bestätigung von CA über Gültigkeit
                - Bestätigung ist über mehrere Requests gültig
                - Server liefert Bestätigung beim Verbindungsaufbau mit
- Single Sign on (SSO):
    - Situtation:
        - Viele Dienste Angeboten
        - Nutzung dieser Dienste erfordert Authentisierung und Authentisierungsinformationen
    - Problem:
        - Nutzer muss für jeden Dienst Credentials übertragen
        - Sensitive Zugangsdaten auf vielen Rechnern
        - Kompromietierung eines Passworts → Zugang auf Services mit gleichem Passwort
    - Konzept:
        - Nutzer hinterlegt Credentials bei vertrauenswürdiger Instanz/Service AuC
        - Für Zugriff auf Service B besorgt sich Alice vom AuC eine Authentizitätsbescheinigung: Ticken/Token/Assertion
        - AuC prüft Authentizität von Alice und erstellt Ticket für B
        - Alice reicht Ticket an B weiter
        - B prüft ticket, keine eigene Authentisierung
    - Konsequenz: Trennung
        - Authetisierung durch AuC (Authentication Center): Policy Desision Point (PDP)
        - Prüfung auf Gültigkeit durch B: Policy Enforcement Point (PEP)
    - Kerberos-Protokoll:
        - AuC heißt Key Distribution Center (KDC), augeteilt:
            - Ticket Granting Service (TGS)
            - Authentication Service
        - Verwendet symmetrische Kryptografie
        - Pre-Shared Master Key mit KDC für jeden Nutzer und Dienst
            - Für Nutzer A: k_a aus Passwort von A abgeleitet
            - Für Dienst F: k_f zufällig generiert
        - TGS des KDC stellt auf Anfrage ein Authentizitäts-Ticket für A zur Vorlage bei einer Instanz/Service S aus

        - Ticket enthält u.a. Identitäten A und S und einen symmetrischen Shared Key k_as:
            - T^{as} = (S, A, addr, timestamp, lifetime, k_as)
                - S Servername, A anfordernde Instanz und addr ihre IP-Adresse, Key-lifetime, Schlüssel k_as für S und A
            - Ticket wird von TGS mit Master-Key k_s von S verschlüsselt:
                - E_{k_S} (T^{as}) (idR AES)
            - Ticket T^{as} dient der Instanz A zur Vorlage bei S:
                - A muss gegenüber S nachweisen, er berechtigter Besitzer des Tickets T^{as} ist
                - => Authenticator
        - Authenticator: $Authent^{A}$: von Instanz A erzeugt
            - $Authent^{A}$ = (A, addr, timestamp)
            - Authenticator mit Shared-Key k_as verschlüsselt: E_k ($Authent^{A}$)
            - Nachweis für S, dass A berechtigt ist, T^{as} vorzulegen => Kenntnis von k_as

Kapitel 7: Autorisierung und Rechtemanagement

- Autorisierung (authorization):
    - Vergabe von Zugriffsberechtigungen und Kontrolle: Verhindern von unautorisierter Ressourcennutzung
- Einführung:
    - Basis für Authorisierung:
        - Kenntnis, an wen Berechtigungen vergeben werden sollen: Authentisierung der Subjekte
    - Aufgabe:
        - Festlegen der Menge der Zugriffsrechte für Ressourcen
        - Festlegen eines Regelwerks und Kontrolle der Zugriffe
        - (Dynamisches) Management von Rechten:
            - Erteilen
            - Einschränken
            - Entziehen
            - Zugriff verbieten
        - Regelwerk: wer darf was, auf welche Weise, unter welchen Bedinungen nutzen (permit, deny)
            - Wer: Zugreifende Instanz identifizieren:
                - Subjekt (z.B. Nutzer, Prozess, Server, Service)
            - Auf was: Objekt/Ressource identifizieren:
                - Objekt/Ressource (z.B. Datei, Datenbankeintrag, Page
            - Auf welche Weise: Zugriffsrechte spezifizieren:
                - Zugriffsrecht: z.B. r,w,x, Suchen, Kamera_an, Internet
            - Unter welchen Bedingungen: Kontexte spezfizieren:
                - Kontexte/Attribute: z.B. Zweck, Rolle, Tageszeit, Alter
    - Prinzipien der Rechtekontrolle
        - Prinzip der minimalen Rechte (need-to-know), least priviledge: Nur Rechte vergeben, die zur Aufgabenerfüllung erfordelich sind
        - Complete Mediation, Zero Trust: Jeder Zugriff auf eine geschützte Ressource ist zu kontrollieren
    - Identity und Access Management (IAM):
        - Aufgaben: Zugriffskontrolle und Rechtemanagement, Authentisierung, Autorisierung, Audit (Protokollieren)

        - Modellierungsansätze zur Vergabe von Berechtigungen:
            - Discretionary Access Control (DAC): benutzerbestimmbare Vergabe/Rücknahme, Ressourcen-Owner bestimmt
            - Role-based Access Control (RBAC): Aufgaben orientiert
            - Mandatory Access Control (MAC): systembestimmte Vergabe/Rücknahme, globale, systemweite Regelungen
- Modelle für das Rechtemanagement:
    - Zugriffsmatrix-Modell (ZM)
        - (Dynamische) Menge von Objekten O
        - (Dynamische) Menge von Subjekten S, mit S untermenge von O
        - Menge von Rechten R
        - Zugriffsmatrix M: S x O → 2^R, Schutz-Zustand zur Zeit t
        - Dynamik:
            - neue Subjekte
            - neue Objekte
            - Rechtevergabe
            - Rechte-Entzug
            - Löschen
        - Positiv:
            - Sehr einfaches Modell
            - Einfach zu Implementieren
        - Negativ:
            - Skaliert schlecht, Rechtevergabe orientiert an Subjekten, nicht an zu erledigen Aufgaben → Verbesserung durch Gruppenkonzept
            - Rechterücknahme ist aufwendig
    - RBAC (Role-based Access Control):
        - Rolle: beschreibt Aufgabe und damit Verbundene Verantwortlichkeiten, Pflichten und Berechtigungen
        - Modell:
            - Menge von Subjekten, Objekten
            - Menge von Rollen
            - Subjekte sind Rollen zugeordnet: sr: S → 2^R
            - Menge von Zugriffsrechten P (permission) für Objekte
            - Rollen erhalten Rechte: pr: Role → 2^P
            - Nutzer s meldet sich mit einer Rolle an: RL untermenge von sr(s)
                - RL: Menge der aktiven Rollen von s
        - Rollenhierarchien:
            - Vererbung von Zugriffsrechten entlang der Rollenhierarchie
            - Basis: Definition einer partiellen Ordnung <= auf Role
            - Vererbung der Rechte: hierarchisch höhere Rollen erben die Rechte von tieferen Rollen
        - Positiv:
            - Gute Modellierung unternehmerischer Strukturen und Prozesse
            - Gut skalierend in Bezug auf Subjekt Dynamik
            - Rollen sind eher statisch, Rollenmitgliedschaft dynamisch:
                - Rollen-Zuweisung: automatisierte Vergabe der Rechte
                - Rollen-Entzug: automatisierter Entzug der Rechte
            - Integriert in vielen Produkten (z.B. SAP)
        - Negativ:
            - Vererbungskonzept verstoßt idR gegen need-to-know
            - Umsetzung: häufig Hunderte von Rollen
            - Keine Kontrolle von Informationsflüssen

- Bell-LaPadula-Modell (BLP):
  - Ziel: Einfach zu prüfende, systemglobale (mandatory) Regeln zur Beschränkung von Informationsflüssen
  - Lösung: Multi-Level-Security (MLS):
    - Klassifizieren von Subjekten und Objekten mit Label
    - Definition einer Fluss-Relation, die die erlaubten Flüsse zwischen klassifizierten Objekten und Subjekten beschreibt
  - BLP-MAC-Policy: no-read-up, no-write-down
  - Vereinfachtes Model:
    - Menge der Zugriffsrechte (ZM)
    - Das Label eines Subjekts heißt Clearance von s
    - Das Label eines Objekts heißt Classification von o
    - Partielle Ordnung auf der Menge der Labels
  - BLP Regeln:
    - (1) Simple security Property (no-read-up-Regel):
      - Subjekt s kann Recht z (read-only r, execute x) auf Objekt o nutzen wenn:
        - z in M(s, o) && sc(o) <= sc(s)
    - (2) *-Property (no-write-down-Regel):
      - z (append a, read-write rw)
      - Subjekt s kann Recht a auf Objekt o nutzen wenn:
        - a in M(s, o) && sc(s) <= sc(o)
      - Subjekt s kann Recht rw auf Objekt o nutzen wenn:
        - rw in M(s, o) && sc(s) = sc(o)
  - Positiv:
    - Einfach zu implementieren
    - Formales Modell, potentiell beweisbare Eigenschaften
    - Gut geeignet zum Nachbilden hierarchischer Informationsflussbeschränkungen
    - Varianten der MLS-Policy in der Praxis stark verbreitet (z.B SE-Linux)
  - Negativ:
    - Problem des Blinden Schreibens: s darf o modifizieren aber anschließend nicht lesen
    - Information/Objekte werden sukzessive immer höher eingestuft
    - Sanitizing Konzept in der Praxis: Zurückstufen
- Konzepte zur Verwaltung von Zugriffsrechten:
  - Basis: Varianten einer Zugriffsmatrix, da idR dünn besetzt und als Tabelle ineffizient
  - Zugriffskontrollliste: Access Control List (ACL):
    - Rechte an Objekten
    - Spaltenweise Realisierung der Zugriffsmatrix:
      - ACL (Datei 1) = ((Bill, {owner, r, w}), (Joe, {r, w}), (Alice, {}))
    - Beispiel Unix/Linux:
      - Subjekte/Prozesse: identifiziert über UID, GUID
      - Zu schützenden Objekte: Dateien, Verzeichnisse.
      - Dateien/Verzeichnisse werden Betriebssystem-intern über einen Datei-Deskriptor, den i-node (index-node), beschrieben:
        - i-node enthält Name des Datei-Owners und die ACL
        - i-nodes auf der Festplatte verwaltet
        - beim öffnen der Date (open-call) wird i-node eingelagert

        - Positiv:
            - Einfache Implementierung:
                - z.B. Erweiterung der BS-Datenstruktur um ACL
            - Einfaches Enforcement/Kontrolle:
                - Bei Zugriff auf Objekt: prüfen der ACL des Objekts
            - Rechtemanagement aus Objekt-Sicht einfach: Einfügen/Löschen
        - Negativ:
            - Fehleranfälliges Rechtemanagement aus Subjekt-Sicht:
                - Subjekt ändert seine Aufgaben → Welche Objekte sind betroffen?
    - Capability-Konzept:
        - Ticket, Token, Handle
            - Zugriffstiket enthält Objekt-UID und Rechte-Bits
            - Capability-Besitz berechtigt zur Wahrnehmung der Rechte
        - Zeilenweise Realisierung der Zugriffsmatrix:
            - Für jedes Subjekt s wird eine Capability-Liste verwaltet
        - Management von Capabilities/Token:
            - Erstellen und Verwalten als BS-Datenstruktur (geschützt durch BS)
            - Erstellung von Access Token durch Authorisierungs-Server und Verteilung an Client Rechner (OAuth-Framework im Web-Umfelt)
        - Positiv:
            - Subjekt-bezogene Rechteverwaltung ist einfach
            - Ticketweitergabe/Delegation möglich
        - Negativ:
            - Rechterücknahme aus Objektsicht schwierig
            - Stehlen oder kopieren von Tickets ist möglich
- Zugriffskontrolle:
    - IdR Aufteilung in Berechtigungs- und Zulässigkeitskontrolle
    - Berechtigungskontrolle: PDP Policy Decision Point
        - Prüfung beim erstmaligem Zugriff auf ein Objekt (z.B. open)
        - Ausstellung einer Bescheinigung (z.B. Capabilitie, File-Handle)
    - Zulässigkeitskontrolle: PEP Policy Enforcement Point
        - durch Objektverwalter (z.B. Server) oder BSKernel
        - bei Objektzugriff: prüfung der Gültigkeit der Bescheinigung
        - kein Zugriff auf die Rechteinformation z.B. ACL, notwendig
    - Hybride Konzepte:
        - PDP auf Basis von ACLs, Ausstellen einer Capability
        - PEP bei Vorlage von Capabilities

Kapitel 8: Betriebssystemsicherheit

- Speicherschutz:
    - Von-Neumann Architektur: universelle Interpretierbarkeit
    - Prozessadressraum: Stack, Heap, Text (Code)
        - Stackbereich reservieren
        - Mit Werten beschreiben
        - Rücksprungadresse: Befehl im Code-Segment, Ausführung nach Return
        - => Heap/Stack häufige Angriffsziele
    - Return orientet Programming:
        - Rücksprung nicht zu Funktionsanfang sondern zu beliebiger Instruktionsfolge im Programm, die mit einem Return endet (ROP-Gadget)
        - Verknüpfung von ROP-Gadgets zu ROP-Chain mit beliebiger Funktionalität
    - Stack-Shielding, Stack Canary:

        - Einfügen eines zufälligen, zur Laufzeit erzeugten Canary-Werts auf dem Stack vor der gespeicherten Rücksprungadresse
        - Vor dem Return: Programm vergleicht Wert mit Referenzwert. Veränderung → Programmterminierung
        - Erzeugungs- und Prüflogik wird durch Compiler ins Programm eingebracht
    - DEP Data Execution Prevention:
        - Problem:
            - Universelle Interpretierbarkeit von Neumann Architektur
            - Nur .text Segment sollte ausführbar sein
        - Lösungskonzept:
            - CPU-Feature: NX-Bit (No-eXecute):
                - Seiten als nicht ausführbar markiert
            - Ausführbare Seiten nicht schreibbar:
                - W xor X Policy
            - => Verhindert dass Angreifer Shellcode ausführt
    - ASLR Address Space Layout Randomization
        - Problem: DEP schützt nicht vor ROP
        - Lösungskonzept:
            - Address Randomization von Stack, Heap und Bibliotheksfunktionen individuell für jeden Programmstart (Linux) / pro Systemstart (Windows)
            - Text-Segment nicht Randomisiert
        - Erweiterung:
            - PIE Position Independent Executable
- Virtualisierungskonzepte:
    - Virtual Machine Monitor (VMM) bzw. Hypervisor:
        - Software die HW-Schnittstelle zur Verfügung stellt
        - Sicherheitsrelevante Eigenschaften:
            - Isolation:
                - VMs sind besser von einander und vom VMM abgeschottet als Prozesse in einem BS
            - Überwachung:
                - VMM hat vollen Zugriff auf Ressourcen und Zustand der VMs und der virtuellen Hardware („sieht“ alles)
            - Kontrolle:
                - VMM kann VMs kontrollieren und verändern (anhalten, neu starten, Netzwerkverkehr filtern)
    - Betriebssystem-Level Virtualisierung: Container:
        - Beobachtungen:
            - Wenige Anwendungen auf gleichen BS voneinander abgeschirmt auf Sandbox
            - VM pro Sandbox sorgt für Isolierung
        - Aber:
            - VM umfasst Anwendungsprogramm plus Betriebssystem
            - BS und Bibliotheken belegen Speicher in RAM und auf Festplatte mehrfach
        - Lösung:
            - BS-Level Virtualisierung
            - Container-Lösung
        - Konzept:
            - Leichtgewichtige Virtualisierung
            - Anwendungsprogramm mit Bibliotheken und Abhängigkeiten in einer Einheit gebündelt
            - Container als isolierter Prozess im User-Space
            - Mehrere Container nebenläufig auf einem Rechner (teilen sich BS)

Kapitel 9: Netzwerksicherheit

- Schutz der Endsysteme:
  - Firewall: Perimeter-Schutz, Filtern von Daten-Paketen
- Transportsicherheit:
  - Aufbau und Nutzung eines sicheren Kommunikationskanals: TLS/SSL, SSH, IPSec, Bluetooth
- Firewall-Architekturen:
  - Netzwerk in Netzsegmente unterteilt
  - An Segmentgrenzen Kontrolle durch Firewall:
    - Firewall-Regeln: erlaubte Datenflüsse festlegen
    - Kontrolle der Einhaltung der Regeln:
      - Ein- und Auslasskontrolle für Datenpakete
  - Format von Datenpaketen:
    - Header: Absender/Empfänger Info, verwendetes Protokoll
    - Payload: Nutzdaten, die gesendet werden
  - Paketfilter: einfache Form einer Firewall
    - Analyse der Header nach Firewall-Regeln:
      - Festgelegt z.B. in iptables (unter Linux)
      - Bsp. Quelle (src), Ziel (dst) IP-Adresse
    - Aktion pro Datenpaket: Paket accept, drop, reject
  - Deep Paket Inspection (DPI):
    - Analyse der Paketheader und Nutzdaten (Deep Inspection)
    - Prüfung auf Protokollverletzung und Malware
    - Missbrauchspotential: Zensur (z.B. Verbindungsaufbau mit TOR-Netzwerk erkennen)
  - Application Layer Gateway (ALG) / Proxy-Firewall:
    - Verbindung in Firewall terminiert und neue Verbindung aufgebaut (Relay-Konzept)
    - Firewall hat Zugriff auf kompletten Verbindungszustand
    - Firewall kann aktiv in die Verbindung eingreifen und Daten verändern
- Transportsicherheit: Sichere Verbindungen
  - Protokolle: (SSL/TLS, IPSec, SSH, WPA2):
    - Unterschiedliche Ausprägungen umgesetzter Schutzziele
      - Authentifikation:
        - nur Gerät oder pro Dienst/Nutzer, einseitig oder wechselseitig
      - Vertraulichkeit:
        - 1 Schlüssel pro Verbindung oder Schlüssel pro Datenpaket
      - Integrität:
        - Integrität des Verbindungsaufbaus oder pro Datenpaket
      - Schlüsselaustausch:
        - Pre-Shared Key, DH-Verfahren, RSA-Verfahren
      - Verfahren:
        - Statisch festgelegte Verfahren oder konfigurierbar
  - TLS 1.3 (Transport Layer Security):
    - Verschlüsselungsverfahren: 5 Ciphersuiten festgelegt
    - Schlüsselaustausch: ECDH auf festgelegten Kurven
    - Authentisierung: X509.v3 Zertificate
    - Digitale Signaturen: RSA, ECDSA

- Ablauf:
  - Aufbau des sicheren Kanals: Handshake Protokoll
    - Abstimmung des Verfahren, der Cipher-Suite
    - Authentisierung (einseitig, beidseitig)
    - Erzeugung von zwei Kommunikationsschlüssel k_ab, k_ba pro Richtung und eines MAC-Schlüssel pro k_mac für den Handshake
  - Verschlüsselte Kommunikation: Application Data Protokoll
    - Datenpakete der jeweiligen Anwendung (z.B. HTTP) werden verschlüsselt und versandt
    - Empfänger entschlüsselt Paket und die Anwendung verarbeitet Daten weiter
- Erläuterungen zum Handshake:
  - Client (Rechner A): Startet Verbindungsaufbau: ClientHello
    - ID bezeichnet neue Verbindung (sessionID)
    - cs_a Liste von Cipher-Suites, die A unterstützt, Zufallszahl R_a
  - Server B: hat statischen Signaturschlüssel und Serverzertifikat für Server-Key e_b
    - B weist nach, dass er auf aktuelle Hello Nachricht antwortet → signiert m1
    - B signiert eigene Daten in m2
    - Mit DH-e_a und DH-d_b → B berechnet DH-Secret s
    - B wählt aus cs_a Suite aus: cs
    - B weist nach, dass er DH-d_b und damit s kennt → generiert k_AB, k_BA, k_mac
  - B antwortet mit ServerHello und finish:
    - Sendet eigenen DH-e_b, die gewählte Suite cs, seine R_b
    - Signierte Daten und Zertifikat cert(e_b) , sodass A die Signatur prüfen kann
    - Server kann bereits verschlüsselte Anwendungsdaten zurücksenden (k_BA bereits berechnet) → Variante von TLS1.3 1RRT (Round Trip Time)
  - A prüft Signatur und Integrität von m1, m2, m3
    - Signaturvalidierung mit e_b aus dem Zertifikat cert(e_b)
    - A berechnet mit DH-d_a und DH-e_b das DH-Secret s und generiert die notwendigen Schlüssel:
      - k_AB, k_BA, k_mac = KDF (s, R_a, R_b)
    - A entschlüsselt D_kAB(m4) und prüft Integrität von m1, m3
    - A sendet finish und informiert B damit, dass alles ok ist
- Weitere Varianten des Handshakes:
  - TLS1.3 mit beidseitiger Authentisierung:
    - Server B sendet einen Certificate Request
    - A muss Client-Zertifikat cert(e_a) vorweisen
    - Server prüft Zertifikat, prüft Signatur und entschlüsselt
  - TLS1.3 anonym: keiner muss sich authentisieren
  - TLS1.3 0-RTT:
    - Client cashed schlüssel aus früheren Handshake: PSK
    - Client kann direkt mit Hello Nachricht verschlüsselte Daten senden: AEAD_PSK(ApplicationData)

- Virtual Private Networks (VPN):
  - Netzinfrastruktur bei dem Komponenten eines privaten Netzes über ein öffentliches Netz kommunizieren und die Illusion besitzen, dieses Netz zur alleinigen Verfügung zu haben:
    - Bsp. End-to-Site-VPN / Remote-Access-VPN
      - Anbindung von einzelnen PCs in ein größeres Intranet (z.B. Home-Office)

Kapitel 10: Secure Programming/ Secure Coding

- Entwicklung von Code/Programmen unter Einhaltung von:
  - Best Practice Regeln
  - Guidelines
  - Prozessen
  - => Um mögliche Schwachstellen in Programmen zu reduzieren
- Maßnahmen:
  - Secure Coding Guidelines: häufige Programmierfehler z.B. fehlende Eingabefiltrierung zu vermeiden
  - Bei Programmiersprachen vorhandene bzw. fehlende Sicherheitskonzepte beachten (Memory safe vs Unsafe konzepte)
  - Sicheren Entwicklungszyklus etablieren: Testen, Code-Review, Analyse…
  - Toolchain: Werkzeuge u.a. zur Prüfung, Filterung einsetzen

Kapitel 11: Informationssicherheitsmanagementsystem (ISMS)

- Festlegung von Verfahren und Regeln innerhalb einer Organisation, zur Steuerung, Kontrolle und zum Aufrechterhalten sowie fortlaufenden Verbesserung der Informationssicherheit
- Lebenszyklus der Informationssicherheit: Plan, Do, Check, Act
  - Problemstellung → Plan:
    - Funktionale Anforderungen
    - Einsatzumgebung
    - Bedrohungs- und Risikoanalyse
    - Auswahl von Maßnahmen
  - Maßnahmen → Do
    - Sicherheitsregelwerk
    - Definition von Kontrollen
    - Umsetzung von Sicherheitsmaßnahmen
  - funktionsfähiges System → Check:
    - Überwachung der Zielerreichung:
      - Testen, Pentesten
      - Validieren
      - Code-Review
  - Bewertung → Act → Problemstellung:
    - Beseitigung von Mängeln, Schwächen bzw. Optimierungen
    - Anpassen, Patchen
    - Update
- STRIDE:

Klasse	Bedrohung	Abwehr
Spoofing:	Phishing, Klartextpassworte, Identitäts-, Credentialdiebstahl, gefälschte Identität (IP-, Mail-Adr)	Mehrfaktor Authentisierung, Sichere Credential-Verwaltung, SSO-Protokolle (u.a. Kerberos), PKI
Tampering	Unautorisierte Datenänderungen, z.B. Datenbank, Nachrichtenpaket	Hashfunktion; MAC-Verfahren, Zugriffskontrollen, ACLs, Zugriffs-Logs
Repudiation	Abstreiten von durchgeführten Aktionen, z.B. nicht zuordenbare Zugriffe (modifizierend, lesend)	Digitale Signatur, Authentisierung und Protokollierung, MAC-Verfahren (nur eingeschränkt)
Information Disclosure	Unberechtigte Informationsweiter-gabe, offene Übertragung, unkontrollierte Zugriffe	Verschlüsselte Kommunikation (TLS), verschlüsselte Dateien, RBAC/BLP Zugriffskontrolle Authentisierung,
Denial of Service	Berechtigte Nutzer behindern; z.B. Ransomware, Web-Server mit Anfragen fluten, Spam	Zugriffe kontrollieren und filtern (u.a. Firewall, SPAM-Filter), Priorisierung, Quoten
Elevation of Privilege	Rechteerweiterung, privilegierte Zugriffe erlangen, z.B. über Buffer Overflow, administrator-Rechte	Eingabeprüfung (BO), Isolierung, Schutz-Zonen, Sandboxing, Mehrfaktor-Auth. Hardware-Schutz