Sichere Webanwendungen

Lerneinheit 4: Authentisierung

Prof. Dr. Christoph Karg

Studiengang Informatik
Hochschule Aalen

Sommersemester 2025

11.4.2025

Gliederung

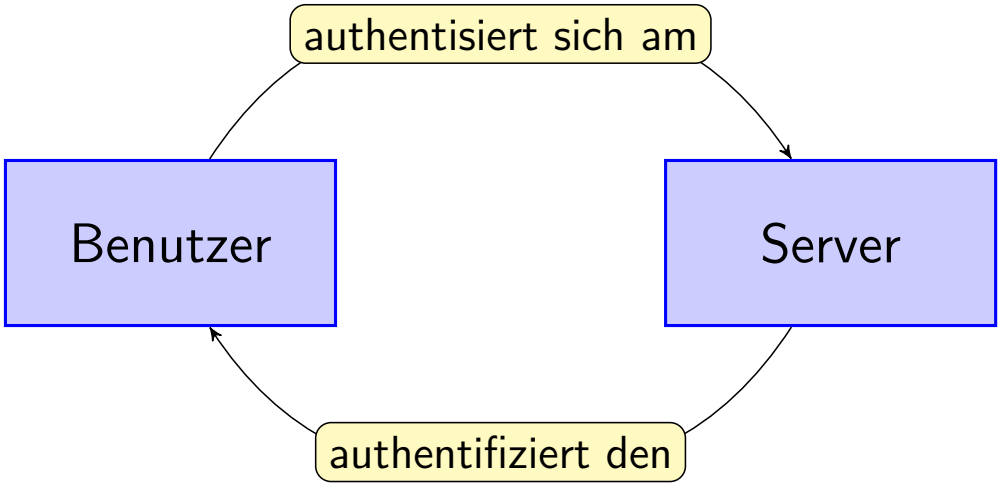
Authentizität

- Mit dem Begriff Authentizität wird die Eigenschaft bezeichnet, die gewährleistet, dass ein Kommunikationspartner tatsächlich derjenige ist, der er vorgibt zu sein.
- Bei authentischen Informationen ist sichergestellt, dass sie von der angegebenen Quelle erstellt wurden.
- Der Begriff wird nicht nur verwendet, wenn die Identität von Personen geprüft wird, sondern auch bei IT-Komponenten oder Anwendungen.

Authentisierung

- Authentisierung bezeichnet den Nachweis eines Kommunikationspartners, dass er tatsächlich derjenige ist, der er vorgibt zu sein.
- Im Deutschen wird manchmal zwischen den Begriffen Authentisierung und Authentifizierung unterschieden.
- Mit Authentisierung wird dann die Vorlage eines Nachweises zur Identifikation bezeichnet, mit Authentifizierung die Überprüfung dieses Nachweises.
- Der Einfachheit halber wird im folgenden auf diese Unterscheidung verzichtet und ausschließlich der Begriff Authentisierung verwendet.

Authentisierung und Authentifizierung



Arten der Authentisierung

- Authentisierung durch Wissen (What you know!)
  - Passwort
  - PIN
  - Sicherheitsfrage
- Authentisierung durch Besitz (What you have!)
  - Physischer Schlüssel
  - Chipkarte
  - USB Token
  - PIN Generator
- Authentisierung durch biometrische Merkmale (What you are!)
  - Fingerabdruck
  - Gesichtserkennung
  - Iriserkennung
  - Stimmerkennung

Einsatz von Passwörtern

- Passwörter sind die am häufigsten eingesetzte Authentisierungsmethode.
- Vorteile:
  - Ein Passwort ist einfach zu erzeugen.
  - Ein Passwort ist einfach zu benutzen.
  - Die Infrastruktur zur Verarbeitung von Passwörtern ist kostengünstig.
- Nachteile:
  - Gute Passwörter sind schwer zu merken.
  - Die Sicherheit hängt nicht nur vom Eigentümer des Passworts ab.
  - Das Passwort muss am System eingegeben werden.

RockYou Vorfall 2009

rockyou

- RockYou war eine Plattform für Social Networking.
- Benutzer konnten über RockYou auf andere Plattformen wie MySpace oder Facebook zugreifen.
- Die Passwörter der Benutzer wurden von RockYou im Klartext in einer Datenbank gespeichert.
- Dezember 2009: SQL Injection Angriff auf RockYou.
- Konsequenz: Diebstahl von 32 Millionen Passwörtern.
- Die Passwörter wurden über eine anonymisierte Liste veröffentlicht.
- Für weitere Details siehe: [Cub09]

Analyse der Passwörter von RockYou Nutzern

- 30% der Passwörter hatten eine Länge von höchstens 6 Zeichen.
- 20% der Passwörter waren in einer Liste von 5000 einfach zu erratenden Passwörtern enthalten.
- 40% der Benutzer wählten Passwörter, die nur Kleinbuchstaben enthielten.
- Nur 0.2% der Passwörter erfüllten die folgenden (gängigen) Anforderungen:
  - Länge von mindestens 8 Zeichen
  - Mischung aus Klein- und Großbuchstaben sowie Ziffern und Sonderzeichen
- Weitere Informationen findet man hier: [ADC2014]

Häufig verwendete Passwörter

Rang	*Passwort*	*Häufigkeit*
1	123456	290731
2	12345	79078
3	123456789	76790
4	Password	61958
5	iloveyou	51622
6	princess	35231
7	rockyou	22588
8	1234567	21726
9	12345678	20553
10	abc123	17542

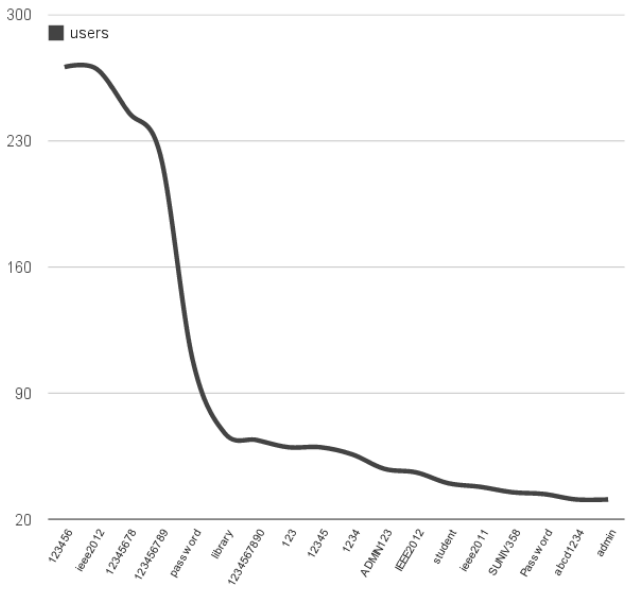
Rang	*Passwort*	*Häufigkeit*
11	Nicole	17168
12	Daniel	16409
13	babygirl	16094
14	monkey	15294
15	Jessica	15162
16	Lovely	14950
17	michael	14898
18	Ashley	14329
19	654321	13984
20	Qwerty	13856

Top 20 der Passwörter von RockYou Nutzern

Beispiel 2: IEEE Log Vorfall 2012

- Im September 2012 wurde eine Fehlkonfiguration des Webservers der IEEE entdeckt.
- Die Log-Dateien des Webservers waren öffentlich über FTP zugänglich.
- Der Zeitraum der Log-Daten war vom 1. August 2012 bis zum 18. September 2012.
- Enthaltene Informationen:
  - Log Einträge insgesamt: 376.021.496.
  - Log Einträge mit Informationen zu Passwörtern: 411.308.
  - 99.979 Zugangsdaten von IEEE-Mitgliedern.
- Konsequenz: annähernd 100.000 kompromittierte Nutzer.
- Betroffen: Apple, Google, IBM, Oracle, Samsung, NASA, Stanford University, ...
- Für weitere Details siehe: [Ras12]

Häufigste Passwörter



Häufigste Passwörter (Forts.)

Rang	*Passwort*
1	123456
2	ieee2012
3	12345678
4	123456789
5	password
6	library
7	1234567890
8	123
9	12345
10	1234

Empfehlungen

- Eigenschaften eines guten Passworts:
  - Das Passwort ist mindestens 10 Zeichen lang.
  - Kombination aus Groß- und Kleinbuchstaben sowie Ziffern und Sonderzeichen.
  - Enthält weder den Namen der Person noch ihre E-Mailadresse.
- Die Qualität des Passworts sollte vom Einsatzzweck abhängen.
- Niemals ein Passwort zweimal benutzen.
- Niemals ein Passwort an Dritte weitergeben.
- Passwörter sollten regelmäßig geändert werden.
- Zur Verwaltung der Passwörter bietet sich ein Password Safe an.

Methode von Bruce Schneier

Idee: Denke Dir einen langen Satz aus und konstruiere daraus ein gutes Passwort.

Beispiel: Satz:

> Sheldon Cooper ist ein hochbegabter Physiker mit zwei Doktortiteln.

Mögliche Passwörter:

- `SCi1hPm/2D`
- `ShC0pi1hPm2D+`
- `7nCriT1HrPrm%zID9`

Für weitere Details siehe: [Sch08]

Diceware Methode

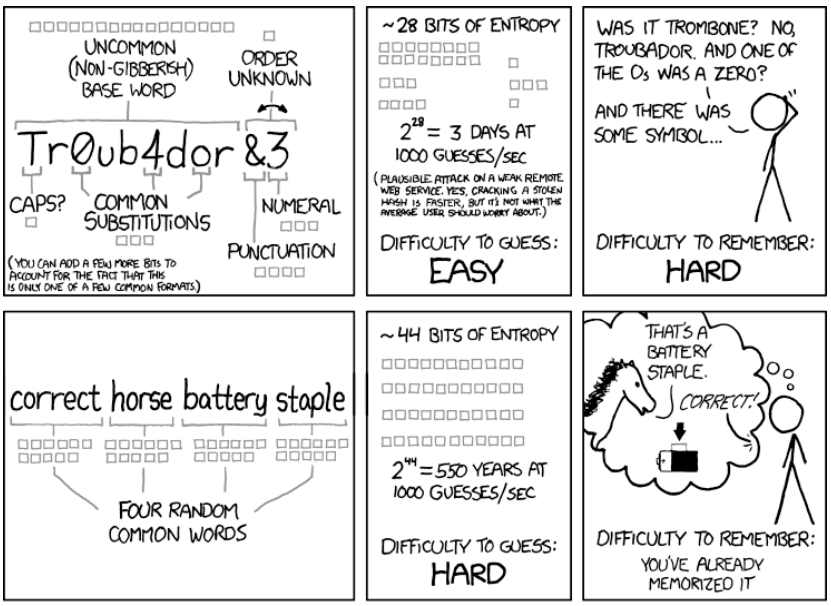
Idee: Nutze einen Würfel und eine Wortliste zur Erstellung eines Passworts.

Vorgehen:

- Lege die gewünschte Länge ℓ des Passworts fest (Empfehlung: 6 Teilwörter).
- Würfele $5 \cdot \ell$-mal und notiere die Ziffern in 5-er Blöcken.
- Wähle pro 5-er Block ein Wort aus der Liste.
- Hänge die Wörter hintereinander und erstelle so das Passwort.
- Merke Dir das Passwort und vernichte die Notizen.

Für Details siehe: [Rei17]

Stärke von Passwörtern



Quelle: `https://xkcd.com/936`

Angriffe gegen Passwörter

- Wenn man den Passwort Hash kennt, führt oft ein Brute Force Angriff zum Ziel.
- Ein moderner PC kann Millionen von Hashes in der Sekunde berechnen.
- Der Einsatz von sogenannten Rainbow Tables beschleunigt die Suche nach Passwörtern.
- Cloud Dienste wie die Amazon Elastic Cloud stellen große Rechenleistung für wenig Geld zur Verfügung.

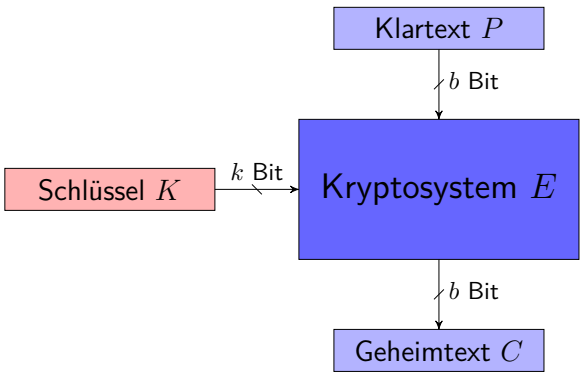
Time-Memory Trade-Off

- Time-Memory Trade-Off (TMTO) ist eine Technik zur Verbesserung der Suche nach Schlüsseln bei einer Kryptoanalyse mit bekanntem Klartext.
- Idee: Nutze eine im Voraus berechnete Tabelle mit Passwörtern, um die Brute Force Suche zu beschleunigen.
- Die grundlegende Arbeit stammt von Hellman [Hel80].
- Weiterentwicklungen:
  - Rivest: Einsatz von Distinguished Points [Den82, Seite 100]
  - Oechslin: Einsatz von Rainbow Tables [Oec03]

Das Verfahren von Hellman

Ausgangspunkt: Kryptosystem $E : \{0,1\}^k \times \{0,1\}^b \mapsto \{0,1\}^b$ mit einer Blocklänge von b Bit und einer Schlüssellänge von k Bit.

Schaubild:



Beispiel: Data Encryption Standard (DES): Blocklänge 64 Bit, Schlüssellänge 56 Bit.

Die zentrale Idee

Aufgabenstellung: Finde für ein gegebenes Klartext-Geheimtext-Paar (P_0, C_0) einen Schlüssel K, so dass $E(K, P_0) = C_0$.

Parameter:

- $N \rightsquigarrow$ Anzahl der zu untersuchenden Schlüssel $(N = 2^k)$.
- $T \rightsquigarrow$ Anzahl der erlaubten Rechenoperationen.
- $M \rightsquigarrow$ Anzahl der zur Verfügung stehenden Speicherzellen.

Bemerkungen:

- Die Rechenzeit wird asymptotisch gemessen.
- Logarithmische und konstante Faktoren werden der Einfachheit halber bei der Analyse nicht berücksichtigt.
- Eine Rechenoperation entspricht einer Entschlüsselung.
- Eine Speicherzelle kann genau einen Schlüssel speichern.

Time-Memory Trade-Off

Ansatz: Suche des Schlüssels mit T Rechenoperationen unter Verwendung von M Speicherzellen

Komplexitätsmaß: $T + M$.

Szenarien:

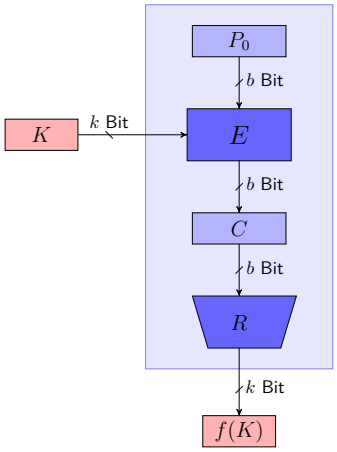
- Exhaustive Search $(T = N, M = 1)$: bei großem Schlüsselraum aufwändig und daher nicht praktikabel.
- Table Lookup $(T = 1, M = N)$: bei großem Schlüsselraum sehr speicherintensiv und daher nicht praktikabel.
- Mischform: durch geeignete Kombination von Rechenzeit und Speicherplatz mit geringerem Aufwand durchführbar.

Bemerkung: Speicherplatz ist in der Regel teurer als Rechenzeit.

Funktion f

Beobachtung: ein gutes Kryptosystem ist auch ein guter Pseudozufallszahlengenerator.

Umsetzung für das Klartext-Geheimtext-Paar (P_0, C_0):



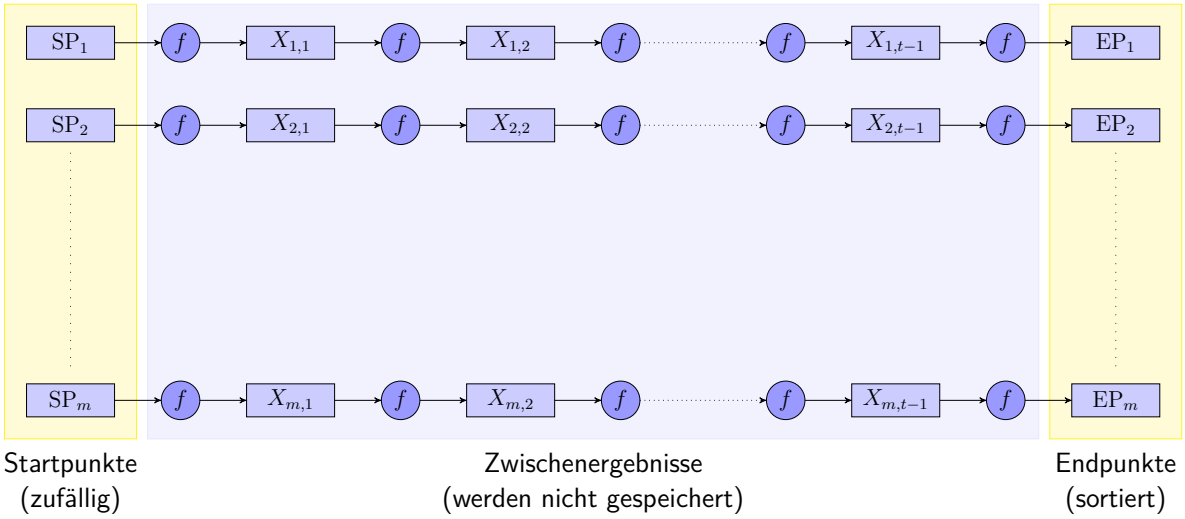
Die Reduktionsfunktion R ist eine Abbildung von $\{0,1\}^b$ nach $\{0,1\}^k$.

Berechnung der Lookup Table

CreateLookupTable(m, t)
Input: Anzahl Elemente m, Anzahl Zwischenschritte t
Output: Lookup Table L mit m Elementen

1: **for** $i \leftarrow 1$ **to** m **do**
2: Wähle SP_i zufällig unter Gleichverteilung aus der Menge aller Schlüssel.
3: $X_{i,0} \leftarrow SP_i$
4: **for** $j \leftarrow 1$ **to** t **do**
5: $X_{i,j} \leftarrow f(X_{i,j-1})$
6: $EP_i \leftarrow X_{i,t}$
7: Füge (SP_i, EP_i) in die Tabelle L ein.
8: Sortiere L anhand der EP-Werte.
9: **return** L

Berechnung der Lookup Table (Forts.)



Suche nach dem Schlüssel

SearchKey$((P_0, C_0), L)$
Input: Klartext-Geheimtext-Paar (P_0, C_0), passende Lookup Table L
Output: Schlüssel K

1: $Y_1 \leftarrow R(C_0)$
2: **for** $i \leftarrow 1$ **to** t **do**
3: **if** Y_i ist der Endpunkt EP_j in L **then**
4: $X_{j,t-i} \leftarrow f^{t-i}(SP_j)$
5: **return** $X_{j,t-i}$
6: **else**
7: $Y_{i+1} \leftarrow f(Y_i)$
8: **return** *Failure*

Analyse der Schlüsselsuche

- Für den korrekten Schlüssel K gilt:

$$R(C_0) = E(K, P_0) = f(K).$$

- Der Algorithmus sucht in der Lookup Table durch iteratives Berechnen von $f^i(R(C_0))$ nach einem Endpunkt EP_j.
- Wird ein passender Endpunkt EP_j gefunden, dann wird ausgehend vom Startpunkt SP_j der Schlüssel K berechnet.
- Insgesamt muss die Funktion f t-mal berechnet werden.
- Da die Suche auf Basis eines reduzierten Werts von C_0 stattfindet, kann der gefundene Schlüssel falsch sein.
- Ist $t \cdot m < N$, dann besteht die Möglichkeit, dass der gesuchte Schlüssel nicht gefunden wird.

Erfolgsaussichten

Annahmen:

- Die $m \cdot t$ Werte der Time-Memory Trade-Off Table sind paarweise verschieden.
- Der Schlüssel K wird zufällig unter Gleichverteilung gezogen.

Erfolgswahrscheinlichkeiten:

- Exhaustive Search mit t Berechnungen: $Prob[S] = \frac{t}{N}$
- Table Lookup mit m Einträgen: $Prob[S] = \frac{m}{N}$
- Time-Memory Trade-Off: $Prob[S] = \frac{t \cdot m}{N}$

Bemerkung: Eine „gewisse“ Anzahl von doppelten Einträgen beim Time-Memory Trade-Off kann toleriert werden.

Abschätzung der Erfolgschancen

Satz 1. Angenommen, $f : \{1, 2, \ldots, N\} \mapsto \{1, 2, \ldots, N\}$ ist eine Abbildung und $K \in \{1, 2, \ldots, N\}$ wird zufällig unter Gleichverteilung gewählt.

Dann ist die Schlüsselsuche mittels Time-Memory Trade-Off erfolgreich mit der Wahrscheinlichkeit

$$Prob\,[S] \geq \frac{1}{N} \sum_{i=1}^{m} \sum_{j=0}^{t-1} \left(\frac{N - it}{N} \right)^{j+1}.$$

Interpretation des Satzes

- Falls $m \cdot t^2 \ll N$, dann ist jeder Summand ungefähr gleich 1 deshalb
$$Prob\,[S] \geq \frac{m \cdot t}{N}.$$
- Im Beweis wird angenommen, dass f eine Zufallsfunktion ist. Die Zufälligkeit von f wirkt sich auf die Länge der von f erzeugten Zyklen aus.
- Für gängige Werte für m und t ist $Prob\,[S]$ klein. Die Wahrscheinlichkeit lässt sich erhöhen, indem man mehrere Lookup Tabellen erzeugt und verschiedene Reduktionsfunktionen einsetzt.

Anwendung des Satzes

Wahl der Parameter:

- Setze $m = t = N^{1/3}$.
- Generiere $N^{1/3}$ Lookup Tabellen unter Einsatz verschiedener Reduktionsfunktionen R.

Ergebnis:

- Der gesuchte Schlüssel ist mit hoher Wahrscheinlichkeit in einer der Tabellen enthalten.
- Der Aufwand besteht aus:
  - $M = N^{2/3}$ Speicherplatz ($N^{1/3}$ Tabellen mit je $N^{1/3}$ Werten),
  - $T = N^{2/3}$ Rechenzeit ($N^{1/3}$ Operationen pro Tabelle).
- Die Tabellen können bei entsprechender Hardware parallel durchsucht werden.

Anwendungsbeispiel

Beispiel: Data Encryption Standard (DES)

Schlüssellänge: 56 Bit $\rightsquigarrow N = 2^{56}$.

Time-Memory Trade-Off:

- Speicherplatz: $N^{2/3} \approx 2^{37.33}$ Speicherzellen je 14 Byte
- Rechenzeit: $N^{2/3} \approx 2^{37.33}$ Operationen

Fazit: Lässt man den Aufwand zur Berechnung der Lookup Tabellen außer Acht, dann verringert sich der zeitliche Aufwand merklich.

Diskussion

- Die TMTO-Attacke von Hellman war über viele Jahre die beste Methode zur Suche nach Passwörtern.
- Die Qualität der Lookup Tabellen hängt von den erzeugten Passwortketten ab.
- Treten bei der Funktion f Kollisionen auf, dann münden verschiedene Startpunkte im selben Endpunkt.
- Werden die Lookup Tabellen wegen ihrer Größe auf einem externen Datenträger gespeichert, dann wirkt sich dies auf die Zugriffszeit aus.

Distinguishing Points

Problem: Die Suche nach einem Endpunkt auf der Festplatte nimmt viel Zeit in Anspruch.

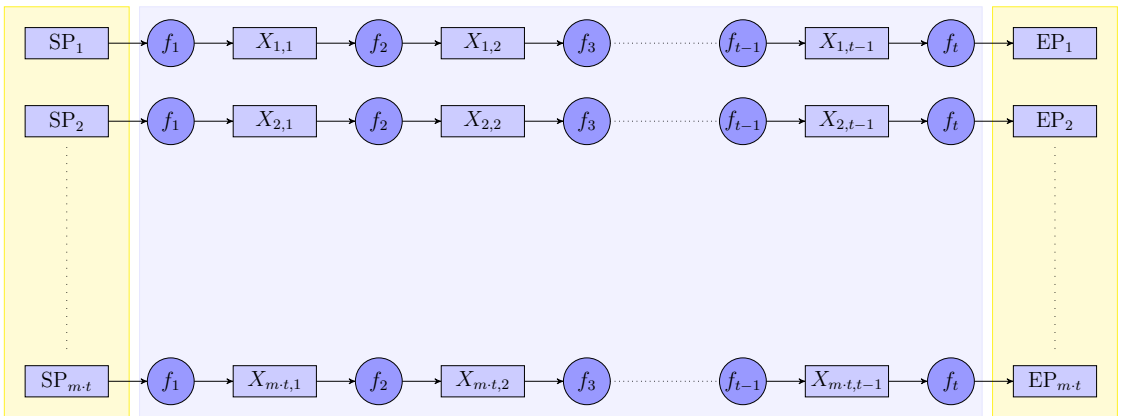
Vorschlag von Ron Rivest: Einsatz von Distinguished Points [Den82].

- Als Endpunkte werden nur Schlüssel gespeichert, die eine vorgegebene Eigenschaft haben (Beispiel: die ersten 8 Bits sind gleich Null).
- Eine Schlüsselkette wird solange berechnet, bis ein passender Schlüssel gefunden wird.
- Bei der Schlüsselsuche wird nur auf die Tabelle zugegriffen, wenn der zu suchende Wert die geforderte Eigenschaft besitzt.

Rainbow Tables

- Rainbow Tables basieren auf einer Arbeit von Philippe Oechslin aus dem Jahr 2003 [Oec03].
- Ansatz:
  - Es wird eine Lookup Tabelle der Dimension $mt \cdot t$ erstellt.
  - Zur Berechnung einer Zeile der Lookup Tabelle wird in jeder Runde eine andere Funktion f verwendet.
- Die Rundenfunktionen werden anhand von f definiert. (Beispiel: Addiere in der i-ten Runde den Wert i zur Ausgabe von f.)
- Vorteil: Der Ansatz von Oechslin verringert die Zahl der Kollisionen deutlich.
- Nachteil: Die Schlüsselsuche ist rechenintensiver.

Berechnung einer Rainbow Table



Startpunkte (zufällig) | Zwischenergebnisse (werden nicht gespeichert) | Endpunkte (sortiert)

Suche nach Passwörtern

- Eine (schlechte) Art der Speicherung von Passwörtern besteht darin, mittels einer kryptographischen Hashfunktion eine Prüfsumme des Passworts zu berechnen und diese in einer Datenbank zu speichern.
- Kennt man den Passwort-Hash, dann kann man die TMTO-Attacke anpassen, um das zugrunde liegende Passwort zu suchen.
- Die Attacke lässt sich einfach verhindern, wenn man an das Passwort einen zufälligen Wert (Salt) anhängt. Dieser Wert muss zusammen mit der Prüfsumme gespeichert werden und muss nicht geheim gehalten werden.

Software zur Passwortsuche (Auswahl)

- Hashcat (`https://hashcat.net/hashcat`)
  - Einer der schnellsten Passwort-Cracker
  - Unterstützung vieler Passwort-Hash-Formate
- John The Ripper (`http://www.openwall.com/john/`)
  - Der Klassiker unter den Passwort-Crackern
  - Unterstützung von verteilter Suche auf mehreren Rechnern
- Ophcrack (`http://ophcrack.sourceforge.net/`)
  - Windows Passwort-Cracker
  - Passwortsuche auf Basis von Rainbow Tables

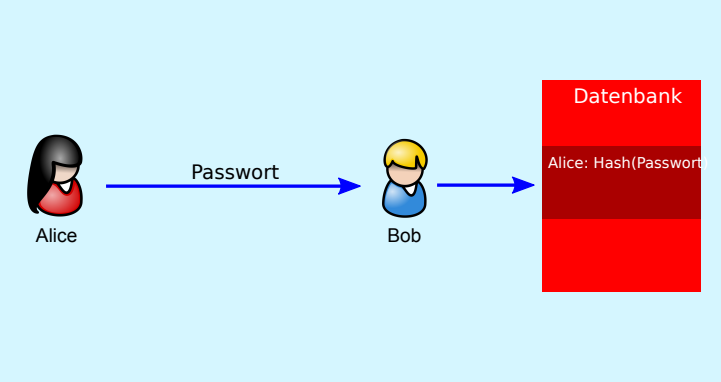
Zusammenfassung

- Die TMTO-Attacke und ihre Verbesserungen können die Suche nach Passwörtern erheblich verkürzen.
- Dank moderner Computer und Grafikkarten ist die Schlüsselsuche als auch die Erstellung der Lookup Tabellen mit moderatem Zeitaufwand möglich.
- TMTO-Attacken wurden in der Vergangenheit erfolgreich zum Passwort-Knacken eingesetzt. Zum Beispiel ist es möglich, die LM-Hashes von Windows Passwörtern innerhalb weniger Sekunden zu analysieren.

Zwei-Faktor-Authentifizierung

- Die Sicherheit bei der Authentisierung mit Passwörtern lässt sich dadurch verbessern, dass man zusätzlich eine zweite Authentisierungsart verwendet.
- In der Regel wird dabei ein technisches Gerät wie zum Beispiel eine Smartcard oder ein PIN-Generator eingesetzt.

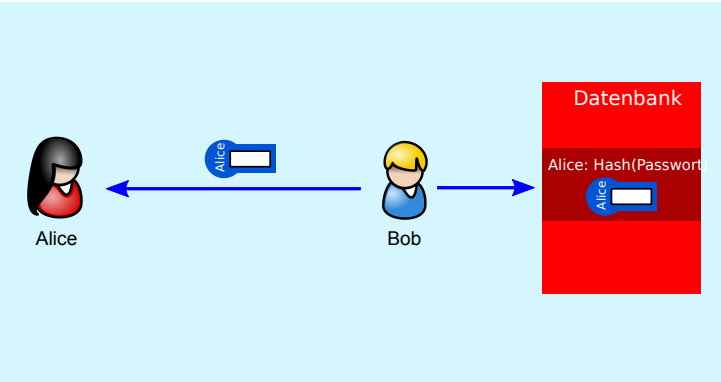
Zwei-Faktor-Authentifizierung – Ablauf



Schritt 1:

1. Alice übergibt Bob das Passwort, mit dem sie sich in Zukunft authentisieren will.
2. Bob prüft die Identität von Alice und speichert ihr Passwort auf geeignete Art und Weise in einer Datenbank.

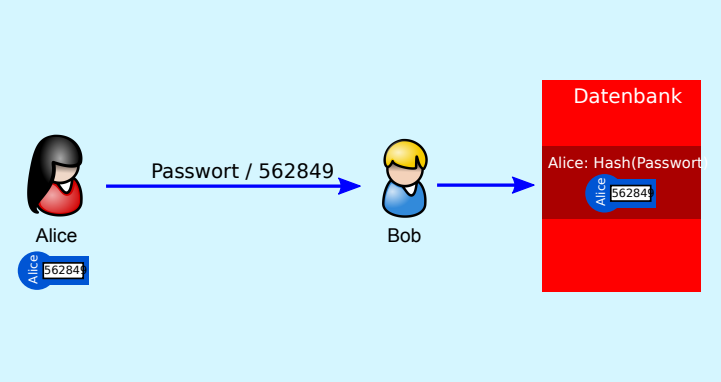
Zwei-Faktor-Authentifizierung – Ablauf (Forts.)



Schritt 2:

1. Bob übergibt Alice einen PIN-Generator.
2. Bob speichert das im PIN-Generator hinterlegte Geheimnis in der Datenbank.

Zwei-Faktor-Authentifizierung – Ablauf (Forts.)



Schritt 3:

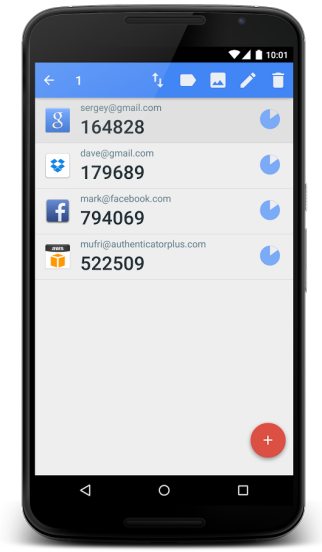
1. Alice authentisiert sich gegenüber Bob mit ihrem Passwort und der generierten PIN.
2. Bob überprüft die Daten und gewährt Alice bei positiver Prüfung Zugang zum System.

Gängige Verfahren

Die IETF hat folgende Verfahren standardisiert:

- HMAC-Based One-Time Password Algorithm (HOTP)
  - Die PIN wird in Abhängigkeit von einem Geheimnis und einem Zähler generiert.
  - Der Zähler wird bei jeder Nutzung inkrementiert.
  - Referenz: [RFC-4226]
- Time-Based One-Time Password Algorithm (TOTP)
  - Die PIN wird in Abhängigkeit von einem Geheimnis und der aktuellen Uhrzeit generiert.
  - Die Uhrzeit auf dem Generator und dem Server muss synchron sein.
  - Referenz: [RFC-6238]

Beispiele für Zwei-Faktor-Authentifizierung



Google Authenticator

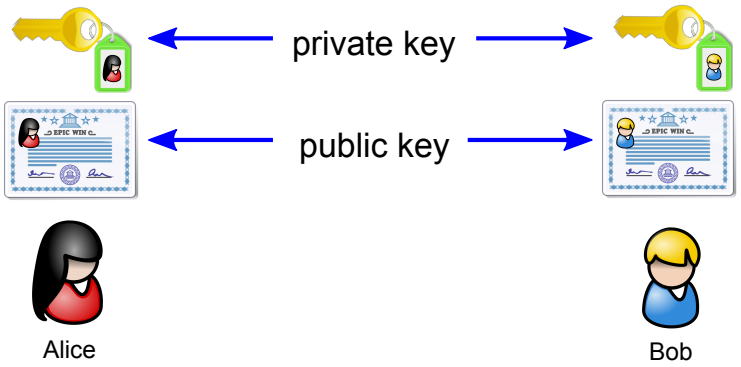
RSA SecurID

Yubikey

Public Key Infrastrukturen

- In der Public Key Kryptographie kommen öffentliche Schlüssel zum Einsatz, die oft über eine Webpage oder einen Verzeichnisdienst bereit gestellt werden.
- Ein wichtiger Aspekt bei der Nutzung der öffentlichen Schlüssel ist deren Authentizität.
- Public Key Infrastrukturen stellen Verfahren zur Echtheitsprüfung dieser öffentlichen Schlüssel bereit.

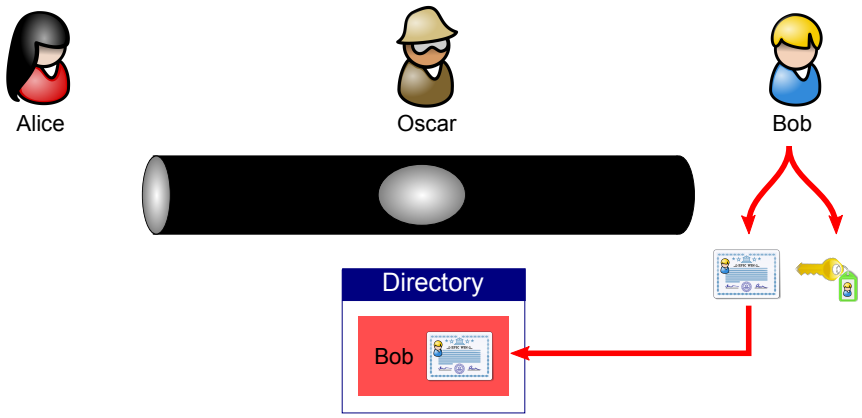
Public Key Kryptosystem – Idee



Szenario:

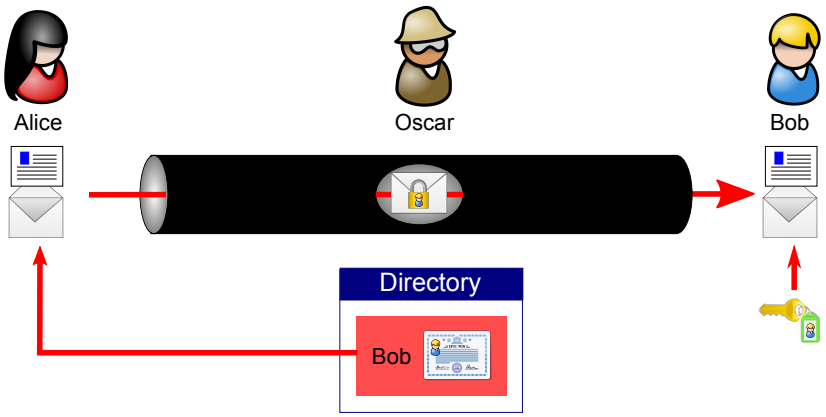
- Jeder Nutzer besitzt ein Paar bestehend aus einem privaten und einem öffentlichen Schlüssel.
- Der private Schlüssel muss geheim gehalten werden, der öffentliche Schlüssel kann auf geeignete Weise publiziert werden.
- Es ist sehr aufwändig, den privaten Schlüssel anhand des öffentlichen Schlüssels zu berechnen.

Nutzung eines Public Key Kryptosystems



Schritt 1: Bob generiert sein Schlüsselpaar und veröffentlicht seinen öffentlichen Schlüssel in einem Verzeichnis.

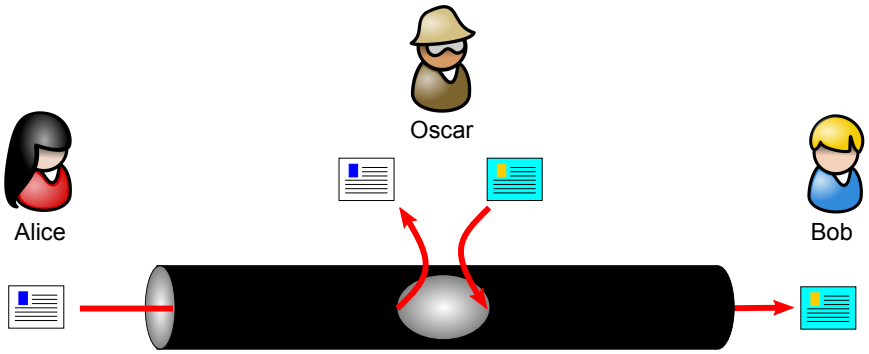
Nutzung eines Public Key Kryptosystems (Forts.)



Schritt 2:

- Alice nutzt Bob's öffentlichen Schlüssel zur Verschlüsselung der Nachricht.
- Bob nutzt seinen privaten Schlüssel, um die Nachricht zu entschlüsseln.

Digitale Signaturen



Herausforderung: Wie kann Bob überprüfen, ob eine empfangene Nachricht

- von Alice versendet wurde und
- von Oskar nicht verändert wurde?

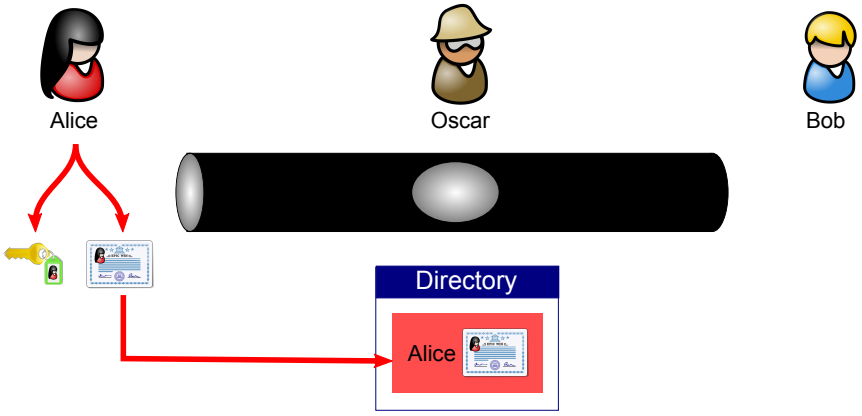
Anforderungen für digitale Signaturen

Ziel: Bereitstellung eines Mechanismus um die Authentizität eines Dokuments oder einer Nachricht zu überprüfen.

Anforderungen:

- Eine digitale Signatur muss fälschungssicher sein.
- Die Echtheit einer Signatur muss effizient überprüfbar sein.
- Die Modifikation eines signierten Dokuments muss effizient erkennbar sein.
- Eine Signatur darf nicht von einem Dokument auf das andere übertragbar sein.

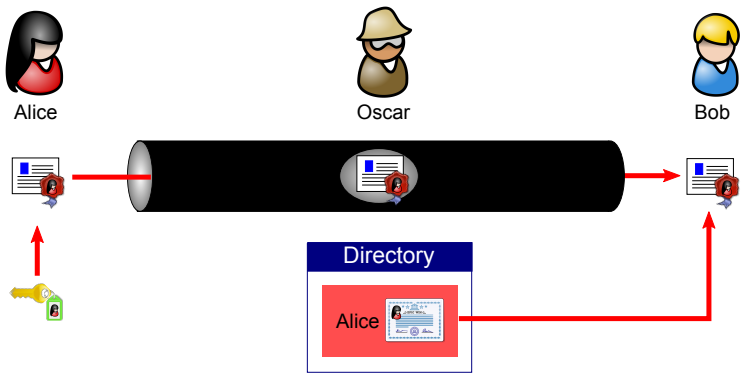
Einsatz einer digitalen Signatur



Schritt 1:

- Alice erstellt ein Paar bestehend aus einem privaten und öffentlichen Schlüssel.
- Alice veröffentlicht ihren öffentlichen Schlüssel in einem Verzeichnis.

Einsatz einer digitalen Signatur (Forts.)



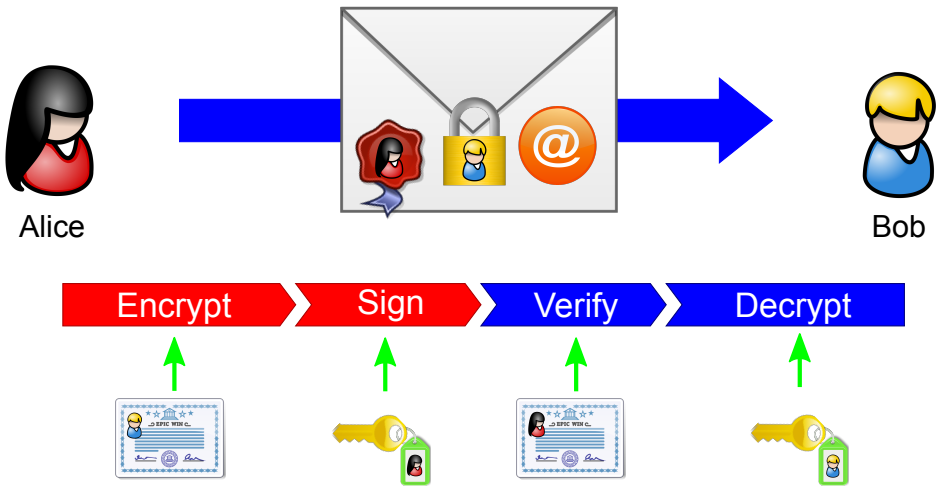
Schritt 2:

- Alice signiert mit ihrem privaten Schlüssel die Nachricht.
- Alice sendet die Nachricht mit der Signatur zu Bob..
- Bob nutzt Alice's öffentlichen Schlüssel, um die Signatur zu überprüfen.

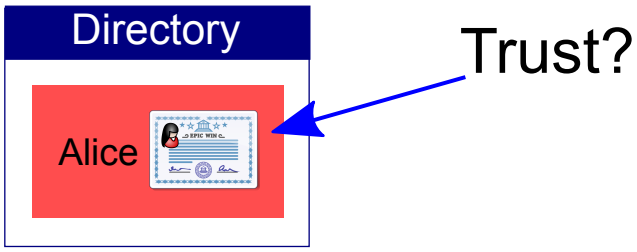
Kombination von Verschlüsselung und Signatur

Frage: In which order shall encryption and signature be applied?

Antwort:



Die Public Key Challenge



Herausforderung: Wie kann man überprüfen, ob das Zertifikat einer Person gültig ist?

Aufgaben einer Public Key Infrastruktur (PKI)

- Erstellen von Zuordnungen von Identitäten zu Public Key Schlüsseln
- Bereitstellung von Mechanismen zur
  - Erzeugung von Schlüsseln
  - Speicherung von Schlüsseln
  - Verifikation der Echtheit von Schlüsseln
  - Sperrung von Schlüsseln

Zwei Ansätze zur Echtheitsprüfung

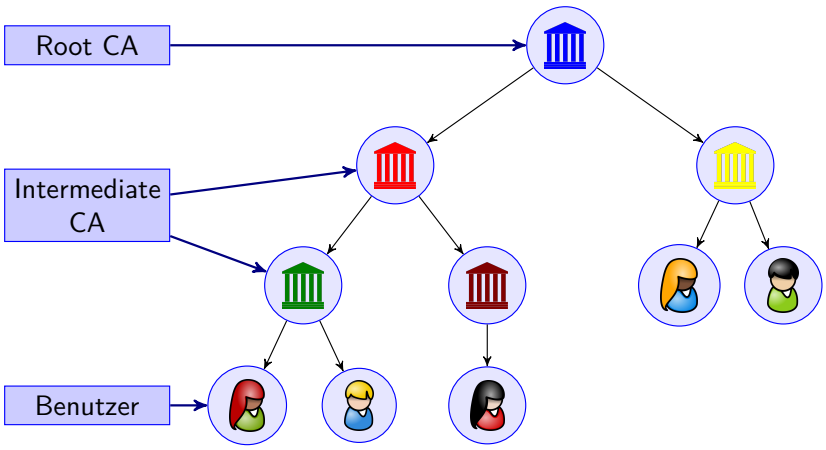
Hierarchische PKI

- Zertifikate auf Basis des X509 Standards
- Hierarchische Struktur für die Prüfung der Echtheit von Zertifikaten
- Beglaubigung der Zertifikate durch eine Certificate Authority (CA)

Web Of Trust

- Pretty Good Privacy (PGP)/GNU Privacy Guard (GPG)
- Gegenseitige Bestätigung der Echtheit der Zertifikate durch die Nutzer
- Qualität eines Zertifikats abhängig vom Vertrauen in die unterzeichnenden Nutzer

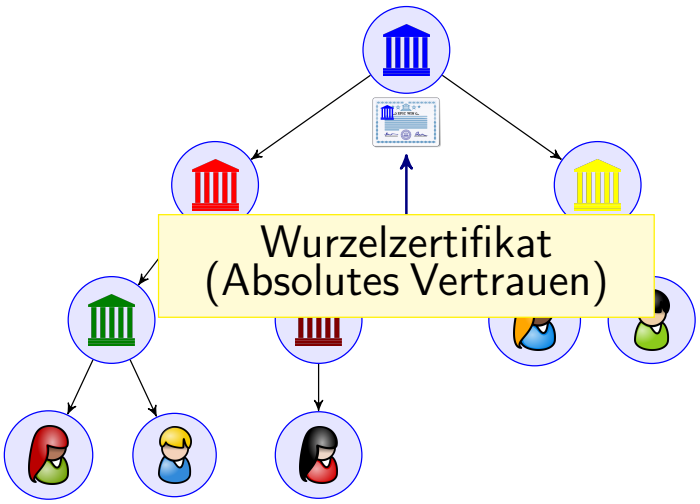
Hierarchische PKI



Entitäten:

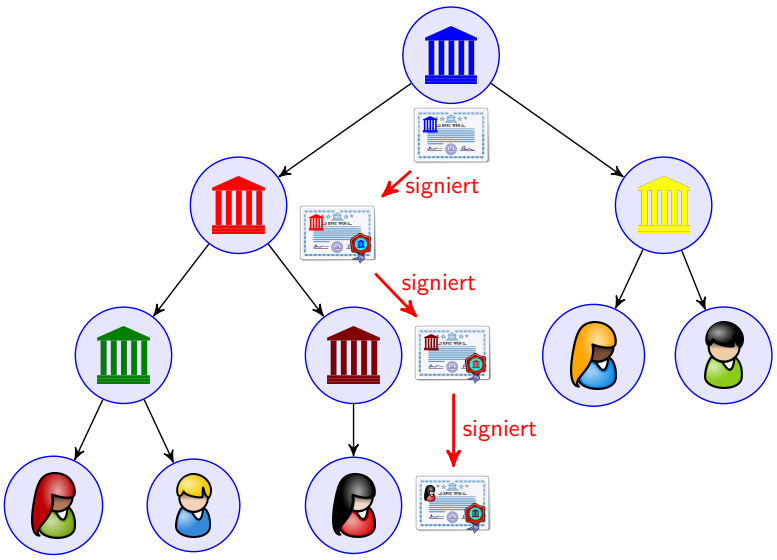
- Die Certificate Authority (CA) $\rightsquigarrow$ authentisiert Benutzer und stellt Zertifikate aus.
- Der Benutzer $\rightsquigarrow$ beantragt und benutzt Zertifikate.

Hierarchische PKI (Forts.)



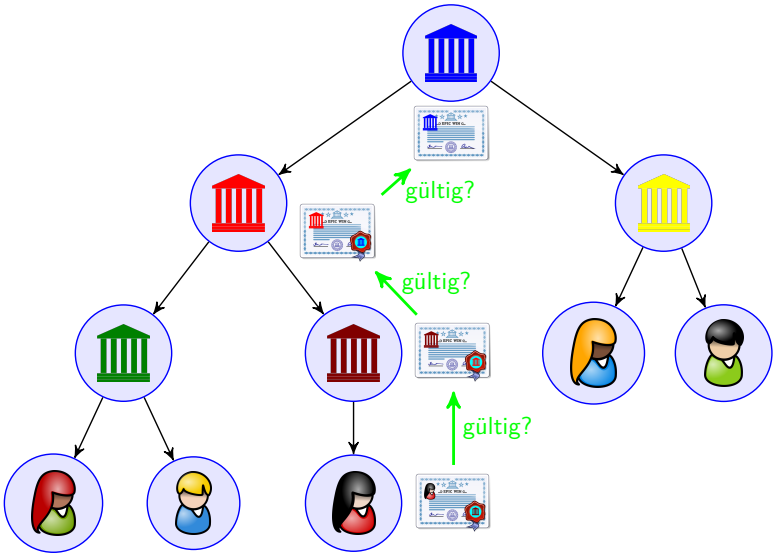
Das Zertifikat der Root CA besitzt absolutes Vertrauen.

Hierarchische PKI (Forts.)



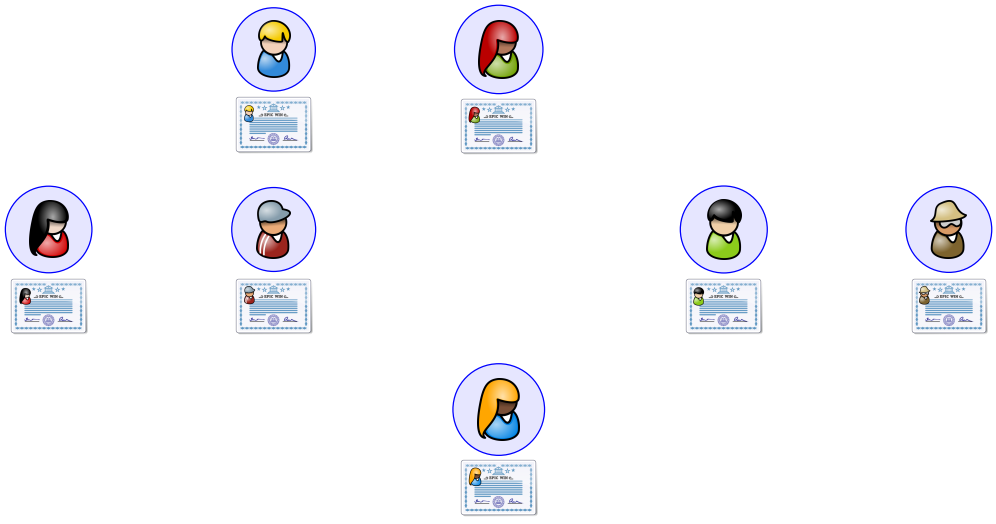
Die Zertifikate entlang des Pfades von der Root CA zu einem Benutzer bilden eine Zertifikatskette.

Hierarchische PKI (Forts.)



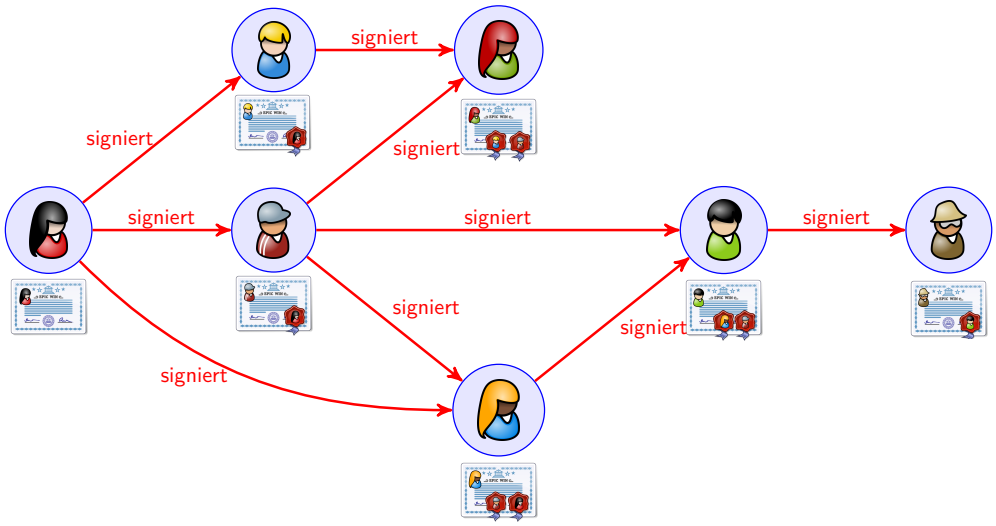
Die Zertifikatskette wird zur Echtheitsprüfung eines Benutzers eingesetzt.

Web of Trust



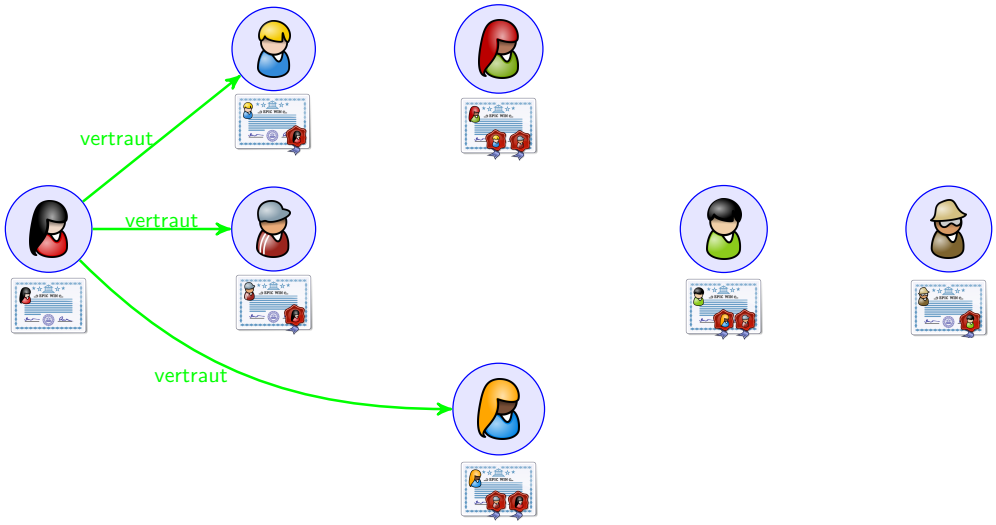
Im Web of Trust gibt es keine CA's.

Web of Trust (Forts.)



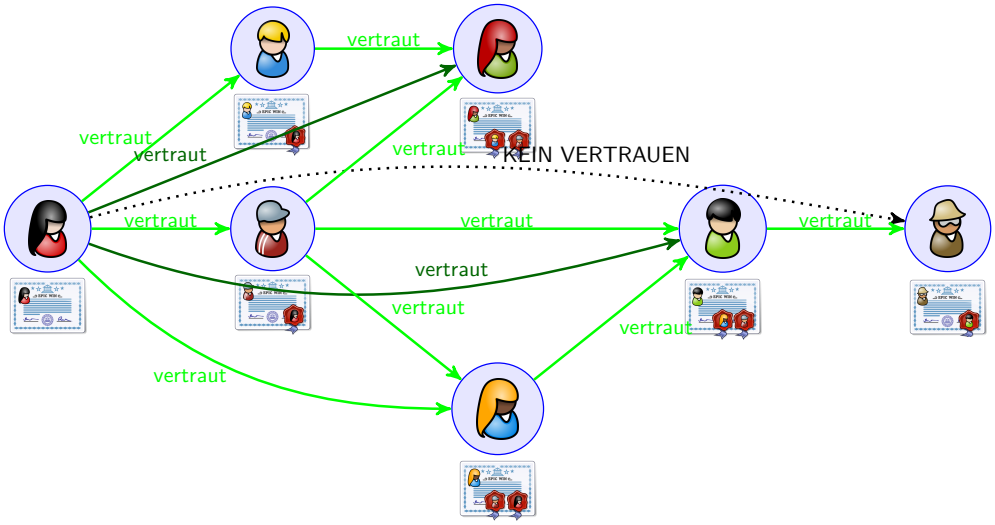
Die Nutzer prüfen gegenseitig deren Echtheit und signieren die öffentlichen Schlüssel der Personen, denen sie vertrauen.

Web of Trust (Forts.)



Alice vertraut allen öffentlichen Schlüsseln, die sie signiert hat.

Web of Trust (Forts.)



Alice vertraut einem öffentlichen Schlüssel, wenn dieser von zwei Personen signiert wurde, denen Alice vertraut.

Das X-509 Zertifikatsmodell

- Weitverbreitester Standard für Zertifikate
- Einsatz bei SSL/TLS, vielen VPN-Lösungen und vielen PKIs
- Teil des X.500 Standards, der vom International Telecommunications Union Telecommunications Standardization Sector (ITU-T) entwickelt wurde
- Einsatz eines hierarchischen Namenssystems für eine eindeutige Bezeichnung der Zertifikate
- Zahlreiche Anpassungen durch die IETF
- Aktuelle Version X.509v3 [RFC-5280; RFC-6818]

Ansatz

- Ausstellung der Zertifikate durch Certificate Authorities (CAs)
- Wichtige Bestandteile eines Zertifikats:
  - Seriennummer
  - Name der Entität
  - Name der CA
  - Öffentlicher Schlüssel der Entität
  - Gültigkeitsdauer
  - Einsatzgebiet des Zertifikats
- Hierarchische Anordnung von CAs
- Einsatz von Zertifikatsketten zur Abbildung der Hierarchien

Prozesse

Ausstellen eines Zertifikats:

1. Eine Entität generiert ein Paar bestehend aus einem privaten und öffentlichen Schlüssel und speichert diese in einem Zertifikatsantrag.
2. Die CA kontrolliert den Inhalt des Zertifikatsantrag.
3. Die CA signiert mit ihrem privaten Schlüssel den öffentlichen Schlüssel der Entität und erstellt anschließend das Zertifikat.

Überprüfen eines Zertifikats:

- Jeder kann die Echtheit des Zertifikats unter Einsatz des öffentlichen Schlüssels der CA überprüfen.

Erstellen einer PKI mit OpenSSL

- OpenSSL ist ein Open Source Werkzeug, dass verschiedene kryptografische Funktionen bereitstellt.
- OpenSSL kann für folgende Zwecke einsetzen:
  - Erzeugen von RSA, DH und DSA Schlüsseln
  - Erzeugen von X.509 Zertifikaten und Revokation Lists
  - Berechnung von kryptografischen Prüfsummen
  - Verschlüsselung von Dateien
  - Tests von SSL/TLS Übertragungen
  - Verarbeitung von mit S/MIME signierter oder verschlüsselter E-Mail

X.509 Zertifikate

- Spezifikation eines Datensatzes für ein digitales Zertifikat.
- Der Datensatz besteht aus Pflichtfeldern und optionalen Feldern.
- Formale Definition in der Abstract Syntax Notation (ASN.1).
- Erzeugung der Binärdaten mit den Determined Encoding Rules (DER).
- Versionen:
  - X.509 v1/v2 $\rightsquigarrow$ veraltetes statisches Format
  - X.509 v3 $\rightsquigarrow$ aktuelles Format
- Aktueller Stand der Standardisierung: [RFC-5280; RFC-6818]

Datenfelder von X.509 v3

Version	Version des Standards, auf dem das Zertifikat basiert
Serial Number	Bezüglich des Ausstellers eindeutige Nummer des Zertifikats
Signature Algorithm	Verfahren, welches zur Signatur des Zertifikats benutzt wurde
Issuer	Aussteller des Zertifikats
Validity	Gültigkeitsdauer des Zertifikats
Subject	Eigentümer des Zertifikats
Subject Public Key	Öffentlicher Schlüssel des Eigentümers
Extensions	Eine Liste mit weiteren Attributen
Signature	Digitale Signatur des Ausstellers über die gesamten Daten des Zertifikats

Zusammenfassung

- Authentisierung ist die Überprüfung der Echtheit eines Gegenstands, einer Person oder eines Computers.
- Man unterscheidet zwischen:
  - Authentisierung durch Wissen
  - Authentisierung durch Besitz
  - Authentisierung durch körperliche Eigenschaften
- Passwörter ist die am häufigsten eingesetzte Art der Authentisierung.
- Public Key Infrastrukturen ermöglichen die Echtheitsprüfung von digitalen Zertifikaten.

Literatur I

[ADC2014] Imperva Application Defense Center (ADC), Hrsg. *Consumer Password Worst Practices*. Imperva. 2014. URL: http://www.imperva.com/docs/WP_Consumer_Password_Worst_Practices.pdf (besucht am 11. 04. 2025).

[Cub09] Nik Cubrilovic. *RockYou Hack: From Bad To Worse*. Techcrunch. 14. Dez. 2009. URL: https://techcrunch.com/2009/12/14/rockyou-hack-security-myspace-facebook-passwords/ (besucht am 11. 04. 2025).

[Den82] Dorothy Denning. *Cryptography And Data Security*. Addison-Wesley, 1982.

Literatur II

[Hel80] Martin E. Hellman. "A Cryptanalytic Time-Memory Trade-Off". In: *IEEE Transactions on Information Theory* 26.4 (1980), S. 401–406.

[Oec03] Philippe Oechslin. "Making a Faster Cryptanalytic Time-Memory Trade-Off". In: *Advances in Cryptology - CRYPTO 2003: 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003. Proceedings*. Hrsg. von Dan Boneh. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, S. 617–630. DOI: 10.1007/978-3-540-45146-4_36. URL: http://dx.doi.org/10.1007/978-3-540-45146-4_36.

Literatur III

[Ras12] Fahmida Y. Rashid. *IEEE Exposed 100k Plaintext Usernames, Passwords on FTP Server*. SECURITYWEEK. 26. Sep. 2012. URL: https://www.securityweek.com/ieee-exposed-100k-plaintext-usernames-passwords-ftp-server (besucht am 11. 04. 2025).

[Rei17] Arnold G. Reinhold. *The Diceware Passphrase Home Page*. 2017. URL: http://world.std.com/~reinhold/diceware.html (besucht am 11. 04. 2025).

Literatur IV

[RFC-4226] D. M'Raihi u. a. *HOTP: An HMAC-Based One-Time Password Algorithm*. RFC 4226. Internet Engineering Task Force (IETF), 2005. URL: https://tools.ietf.org/html/rfc4226.

[RFC-5280] D. Cooper u. a. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. RFC 5280. Internet Engineering Task Force (IETF), 2008. URL: https://tools.ietf.org/html/rfc5280.

[RFC-6238] D. M'Raihi u. a. *TOTP: Time-Based One-Time Password Algorithm*. RFC 6238. Internet Engineering Task Force (IETF), 2011. URL: https://tools.ietf.org/html/rfc6238.

Literatur V

[RFC-6818] P. Yee. *Updates to the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. RFC 6818. Internet Engineering Task Force (IETF), 2013. URL: https://tools.ietf.org/html/rfc6818.

[Sch08] Bruce Schneier. *Passwords Are Not Broken, but How We Choose them Sure Is*. Schneier on Security. 13. Nov. 2008. URL: https://www.schneier.com/essays/archives/2008/11/passwords_are_not_br.html (besucht am 11. 04. 2025).