Access Control

Standart Access Control Systeme

Preventive

Unauthorisierte Aktivität wird bereits im Vornherein verhindert.

- Zaun, Schloss, Biometrie, Encrpytion, Firewalls, security-awareness training, ...

Detective

Ungewollte/unauthorisierte Aktivität wird entdeckt. Detective Access Control funktioniert nach der Tat und kann Aktivität nur entdecken, wenn/nachdem sie passiert ist.

- Wächter, Überwachungskameras, Intrusion Detection Systeme (IDSs), Bewegungsmelder ...

Corrective

Wird nach der Tat angewendet, um das System zurück in den normalen Zustand zu versetzen. Beispiel: Backup

- System-Reboot, Antivirus-Software (die den Virus entfernt), Backup, ...

Weitere Access Control Systeme

Deterrent Access Control

Ähnlich wie Preventive Access Control. Hier wird der Angreifer vor der Tat abgeschreckt.

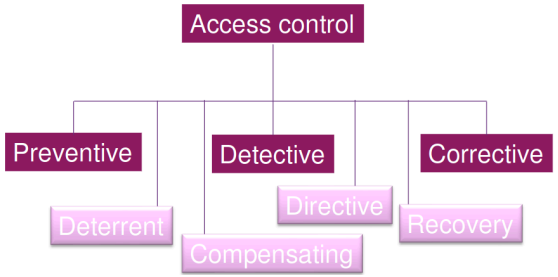

Abbildung 1: Access Control Types

- Policies, Security Awareness Training, Schloss, Zaun, Security Badges, Sicherheitsbeamte, Sicherheitskameras ...

Compensating Access Control

Wenn andere Access Control Systeme nicht ausreichen, wird dieses System eingesetzt. Es unterstützt und verstärkt die anderen Systeme.

- Policy, die besagt, dass alle PII (Personal Identifiable Information) verschlüsselt werden muss. Zum Beispiel wird PII in einer Datenbank gespeichert, die verschlüsselt ist, jedoch werden die Daten in Klartext über das Netzwerk übertragen. Hier kann ein Compensation Control System verwendet werden.

Recovery Access Control

Eine Erweiterung von Corrective Access Control, mit fortgeschritteneren oder komplexeren Möglichkeiten.

- Backups, System Imaging, Server Clustering, Antivirus Software, Database/VM-Shadowing, hot/cold sites, ...

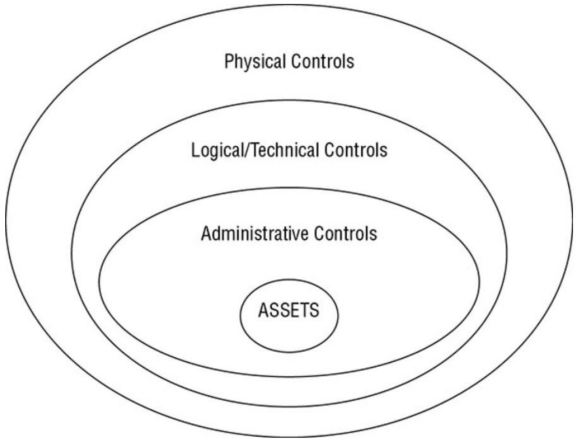

Abbildung 2: Access Control Layers

Directive Access Control

Hier wird dem Subjekt gesagt, was er tun soll, und was nicht. Beispiel: „Bitte geben Sie Ihr Passwort ein.“

- Policies, Excape Route Exit Signs, Systemüberwachung, ...

Zugriff auf Assets konrtollieren

Physical Controls

Physikalische Barrieren innerhalb einer Einrichtung:

- Schloss, Zaun, Türen, Fenster, ...

Technical/logical Controls

Hardware/Software-Mechanismen, die den Zugriff auf Systeme und Daten kontrollieren:

- Passwörter, Biometrie, Firewalls, Intrusion Detection Systems, Routers, ...

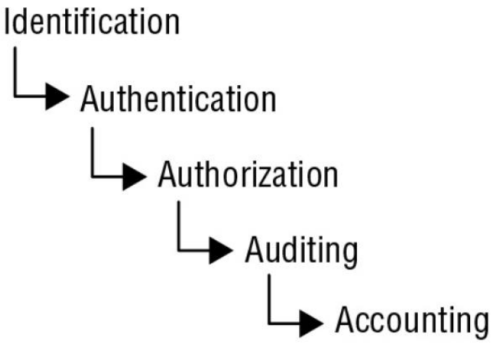

Abbildung 3: Access Control Steps

Administrative Controls

Policiesm, Verfahren und Richtlinien einer Organisation, die den Zugriff auf Systeme und Daten kontrollieren:

- Security Awareness Training, Security Policies, Security Procedures, Security Guidelines, Personalkontrollen, ...

Schritte der Zugriffskontrolle

Identifikation

Unter Identifikaton versteht man den Prozess, bei dem das Subjekt seine Identität „behauptet“ (claimt). Dabei müssen alle Subkekte eindeutige Identitäten haben. Die Identiät eines Subjekts ist normalerweise Public Information.

- Username, Smart Card, Token Device, Phrase, Gesichtserkennung, Fingerabdruck, ...

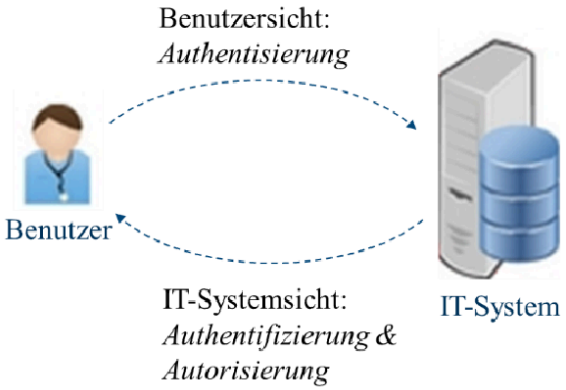

Abbildung 4: Authentisierung und Authorisation

Authentication

Bei der Authentication wird eine weitere Information benötigt, die zur Identität des Subjekts gehört. Dies ist meist ein Passwort. Identifikation und Authentisierung werden oft zusammen als einen einzigen Two-Step Prozess betrachtet.

Authentisierungsinformationen sind privat.

Passwort

Passwörter sind normalerweise statisch und die schwächste form der Authentisierung. Einfache Passwörter sind leicht zu erraten, komplexere Passwörter sind schwer zu merken und werden aufgeschrieben. Passwörter werden normalerweise hashed gespeichert.

Starke Passwörer:

- Keine Identifikationsinformationen (Name, Geburtsdatum, ...)
- Keine Wörter aus dem Wörterbuch
- Keine Namen von Personen, etc. aus Social Network-Verbindungen
- Verwendung von nicht-standart Gross- und Kleinbuchstaben (z.B. „stRongsecuRity“)
- Verwendung von Zahlen und Sonderzeichen (z.B. „stR0ng$ecuR1tee“)

Passphrases

Statt normalen Passwörtern bieten Passphrases eine bessere Alternative. Sie sind einfacher zu merken und Benutzer tendieren dazu, längere Passphrases zu verwenden. (Bsp: „1P@ssedTheCySecEx@m“)

Cognitive Passwords

Cognitive Passwords sind Passwörter, die auf dem Wissen des Benutzers basieren. Zum Beispiel:

- „Wie heisst Ihr erstes Haustier?“
- „Was ist Ihr Lieblingsfilm?“

Am besten erlauben Cognitive Passwords dem User selbst eine Frage zu stellen, die nur er beantworten kann.

Smart Cards

- Smart Cards sind so gross wie eine Kreditkarte und enthalten einen Circuit Chip.
- Smart Cards beinhalten Informationen über den Benutzer, die für die Identifikation und/oder Authentisierung verwendet werden.
- Die meisten Smart Cards haben einen Mikroprozessor und ein oder mehrere Zertifikate. Die Zertifikate werden für assymetric Cryptography ver-

wendet. So können z.B. Daten verschlüsselt werden, oder E-Mails digital signiert werden.
- Smart Cards sind tamper-resistant, d.h. sie sind schwer zu manipulieren.

Token

Ein Token Device/Hardware Token ist ein kleines Gerät, das Passwörter generiert. Ein Authentication Server speichert die Details des Tokens, somit weiss der Server immer, welche Zahl gerade auf dem Token angezeigt wird.

- Synchronous Dynamic Password Tokens:
  - Hardware Tokens, die asynchrone dynamische Passwords generieren. Sie sind Time-based und synchronisiert mit einem Authentication Server.
- Asynchronous Dynamic Password Tokens:
  - Ohne Zeit-Synchronisation. Das Hardware Token generiert Passwörter, die auf einem Algorithmus und einem aufzählendem Counter (Incrementing Counter) basieren.
  - Wenn ein Incrementing Counter verwendet wird, wird ein dynamisches Onetime Password generiert, welches dasselbe bleibt bis zur Authentisierung.

One Time Passwords

- Dynamische Passwörter, die nach einmaliger Verwendung neu generiert werden.
- OTP-Generators sind Token Devices.
- Der PIN kann via eine Applikation auf z.B. dem Smartphone generiert werden.
- TOTP (Time-based One Time Password):
  - Verwendet einen Timestamp, bleibt valid für eine bestimmte Zeit (z.B. 30 Sekunden).
  - Ähnlich wie die synchronous dynamic Passwords, verwendet durch Tokens.
- HOTP (HMAC-based One Time Password):
  - Beinhaltet eine Hash-Funktion, um one time Passwords zu generieren. Es werden HOTP-Werte mit sechs bis acht Ziffern generiert.
  - Ähnlich wie die asynchronous dynamic Passwords, verwendet durch Tokens. HOTP-Werte bleiben valid bis zur Verwendung.

Hijacked People

- Customer, der in die Bank-Website eingeloggt ist
- Der Hacker hat ihre Session übernommen und hat die Credentials gesnifft
- Der Hacker hat nun Access zu den Bank-Konten

Authentication Factors

- Type 1: Something you know
  - Passwort, PIN, Passphrase, Cognitive Passwords
- Type 2: Something you have
  - Smart Card, Token, Memory Card, USB-Drive
- Type 3: Something you are / you do
  - Biometrie, Fingerabdruck, Gesichtserkennung, Stimmerkennung, Iris-Pattern, Retina-Pattern, Palm Topology, Heart/Pulse-Patterns, Keystroke-Patterns, ...
- Somewhere you are
  - GPS, IP-Adresse, MAC-Adresse, ...

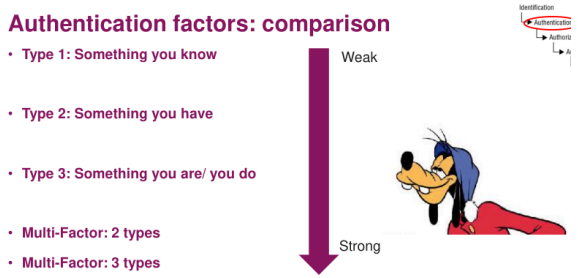

Abbildung 5: Authentication Factors Comparison

- Somewhere you aren't
  - Geofencing, ...
  - Zugriff kann verweigert werden, wenn der Benutzer sich nicht am gewöhnlichen Ort befindet.

Multifactor Authentication

- Zwei oder mehrere Faktoren
- Wenn zweimal derselbe Faktor verwendet wird, ist dies nicht sicherer, als nur ein Faktor (Siehe Abbildung 5).
- Beispiel: Bancomat

Authorisation

- Nachdem das Subjekt sich authentisiert hat, wird ihm Zugriff auf bestimmte Ressourcen gewährt (Authorisierung).
- Identifikation und Authentisierung sind all-or-nothing, während Authorisierung granular ist.

Access Control Models

- Discretional Access Control (DAC)

- Der Eigentümer der Ressource entscheidet, wer Zugriff hat.
- Der Eigentümer kann die Rechte an andere Benutzer delegieren.
- Beispiel: Windows File System

- Role Based Access Control (RBAC)
  - Zugriff wird nicht direkt dem Benutzer zugeordnet
  - User Accounts sind in Rollen plaziert und Administratoren weisen Rollen Zugriffsrechte zu.
  - Rollen sind normalerweise nach Job-Funktionen definiert.
  - Beispiel: Active Directory
- Rule Based Access Control (RuBAC)stylebeinhalten können
  - Flexibler als RuBAC, da verschiedenen Subjekten unterschiedliche Rechte zugewiesen werden können.
  - Beispiel: XACML (eXtensible Access Control Markup Language)
- Mandatory Access Control (MAC)
  - Zugriff wird durch die Sicherheitsrichtlinien des Systems bestimmt.
  - Labels für Subjekte und Objekte.
  - Beispiel: Military Systems. Ein Benutzer hat das Label „Top Secret“, somit darf er auf Dokumente mit demselben Label zugriffen.

Authorisation Mechanisms

- Implicit Deny
  - Basic Principle of Access Control
  - Meistverwendeter Mechanismus
  - Zugriff wird verweigert, wenn keine explizite Erlaubnis erteilt wurde.
- Constrained Interface
  - UI wird so gestaltet, dass nur erlaubte Aktionen sichtbar sind.
  - Benutzer mit voller Berechtigung sehen alle Optionen.
- Access Control Matrix (ACM)
  - Tabelle, welche Subjekte, Objekte und Zugriffsrechte beinhaltet.
  - Wenn ein Subjekt eine Aktion auf ein Objekt ausführen will, wird die Matrix überprüft.
  - ACLs (Access Control Lists) sind Object Focused, sie identifizieren die Zugriffsrechte zu Subjekten für irgendein spezifisches Objekt.
- Capability Tables
  - Subjekt-Focused, sie identifizieren Objekte, auf welche die Subjekte zugreifen können.
- Content-Dependent Access Control
  - Zugriff wird basierend auf dem Inhalt des Objekts verweigert.
  - Beispiel: Database-View. Eine View ruft spezifische Columns fvon einer oder mehreren Tabellen (Virtual Table) ab.
  - Beispiel: Eine Benutzertabelle enthält Name, E-Mail, Kreditkartennummer. Ein Benutzer hat nur Zugriff auf Name und E-Mail.
- Need to know
  - Zugriff wird nur gewährt, wenn der Benutzer für seine Work-Tasks und Job-Functions das Wissen benötigt.
  - Beispiel: Ein Benutzer in der Buchhaltung benötigt keine Zugriff auf die Kundendatenbank
- Least Privilege
  - Benutzer erhalten nur die Rechte, die sie für ihre Arbeit benötigen.
  - Wichtig, dass alle Benutzer Well-Defined Job-Beschreibungen haben, welche das Personal versteht.
  - Für Data: Create, Read, Update, Delete (CRUD)
  - Beispiel: Ein Benutzer in der Buchhaltung benötigt nur Lesezugriff auf die Kundendatenbank.
- Separation of Duties and Responsibilities
  - Verhindert, dass ein Benutzer alleine eine kritische Aufgabe ausführen kann.
  - Beispiel: Ein Benutzer kann eine Transaktion erstellen, jedoch nicht genehmigen.

Auditing und Accountability

Auditing

- Überwachung von Aktivitäten eines Subjekts
  - Somit kann ein Subjekt bei einem Missbrauch zur Rechenschaft gezogen werden.

Accountability

- Wichtig, dass die Identität eines Subjekts bewiesen werden kann.
- Verantwortlichkeit
  - Ein Subjekt ist für seine Aktivitäten verantwortlich.

- Ein Subjekt kann zur Rechenschaft gezogen werden, wenn es gegen die Richtlinien verstösst.

Common access control attacks

- Access Aggregation Attacks (Passive Attack)
  - Mehrere nicht-sensitive Informationen werden zusammengeführt, um sensible Informationen zu erhalten.
- Password Attacks (Brute-Force Attack)
  - Online: Attacks gegen Onlinekonten
  - Offline: Stehlen einer Account-Datenbank und Cracken der Passwörter
- Dictionary Attacks (Brute-Force Attack)
  - Verwendung von Wörterbüchern, um Passwörter zu erraten
  - Scannen oft für one-upped Passwörter, wie *Password*, *password1*, *passXword*, ...
- Birthday Attacks
  - Kollisionsangriff auf Hash-Funktionen (Siehe Abbildung 6)
  - Geburtstagsparadoxon: Wahrscheinlichkeit, dass zwei Personen am selben Tag Geburtstag haben
  - Kann durch Hashing-Algorithmen mit genügend Bits oder durch Salting verringert werden.
    - MD5 ist nicht Collision-Free
    - SHA-3 kann bis zu 512 Bits verwenden und wird (aktuell) als sicher gegen Birthday-Attacks angesehen.
- Rainbow Table Attacks
  - Ein Passwort zu erraten, es dann zu hashen und dann vergleichen braucht eine lange Zeit.
  - Rainbow Tables enthalten bereits die vorberechneten Hashes.
  - Es wird Zeit gespart
- Sniffer Attacks
  - Ein Sniffer (Packet-/Protocol-Analyzer) ist eine Applikation, die Network-Traffic überwacht.
  - Schutzmassnahmen:
    - Encryption von Daten
    - OTPs verwenden (Können nach dem „Sniffen“ nicht vom Attacker wiederverwendet werden)
    - Physical Security: Zugriff auf Routers und Switches physikalisch verhindern
- Spoofing (Masquerading) Attacks
  - Sich als jemand/etwas Anderes ausgeben
    - IP Spoofing, valide Source IP wird mit einer falschen ersetzt.
    - Idenität verschleiern oder sich als trusted System ausgeben.
    - E-Mail Spoofing, Phonenumber-Spoofing
- Social Engineering Attacks
  - Manchmal ist es am Einfachsten, an ein Passwort zu kommen, indem man danach fragt.
  - Attacker versucht, Vertrauen gewinnen
  - Risiko kann durch Trainings verringert werden
- Shoulder Surfing
  - Social Engineer schaut jemandem über die Schulter
  - Screen Filters können dies verhindern
- Phishing
  - Art von Social Engineering
  - Sensitive Informationen weitergeben via malicious Attachment oder Link
  - Werden als Spam versendet, in der Hoffnung, dass jemand trotzdem antwortet
  - Simple Fishing: Es wird direkt nach Passwort, Username, etc. gefragt
  - From-Adresse oft spoofed, Reply-Adresse ist Account des Attackers
  - Sophisticated Phishing:
    - Link sieht korrekt aus
    - Infiziertes File als Attachment
    - Drive-By Download: Lädt Dateien herunter ohne Wissen des Users, Sicherheitslücken des Browsers, Extensions
    - Oft wird Social Media verwendet, um sich über Freundschaften und Verhältnisse der Opfer zu informieren.
  - Spear Phishing:
    - Phishing auf spezifische Personen gezielt
  - Whaling: Ziel auf High-Level Executives, wie CEOs
  - Vishing: Instant Messaging (IM), VoIP anstelle E-Mails

Hash Function

Schutzmechanismen

- Layering (defense in depth)
  - Mehrere Kontrollen seriell

Abbildung 6: Hash Function

  - So kann eine oder mehrere Kontrollen immernoch fehlschlagen, ohne dass die Attacke unbedingt gelingt
- Abstraction
  - Classifying Objects, Rollenzuweisung an Subjekte
  - Zuweisung von Security Controls an eine Gruppe von Objekten
- Data Hiding
  - Speicherung von Daten in einer logischen Speichereinheit, auf das ein unbefugtes Subjekt keinen Zugang besitzt
- Security through Obscurity
  - Ein Subjekt wird nicht über ein Objekt informiert
  - Hoffen, dass das Subjekt das Objekt nicht entdeckt
  - Keine Art von „Schutz“
- Encryption
  - Das Verschleiern der Bedeutung oder Absicht einer Nachricht von Unbefugen
  - Schleche Encryption entspricht Security through Obscurity

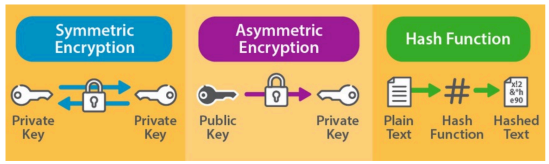

Abbildung 7: Drei Typen der Kryptographie

Symmetric Encryption and Key Exchange

Die drei Typen der Kryptographie

Konzepte

Zunächst liegt eine Nachricht in Plaintext vor. Diese kann der Sender mithilfe eines kryptografischen Algorithmus in Ciphertext umwandeln (Encryption). Der Empfänger kann den Ciphertext durch Decryption wieder in Plaintext umwandeln.

- Message/Plaintext
- Ciphertext
- Cipher
  - Der Encryption-Algorithmus wird auch als Cipher bezeichnet.
- Cryptographic Key
  - Eine (oft sehr grosse) Binärzahl

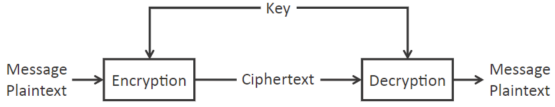

Abbildung 8: Encryption und Decryption

  - Jeder Algorithmus hat einen spezifischen Keyspace (die Menge aller möglichen Schlüssel, von der Anzahl Bits abhängig)
  - Keyspace in der Range von 0 bis $2^n - 1$, n ist die Anzahl der Bits
  - Private Keys müssen geschützt werden
- One Way Functions
  - Mathematische Funktion, die leicht zu berechnen ist, jedoch schwierig oder unmöglich, umzukehren
  - Es wurde nie bewiesen, dass eine wirkliche One-Way-Function existiert
  - Cryptographers verlassen sich auf Funktionen, die sie als One-Way erwarten
  - Könnten in der Zukunft gebrochen werden
- Reversability
  - Sehr wichtig in der Kryptographie, eine Encyrption muss reversibel sein (Decryption)
- Nonce[1]
  - Einmaliger Wert, der nur einmal verwendet wird
  - Versichert, dass derselbe Key nicht mehrmals verwendet wird
  - Oft in Kombination mit einem Counter verwendet
  - Nonce ist public, Key ist private
  - Verhindert Replay-Attacken
- Initialization Vector (IV)
  - Zufälliger Bit-String
    - Wird oft in Block-Ciphers verwendet
    - Gleiche Grösse wie die Block-Size, XOR ($\oplus$) mit dem Plaintext
    - Werden dafür verwendet, um denselben Plaintext mit demselben Key in unterschiedlichen Ciphertext zu verschlüsseln
- Confusion
  - Relationship zwischen Plaintext und Ciphertext so komplex, dass der Attacker den Ciphertext nicht einfach analysieren kann
  - Input ↔ Output-Mapping ist komplex
  - Substitution von Bytes
  - Beispiel: Enigma, Caesar Cipher: Nur Confusion, keine Diffusion
- Diffusion
  - Eine Änderung im Plaintext sollte sich auf den gesamten Ciphertext auswirken
  - Kleine Änderung resultiert in grosser Änderung im Ciphertext
  - Permutation von Bytes

Kerckhoffs' Prinzip

- Security through Obscurity
  - Die Sicherheit eines Systems basiert auf der Geheimhaltung der Geheimnisse
- Die Sicherheit eines Systems ist nicht von der Geheimhaltung des Algorithmus, sondern von der Geheimhaltung des Schlüssels abhängig
- Ein kryptographisches System sollte sicher sein, auch wenn alles ausser dem Schlüssel über das System bekannt ist
  - Algorithmen können public sein und von jedem getestet werden

[1] Abkürzung für 'Number used once'.

- *„The Enemy knows the system“*
- Public Exposure kann Weaknesses schneller aufdecken, schnellere Adoption guter Algorithmen
- Viele Kryptographen passen sich diesem Prinzip an, aber nicht alle sind derselben Meinung
- Einige Kryptographen glauben, dass die Sicherheit eines Systems erhöht wird, wenn der Algorithmus auch geheim gehalten wird

Substitution Permutation Network (SPN)

Unter einem SPN versteht man einen Algorithmus, der wiederholend Substitution und Permutation anwendet. - Substitution: Bytes durch andere ersetzen

- Permutation: Bytes swappen
- Substitutionen und Permutationen werden zusammengefasst in einer Runde (Round)
- Runden werden viele Male wiederholt

Caesar-Cipher

- A wird zu D, B wird zu E, X wird zu A, Y wird zu B, ...
- ROT3 (Rotate by 3)
- Monoalphabetic Substitution Cipher

Die Bedeutung von XOR

Eine Bitfolge B kann bestimmen, wie sich de Plaintext A verhält: Ist an einer Position eine 1, wird das Gegenteil vom Bit von A genommen. Ist an einer Position eine 0, bleibt das Bit von A unverändert. B

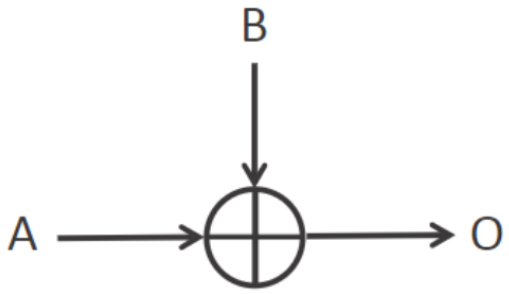

Abbildung 9: XOR

ist ein Key, welcher entscheidet ob A verändert wird oder nicht.

A	B	O
0	0	0
0	1	1
1	0	1
1	1	0

Doppeltes Anwenden von XOR kehrt die Operation um: $A \oplus B \oplus A = B$, $A \oplus B \oplus B = A$.

One Time Pad (OTP)

- Jeder Key ist so lang wie der Plaintext
- XOR jedes Bit des Plaintexts mit dem Key
- Perfect Secrecy:
  - Wenn man den Key wegnimmt, kann die Nachricht nicht entschlüsselt werden, da kein statisches Mapping von Input zu Output vorhanden ist.
  - Diese Cipher kann nicht gebrochen werden.
- Aber:
  - OTP ist nicht praktisch
  - Ein 1GB File benötigt einen 1GB Key
  - Keys können nicht wiederverwendet werden

Symmetric Cryptography

- Shared Secret Key
- Shared Key wird für Encryption und Decryption verwendet
- Mit grossen Keys kann eine starke Verschlüsselung erreicht werden
- Nur Confidentiality
- Key distribution
  - Die Parteien brauchen eine sichere Methode, um den geheimen Schlüssel auszutauschen
- Implementiert keine *nonrepudiation*
  - Jede Partei kann Nachrichten verschlüsseln und entschlüsseln, so kann nicht bewiesen werden, von wo die Nachricht kommt
- Keine Integrität der Nachricht
- Sehr schnell (1000 bis 10000 mal schneller als asymmetrische Verschlüsselung)
  - Viele Prozessoren haben ein AES Instruction Set eingebaut
- Alternative zu AES: Chacha20

Stream Ciphers

Es kann ein One-Time Pad approximiert werden, mit einem undendlichen pseudo-random Keystream. Stream-Ciphers funktionieren auf Nachrichten mit beliebiger Länge.

Vorteile:

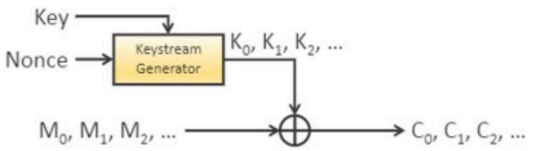

Abbildung 10: Stream Cipher

- Encryption von langen, kontinuierlichen Streams, auch möglich für unbekannte Längen
- Sehr schnell, mit kleinem Memory-Footprint, ideal für low-power Geräte
- Kann zu einer beliebigen Position im Stream springen

Nachteile:

- Keystream muss statistisch zufällig sein
- Key + Nonce dürfen nicht wiederverwendet werden
- Streamciphers schützen den Ciphertext nicht vor Modifikation (keine garantierte Integrität)

Block Ciphers

Block Ciphers nehmen einen Input von einer fixen Länge und geben einen Output von derselben Länge. Dabei findet Confusion und Diffusion statt. Sie sind oft SPNs.

Der Advanced Encryption Standard (AES) ist ein Block Cipher, der 128-Bit Blöcke verwendet und ein SP-Network. Es ist der meistverbreitete Block Cipher. Es gibt aber auch andere, wie Feistel Ciphers. Man kann von einem Mapping eine Basic Permutation Box zeichnen wie in Abbildung 11.

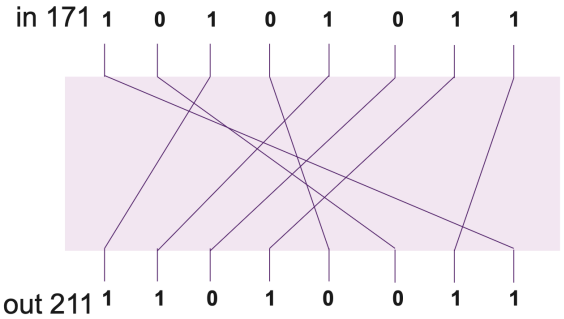

Abbildung 11: Basic Permutation Box

Advanced Encryption Standard (AES)

AES ist ein Standart basiert auf dem Rijndael-Algorithmus. Er hat 2002 den DES als Standart abgelöst.

- SPN mit einer Block-Size von 128-Bit
- Key-Size von 128, 192 oder 256 Bit
- 10, 12 oder 14 Runden
- Jede Runde besteht aus SubBytes, ShiftRows, MixColumns und KeyAddition

XOR

Zunächst wird der Key mit dem Plaintext per XOR transformiert.

SubBytes

Die einzelnen Bytes werden per Lookup Table ersetzt. Die Lookup Table ist eine 16x16 Matrix, wobei die Reihen und Spalten jeweils die HEX-Zahlen von 0 bis F darstellen. Dabei gibt es keinen Fixed-Point, d.h. keine Zahlen bleiben gleich.

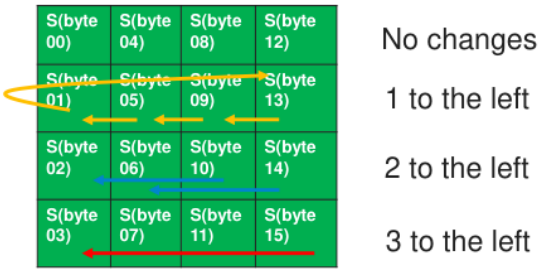

Abbildung 12: Shift Rows

ShiftRows

In diesem Schritt wird:

- In der zweiten Zeile alle Bytes jeweils um 1 nach links verschoben
- In der dritten Zeile alle Bytes jeweils um 2 nach links verschoben
- In der vierten Zeile alle Bytes jeweils um 3 nach links verschoben
- Die erste Zeile bleibt gleich

MixColumns

Im vierten Schritt werden die Werte mittels „Matrixmultiplikation“ berechnet. Die Column für jeden Wert wirt mithilfe einer Lookup Table ermittelt (Siehe Abbildung 13). Möchte man den Wert in der 2. Reihe und 3. Spalte berechnen, multipliziert den ersten Wert der 3. Spalte der originalen Matrix mit dem 1. Wert der 2. Reihe der Lookup-Matrix, usw. Die Produkte werden dann mit XOR verküpft (Siehe Abbildung 14).

Mode of operations

2	3	1	1
1	2	3	1
1	1	2	3
3	1	1	2

Abbildung 13: MixColumns Lookup Table

				c0	
				c1	
				c2	
				c3	
2	3	1	1	2c0+3c1+c2+c3	
1	2	3	1	c0+2c1+3c2+c3	
1	1	2	3	c0+c1+2c2+3c3	
3	1	1	2	3c0+c1+c2+2c3	

Abbildung 14: MixColumns

- Nachrichten mit genau 128-Bits sind unwahrscheinlich
- Mode of operation = Kombination mehrerer Block-Encryption-Instanzen in zu einem nutzbaren Protokoll
- Es gibt drei mode of operations:
  - Electronic Code Book (ECB)
  - Cipher Block Chaining (CBC)
  - Counter Mode (CTR)

Electronic Code Book (ECB)
- Seriell Block nach Block verschlüsseln
- Schwach für reduntante Daten: Gibt mit relativ grosser Wahrscheinlichkeit dasselbe Pattern (ECB-Pinguin, siehe Abbildung 15)
- ECB wird nicht empfohlen

Cipher Block Chaining
- Der Output jedes Cipher-Blocks XOR dem nächsten Input
- Nicht parallelisierbar
- Besser als ECB

Counter Mode (CTR)
- Einen Counter (Nonce) verschlüsseln
- Kann parallelisiert werden
- Es wird nicht die Nachricht selbst verschlüsselt, sondern die Nonce und wenden XOR auf die Nachricht an
- Heutzutage Standart

Diffie-Hellmann
- Geteiltes Geheimnis über einen unsicheren Kanal

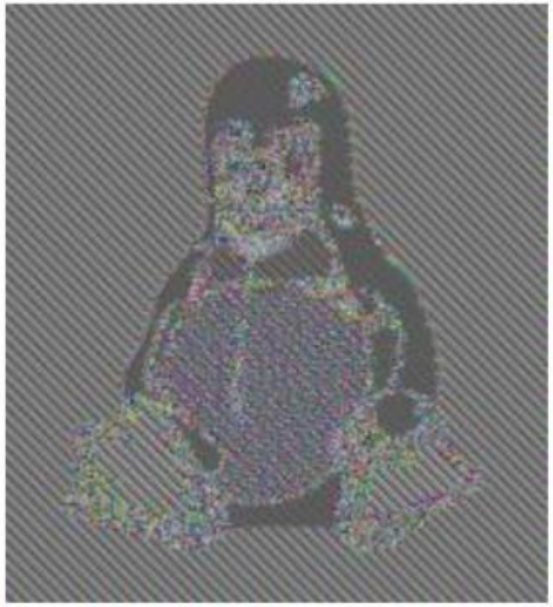

Abbildung 15: ECB Pinguin

- Jeder Kommunikationshandshake im Internet verwendet Diffie Hellmann (z.B. TLS)
- Kein direkter Schlüsselaustausch, nur Teile eines Schlüssels

Diskreter Logarithmus

$$a^b \bmod n = c$$
$$b = \log_{a,n}(c)$$

$$3^{29} \bmod 17 = 12 \text{ einfach}$$

Beispiel: $3^x \bmod 17 = 12$ schwierig, was ist x?

Diskrete Logarithmen sind sehr schwierig zu berechnen.

Primitive Root eine Primzahl

Primzahl p, g ist primitive Root, wenn $g \not\equiv g^2 \not\equiv g^3, ..., g^{p-1} \bmod p$

Key Exchange

1. Alice und Bob eignen sich auf eine grosse Primzahl p (Normalerweise 2048 oder 4096 Bits) und eine Primzahl g, die eine Primitve Root von p ist.
2. A und B wählen zufällige Zahlen a und b als ihre private Keys (Zahlen zwischen 1 und p).
3. A und B berechnen je einen public Key:
   - A: $g^a \bmod p$
   - B: $g^b \bmod p$
4. Sie tauschen diese public Keys über daz Netzwerk miteinander aus
5. Sie berechnen den shared secret Key:
   - A: $\left(g^b\right)^a \bmod p = g^{ab} \bmod p$
   - B: $\left(g^a\right)^b \bmod p = g^{ab} \bmod p$
6. Dieser secret Key wird auch *pre-master secret* genannt, er wird für das Erstellen des Session Keys verwendet
   - Der secret Key ist oft sehr gross (2048 Bit), deshalb nicht optimal für einen Session Key.
   - Master Secret wird durch *hashed-key derivation function (HKDF)* generiert, z.B. **SHA-256**.

Beispiel

A und B einigen sich auf $g = 3$ und $p = 29$

- A wählt $a = 23$, $3^{23} \bmod 29 = 8$
- B wählt $b = 12$, $3^{12} \bmod 29 = 16$
- A berechnet $\left(g^b\right)^a \bmod 29 = 16^{23} \bmod 29 = 24$
- B berechnet $\left(g^a\right)^b \bmod 29 = 8^{12} \bmod 29 = 24$

Der shared secret Key ist somit 24.

Nur g, p, $g^a \bmod p$ und $g^b \bmod p$ wurden öffentlich übertragen.

Um den private Key zu knacken, müsste man folgendes lösen:

$a = \log_{g,p}(gb)$

$b = \log_{g,p}(ga)$

Elliptic Curve Cryptography (ECC)

Elliptic Curves sind ein drop-in Replacement für die Mathematik des Diffie-Hellmann-Algorithmus. Im Browser wird dies als ECDHE (Elliptic Curve Diffie-Hellmann Ephemeral-Verfahren) bezeichnet. Es handelt sich um eine zweidimensionale Kurve:

$$y^2 = x^3 + ax + b$$

- Der private Key ist eine Zahl
- Der public Key wird durch zwei Zahlen (x, y) komposiert
- Es handelt sich wiederum um ein diskretes Logarithmus-Problem (ECDLP), es ist aber ein wenig schwieriger als das herkömmliche Verfahren.

Für dieselbe Key-Länge sind Elliptic Curves viel stärker (Siehe Abbildung 17).

Ephemeral Mode

- Neuer Key-Exchange für jede Session
  - Perfect Forward Secrecy
- Jedes Mal ein neuer Key

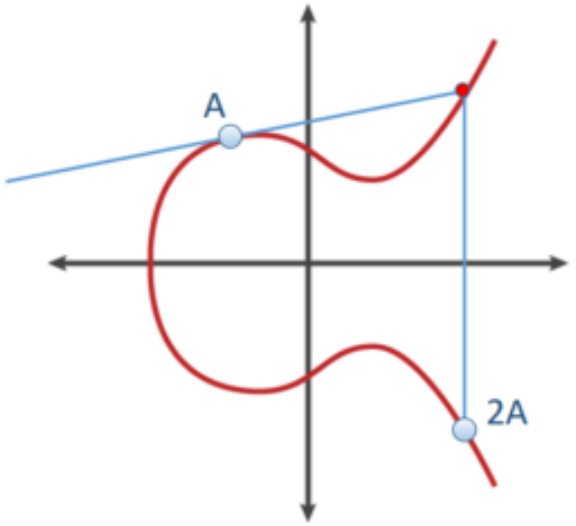

Abbildung 16: Elliptic Curve

- Neuer Diffie-Hellmann-Algorithmus nicht für jede Nachricht, aber sehr oft:
  - Seite neu laden
- Wenn Keys gebrochen werden, hält dies nicht für lange, neue Keys werden bald wieder generiert.
  - Self-Healing Property
- Kein Handshake mit denselben Keys für Monate.

Symmetric	Diffie-Hellman and RSA	Elliptic Curve
56	512	112
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	512

Abbildung 17: ECC vs. traditionelles Verfahren

Asymmetric Cryptography

Drei Typen der Kryptographie:

- Symmetrische Versschlüsselung
  - Ein private Key für das Verschlüsseln und Entschlüsseln
- Asymmetrische Verschlüsselung
  - Zwei Keys: public/private
  - Ein Key verschlüsselt und der andere Key entschlüsselt
- Hash-Funktion
  - Plaintext → Hashed Text

RSA

RSA ist die meistverwendete Methode für public Cryptography. Es wird unter anderem verwendet für:

- Verschlüsselung mithilfe des public Keys, nur der Besitzer des privaten Schlüssels kann die Nachricht entschlüsseln
- Signaturen: Die Message wird mit dem private Key verschlüsselt, mit dem public Key entschlüsselt
  - Server möchte beweisen, dass es sich um ihn handelt → Authentication

→ Encryption und Authentication

Wie man in Abbildung 18 sieht, müsste Alice bei der symmetrischen Verschlüsselung für jeden Kommunikationspartner einen neuen Key generieren. Es skaliert nicht.

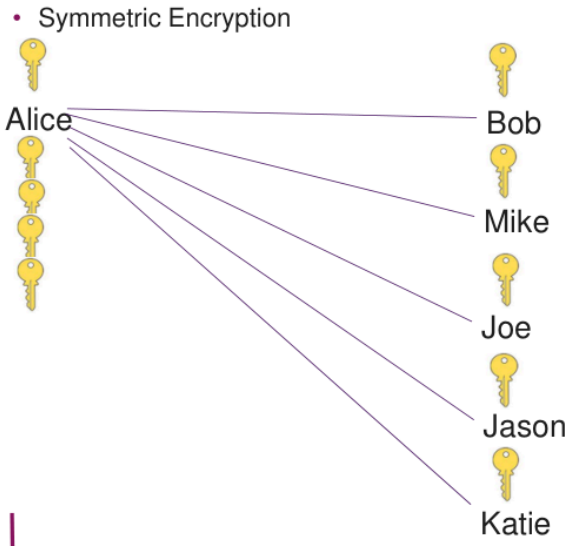

Abbildung 18: Symmetrische Verschlüsselung

Dies kann sie durch asymmetrische Verschlüsselung lösen. Sie generiert einen private Key für sich selbst und übergibt den public Key an alle Anderen. So können alle verschlüsselte Nachrichten an Alice senden und Alice kann Nachrichten mit ihrem private Key signieren.

Public Key

- e: sehr kleine Zahl (normalerweise 2 oder 3)
  - Wird für die Encryption verwendet
- n: sehr grosse Semi-Prime Zahl (Multiplikation von zwei grossen Primzahlen p, q)

RSA basiert darauf, dass die Primfaktorzerlegung von n sehr schwierig ist.

Private Key

- d
  - Wird für die Decryption verwendet

Mathematik

- Cyphertext: $c = m^e \bmod n$
- Message: $m = c^d \bmod n$

$\rightarrow m^{ed} \bmod n = m$

$d = \frac{k \cdot \Phi(n)+1}{e}, k \in \mathbb{Z}$

Prozess

- Wähle e, generiere n zufällig, berechne d (Euklidischer Algorithmus)
- Eher aufwändig, sollte selten durchgeführt werden
- e ist fast immer 3 oder 65537, n ist 4096 bits

Verschlüsselung mit RSA

RSA ist sehr schwach für kurze Nachrichten:

- Padding
- Optimal Asymmetric Encryption Padding (OAEP)
- Pseudo-Random Padding, fügt einen IV (Initialization Vector) dazu und hashed
- Der Empfänger muss nach der Entschlüsselung dasselbe Padding beachten
- Verschlüsselung mit RSA ist selten
  - TLS hat früher RSA verwendet, aber verwendet heutzutage DH
  - Signaturen geschehen mit RSA
  - RSA ist 1000x langsamer als symmetrische Kryptographie

Signieren mit RSA

- Verschlüsselung mit dem privaten Key
- Integritätsverifikation der Nachricht
  - Nachricht wird gehashed
  - Hashing für Kürzen der Nachricht
- Hash wird signiert und zusammen mit der Nachricht versendet
- Empfänger hasht die Nachricht, entschlüsselt die Signatur und checkt ob die Hashes dieselben sind
- Hashes können **nicht** entschlüsselt werden
- Wird oft als Challenge gesendet
  - Der Server soll seine Idenität bestätigen
  - Der Client schickt eine Nachricht, die signiert werden soll
  - Server signiert diese und schickt sie zurück mit dem public Key
  - Challenge-Response-Verfahren
  - Wichtiger Teil von TLS

Digital Signature Algorithm (DSA)

- RSA benötigt immer grössere Keys → bald zu langsam
- DSA als Alternative
- EC DSA (oder DSS) ist viel schneller als RSA, wird bald Standart
  - Elliptic Curve Digital Signature Algorithm/Digital Signature Standard
- DSA kann nur für Signaturen verwendet werden, nicht für Verschlüsselung
- Wie RSA, aber Mathematik wie DH.
  - Elliptic Curves

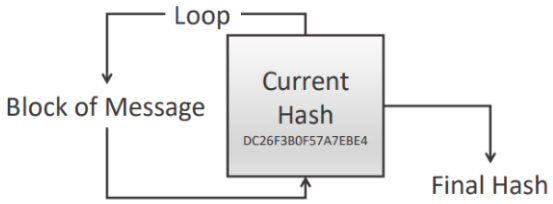

Abbildung 19: Hash

Hash Functions

Eine Hash-Funktion nimmt eine Nachricht irgendeiner Länge und erstellt von dieser einen pseudorandom Hash einer festen Länge. So wird eine 128-Bit-Hash-Funktion immer 128-Bits produzieren. Sie ist eine one-way Function, d.h. sie sind **irreversibel**. Dabei wird immer ein Block der Message genommen und gehashed, iterativ. Wenn die Nachricht fertig ist, ist der Hash fertig (Siehe Abbildung 19).

Starke Hashing Functions:

- Generieren Output, der nicht unterscheidbar ist von random Noise
  - Output soll nicht aussehen, als würde er auf dem Input basieren
- Bitänderungen müssen den gesamten Output verändern (Diffusion)
  - Avalanche Effect

Hash-Kollision

Man versteht unter einer Hash-Kollision zwei verschiedene Inputs, die denselben Hash produzieren. MD5 ist komplett broken und man kann beliebige Collisions erzwingen.

Name	Output Length	Rounds	Security	
MD5	128-bit	4	Broken	Cryptography
SHA-1	160	80	Recent collision found, not trusted	
SHA-2	224, 256, 384, 512	64, 80	Some theories, currently considered safe	
SHA-3 (Keccak)	224, 256, 384, 512 SHAKE128, 256	24	Secure, but relatively untested, strength variable	
PBKDF2	Varies		Iterates another hash function	Password Storage
bcrypt	184-bit		GPUs struggle to crack it	

Anforderungen an eine Hash-Funktion

- Schnell (aber sicher, d.h. nicht zu schnell)
- Diffusion
- Irreversibel
- Keine Collisions
  - Wichtig, da Hashes dazu verwendet werden, um zu verifizieren, dass Daten nicht verändert wurden.
  - https://shattered.io

Übersicht

SHA-X

- SHA-1 ist bereits viel besser als MD5. Nicht komplett broken aber viel schwächer mittlerweile
- SHA-2 256 bits und 512 bits sind heutezutage Standart
  - SHA-2 hat dieselben Funktionalitäten wie SHA-1, aber der Output ist länger, momentan keine Probleme
- SHA-3 als Backup für SHA-2
  - Komplett andere Funktion (Keccak Algorithmus)

Passwörter

Für Passwörter sind SHA-X-Algorithmen nicht gut, sie sind zu schnell. SHA wird für eine schnelle Zusammenfassung der Daten verwendet und sind verletzlich gegen Brute-Force-Attacken. Somit werden die Hashes mehrfach wiederholt:

- PBKDF2 (Password-Based Key Derivation Function 2) verwendet einen ähnlichen Algorithmus wie SHA-2 aber wendet diesen 5000-mal an
  - Exklusiv für Logins und Passwörter
  - Komplett useless für andere Hash-Zwecke
  - Anzahl Iterationen kann angepasst werden
- Bcrypt ist eine Alternative zu PBKDF2
  - Komplett andere Funktion, basierend auf der Cipher **blowfish**
  - Nicht gut auf GPU (→ Kann nicht so einfach Brute-Forced werden, wie andere die parallelisierbar sind)

Anwendungszwecke

1. Digitale Signatur
2. Integrity (Symmetric Cryptography ist verletzlich für Tampering)
   - Hashes können zeigen, dass die Nachricht nicht verändert wurde
   - Message Authentication Code (MAC)

MAC

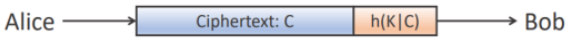

Abbildung 21: MAC

- Alice fügt dem Ciphertext den Key hinzu und hasht den Ciphertext mit dem Key (→ h(K|C))
- Sie sendet den Ciphertext mit h(K|C) an Bob
- Bob fügt ebenfalls dem Ciphertext den Key hinzu und hasht den Ciphertext mit dem Key (→ h(K|C))
- Wenn h(K|C) derselbe ist, wie der von Alice, wurde der Ciphertext nicht modifiziert \
- Standart-MACs sind gelegentlich gefährdet durch SHA-1/2 Length-Extension-Attacks
- Hash-Based MAC (HMAC) ist der meistverwendete Approach, er splitted den Key in zwei und hasht zweimal
  - Sicherer
  - Split Key in zwei Teile und man hasht zweimal mit jedem Key
  - Somit nicht gefährdet durch Length-Extension-Attacks

Complete Cryptographic Systems

Digitale Signaturen

- Hashes werden zusammen mit RSA/DSA verwendet, um Signaturen zu bilden
- Mit Signaturen kann der Sender seine Authenticity beweisen

Digitale Zertifikate

Die Signatur beweist nur, dass der Server den private Key hat, aber jeder kann einen private Key erstellen. Deshalb braucht es Zertifikate, welche über eine Drittpartei (normalerweise via PKI) die Herkunft des Schlüssels beweist.

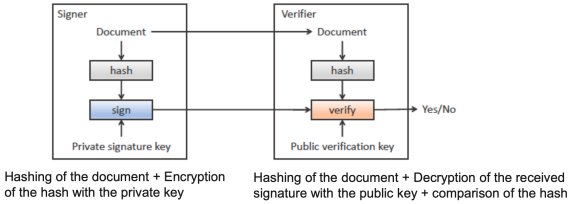

Abbildung 22: Digitale Signatur

Der Server erstellt eine Certificate Signing Request (CSR) und sendet diese an eine Certification Authority (CA)[2].

Die CA macht ein paar Identitätschecks und signiert das Zertifikat mit seinem private Key. Dann schickt es das signierte Zertifikat zurück an den Server.

$$\text{Cert}(\text{CA}, A) = \{\text{ID}_{\text{CA}}, \text{ID}_A, e_A, T, \text{Ext}, \text{sig}_{\text{CA}}\}$$

$$\text{sig}_{\text{CA}} = d_{\text{CA}}[h(\text{ID}_{\text{CA}}, \text{ID}_A, e_A, T, \text{Ext})]$$

- ID_{CA} = Eindeutiger Name der CA
- ID_A = Eindeutiger Name des Teilnehmers A (server.com)
- e_A = Öffentlicher Schlüssel
- T = Gültig bis T
- Ext = Optionale Extensions

Der Server sendet dann beim TLS-Handshake die Signatur. Dazu entschlüsselt er zunächst die Signatur mithilfe des public Keys e_A.

[2] E.g. Geotrust, Globalsign, Digicert, Godaddy, letsencrypt, Google

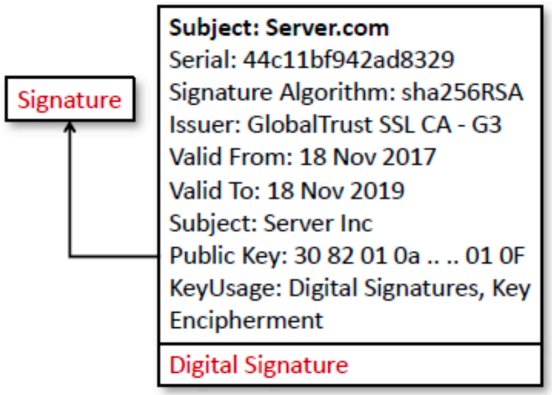

Abbildung 23: Digitales Zertifikat

Transport Layer Security (TLS)

TLS unterstützt:

- Confidentiality (Encryption)
- Integrity (HMAC)
- Authentication (Server, optional Client, Zertifikate)

Übersicht

Terms

- Secure Socket Layer (SSL) und TLS sind Network Security Protocols für die Authentisierung, Verschlüsselung von Daten und Datenaustausch
- SSL wurde nach SSL v3.0 zu TLS (momentan v1.3)
- TLS ist der primäre Mechanismus für die Verschlüsselung von HTTP-Kommunikation (HTTPS)

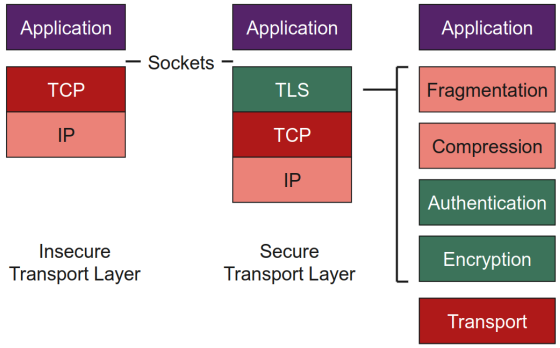

Abbildung 24: TLS Übersicht

- Kann auch perfekt mit anderen Protokollen verwendet werden
- TLS kann für Applikationen transparent sein
- TLS kann embedded werden in spezifische Pakete

Ursprünglich hiess TLS „Secure Socket Layer (SSL)“, heisst aber seit SSL v3.0 „TLS“ (aktuell v1.3). SSL/TLS ist ein **Netzwerksicherheitsprotokoll** für das Aufsetzen von authentisierten und verschlüsselten Verbindungen und Datenaustausch.

> TLS ist der primäre Mechanismus für die Verschlüsselung von HTTP-Kommunikation (HTTPS)

- TLS kann auch für andere Protokolle verwendet werden
- Kann für Applikationen tranparent sein
- Kann embedded werden in spezifische Pakete

TLS-Architektur

TLS Connection

- Transport, welcher einen passenden Type of Service anbietet
- Peer-to-Peer
- Connections sind flüchtig (transient)
- Jede Verbindung ist mit einer eigenen Session verknüpft

TLS Session

- Eine Verbindung (association) zwischen Cleint und Server über ein Handshake Protokoll
- Definiert ein Set von kryptographischen Sicherheitsparametern, welche an mehrere Verbindungen geteilt werden können
- Es müssen nicht für jede Verbindungen neue (teure) Sicherheitsparameter verhandelt werden

Geschichte

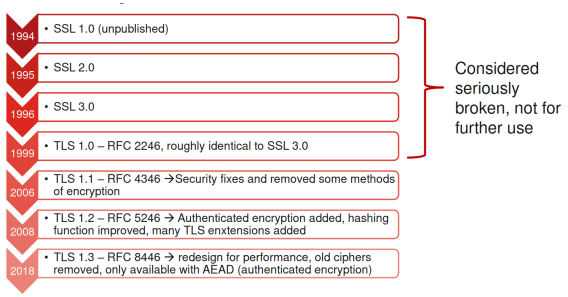

Abbildung 25: TLS Geschichte

TLS 1.3 (RFC 8446, August 2018) – Verbesserungen

- Clean up: Unsichere oder nicht verwendete Funktionen entfernt

  - Legacy & Broken Crypto: (3)DES, RC4, MD5, SHA1, Kerberos, RSA PKCS#1v1.5. Key Transport
  - Cipher Suites von > 100 auf 5 reduziert
  - Broken Features: Compression und Renegotiation
  - Statischer RSA/DH entfernt
- Performance: 1-RTT und 0-RTT Handshakes
- Sicherheit verbessert
- Privacy: Fast alle Handshake-Messages werden verschlüsselt
- Continuity: Backwards Compatibility

TLS-Record-Structure

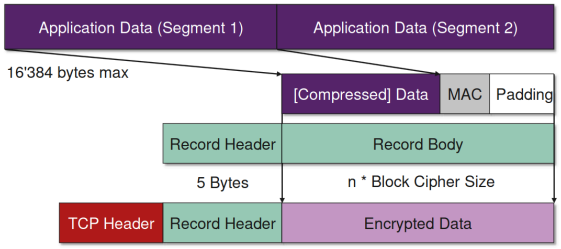

Abbildung 26: TLS Frame Structure

Abbildung 27: TLS Framestruktur (Wireshark)

HTTPS-Content-Types:

- Handshake Protocol (22): ClientHello, ServerHello, Certificate, ServerHelloDone
- Change CipherSpec Protocol (20): Wechsel der Cipher Suite
- Alert Protocol (21): Warning, Fatal (Session wird auf der Stelle beendet)
- Application Protocol (23): Verschlüsselter Payload wird versendet (Transmission)

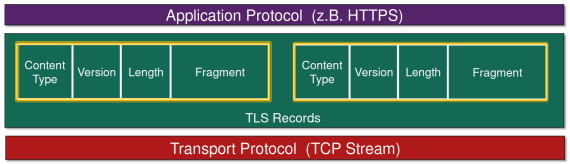

Abbildung 28: TLS Records

TLS-Handshake

- Bevor Applikationsdaten übermittelt werden
- Server und Client:
  - Authentisieren sich gegenseitig
  - Verhandeln Verschlüsselungs- und MAC-Algorithmen
  - Verhandeln Schlüssel
- Vier Phasen:
  1. Sicherheitsfähigkeiten aushandeln (TLS Version, etc.)
  2. Server kann Zertifikat schicken, Schlüsselaustausch und Anforderung eines Zertifikats
  3. Client sendet Client-Zertifikat, wenn angefordert. Client sendet Schlüsselaustausch.
  4. Änderung der Cipher Suite (ChangeCipherSpec) und Abschluss des Handshake Protokolls

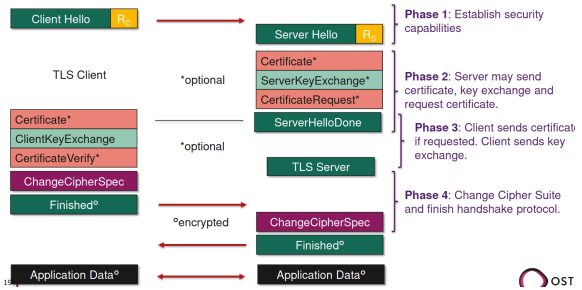

Abbildung 29: TLS Handshake

RSA Public Key Encryption ohne Perfect Forward Secrecy (Legacy, v1.2)

- Problematisch, da Pakete jetzt mitgehört werden können und später entschlüsselt werden können (Quantencomputing)
- RSA Keyaustausch

Ephemeral Diffie-Hellmann (DHE) Key Exchange (Perfect Forward Secrecy, v1.3)

- Diffie-Hellmann Key Exchange

Public Key Infrastructure (PKI)

Alles basiert auf *Trust* im digitalen Zeitalter. Momentane Probleme in der digitalen Ökonomie sind:

- Fragmentation
- Lack of interoperability
- Steigerung der Cyberkriminalität
- eID und Trust Services haben ein gemeinsames Sicherheitsfundament

Wenn eine HTTPS-Verbindung auf ost.ch stattfindet, wird Public Key Cryptography für die Authentication, Verschlüsselung und digitale Signaturen verwendet. Wichtige Fragen sind:

- Ist dies tatsächlich der Schlüssel für www.ost.ch?
- Ist der Key verwendet worden, um zu signieren?

Public Key als String von Bits:

- Wem gehört dieser Bit-String?
- Für welche Zwecke kann er verwendet werden?
- Ist er noch immer valid?

PKI soll eine Antwort auf all diese Fragen liefern.

- PKI wird verwendet, um einen Public Key zu einer Identität zu binden
- Die Bindings passieren über:
  - Einen Registrierungsprozess einer Registration Authority (RA) und
  - Eine Issuance eines Zertifikats durch eine Certificate Authority (CA)
- Die CA kann durch eine unabhängige Validation Authority (VA) validiert werden
- Das Artifact einer Bindung nennt sich das Zertifikat

Geschichte

- X.500 ist die Menge der Netzwerkstandarts, die elektronische Directory-Services abdeckt (1988)
- X.509 ist der Standart für das Format von Public Key Zertifikaten (1988)
- X.509v3 Profiles sind die meistverwendeten.

Public Key Certificate

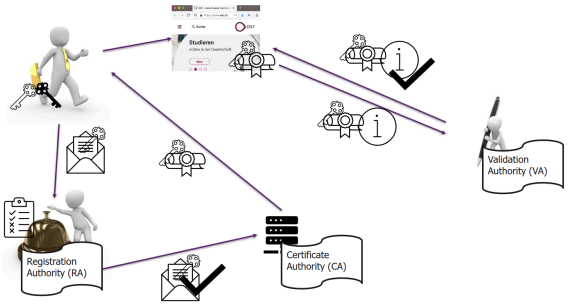

Abbildung 30: Public Key Infrastructure (PKI)

- Ein elektronisches Dokument, welcher die Ownership eines Public Keys beweist
- Enthält Informationen über den Schlüssel
- Enthält Informationen über den Besitzer des Schlüssels (Subjekt)
- Enthält Informationen über die CA, die das Zertifikat signiert hat (Issuer)

Felder eines Zertifikats

- Version
- Serial Number: Eindeutige Nummer des Zertifikats
- Signature Algorithm: Algorithmus, der für die Signatur von der CA verwendet wurde
- Issuer Distinguished Name (DN): Name der CA
- Gültig von/bis
- Public Key
- Key Usage: Für welche Zwecke der Schlüssel verwendet werden kann
- Extended Key Usage (EKU): serverAuth, clientAuth, codeSigning, emailProtection, timeStamping, OCSPSigning

Vier Kategorien

Abhängig vom Typ der Checks, die durchgeführt werden:

- Domain Validated (DV): Issued nachdem der Domain-Besitz bestätigt wurde
- Organisation Validated (OV): Issued nachdem die Organisation bestätigt wurde (Company name, domain name, etc. durch öffentliche Datenbanken)
- Extended Validation (EV): Issued nachdem die Entität eine strikte Authentisierungsprüfung durchlaufen hat
- Qualified Website Authentication Certificate (QWAC): Ein qualifiziertes Zertifikat (eIDAS, PSD2 Regulation)

Validitätsinformatinonen erhalten:

- Certificate Revocation List (CRL): Liste von Zertifikaten, die nicht mehr gültig sind
  - CRL DIstribution Point im Zertifikat
  - Wird von den meisten Browsern geprüft
- Online Certificate Status Protocol (OCSP):
  - Authority Information Access (AIA)-Feld im Zertifikat zeigt, wie Informationen über die CA erhalten werden können

Desweiteren

- Certificate Policies:
  - Link zu den Regeln der CA
  - Jedes CA hat eigene Policies
- Authority Key Identifier (AKI):

- Key-Identifier des CA-Zertifikats, welches das TLS-Zertifikat signiert hat
- Subject Alternative Name (SAN): Weitere Informationen über den Besitzer des Zertifikats
- Subject Key Identifier (SKI): Hash-Wert des Zertifikats, wird verwendet, um die Identität des Zertifikats zu überprüfen

Trust Service Provider (TSP)

Erstellt Trust zwischen Kommunikationspartnern:

- Trusted Identity Information
- Secure Authentication
- Integrity Protected Communication
- Encrypted Communication

Certificate Issuance

- Der Subscriber (Die Entität, die das Key-Pair besitzt, wessen public Key ein Zertifikat erhalten soll) sendet eine Certificate Signing Request (CSR) an die CA
- CSR enthält Informationen über das Onjekt des Subscribers, den public Key (welcher signiert werden soll) und Informationen über Key-Type und Länge
- CSR wird im Base64-Format an die CA gesendet
- CA registriert den Request, prüft die Daten und signiert den CSR (→ X.509-Zertifikat)

CA Hierarchy

- Root CA: Trust-Anchor der PKI.
- Issuing CA: CA, die Zertifikate für End-Entities signiert (Auch Intermediate CA, Subordinate CA)

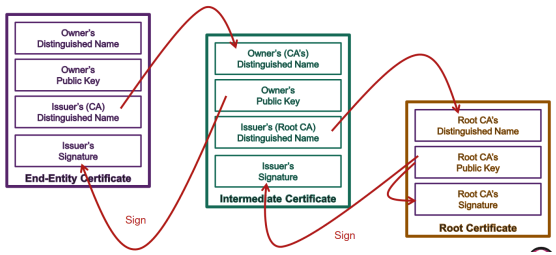

Abbildung 31: CA Hierarchy

Trust mit Zertifikaten

- PKI ist vertrauenswürdig mit allem im Trust Store
- Es können hunderte von Root-Zertifikaten auf der Client-Maschine sein
- Certificate Pinning: Client speichert den public Key des Servers und vertraut nur diesem Key
  - Operatoren pinnen die CA issuers, public Keys oder End-Entity-Zertifikate. Clients vertrauen nur diesen Zertifikaten.

Certificate Pinning

- 2011 Google Chrome
- HTTP Public Key Pinning (HPKP)
- Risiken:
  - Key Compromise nicht entdeckt
  - Hackers können eigene HPKP auf komprimierten Servern setzen

Ethical Hacking und Penetration Testing

Attackkonzepte

- Hacking
  - Vulnerabilities ausnützen
  - Security control compromise
  - Verhalten von Systemen ändern
- Ethical Hacking
  - Hacking, aber legal
  - Tools und Techniken verwenden, um Schwachstellen zu finden, validieren, doumentieren und melden
  - Vulnerability existence reporting

Hacker-Typen

- Black Hat
  - Böse Hacker, malicious, bleibt normalerweise anonym
- Grey Hat
  - Hacker mit Black Hat Skills, die ihre Fähigkeiten offensiv und defensiv einsetzen
- White Hat
  - Hacker mit Black Hat Skills, die ihre Fähigkeiten nur defensiv einsetzen
- Script Kiddie
  - Verwendet Tools, ohne zu verstehen, was sie tun
- Cyber Terrorist
  - Skilled Attacker, der mit Hacking einer Ideologie vorantreibt
- State Sponsored
  - Vom Staat angestellt für offensive und defensive Aktivitäten
- Hacktivist
  - Ein Hacker für soziale oder politische Zwecke

Hacktivism

Authentication Schemes

Basic Authentication

Übertragung der Credentials via Plaintext: Man in the Middle Attacks

One Time Passwords

Passwörter, die generiert werden und nur einmal verwendet werden können. MITM möglich, jedoch nur kurzfristig, z.B. Session-Hijacking. Dasselbe Passwort kann aber nicht wiederverwendet werden.

Challenge-Response-Verfahren

Der User kennt bereits seine ID_U und generiert einen Random-Wert R_U. Der Server sendet einen Random-Wert R_S an den User. Dieser generiert dann aus R_S, R_U, ID_U und einer vordefinierten **Keyed Hash Function** und einem Key eine MAC. Er sendet diese MAC dann zurück an den Server. Der Server macht denselben Durchlauf und überprüft, ob die beiden MACs dieselben sind (siehe Abbildung 32).

Challenge-Response based on Digital Signatures

Derselbe Prozess wie beim herkömmlichen Challenge-Response-Verfahren, nur wird hier der Hash vor der Übertragung verschlüsselt (Siehe Abbildung 33).

Kerberos

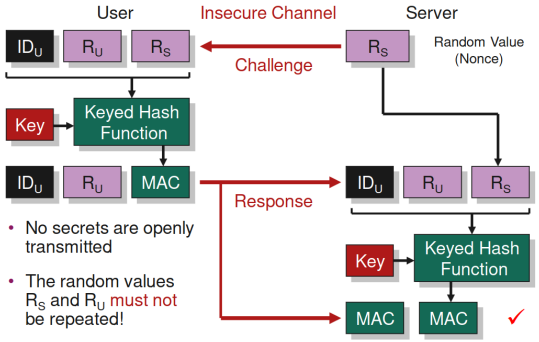

Abbildung 32: Challenge-Response-Verfahren

Ablauf (Simplifiziert)

Siehe Abbildung 34.

1. Alice hashed ihr Passwort: MKey_A
2. A. verschlüsselt mit diesem Hash die aktuelle Uhrzeit: $E(\mathrm{time}, \mathrm{MKey}_A)$
3. A. sendet diesen Wert an den Key an das Key Distribution Center (KDC)

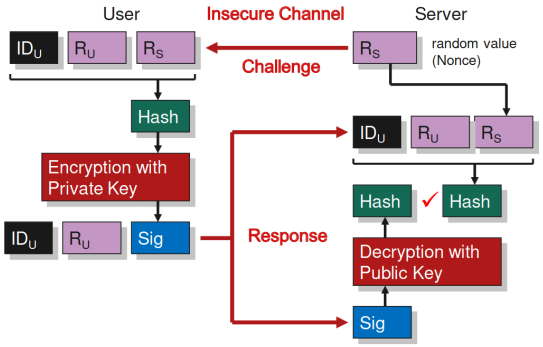

Abbildung 33: Challenge-Response-Verfahren mit Digitaler Signatur

4. KDC entschlüsselt den Wert mit dem Hash von Alice: $D(E(\mathrm{time}, \mathrm{MKey}_A), \mathrm{MKey}_A)$ und prüft, ob die Zeit gültig ist
5. KDC generiert einen Session Key S_{AB} und verschlüsselt diesen mit dem Hash von A.: $E(S_{\mathrm{AB}}, \mathrm{MKey}_A)$
6. KDC gen. Ticket: $E(\{\mathrm{Alice}, S_{\mathrm{AB}}\}, \mathrm{MKey}_B)$
7. KDC sendet verschlüsselten Session Key und Ticket an A.
8. A. entschlüsselt den Session Key
9. A. verschlüsselt die aktuelle Zeit mit dem Session Key und sendet dies zusammen mit dem Ticken an B.

Anonymous Key Exchange

Generieren eines Keys mithilfe von Diffie-Hellman.

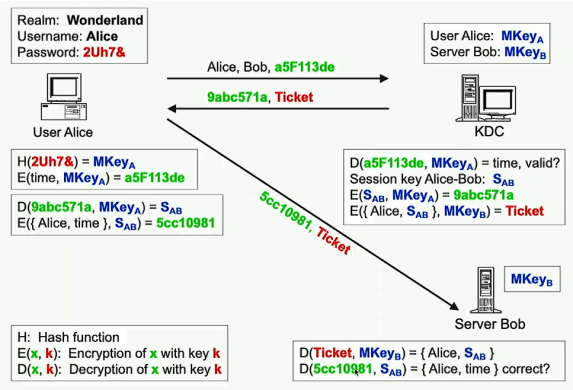

Abbildung 34: Kerberos

Certificate-Based Server Authentication

Der Server sendet sein Zertifikat an den Client. Der Client überprüft die Idenität des Servers und sendet dann das Passwort verschlüsselt an den Server, wenn der Server authentifiziert ist.

Mutual Public Key Authentication

Der Client authentifiziert sich mit seinem Client-Zertifikat und macht eine Signatur mit seinem private Key. Der Server überprüft die Signatur des Clients. Gleichzeitig sendet der Server sein Zertifikat an den Client, welcher die Signatur des Servers überprüft.