Grundlagen

Schutzziele kryptographischer Verfahren

- Vertraulich: Nur Empfänger kann Nachricht lesen
- Integrität: Nachricht nicht manipulierbar
- Authentizität: Empfänger kann sich von der Identität des Absenders zweifelsfrei überzeugen
- Zurechenbarkeit: Empfänger kann beweisen von wem die Nachricht stammt
- Anonymität der Sender/Empfänger

Grundbegriffe

- Key Space: Menge von Schlüsseln
- Plaintext Space: Menge von Klartexten
- Ciphertext Space: Menge von Schlüsseltexten
- Key Generation: Schlüsselerzeugung
- Encryption: Verschlüsselung
- Decryption: Entschlüsselung
- Konfusion: Beziehung zw. Klartext und Chiffre verschleiern
- Diffusion: Bit aus dem Klartext soll viele Bits aus dem Chiffretext beeinflussen

Symmetrische Verfahren

- Ver- und Entschlüsselung mit gleichem Schlüssel
- Beispiele: Cäsar, Vigenere, Vernam, DES, AES
- Vorteil: Hohe Geschwindigkeit
- Nachteil: Schlüssel muss geheim bleiben

Asymmetrische Verfahren

- Verschlüsselung mit öffentlichem Schlüssel
- Entschlüsselung mit privatem Schlüssel
- Beispiele: RSA, ELGamal, DSA
- Vorteile: Ermöglicht digitale Signaturen, einfacher Nachrichtenversand
- Nachteil: höherer Rechenaufwand, Anfällig für Implementierungsfehler

Weitere Verfahren

- Hybrid: Asymmetrischer Schlüsselaustausch und symmetrisches Verfahren
- Verschiebungschiffre: Symmetrische Verfahren mit verschobenem Alphabet
- Transpositionschiffre: Symmetrisches Verfahren mit Permutation
- Substitutionschiffre: Jedes Zeichen wird durch festgelegtes Zeichen ersetzt
- Blockchiffre: Text wird in Blöcke aufgeteilt
- Stromchiffre: Zeichen aus Klartext werden mit Zeichen aus Schlüsseltext verrechnet
- Perfekte Geheimhaltung: Nicht entschlüsselbar ohne Schlüssel
- Mehrfachverschlüsselung: Kombiniert mehrere Verfahren (Keine Garantie für mehr Sicherheit)

Kerkhoff's Grundregeln

- System ist unentzifferbar
- System darf keine Geheimhaltung erfordern
- Schlüssel ist leicht zu merken und auswechselbar
- System ist telegraphisch kompatibel
- System muss transportierbar sein
- System muss einfach anwendbar sein

Angriffsmethoden

Grundbegriffe:

- Existential break: Erlangt Schlüssel-Klartext Paar
- Selective break: Einzelne Chiffren entschlüsselbar
- Universal break: Universelle Entschlüsselung ohne Schlüssel
- Total break: Angreifer hat den Schlüssel

Angriffstypen:

- Bruteforce: Probier alles aus
- Ciphertext-Only: Nur Schlüsseltext bekannt
- Known-Plaintext: Kennt andere Klartext-Chiffretext Paare
- Chosen-Plaintext: Selbstgewählte Texte können verschlüsselt werden
- Chosen-Chiphertext: Schlüsseltext wird vom Opfer entschlüsselt
- Replay-Angriff: Angreifer wiederholt abgefangene Schlüsseltexte
- Substitutionsangriff: Nachricht umkehrbar
- Häufigkeitsanalyse

Blockchiffren

Stromchiffren

- Schlüssel ist solange wie Klartext
- Element aus Schlüssel wird mit Element aus Klartext verrechnet (z.b. XOR)
- Beispiel Schlüssel: Gedicht
- Synchron: Schlüssel vorher bestimmt
- Asynchron: Schlüssel auch abhängig von Chiffrat

Blockchiffren

- Zerlege Klartext in Blöcke gleicher Länge (Padding)
- Verschlüsselt Blöcke gemäß Betriebsmodi

Betriebsmodi

- ECB: Blöcke unabhängig voneinander mit derselben Funktion verschlüsseln
- CBC: Blöcke abhängig vom vorherigen Block verschlüsseln
- CFB: Realisiert asynchrone Stromchiffre
- OFB: Realisiert synchrone Stromchiffre

Sicherheit

- Verschlüsselung kann auf Gleichungssysteme reduziert werden und somit algebraisch gelöst werden
- Zweifachverschlüsselung unpraktisch
- Dreifachverschlüsselung sinnvoll

Affin Lineare Blockchiffren

Affine Chiffre

- Blockchiffre der Länge 1
- Schlüsselraum aus Paaren $a, b \in \mathbb{Z}_m$, a relativ prim zu m wegen Eindeutigkeit
- Klartext wird mit affiner Funktion verschlüsselt: Funktion mit additivem Faktor ungleich 0 (z.b. 8*x + 4)
- Entschlüsselung über Inverse zu a
- Einfach lösbar durch Known Plaintext Angriffe (Gleichungssystem lösen)
- Geringer Schlüsselraum (BruteForce-Angriff)

Affin Lineare Blockchiffre

- Klartext wird in Blöcke unterteilt, welche mit einer affin linearen Funktion verschlüsselt werden
- Beispiel Viginere-Chiffre

Sicherheit:

- Durch Known-Plaintext Angriffe knackbar
- Schlüssellänge ist Sicherheitsfaktor: Schlüssellänge = Klartextlänge macht das System sicher

Perfekte Geheimhaltung

Definition

Perfekte Geheimhaltung bedeutet, dass Klartext und Schlüsseltext stochastisch unabhängig sind!
Nach Shannon:
Perfekt geheim, wenn Schlüsselraum gleichverteilt ist und jeder Klartext genau einen Schlüssel hat.

Vernam-Chiffre

- System mit perfekter Geheimhaltung
- Spezialfall der Vigenere-Chiffre
- Schlüssel und Klartext gleich lang
- Schlüssel ist rein zufällig und einmalig
- Elemente des Klartextes und des Schlüssels werden paarweise verrechnet
- Schlüssel bringt Probleme: Sehr kompliziert und Übermittlung schwierig
- Kerkoff erfüllt ausgenommen leichter Schlüssel

Sicherheit:

- Bekannte Angriffsmethoden nur möglich, wenn man zweimal den gleichen Schlüssel für unterschiedliche Texte verwendet!

Perfekte Authentizität

- Nachricht soll vor Fälschung geschützt sein
- Nachricht soll von erwartetem Absender kommen
- Möglicher Ansatz: Teile des Schlüssels preisgeben. Somit muss der dazugehörige Klartext gleich bleiben, wodurch fälschen schwierig wird.

DES-Algorithmus

DES-Algorithmus

Grundlage:

- Symmetrische Blockchiffre
- Basiert auf der Feistel-Chiffre, wobei die Anzahl der Rundenschlüssel auf 16 festgelegt ist
- Schlüssellänge 64 Bit, wobei 56 nutzbar sind (8 Bit Parität)
- Es werden aus dem Schlüssel 16 Rundenschlüssel je 48 Bit erzeugt
- Erfüllt Diffusion und Konfusion

Ablauf:

- Klartext wird in 64 Bit Große Blöcke geteilt
- Initiale Permutation: Bits eines Blockes werden gemäß Tabelle permutiert
- Mit den 16 Rundenschlüsseln werden die Blöcke verschlüsselt
- Ausgangspermutation: Inverse zur Initialen Permutation

Triple-DES:

- Verwendet 3 Schlüssel K1, K2, K3
- Verschlüsselung: Enc. mit K1, Dec. mit K2, Enc. mit K3
- Entschlüsselung: Dec. mit K1, Enc. mit K2, Dec. mit K3

Sicherheit des DES-Algorithmus

- Mit BruteForce gebrochen
- Schwache Schlüssel reduzieren die Komplexität der Verschlüsselung und sollten vermieden werden
- Triple-DES weiterhin sicher
- Triple-DES kann mit Meet in the Middle angegriffen werden
- AES ist heute besser als DES

AES-Algorithmus

AES-Algorithmus

Grundlage:

- Symmetrische Blockchiffre
- 128 Bit Blöcke bzw. 16 Byte
- Schlüssellänge ist variable (128,192,256)
- Schlüssellänge legt Rundenanzahl fest
- Besonders effizient in Hardware

Vorgehensweise:

- Vorbereitung:
  - Aus Schlüssel nötigte Rundenschlüssel erzeugen (Key Expansion)
  - Substitutionsbox (S-Box) erzeugen/ bereitlegen
  - Blöcke in 2-dim Tabelle umwandeln
  - Block einmal mit Schlüssel verrechnen
- Verschlüsselungsablauf je Runde:
  - SubBytes: Substitutions mittels S-Box
  - ShiftRows: Zeilen permutieren, festgelegte Reihenfolge
  - MixColumns: 4 Bytes permutieren, anhand von

Galois Körpern
  - Schlüssel mit Block verrechnen
  - Abschließend:
  - SubBytes, ShiftRows, Schlüssel mit Block verrechnen
- Entschlüsselung: Alle Verschlüsselungschritte besitzen Inverse, mit welchen entschlüsselt wird

Sicherheit

- Mehr Runden steigern Widerstandsfähigkeit, größere Blöcke verringern diese
- Meet in the Middle Angriffe möglich
- Angriff auf algebraischer Ebene mittels der Lösung von Gleichungssystemen
- Seitenkanalangriffe durch schlechte Implementierung

Primzahlen

Probabilistische Primzahltests
Nur bestimmte Wsk, dass Zahl prim ist!

Primzahltests

Probedivision:

- Teile Zahl n durch alle Primzahlen in $\{2, ..., \sqrt{n}\}$
- Lässt sich n durch keine dieser Zahlen teilen, hat man eine Primzahl

Fermat-Test:

- Probabilistischer Primzahltest
- n ist prim, wenn: $a^{n-1} \equiv 1 \ mod \ n$ und $ggT(a, n) = 1$
- Anhand des kleines Satzes von Fermat
- Carmichael Zahlen: Es gibt kein zulässiges a, sodass der Test richtig funktioniert

Miller-Rabin-Test:

- Probabilistischer Primzahltest
- Bessere Fehlerrate als der Fermat-Test: $\leq 1/4$ pro Durchlauf
- Verwendet Verschärfung des kleinen Satzes von Fermat

Solovay-Strassen-Test:

- Probabilistischer Primzahltest
- Auf Grundlage des Jacobi-Symbols

Faktorisierung

Grundlage:

- Faktorisieren: Aufspaltung einer Zahl in Primfaktoren
- Nicht in polynomialer Zeit möglich
- Probedivision ist die einfachste Variante

P-1 Methode:

- Anwendbar, wenn p-1 nur kleine Primfaktoren hat
- Sei N zu faktorisieren, m Variable und p prim
- Nach Fermat: $a^m \equiv 1 \ mod \ p$, wobei m Vielfaches von p ist
- D.h. $a^m - 1$ ist Vielfaches von p
- p kann mittels $ggT(a^m - 1, N) = p$ berechnet werden
- m lässt sich leichter bestimmen, wenn p-1 kleine Primfaktoren hat

Quadratische Sieb:

- Sei N zu faktorisieren
- Finde x,y sodass: $x^2 \equiv y^2 \ mod \ N$
- Daraus folgt: $(x - y) \cdot (x + y) \equiv 0 \ mod \ N$
- D.h. $ggT(x - y, N)$ ist Teiler von N
- x und y zu finden ist schwierig

Primzahlerzeugung

Sieb des Eratosthenes:

- Liste von 2 bis n erzeugen
- Nacheinander nicht Primzahlen löschen
- Ergebnis: Liste von Primzahlen

Shor-Algorithmus:

- Zur Faktorisierung von Zahlen auf einem Quantencomputer
- Somit ein exakter und polynomieller Primzahlbeweis
- Grundlage bildet Fourier Transformation
- Technisch noch nicht ausgereift

RSA

Vorgehensweise

Schlüsselerzeugung:

- Wähle 2 Große Primzahlen p und q und sei $n = p \cdot q$
- Bestimme $\varphi(n) = (p - 1) \cdot (q - 1)$
- Wähle ein e, das teilerfremd zu $\varphi(n)$ ist
- Berechne d sodass: $d \cdot e \equiv 1 \ mod \ \varphi(n)$
- Public key: (e,n) und Private key: d

Ver- und Entschlüsselung:

- Klartext in Zahlenfolge umwandeln
- Verschlüssle jedes Element mit: $c = m^e \ mod \ n$
- Entschlüssle jedes Element mit: $m = c^d \ mod \ n$
- Zahlenfolge wieder in Klartext umwandeln

Sicherheit

Allgemein:

- Basiert auf dem Faktorisierungsproblem
- Daher werden immer größere Primzahlen verwendet

Angriffe:

- Low Exponenten Angriff möglich
- Man in the Middle Angriff möglich
- Faktorisierungsproblem lösen

Vorteile:

- Praktisch sehr sicher
- Als digitale Signaturen anwendbar

Nachteile:

- Sicherheit theoretisch nicht bewiesen
- Sehr rechenintensiv
- Nicht Post Quantum sicher

Kryptografische Hashfunktionen

Hash Funktionen

- Hash Funktionen bilden beliebig Lange Strings auf Strings fester Länge ab
- Hash-Wert ist von allen Eingangsbits abhängig
- Hash Funktion: Deterministisch, effizient, surjektiv, kein Schlüssel benötigt
- Kompressionsfunktion: Bildet String fester Länge auf kürzere Länge ab. Informationen gehen verloren. Ist kollisionsresistent
- Merkle Damgard Konstrukt: Nachricht wird gepaddet und dann Blöcke fester Länge gehasht
- Einwegeigenschaft: Ausgangswert nicht berechenbar
- Schwache Kollisionsresistenz: Es ist unmöglich zu einer Eingabe mit Hash eine zweite Eingabe zu finden
- Starke Kollisionsresistenz: Keine zwei Eingaben haben gleichen Hash-Wert
- Hash Funktionen werden zur Signatur verwendet

Geburtstagsangriff

- System ist nicht kollisionsresistent
- Erzeuge solang eine Nachricht, bis man gleichen Hash-Wert hat
- Sind in etwa $\sqrt{2^n}$ Hashs notwendig
- Man kann nun eine abgefangene Nachricht austauschen
- Bsp: Gegenteil der Nachricht solange variieren, bis man passenden Hash findet

Secure-Hash-Algorithm

- SHA 0, 1 sind unsicher, SHA 2, 3 sind sicher
- Vorgestellt wird hier SHA 1
- Eingangsnachrichten haben max. Größe
- Zuerst Padding sodass Text in 512 bit Blöcke zerteilt werden kann
- Kompressionsfunktion mehrfach anwenden (80x)
- 160 Bit Ausgabewert

Message Authentication Codes

- Stellen sicher, dass Nachricht unverändert und Urheber korrekt ist
- Symmetrischer Schlüssel für Erzeugung der Prüfsumme und Verifikation, wird zusammen mit Nachricht gehashed
- Keine Beweisbarkeit, da MACs nicht zuordenbar
- Wird Schlüssel vorne angehangen bei Hashing, kann Nachricht beliebig verlängert werden
- Wird Schlüssel hinten angehängt: nur unsicher, falls nicht kollisionsresistent
- Allgemeine Lösung sind Keyed-Hash Message Authentication Codes: Unterteilen in innere und äußere Hashfunktion, da Nachricht so nicht beliebig verlängert werden kann.

Diffie-Hellmann

Schlüsselaustausch

- Basiert auf Primitivwurzeln und diskretem Logarithmus
- Öffentliche Ausgangsparameter: Primzahl und Primitivwurzel
- Beide Seiten haben einen privaten und einen öffentlichen Schlüssel
- Daraus wird ein gemeinsamer geheimer Schlüssel gebildet
- Parameterwahl kleiner als bei RSA

Diffie-Hellman Problem

- Diffie Hellman Problem: Bestimmung des geheimen Keys aus den öffentlichen Parametern. Basiert auf den diskretem Logarithmus
- Decision DH-Problem: Teilproblem, 3 Teilnehmer, einer berechnet privaten Schlüssel abhängig von den anderen
- Das Decision DH-Problem ist lösbar, falls Primitivwurzel vorhanden ist

El-Gamal Verschlüsselung

Vorgehensweise:

- Schlüssel wird mit Diffie Hellman erstellt
- Verschlüsselt wird mit gemeinsamen geheimen Schlüssel K
- Verschlüsselung mittels Multiplikation innerhalb einer Restklasse
- Verschicke eigenen öffentlichen Schlüssel und Chiffretext
- Entschlüsselt wird mittels privatem Schlüssel
- ElGamal ist schneller, RSA hat keine Nachrichtenexpansion

Angriffsmethoden:

- Known Plaintext möglich, wenn zweimal der gleiche Private Schlüssel verwendet wird
- Man in the Middle möglich: Vermeidbar durch Signaturen
- Chosen-Chiphertext möglich
- Chosen Plaintext: Möglich, falls Primitivwurzel verwendet wird

Digitale Signaturen

Allgemeines

Was soll gewährleistet werden:

- Integrität des Dokumentes
- Authentizität des Dokumentes
- Urheberschaft nicht Abstreitbar

Allgemeiner Aufbau:

- Schlüsselerzeugungsalgorithmus
- Signieralgorithmus
- Verifikationsalgorithmus

Verschiedene Signaturen

Lamport Diffie Einmal Signatur:

- Benötigt Sicherheitsparameter und Einwegfunktion
- Jeden Schlüssel nur einmal verwenden
- Einzige Sicherheit liegt in Einwegfunktion
- Signatur ist Matrix, welche Nachrichtenbits und Stelle speichert
- Verifikationsschlüssel ist gehashte Signatur
- Verifizierer kennt Hash-Funktion, Nachricht, Verifikationsschlüssel und Signatur

  $\Rightarrow$ Gehashte Signatur $\stackrel{!}{=}$ Verifikationsschlüssel

RSA-Signaturverfahren:

- Schlüsselerzeugung wie bei RSA-Verfahren
  $\Rightarrow$ Public Key (n,e), private Key d
- Signatur: $message^d \ mod \ n$
- Verifizierer kennt öffentlichen Schlüssel, Nachricht und Signatur
  $\Rightarrow$ Verfizieren mittels $m = s^e \ mod \ n$
- Einfache existentielle Fälschung, falls Nachricht unbekannt durch erfinden einer Signatur
- Einfacher Chosen-Message-Angriff durch Multiplikativität von RSA
- Verbesserung durch Signatur von $m \circ m$ statt m oder Signatur von Hashwert

ElGamal-Signaturverfahren:

- Schlüsselerzeugung wie in ElGamal-Verfahren
- Signatur benötigt Hash-Funktion und Zufallszahl
- Verifizierer kennt Nachricht, Signatur, Schlüssel und Hash-Funktion

Digital Signature Algorithm:

- Effizienteres ElGamal mit vorgeschriebenen Parametern
- Wahl eines von 4 Tupeln für Primzahllänge
- Signaturalgorithmus gleicht ElGamal-Signatur

Merkle-Signatur:

- Verwendet binären Hash-Baum, um Gültigkeit vieler Einmal-Verifikationsschlüssel auf Gültigkeit eines öffentlichen Schlüssels zurückzuführen
- Voraussetzungen: Hash-Funktion, Einmal-Signaturverfahren, Zahl die Anzahl verifizierbarer Signaturen festlegt
- Schlüssel: Wähle N Schlüsselpaare des Einmal-Signaturverfahrens und konstruiere binären Merkle-Hash-Baum
- Signatur besteht aus Index des ersten unbenutzten Signierschlüssels, zugehörigem Verifikationsschlüssel der ES, der ES und dem Pfad
- Verifizierer benötigt nur Nachricht und Signatur
- Sicherheit beruht auf sicheren Hash-Funktionen

Identifikation

Passwörter

- Zur Identifikation wird Passwort benötigt
- Passwörter können abhörbar, leicht zu erraten, leicht aufzufinden sein
- Problem wenn Passwort vergessen wird

Challenge-Response

- Identifikation durch Lösen einer Challenge
- Challenge nur mit "Geheimnis"lösbar
- z.b. persönliche Fragen

Zero-Knowledge-Beweis

- Beispiel Dungeon: Gegeben ist ein Dungeon mit zwei Eingängen, die von einer geschlossenen Tür getrennt werden. Bob geht in das Dungeon hinein, danach erst stellt sich Alice vor die Eingänge. Alice sagt Bob aus welchem Eingang er wieder herauskommen soll. Kennt Bob den Schlüssel, kann er immer die Tür öffnen und richtig herauskommen. Kennt Bob den Schlüssel nicht, hat er nur eine 50% Wahrscheinlichkeit richtig herauszukommen
- Eigenschaften:
  - Completness: Antwort mit Kenntnis des Geheimnis immer richtig
  - Soundess: Verifikation durch Schummeln unmöglich
  - Zero-Knowledge: Geheimnis wird nicht preisgegeben

Fiat-Shamir-Identifikationsverfahren

- Setzt Zero-Knowledge korrekt um
- Bob wählt: $r \in \mathbb{Z}_n$, berechnet $x = r^2 \ mod \ n$ und schickt x an Alice
- Alice schickt zufällig 0 oder 1 an Bob
- Bob berechnet $y = r \cdot s^e \ mod \ n$ und verschickt y
- Erfolg: $y^2 = x \cdot v^e \ mod \ n$ gilt

Feige-Fiat-Shamir:

- Wird für Smartcard verwendet
- Basiert auf Fiat-Shamir

Public-Key-Infrastrukturen

Persönliche Sicherheitsumgebung:

- Zum Abspeicher der Keys, z.b. auf Festplatten
- Sicherheit durchs Betriebssystem oder durch Passwörter gegeben

Zertifizierungsstellen:

- Zertifikat bindet Public Key an eine echte Person
- Werden von bestimmten Institutionen (Certification Authority CA) ausgegeben
- Stellt sicher, dass Public Keys NICHT ausgetauscht werden können
- CA ermöglicht die Archivierung von Keys
- Über ein Verzeichnis der CA können Außenstehende Bob's Zertifikat und Key erhalten
- Bob kann seinen Schlüssel ändern
- Zertifikate können ungültig gemacht werden (Passwort vergessen)
- Cross-Zertifikat, falls über verschiedene CAs zertifiziert werden muss
- Pretty Good Privacy für E-Mail