**Kryptographie:**

Ist ursprünglich die Wissenschaft der Verschlüsselung von Informationen.

**Geschichte der Kryptographie:**

Im Altertum:

* Geheimschriften
* Älteste Funde in Mesopotamien und in Ägypten
* Julius Cäsar verwendete, die nach ihm benannte Cäsar-Verschlüsselung
  + Schutz militärischer Geheiminformationen
  + Buchstaben werden um definierten Wert rotiert

Im Mittelalter:

* In Europa hat sich in dieser Zeit nicht weiterentwickelt
* Im Arabischen Raum, hat Philosoph „al Kindi“ auf diesem Gebiet geforscht
  + Pionier auf dem Gebiet Kryptoanalyse
  + Knacken von Verschlüsselungen mit Hilfe statistischen Häufigkeitsanalyse
  + Seine Abhandlung wurde erst 1987 entdeckt

In der Neuzeit:

* In der Renaissance erlebte die Wissenschaft der Kryptographie wieder einen Aufschwung
* Ab Ender des 14.Jh. werden seit dem Altertum unveränderte Verfahren weiterentwickelt
* Beispielsweise die nach Blaise de Vigenere benannte Vigenere Chiffre
  + Klartext wird durch Schlüsselwort verschoben
  + Mit Schlüssellänge = 1 erhält man Cäsar Verschl.

1.Weltkrieg:

* Einsatz vergleichsweise simpler Verfahren
  + per Hand mit Papier und Bleistift errechnet
* Deutscher Code “ADFGX” wurde kurz vor Kriegsende von den Franzosen geknackt
* Deutsche Frühjahrsoffensive scheiterte
* Entschlüsselung des deutschen Nachrichtenverkehrs war ein maßgeblicher Grund
  + Paris konnte nicht erobert werden

**Cäsar Verschlüsselung**

Cäsar Verschlüsselung:

* Verschlüsselt werden lateinische Nachrichten
* Buchstaben werden um fixen Wert verschoben
* Bekannte Variante ist “ROT13”
  + Buchstaben werden um 13 Stellen verschoben

Knacken der Cäsar-Verschlüsselung:

* Für die Verschlüsselung existieren nur 25 mögliche Schlüssel
* Einfaches durchprobieren max. 25 Versuche
* Cäsar Verschlüsselung funktioniert nur, wenn das Verfahren unbekannt wäre
* Moderne Verschlüsselungsverfahren funktionieren weil der Schlüssel geheim ist.

**Kryptoanalyse:**

* Kryptoanalyse ist die Kunst ohne Kenntnis des Schlüssels den Klartext wiederherzustellen
  + zB. durch Häufigkeit der Buchstaben in der natürlichen Sprache

**Enigma:**

* Bisherige manuelle Verschlüsselungsverfahren waren veraltet, umständlich und unsicher
* 1918 wurde in Deutschland die Enigma erfunden
* Ab 1923 auf Messen zum Verkauf angeboten
* Das Deutsche Militär wurde schnell darauf aufmerksam
* Eine erneute kryptographische Katastrophe sollte verhindert werden
* Im 2. Weltkrieg von den Deutschen tausendfach im Einsatz
* Enigma wurde verbessert und weiterentwickelt
* Tagesschlüssel wurde täglich um Mitternacht gewechselt
* Theoretisch 200 Trilliarden Verschlüsselungsmöglichkeiten
* Manuelles Knacken unmöglich

Bestandteile der Enigma:

* Tastatur zur Eingabe
* Leuchten die den verschlüsselten Buchstaben anzeigen
* Walzen, Rotoren und Verkabelung sorgen für maschinelle Verschlüsselung

**Turing-Bombe:**

* Polen arbeiten bereits vor dem Start des 2. Weltkrieg an der Analyse der Enigma
* Vor der Einnahme Polens durch die Deutschen geben sie ihr Wissen an die Briten weiter
* Codeknacker Teams rund um Alan Turing gelingt es während des Kriegs den Code zu knacken
* Turing-Bombe verringert durch Diagonalbrett den Suchraum des Schlüssels dramatisch
* 2014 verfilmt in “The Imagination Game”

**Kryptographie und Gesellschaft:**

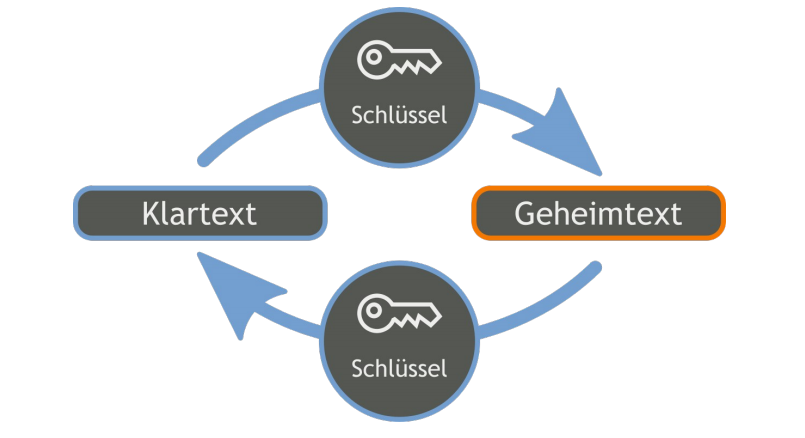
* Kryptographie lange Zeit nur Regierungen und Großunternehmen zugänglich
  + Berechnung nur auf teuren, leistungsstarken Groß Computern möglich.
* 1991 entwickelt amerikanische Physiker Phil Zimmermann RSA-Verschlüsselung für die breite Öffentlichkeit
* Er nennt sie Pretty Good Privacy und veröffentlicht sie
* In den USA gibt es jedoch Exportbeschränkungen für Kryptographie Technologie
* Ein Verfahren wegen illegalen Waffenexports wird gegen Phil Zimmermanns eingeleitet
* Nur Aufgrund öffentlicher Proteste wird dieses eingestellt
* In Frankreich ist es noch bis 1996 verboten kryptographische mit einer Schlüssellänge über 40 Bit einzusetzen
  + >40 bit -> Schlüssel muss beim Staat hinterlegt sein
  + Heute noch sind bestimmte Kryptographie Varianten genehmigungspflichtig
  + Der Regierung ist es ein Dorn im Auge, dass sich ihre Bürger geheim austauschen können

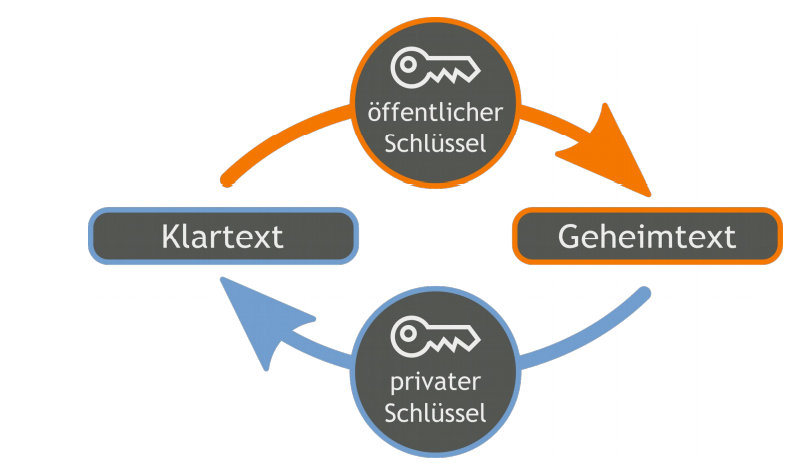
**Kerckhoffs Prinzip:**

* Kerkhoffs Prinzip ist der Grundsatz moderner Kryptographie
* Sicherheit ruht auf geheimhaltung des Schlüssels, nicht dem Verfahren
* Alles andere ist Security by obscurity

**Aktuelle Krypto Standards:**

* 1976 wurde **symmetrische Verschlüsselung** “DES” erfunden
  + 64bit Schlüssel zum ver und entschlüsseln
  + Heutige Weiterentwicklungen 3DES und AES



* 1976 wurde auch das **asymmetrische Verschlüsselungsverfahren** RSA entwickelt
  + Jeweils einen Schlüssel zum Ver- und Entschlüsseln

**Symmetrisch vs. Asymmetrisch**

* Symmetrische Verschlüsselung
  + Gleicher Schlüssel zum Ver- und Entschlüsseln
  + Ermöglicht die Geheimhaltung von Informationen
* Asymmetrische Verschlüsselung
  + 2 zusammengehörige Schlüsselpaare
  + Den privaten Schlüssel kennt nur der Eigentümer
  + Ermöglicht neben Verschlüsselung auch den Nachweis der Urheberschaft
  + Grundlage für die digitale Signatur

**Schutzziele**

* Vertraulichkeit -> Verschlüsselung
  + Daten können nicht von Dritten gelesen werden
  + Möglich mit synchroner + asynchronen Algorithmen
* Integrität -> Prüfsumme/Hash
  + Schutz vor unbefugter Datenmanipulation
* Authentizität -> Signatur
  + Ermöglicht Empfängern zu prüfen, ob die Nachricht wirklich von Person X gesendet wurde
  + Nur mit asymmetrischer Verschlüsselung möglich

**Hashverfahren:**

* Hash ist eindeutiger Fingerprint einer Datei
  + 2 Dokumente mit unterschiedlichen Inhalten dürfen nicht denselben Fingerprint erhalten
* Wird die Datei nur minimal verändert (1Bit) ändert sich auch der Hashwert
* Damit kann geprüft werden, dass die Datei seit Berechnung des Hashes nicht verändert wurde
* Hash-Verfahren sind Einwegfunktionen
* y = f(x) ist mit wenig Aufwand zu berechnen
* Umkehrfunktion x = f-1(y) nicht schwer anwendbar
* Beispiele für Verfahren
  + MD5, SHA-1 -> gelten bereits als unsicher
  + SHA-2 Familie(SHA-256, SHA-512, ...)

**Signatur**

* Alice verschlüsselt Daten mit ihrem privaten Schlüssel und sendet sie Bob
  + Privater Schlüssel nur in ihrem Besitz!
* Empfänger Bob kann mit öffentlichem Schlüssel von Alice entschlüsseln
* Empfänger Bob weiß dadurch, dass die Nachricht von Alice stammen muss
  + Niemand sonst kann Nachrichten erstellen, die mit ihrem öffentl. Schlüssel entschlüsselt werden kann
* Verschlüsselung großer Dateien sehr rechenintensiv
* Um Authentizität und Integrität zu garantieren genügt es, den Hashwert einer Datei zu verschlüsseln
  + Empfänger berechnet Hashwert erhaltener Datei
  + Empfänger entschlüsselt Signatur
  + Beide Hashes müssen übereinstimmen

**Schlüsselaustausch:**

* symmetrisch
  + gemeinsamer Schlüssel
  + Dieser muss geheim bleiben
  + Unverschlüsseltes Senden birgt Risiko
  + Diffie Hellman Schlüsselaustausch
* Asymmetrisch
  + Jeder sendet dem anderen seinen öffentlichen Schlüssel zu
  + Daten werden mit öffentlichem Schlüssel des Empfängers verschlüsselt
  + Empfänger kann Daten mit seinem privaten Schlüssel entschlüsseln

**Public Key Zertifikat**

* Bestätigt Eigentümer eines Public Keys
* Wichtige Bestandteile:
  + Name des Ausstellers
  + Name des Inhabers
  + Der öffentliche Schlüssel
  + Gültigkeitsdauer des Zertifikats
  + Geltungsbereich des Schlüssels
  + Signatur des Ausstellers über alle Informationen

**Zertifikatskette:**

* ist ein Vertrauensmodell
* Am Beginn steht das Vertrauen in die Wurzel Zertifizierungsinstanz
  + z.B. Lets encrypt von Google
* Durch Vertrauen in die Root CA kann ich auch den von ihr abgeleiteten Zertifizierungsstellen vertrauen
* Durch Vertrauen in die Zertifizierungsstellen kann ich auch den ausgestellten Zertifikaten vertrauen

**Public Key Infrastruktur:**

* Ermöglicht öffentliche Schlüssel zu zertifizieren
  + Benutzer stellt Zertifizierungsantrag
  + Nach Prüfung stellt Zertifizierungsstelle ein Zertifikat aus
  + Benutzer kann nun Dokumente signieren
* Ermöglicht Abfrage von Zertifikaten
  + Benutzer können Zertifikate abfragen
  + Mit öffentlichem Schlüssel aus den Zertifikaten können signierte Dokumente geprüft werden

**Authentifizierung vs. Autorisierung**

* Authentifizierung
  + Wer sitzt vor dem Bildschirm
  + Von wem stammt die Nachricht
* Autorisierung
  + Darf Bob die Datei lesen
  + Darf Prozess x darauf zugreifen

Autorisierung braucht immer zuerst Authentifizierung

**Kerberos**

* Standard-Authentifizierungsprotokoll in Active Directory Umgebung
* Zentralisierte Authentifizierung
  + Sonst müsste sich jede Ressource die zugreifenden Clients umständlich selbst authentifizieren
* Funktionsweise:
  + Benutzer nimmt Kontakt mit Authentication Service des Key Distribution Centers auf
    - liegt am Domaincontroller
    - Client weist seine Identität nach (z.B. User/PW)
  + Benutzer erhält Ticket Granting Ticket
  + Für Zugriff auf Ressource wird beim Ticket Granting Server ein Ticket angefordert
    - Benutzer übermittelt dazu TGT in seiner Anfrage
  + Wenn das TGT gültig ist, stellt TGS ein Service Ticket aus für die angefragte Ressource
    - Ticket enthält einen Session Key
  + Benutzer kann sich nun mit Ticket bei der angefragten Ressource authentifizieren
    - Service Ticket ist kryptografisch nachweislich vom Ticket Granting Server signieren
  + Resource autorisiert Anfrage des Benutzers
    - Ist er autorisiert, dann wird der Zugriff genehmigt

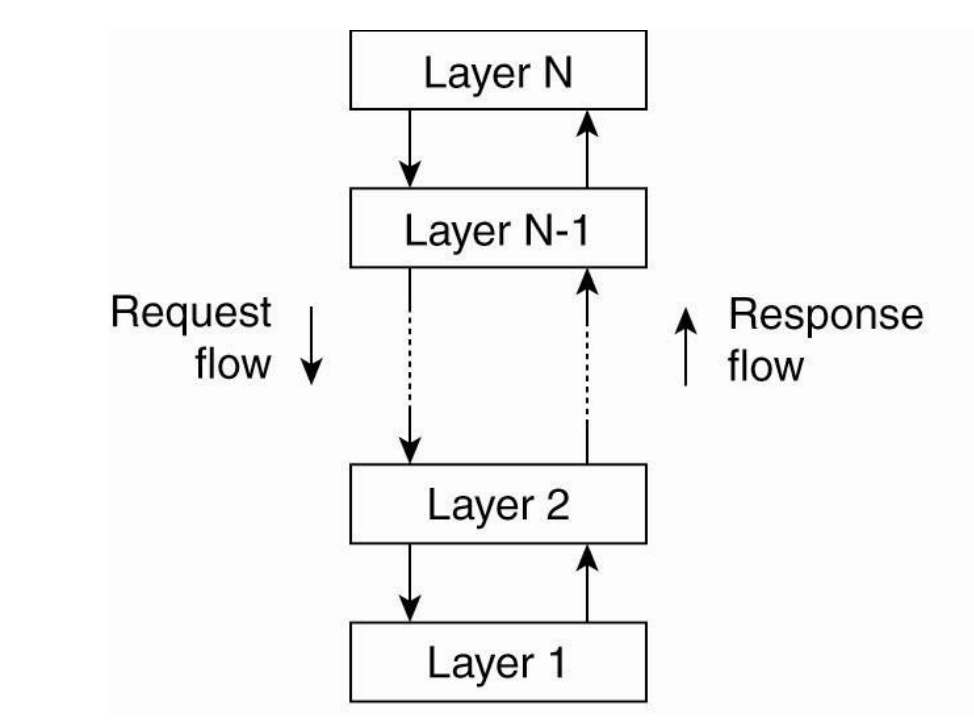
**Architekturen und Middleware**

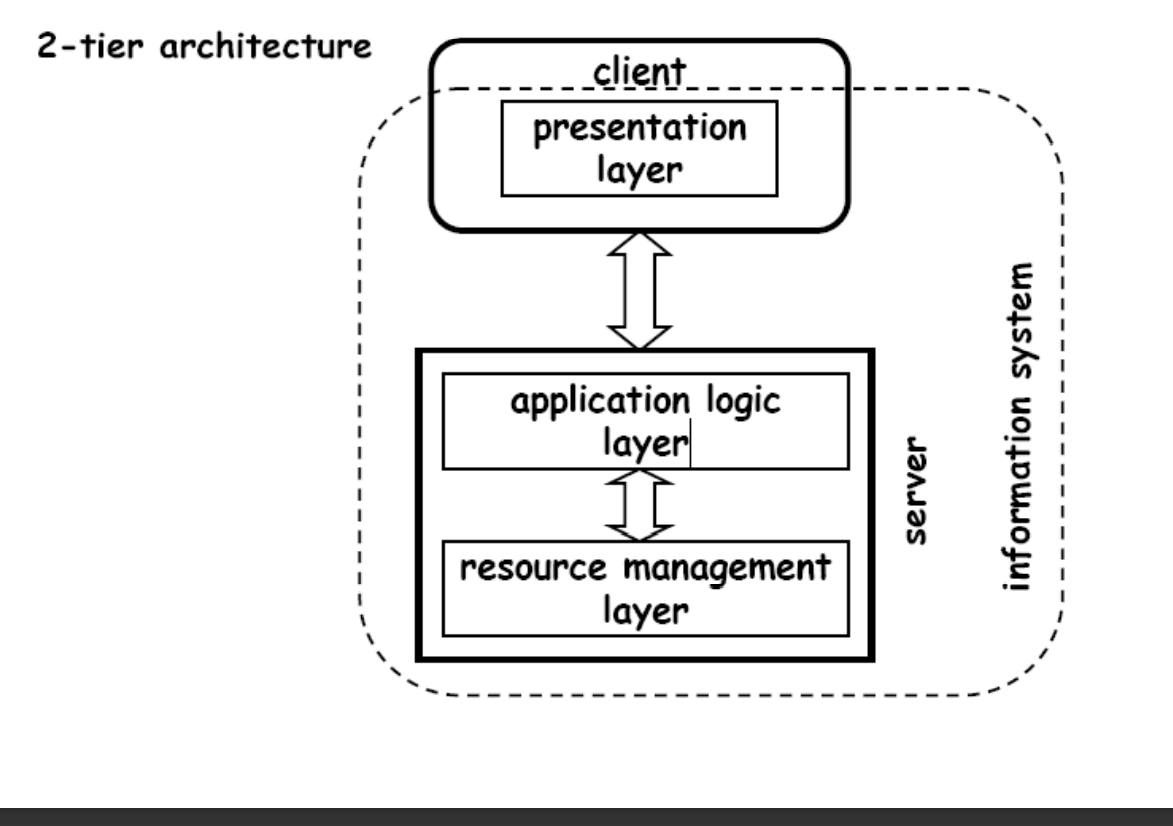
**Bewältigung der Komplexität**:

* Große Systeme haben Millionen Zeilen Code
* Es gilt den Überblick zu bewahren
  + Fehler müssen rasch gefunden werden
  + Neue Features müssen laufend eingearbeitet werden
* Architektur befasst sich mit Aufteilung des Programmcodes in verschiedenen Komponenten und deren Zusammenwirkungen
* Separation of Concerns
  + Einteilung in Client, Server, Service
  + Definierung von Schichten
  + Implementierung aufteilen auf Komponenten
* Datenkapselung
  + Zugriff nur über Schnittstelle
  + Direkter Zugriff auf Implementierung nicht möglich

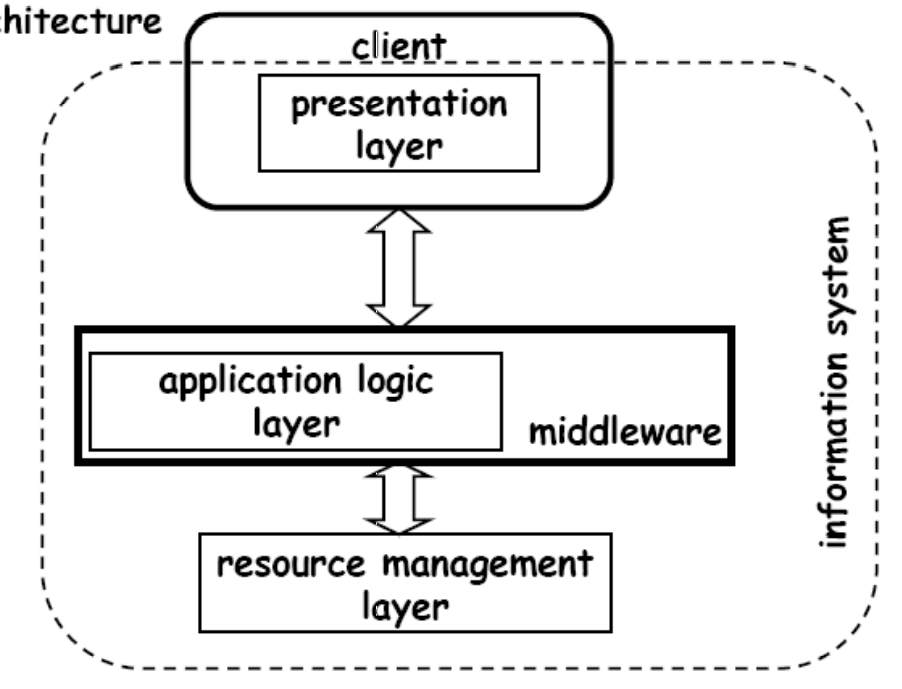
**Client Server Kommunikation**

* Server wartet permanent auf Anfragen
* Client sendet Anfrage an Server und wartet
* Sobald eine Anfrage am Server eintrifft, wird diese abgearbeitet und beantwortet
* Danach wartet der Server auf weitere Anfragen
* Sobald der Client die Nachricht empfängt, kann er weiterarbeiten

Schichtenarchitektur:  


Schichtenarchitektur tier 2:

Schichtenarchitektur tier 3:



**Wie Server Call implementieren?**

* Server lauscht auf TCP Port
* Client packt Parameter in einen String
* Client sendet gewünschten Funktionsnamen und Parameter String
* Server entpackt Request und ruft gewünschte Funktion auf
* Server packt Antwort wieder in String und sendet zurück an Client
* Client entpackt Antwort und verarbeitet sie
* Packen komplexer Objekte in einen String
* Exaktes Protokoll fürs Datenübertagen
* Unterstützung für Nebenläufigkeit
* Fehlerbehandlung
  + Server nicht erreichbar, Verbindung bricht ab
  + Request vom Client fehlerhaft
* Usw.

**Middleware:**

* Typische Geschäftsanwendungen haben:
  + 70-80% Applikations Infrastruktur die sich von Anwendungen zu Anwendung kaum unterscheidet
  + 20-30% Geschäftslogik die spezifisch ist für die jeweilige Anwendung
* Middleware bietet Applikations Infrastruktur für Anwendungen zur Laufzeit

**Middleware Services:**

* Erstellen von Schnittstellen
* Bietet einfache Kommunikationsmöglichkeiten
  + z.B. Remote Procedure Call
* Unterstützung für Nebenläufigkeit
* Erleichtert Skalierbarkeit
* Service für Datenbankzugriff
* Unterstützungen für Replikationen
* Naming Services zum Auffinden der Server
* Security: Authentifizierung, Berechtigungen, …

**Remote Procedure Call:**

* Ein normaler Funktionsaufruf
  + String text = read(“textfile.txt);
  + Ziel des RPC ist es, eine Remote Call aussehen zu lassen wie einen lokalen Aufruf
    - Transparenz!

**Java Remote Method Invocation**

* Java RMI ist die Default Implementierung eines Remote Procedure Calls in Java
* Beispiel - Aufruf der Servermethode “read” auf dem Interface FileReader von Server “Server1”
* try { Registry registry = LocateRegistry.getRegistry(“Server1”); FileReader stub = (FileReader) registry.lookup("FileReader"); String response = stub.read("some\_textfile.txt"); System.out.println("File content: " + response); } catch (Exception e) { System.err.println("Client exception: " + e.toString()); }