

Verschlüsselung im Internet

Christian Bockermann

Überblick

- Kryptographie
 - Was ist das? Warum braucht man das?
 - Wie funktioniert das?
 - Beispiele (Rucksäcke, RSA)
- Anwendungen im Internet
 - Verschlüsselung im Internet
 - Digitale Signaturen

Kryptographie

- *Kryptographie* bezeichnet die Wissenschaft, die sich mit der Verschlüsselung von Nachrichten befasst
- *Kryptanalyse* ist die Kunst, verschlüsselte Nachrichten zu knacken
- *Kryptologie* ist ein Zweig der Mathematik, der sich mit Kryptographie und Kryptanalyse befasst

Kryptographie - Warum?

- **Vertraulichkeit**
 - Nur autorisierte Personen dürfen eine lesen
- **Integrität**
 - Nur autorisierte Personen dürfen schreiben
- **Zurechenbarkeit**
 - kein Verschleiern, kein Abstreiten
- **Anonymität**
 - keine Verfolgbarkeit
- **Identifikation/Authentisierung**

Kryptographie - Wie?

- Zum Schutz von Nachrichten entwickeln Kryptographen Verfahren, mit denen aus Klartext ein Chiffre-Text wird
- Den Chiffre-Text kann nur derjenige zurückwandeln, der entweder
 - das geheime Verfahren kennt oder
 - den benötigten Schlüssel besitzt

Kryptographie - Wie?

Beispiel: geheimes Verfahren - geheimer Schlüssel



Kryptographie - Wie?

- Zusätzlich unterscheidet man *symmetrische* und *asymmetrische* Verfahren
- *symmetrische* Algorithmen benutzen zum Ver- und Entschlüsseln den gleichen Schlüssel K
- *asymmetrische* Verfahren nutzen unterschiedliche Schlüssel - einen zum Verschlüsseln und einen zum Entschlüsseln

Symmetrische Verfahren

Substitutions-Chiffrierung

- Jedes Zeichen in M wird durch entsprechendes Zeichen in C ersetzt
- Cäsar-Chiffre: Jedes Zeichen wird im Alphabet um 3 Stellen verschoben ($A \rightsquigarrow D, B \rightsquigarrow E, C \rightsquigarrow F, \dots$)
- ROT_{13} in Usenet-News, Rotation der Buchstaben um 13 Stellen im Alphabet, dadurch

$$M = ROT_{13}(ROT_{13})(M)$$

Symmetrische Verfahren

Transpositions-Chiffrierung

- Zeichen bleiben erhalten - lediglich die Zeichen-Reihenfolge wird durcheinandergewürfelt
- Beispiel: $M =$ PING IST EIN COOLER VEREIN!

PINGIS

TEINCO PTOR IELE NIEI GNRN ICV! SOE

OLERVE

REIN!

Symmetrische Verfahren

Wie substituiert oder transponiert ein Computer?

Computer arbeiten auf Bit-Folgen, Operationen sind z.B.

- Bitweise Substitution mit XOR
- Transposition z.B. mit Shiften

Symmetrische Verfahren

Substitution mit XOR

$$0 \oplus 0 = 0$$

$$0 \oplus 1 = 1$$

$$1 \oplus 0 = 1$$

$$1 \oplus 1 = 0$$

Dabei gilt vor allem:

$$a \oplus a = 0$$

$$a \oplus \underbrace{b \oplus b}_{=0} = a$$

Symmetrische Verfahren

Substitution mit XOR

Um eine Nachricht M mit einem Schlüssel K zu verschlüsseln, wird M in Blöcke M_i der Länge $|K|$ aufgeteilt

Der Chiffre-Text C ergibt sich dann aus den Blöcken C_i :

$$M_i \oplus K = C_i$$

Zum Entschlüsseln lassen sich auf gleiche Weise die Klartext-Blöcke aus den C_i berechnen:

$$C_i \oplus K = M_i$$

Symmetrische Verfahren

Permutation

Um Bits zu Permutieren, verwenden Computer z.B. Tabellen:

58	50	42	34	26	18	10	2	...
62	54	46	38	30	22	14	6	...

Symmetrische Verfahren

DES - Data Encryption Standard

- 1973: öffentliche Ausschreibung zu einheitlichem Verschlüsselungsalgorithmus
- Kein hinreichend guter Algorithmus als Vorschlag eingereicht
- 1974: erneute Ausschreibung durch NBS (Nation Bureau of Standards)
- IBM schlug selbstentwickelten Algorithmus vor
- 1976: nach Prüfung (u.a. durch NSA) wurde der DES als Standard anerkannt

Symmetrische Verfahren

DES - Data Encryption Standard

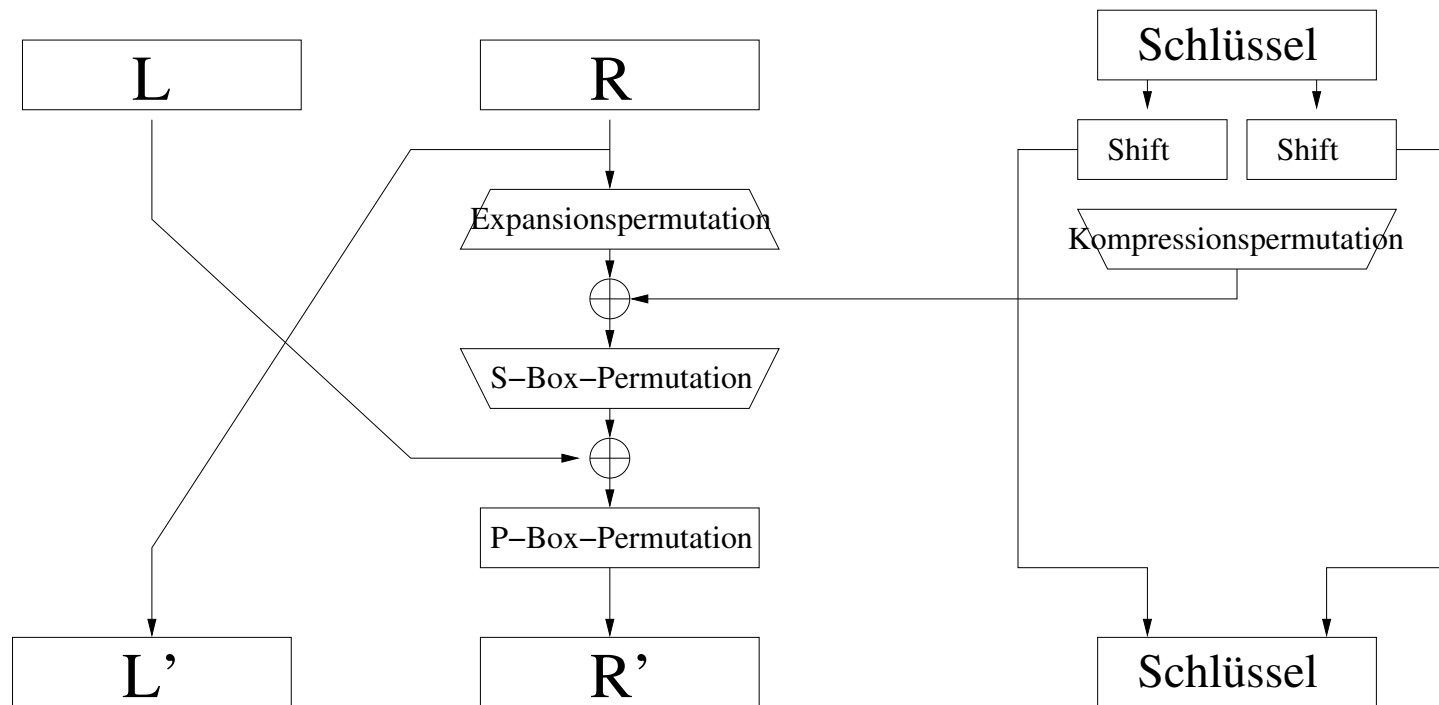
- DES zerlegt Klartext in 64-Bit Blöcke
- Auf jeden Block wird je eine vom Schlüssel abhängig Substitution und Permutation durchgeführt (= 1 Runde)
- Pro Block werden 16 Runden ausgeführt
- Sei B_i ein Block in i -ter Runde, L_i und R_i die Hälften, K_i ein 48-Bit langer Rundenschlüssel, Runde ist dann:

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$$

Symmetrische Verfahren

Symmetrische Algorithmen (Beispiel: DES)



Symmetrische Verfahren

- Es gibt unzählige Beispiele für symmetrische Verschlüsselungsverfahren: DES, Blowfish, AES, CAST, IDEA, RC2, RC4,...
- Viele bieten die Möglichkeit verschiedene Schlüssellängen zu Nutzen: 56Bit, 64Bit, 128Bit, ... (daher dann z.B. AES256)

Symmetrische Verfahren

Sicherheit

- Länge des Schlüssels = Grösse des Schlüsselraumes
- *Güte* des Schlüssels
- *Stärke* des Algorithmus

Symmetrische Verfahren

Schlüssel-Länge

- n -Bit Länge ergibt 2^n mögliche Schlüssel
- Brutforce-Angriff probiert alle möglichen Schlüssel aus
- Angenommen PC kann 1000000 Schlüssel/s testen

Länge (Bit)	Rechenzeit (Jahre)
64 Bit	585000 Jahre
128 Bit	$\sim 10^{25}$ Jahre
⋮	⋮
2048 Bit	$\sim 10^{597}$ Jahre für 1000000 parallele PCs

Symmetrische Verfahren

Güte des Schlüssels

- längster Schlüssel nützt nichts, wenn der Angreifer ihn ableiten oder erraten kann
- gleiches Problem wie bei *guten* Passwörtern
- oft lange Zufallszahlen als Schlüssel, dafür aber *guter Zufallsgenerator* notwendig (z.B. durch externe Rausch-Quelle)

Symmetrische Verfahren

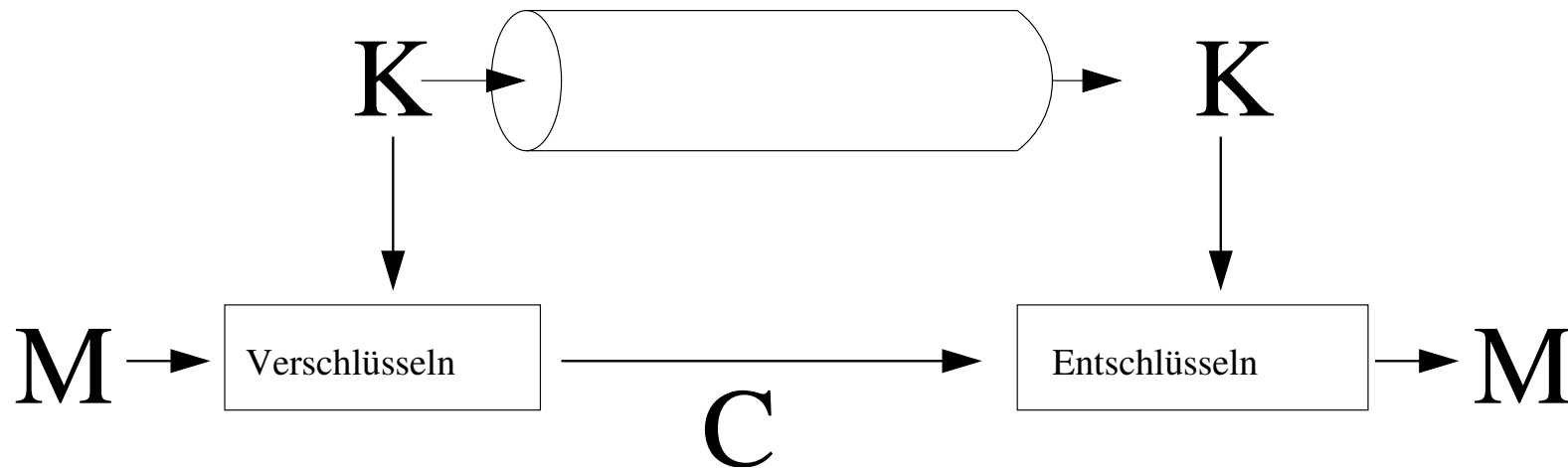
Schwächen des Algorithmus

- Längster Schlüssel nutzt nichts, wenn der Algorithmus nicht stark genug ist
- Es gibt oft *schwache Schlüssel*, bei denen der Algorithmus nicht genug permutiert
- Jeder Chiffre-Text C sollte für jeden Schlüssel K etwa gleich wahrscheinlich sein

Symmetrische Verfahren

Schlüssel-Austausch

Wie einigen sich Alice und Bob auf einen Schlüssel K ?



Symmetrische Verfahren

Schlüssel-Austausch

- Sicherheit symmetrischer Verfahren hängt von der Geheimhaltung des Schlüssels ab, allerdings muss dieser beiden Parteien bekannt sein
- Wenn ein Angreifer den Schlüssel abfängt, kann er die Kommunikation belauschen, gefälschte Nachrichten einspielen usw.
- Daher ist es wichtig *gute Schlüssel* zu wählen und diese *sicher zu übertragen*

Alternativen

- Um dem Problem des Schlüsselaustauschen aus dem Weg zu gehen, schlugen Diffie und Hellman 1976 Public-Key-Verfahren vor
- Public-Key-Verfahren verwenden ein Schlüssel-Paar (K_{priv}, K_{pub}) , mit K_{pub} verschlüsselte Daten können mit K_{priv} entschlüsselt werden
- Im folgenden ein kleiner Einblick in die Welt der Public-Key-Verschlüsselung

Public-Key - Prinzip

- Es gibt eine Funktion (Verschlüsselung)

$$f : X \rightarrow Y$$

deren Berechnung $y = f(x)$ max. polynomielle Zeit benötigt

- Berechnung der Umkehrfunktion f^{-1} möglichst schwierig (nicht-deterministisch polynomiell)
- Mit Zusatz-Information k lässt sich f^{-1} schneller berechnen (Entschlüsselung):

$$y = f(x), \quad x = f_k^{-1}(y)$$

Public-Key - Historie

- Merkle-Hellman: Rucksack-Problem
1972 patentiert, '82 gebrochen, '85 iteriert gebrochen
- RSA (Rivest, Shamir, Adleman)
1977, bis 2000 patentiert, brechbar bei Modulus $< 2^{1024}$
- LUC
1981 von Müller und Nöbauer
- ElGamal
1984, diskretes Logarithmus-Problem
- DSS/DSA
ElGamal-Variante, aktuelle NIST-Empfehlung

Rucksack-Problem

Rucksack-Problem: Gegeben sind Gewichte $G_1, \dots, G_n \in \mathbb{N}$ und Summe $S \in \mathbb{N}$. Gesucht sind $b_i \in \{0, 1\}$ mit

$$S = b_1 G_1 + \dots + b_n G_n$$

Beispiel: Gewichte $\{1, 5, 7, 11, 14, 20\}$, Summe $S = 17$

$$1 + 5 + 11 = 17$$

also ist $(1, 1, 0, 1, 0, 0)$ eine Lösung für dieses Rucksack-Problem

Rucksack-Problem

- Das allgemeine Rucksack-Problem ist schwierig zu lösen (NP-vollständig)
- Das Rucksack-Problem ist leichter zu lösen, wenn die Gewichte, die *super-increasing*-Eigenschaft haben

Definition: G_1, \dots, G_n haben die *super-increasing*-Eigenschaft, wenn jedes G_i grösser ist, als die Summe seiner Vorgänger

$$\sum_{k=1}^{i-1} G_k < G_i \quad \forall i$$

Rucksack-Problem

Beispiel: Ein Rucksack mit *super-increasing*-Eigenschaft ist z.B. $\{2, 3, 6, 13, 27, 52\}$.

Sei $S = 49$, dann ist $(0, 1, 1, 1, 1, 0)$ eine Lösung:

$$G_6 = 52 > S' = 49 \Rightarrow b_6 = 0, S' = S'$$

$$G_5 = 27 \leq S' = 49 \Rightarrow b_5 = 1, S' = S' - 27$$

$$G_4 = 13 \leq S' = 22 \Rightarrow b_4 = 1, S' = S' - 13$$

$$G_3 = 6 \leq S' = 9 \Rightarrow b_3 = 1, S' = S' - 6$$

$$G_2 = 3 \leq S' = 49 \Rightarrow b_2 = 1, S' = S' - 3 = 0$$

Rucksack-Problem

Merkle und Hellman entwickelten ein Verfahren um aus einfachen Rucksack-Problemen, schwierige zu erzeugen:

$\{2, 3, 6, 13, 27, 52\}$ ist ein super-increasing Rucksack

Wähle n, m mit $n < m$ und berechne n^{-1} mit

$$nn^{-1} \equiv 1 \pmod{m}$$

Rucksack-Problem

Beispiel: Bei $n = 31$, $m = 105$ ergibt sich $n^{-1} = 61$ und es lässt sich folgender Rucksack berechnen

$$2 \cdot 31 \pmod{105} = 62$$

$$3 \cdot 31 \pmod{105} = 93$$

$$6 \cdot 31 \pmod{105} = 81$$

⋮

Damit ergibt sich ein *schweres* Rucksack-Problem mit den Gewichten $\{62, 93, 81, 88, 102, 37\}$

Rucksack-Problem

Verschlüsselung: Seien die Gewichte des schweren Rucksack-Problems $\{62, 93, 81, 88, 102, 37\}$ und der Klartext $M = 011000110101101110$

$$011000 = 93 + 81 = 174$$

$$110101 = 62 + 93 + 88 + 37 = 280$$

$$101110 = 62 + 81 + 88 + 102 = 333$$

Ergibt den Chiffre-Text $C = 174, 280, 333$.

Rucksack-Problem

Entschlüsselung: Mit den Gewichten des einfachen Rucksack-Problems $\{2, 3, 6, 13, 27, 52\}$ und $n^{-1} = 61$ lässt sich der Chiffre-Text $C = 174, 280, 333$ wieder entschlüsseln:

$$174 \cdot 61 \pmod{105} = 9 = 3 + 6 \rightsquigarrow 011000$$

$$280 \cdot 61 \pmod{105} = 70 = 2 + 3 + 13 + 52 \rightsquigarrow 110101$$

$$333 \cdot 61 \pmod{105} = 48 = 2 + 6 + 13 + 27 \rightsquigarrow 101110$$

Rucksack-Problem

Vorgehen: Bob will Alice eine Nachricht M schicken.

1. Alice veröffentlicht die schwierige Gewichts-Folge im Internet.
2. Bob benutzt die schwierige Gewichts-Folge um eine Nachricht an Alice zu verschlüsseln C .
3. Alice empfängt C und nutzt n^{-1} und die einfache Folge, um die Nachricht M zu erhalten.

Rucksack-Problem

- Shamir und Zippel fanden Schwachstelle in der Transformation des super-increasing-Rucksacks
- Krypto-Systeme auf Basis des Rucksack-Problems wurden mehrfach erweitert und oft geknackt
- Da die meisten Varianten schnell geknackt wurde, gelten auch die derzeit noch sicheren Rucksack-Systeme als nicht allzu vertrauenswürdig

RSA

RSA nutzt das Problem der Faktorisierung einer grossen Zahl $n = pq$, wobei p und q Primzahlen sind

Ein Schlüsselpaar (privat, öffentlich) besteht dabei aus Zahlen e (encrypt) und d (decrypt) mit der Eigenschaft

$$\begin{aligned} ed &\equiv 1 \pmod{(p-1)(q-1)} \\ \Leftrightarrow d &\equiv e^{-1} \pmod{(p-1)(q-1)} \end{aligned}$$

RSA

Verschlüsselung: Die Nachricht M wird in Blöcke m_i zerlegt und zur Verschlüsselung die Blöcke c_i des Chiffre-Texts C berechnet

$$c_i = m_i^e \pmod n$$

Entschlüsselung: Aus dem Chiffre-Text C mit den Blöcken c_i wird der Klartext M folgendermassen entschlüsselt:

$$m_i = c_i^d \pmod n$$

RSA

Das alles ist keine Magie, sondern funktioniert, da

$$\begin{aligned}c_i^d &\equiv (m_i^e)^d \pmod{n} &&\equiv m_i^{ed} \pmod{n} \\ &&&\equiv m_i^{k(p-1)(q-1)+1} \pmod{n} \\ &&&\equiv m_i m_i^{k(p-1)(q-1)} \pmod{n} \\ &&&\equiv m_i \cdot 1 \pmod{n} \\ &&&\equiv m_i \pmod{n}\end{aligned}$$

RSA - Beispiel

Seien $p = 47$ und $q = 71$ zufällig gewählte Primzahlen, dann

$$n = pq = 3337$$

d darf keine gemeinsamen Faktoren mit

$$(p - 1)(q - 1) = 46 \cdot 70 = 3220$$

haben. Wähle e zufällig, z.B. 79, dann

$$d = 79^{-1} \pmod{3220} = 1019$$

RSA - Beispiel

Bob verschlüsselt den Klartext $M = \text{PING}$ der als ASCII-Zeichenfolge den Werten 80, 73, 78, 71 entspricht mit $K_{Alice} = (e, n)$

$$80^{79} \pmod{3220} = c_1$$

$$73^{79} \pmod{3220} = c_2$$

$$78^{79} \pmod{3220} = c_3$$

$$71^{79} \pmod{3220} = c_4$$

Die Nachricht $C = c_1, c_2, c_3, c_4$ kann nun nur noch Alice mit dem geheimen Schlüssel $K_{Alice}^{-1} = (d)$ entziffern

RSA - Signaturen

Das RSA-Verfahren eignet sich auch, Dokumente digital zu signieren:

1. Alice möchte M digital signieren. Dazu verschlüsselt sie M mit ihrem *privaten Schlüssel* $K_{Alice}^{-1} = d$ zu C_M
2. Bob kennt den öffentlichen Schlüssel K_{Alice} und weiss, dass dieser zu Alice gehört.
3. Bob entschlüsselt C_M mit Alice' öffentlichem Schlüssel und weiss nun, dass M von Alice stammt, da nur sie C_M mit dem privaten Schlüssel erzeugen kann.

RSA - Probleme

Probleme

- Woher weiss Bob, dass K_{Alice} wirklich zu Alice gehört?
- Wie erkennt Bob, dass $E_{K_{Alice}^{-1}}(C_M)$ wirklich das ist, was Alice meinte und nicht irgendein Kauderwelsch?
- Ausserdem sind asymmetrische Verfahren meist um Faktor 1000 langsamer als symmetrische Verfahren

Zunächst zur Geschwindigkeit...

Hybride Systeme

- Public-Key-Verfahren sind sehr langsam, daher versucht man mit ihnen nur möglichst wenige Daten zu verschlüsseln
- Der Grossteil der Daten soll mit symmetrischen Verfahren verschlüsselt werden
- Es wird dazu ein sogenannter elektronischer Umschlag genutzt, bei dem beide Verfahren kombiniert werden

Hybride Systeme

1. Alice denkt sich eine lange Zahl K_S zufällig aus.
2. Alice verschlüsselt die Nachricht M mit einem symmetrischen Verfahren mit dem Schlüssel K_S .
3. Nun verschlüsselt Alice K_S mit Bob's öffentlichem Schlüssel K_{Bob} und sendet $(\{M\}_{K_S}, \{K_S\}_{K_{Bob}})$ an Bob.
4. Bob entschlüsselt zunächst $\{K_S\}_{K_{Bob}}$ mit seinem geheimen Schlüssel K_{Bob}^{-1} und erhält K_S .
5. Nun kann Bob mit dem Schlüssel K_S auch die eigentliche Nachricht $\{M\}_{K_S}$ entschlüsseln

Digitale Signaturen - Hash

Wie erkennt Bob, dass $Decrypt_{K_{Alice}}(C_M) = M'$ wirklich das M' ist, was Alice meinte und nicht irgendein Kauderwelsch?

Um Dokumente digital zu signieren, wird meist nicht das Dokument M verschlüsselt, sondern ein sogenannter Hash-Wert von M

Dazu wird eine (*Einweg-*) *Hash-Funktion*

$$h : \{0, 1\}^n \rightarrow \{0, 1\}^k, k \ll n$$

benutzt, die jedem Dokument eine möglichst eindeutige Zahl (meist konstanter Länge) zuordnet

Digitale Signaturen - Hash

Eigenschaften von (Einweg-) Hash-Funktionen

- Leicht zu berechnen, auch für grosse Daten-Mengen (z.B. ISO-Images)
- Es ist enorm schwierig, zwei Dokumente M, M' zu erzeugen mit

$$h(M) = h(M') \text{ und } M \neq M'$$

- Es ist sehr schwierig, aus $h(M)$ die ursprüngliche Nachricht M zu ermitteln (Einweg-Hash-Funktion)
- Bekannte Hash-Funktionen sind z.B. MD5, SHA1

Digitale Signaturen - Hash

Um nun M zu signieren, geht Alice wie folgt vor:

1. Alice berechnet $h(M) = h_A$ mit der Hash-Funktion h und verschlüsselt h_A mit ihrem *privaten Schlüssel* zu $\{h_A\}_{K_{Alice}^{-1}}$
2. Nun sendet Alice $(M, \{h_A\}_{K_{Alice}^{-1}})$ an Bob.
3. Bob entschlüsselt $\{h_A\}_{K_{Alice}^{-1}}$ und erhält h_A . Auch Bob berechnet $h_B = h(M)$
4. Ist $h_A = h_B$, kann Bob davon ausgehen, dass Alice M wirklich so geschrieben hat

Zertifikate

Ziel: Person \leftrightarrow öffentlicher Schlüssel

- Bob muss Alice öffentlichen Schlüssel sicher Alice zuordnen können
- Alice könnte Bob den Schlüssel z.B. am Telefon vorlesen
- Alternative: Sie holen sich Hilfe von Trent

Zertifikate

Der vertrauenswürdige Dritte

- Alice und Bob vertrauen Trent
- Beide wissen, dass K_T der öffentliche Schlüssel von Trent ist
- Trent kennt K_{Alice} und K_{Bob}
- Trent signiert $(Alice, K_{Alice})$, d.h. er berechnet $h(Alice, K_{Alice})$ und verschlüsselt das mit K_{Trent}^{-1} zu $\{h(Alice, K_{Alice})\}_{K_{Trent}^{-1}}$

Zertifikate

Der vertrauenswürdige Dritte

- Nun veröffentlicht Trent

$$(Alice, K_{Alice}, \{h(Alice, K_{Alice})\}_{K_{Trent}^{-1}})$$

- Bob kann nun mit K_{Trent} überprüfen, ob Alice wirklich den Schlüssel K_{Alice} hat
 - Bob berechnet $h_B = h(Alice, K_{Alice})$ und entschlüsselt $\{h(Alice, K_{Alice})\}_{K_{Trent}^{-1}}$ mit K_{Trent}
 - Nun vergleicht Bob h_B mit dem entschlüsselten $h(Alice, K_{Alice})$

Zertifikate

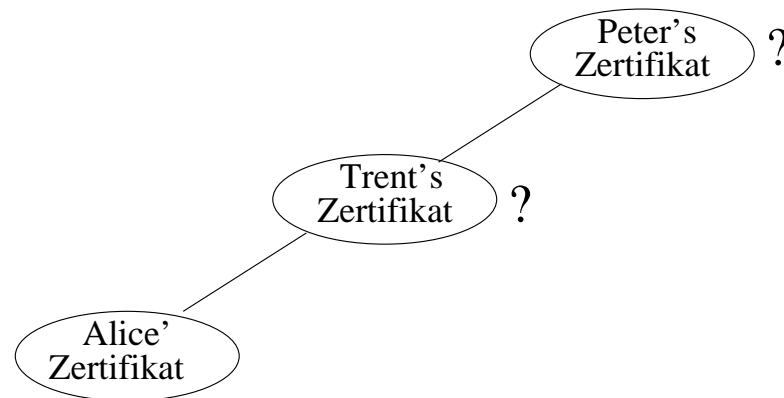
$(Alice, K_{Alice}, \{h(Alice, K_{Alice})\}_{K_{Trent}^{-1}})$ ist ein **Zertifikat**

- Zertifikate binden Identitäten (Alice) an öffentliche Schlüssel (K_{Alice})
- Der Aussteller (Trent) beglaubigt die Verbindung $Alice \leftrightarrow K_{Alice}$ mit seinem privaten Schlüssel
- Den Aussteller bezeichnet man auch als *Certificate Authority* (CA)

Zertifikate

Zertifikats-Pfade

- Um $Trent \leftrightarrow K_{Trent}$ zu verifizieren, müsste man ein Zertifikat haben, das für die Verbindung $(Trent, K_{Trent})$ bürgt, usw.
- Dadurch entstehen Zertifikats- oder Vertrauens-Pfade



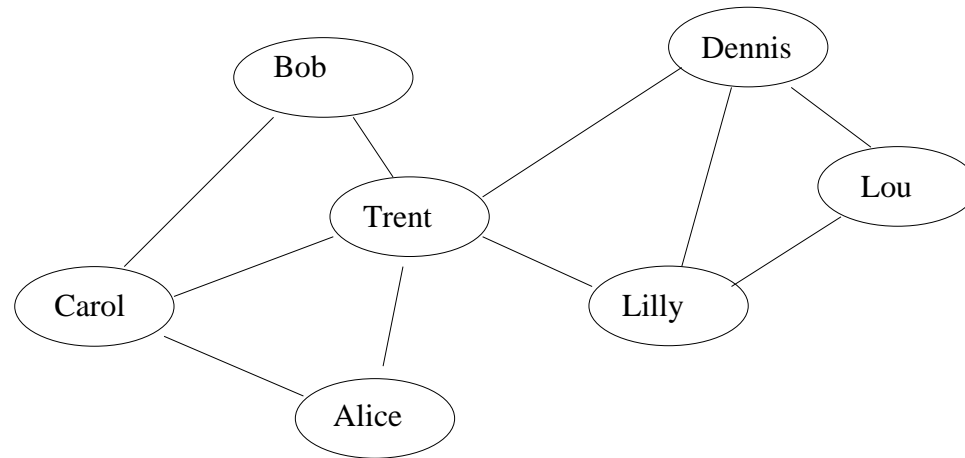
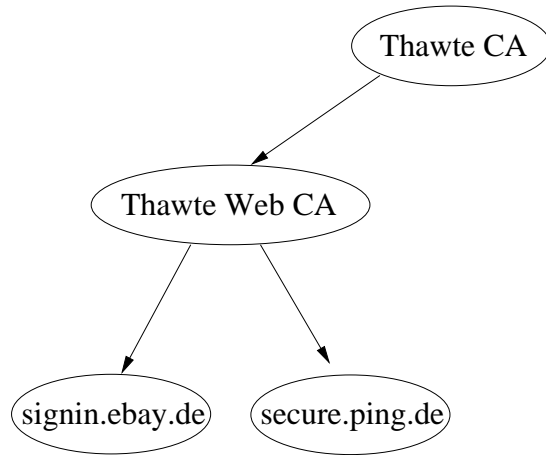
Zertifikate

Vertrauensmodelle

- Es gibt unterschiedliche Vertrauensmodelle, um Zertifikate, bzw. Zertifikats-Pfade zu verifizieren
 - Hierarchische Modelle (bei Behörden, Institute, usw.)
 - Netzwerke, *Web-of-Trust*, z.B. bei PGP

Zertifikate

Vertrauensmodelle



Zertifikate

Hierarchie - Wurzel-Zertifikate

- Die Vertrauenskette endet irgendwann an der Wurzel
- Die Wurzel-Zertifikate der gängigen Zertifizierungsstellen sind meist in der verifizierenden Software (Browser, Betriebssystem) integriert

Zertifikate

Was enthält ein Zertifikat?

- Identität die mit dem Schlüssel verbunden wird
- Natürlich den öffentlicher Schlüssel
- Namen des Ausstellers
- Signatur des Ausstellers
- "Haltbarkeits-Datum"
- Informationen über Algorithmen und Hash-Funktionen die verwendet wurden
- Eindeutige Serien-Nummer des Ausstellers

Zertifikate

X.509-Standard

- 1988 wurde erstmalig ein Standard für das Format von Zertifikaten vorgestellt (X.509)
- Der X.509-Standard wurde in mehreren Versionen weiterentwickelt bis zu derzeit X.509v3
- Die X.509-Zertifikate werden heute bei der Kommunikation über SSL (https), IPsec, SSL-VPN usw. verwendet

Zertifikate

X.509-Zertifikat als Text

Certificate:

Data:

Version: 3 (0x2)

Serial Number:

01:07:ad:36:b6:1e:7e:51:d4:91:39:a6:7a:c6:b7:e2

Signature Algorithm: sha1WithRSAEncryption

Issuer: C=GB, O=Comodo Limited, OU=Comodo Trust Network,
CN=Comodo Class 3 Security Services CA

Validity

Not Before: Dec 4 00:00:00 2003 GMT

Not After : Dec 3 23:59:59 2006 GMT

Subject: C=DE/postalCode=44227, ST=Nordrhein-Westfalen,
L=Dortmund/streetAddress=Emil-Figge-Str. 85,
O=Verein zur Foerderung der privaten Internet Nutzung e.V.,
OU=Systemadministration, OU=EliteSSL, CN=secure.ping.de

Subject Public Key Info:

Public Key Algorithm: rsaEncryption

RSA Public Key: (1024 bit)

Modulus (1024 bit):

00:bd:ad:39:0a:44:fc:11:d3:b1:08:a1:14:bc:81:

88:ea:86:1d:92:02:99:89:bd

Exponent: 65537 (0x10001)

Zertifikate

X.509-Zertifikat als Text

X509v3 extensions:

X509v3 Authority Key Identifier:

keyid:36:E0:E8:7C:6D:9D:45:91:EE:99:E5:42:76:4D:70:B3:50:30:AC:5E

X509v3 Subject Key Identifier:

B2:B6:35:FE:6D:2A:5C:A6:7F:A4:71:EE:1C:A0:20:62:65:39:09:44

X509v3 Key Usage: critical

Digital Signature, Key Encipherment

X509v3 Basic Constraints: critical

CA:FALSE

X509v3 Extended Key Usage:

TLS Web Server Authentication, TLS Web Client Authentication

X509v3 Certificate Policies:

Policy: 1.3.6.1.4.1.6449.1.2.1.3.4

CPS: <https://secure.comodo.net/CPS>

X509v3 CRL Distribution Points:

URI:http://crl.comodo.net/Class3SecurityServices_3.crl

URI:http://crl.comodoca.com/Class3SecurityServices_3.crl

email:Class3SecurityServices_3@crl.comodo.net

Netscape Cert Type:

SSL Client, SSL Server

Signature Algorithm: sha1WithRSAEncryption

95:04:e7:8e:c7:07:c8:16:83:82:85:10:db:4f:83:63:fd:8b:
c7:1d:36:85:e7:e6:71:33:18:a7:d9:d1:7a:7f:e7:71:20:95:
ef:06:b2:e4:86:c5:b4:62:92:db:62:5f:ee:ac:18:f5:6c:48:
60:61:06:d2:80:59:3e:a5:1e:33:96:5a:ed:15:19:8c:03:e1:
ce:75:be:2a:cb:73:1a:09:76:6d:f8:1d:18:1f:d8:28:17:ec:
9c:cf:76:2a:18:b5:0b:eb:0b:cd:8d:d4:7e:03:c5:6d:75:e3:
a1:b6:b4:b8:3c:df:a4:1f:98:31:90:1a:59:e8:3b:50:73:ff:

Zertifikate

Probleme

- Eve installiert einen Virus auf Alice' Laptop
- Der Virus klaut Alice' geheimen Schlüssel K_{Alice}^{-1} und sendet ihn an Eve
- Nun kann Eve sich als Alice ausgeben

Wie kann Trent Alice' Zertifikat für ungültig erklären?

Zertifikate

CRL - Certificate Revoke List

- Jeder Zertifikats-Aussteller verwaltet eine Liste Serien-Nummern von zurückgerufenen Zertifikaten
- Diese Liste signiert er mit seinem privaten Schlüssel und veröffentlicht sie
- Der X.509-Standard definiert auch ein Format für CRLs

Zertifikate

CRL als Text

Certificate Revocation List (CRL):

Version 2 (0x1)

Signature Algorithm: sha1WithRSAEncryption

Issuer: /C=GB/O=Comodo Limited/OU=Comodo Trust Network

OU=Terms and Conditions of use:

[http://www.comodo.net/repository/OU=\(c\)2002 Comodo Limited](http://www.comodo.net/repository/OU=(c)2002 Comodo Limited)

CN=Comodo Class 3 Security Services CA

Last Update: Sep 14 07:29:22 2006 GMT

Next Update: Sep 15 07:29:22 2006 GMT

CRL extensions:

X509v3 Authority Key Identifier:

keyid:36:E0:E8:7C:6D:9D:45:91:EE:99:E5:42:76:4D:70:B3:50:30:AC:5E

X509v3 CRL Number:

1434

Revoked Certificates:

Serial Number: 3E8048516F49E1D99A65C98EC381B994

Revocation Date: Sep 17 17:10:48 2003 GMT

Serial Number: 094907430E71F6539E2B27F873AD378C

Revocation Date: Sep 26 20:22:12 2003 GMT

.....

Serial Number: B470B4D5A1A4BFAC91E8CCA0FC844390

Revocation Date: Sep 12 17:22:16 2006 GMT

Signature Algorithm: sha1WithRSAEncryption

a6:77:9c:72:ad:8e:5d:3e:d0:c4:7f:e3:68:42:a1:c0:5d:d7:
0f:8f:1d:3e:11:e9:fe:ea:a4:a1:98:86:ac:0b:b1:bb:c7:93:
f0:0d:24:7e:f3:f7:7d:ff:7c:8b:ee:26:5d:ea:ab:d3:69:04:
fe:ec:e4:66:9e:32:70:fb:9b:de:03:dd:9a:b1:2a:8d:95:eb:
e0:31:96:3f:bb:44:98:05:8d:7e:c5:46:ea:f6:4f:ce:9b:a7:
e8:56:10:c8:05:fd:ed:72:13:8e:a7:cc:b2:41:7f:0a:57:6f:
f2:8d:7f:e4:1d:c0:7d:b0:5e:6f:59:7c:85:69:3b:7c:71:47:
04:0a:9d:05:45:ee:52:6f:e1:c9:42:bf:bd:ff:d1:2c:d6:56:
97:fa:68:7e:9a:34:6c:e2:90:78:d7:01:df:68:a7:ea:44:60:
bc:5b:f2:8a:80:6a:1e:b0:f5:f4:e7:71:b8:a2:74:dd:e4:32:
41:0c:72:e3:65:65:d9:46:60:67:64:36:ba:79:43:ee:fd:32:
11:37:a9:be:b7:1d:53:3d:00:c3:5f:49:e7:44:01:15:af:53:
d4:4f:cf:3b:9b:90:b4:7e:3b:ad:71:46:06:18:b7:ae:21:09:
0c:31:cf:42:06:23:b5:f0:eb:e5:03:e2:42:bb:a8:82:18:7e:
73:09:25:97

Zertifikate

Verifikation mit CRL

1. Bob schaut nach, wer Alice' Zertifikat ausgestellt hat:
Trent
2. Bob fragt Trent nach seiner *aktuellen* Rückruf-Liste
(CRL)
3. Bob überprüft, ob das Zertifikat *echt* ist (mit
Hash-Algorithmus usw.)
4. Nun muss Bob noch überprüfen, ob die
Zertifikats-Nummer nicht eventuell auf der
Rückruf-Liste steht