
Security

- LV 4120 und 7240 -

Kryptographische Protokolle und Anwendungen

Kap. 8: Kryptographische Protokolle und Anwendungen

Teil 1: Authentifikation und digitale Signatur

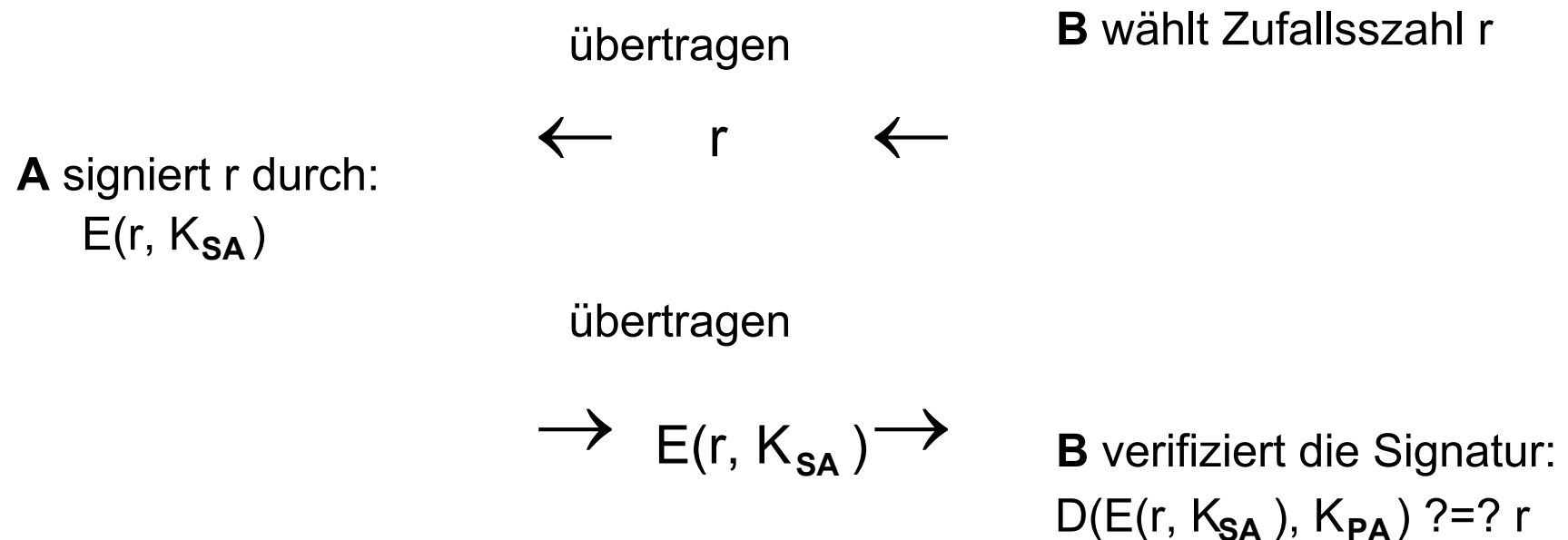
- Digitale Signaturen in der Praxis
- Authentifikation mit digitaler Signatur

Digitale Signaturen in der Praxis:

- Absicherung der gesamten Signaturkomponente durch ein möglichst langes Passwort (passphrase).
 - Mit Hilfe dieses Passwortes wird der geheime (private) Schlüssel symmetrisch verschlüsselt und gespeichert.
 - Der öffentliche Schlüssel (eines Kommunikationspartners) wird mittels eines Zertifikats gesichert.
 - Dieses **Zertifikat** trägt die digitale Signatur eines **Trustcenters** oder der sogenannten **Certification Authority (CA)**.
 - Im ersten Schritt gilt es nun mittels öffentlichen CA-Schlüssels das Zertifikat des Kommunikationsteilnehmers zu verifizieren.
-

Authentifikation mit digitaler Signatur:

Benutzer **A** unsicherer Kanal Server **B**




Kap. 8: Kryptographische Protokolle und Anwendungen

Teil 2: Public-Key-Infrastruktur

- Prüfung öffentlicher Schlüssel und Trustcenter
- Zertifikatshierarchie

Prüfung öffentlicher Schlüssel und Trustcenter:

X.509 Zertifikat von X 

Public Key von X

Seriennummer des Zertifikates

Gültigkeitszeitraum

Eindeutiger Name von X

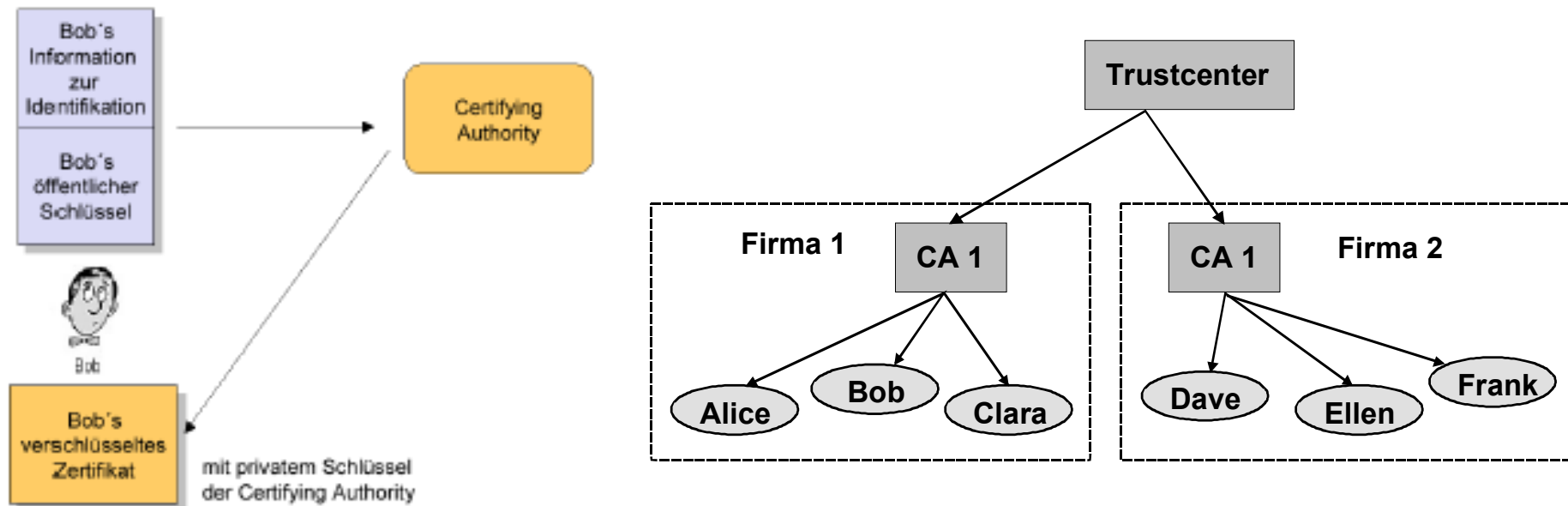
Eindeutiger Name der Zertifizierungsstelle

Digitale Unterschrift der Zertifizierungsstelle

Bestandteile eines Zertifikats:

- Version
- Seriennummer
- Algorithmus
- Aussteller des Zertifikats
- Geltungsdauer des Zertifikats
- Verwendungszweck
- Öffentlicher Schlüssel
- **Signatur des Ausstellers**

128-Bit-Schlüssel und zweistufige Zertifikatshierarchie:



Type	Bits/KeyID	Date	User ID
pub	1024/1C42BD1A	2010/05/29	Berhard Geib <B.Geib_ho@gmx.de>
Key fingerprint = E4 87 BC 23 A9 77 5D E2 E4 87 BC 23 A9 77 5D E2			

Kap. 8: Kryptographische Protokolle und Anwendungen

Teil 3: Secret Sharing und Secret Splitting

- Secret Sharing
- Secret Splitting

- Secret-Sharing-Verfahren wurden bereits 1979 von **Adi Shamir** zur **Aufteilung von geheimen Schlüsseln** eingeführt.
- Dabei wird ein Geheimnis auf eine Gruppe von n Personen so aufgeteilt, dass eine beliebige Teilgruppe von t Personen mit $t < n$ das Geheimnis rekonstruieren kann, $t - 1$ oder weniger jedoch nicht.
- Das Einrichten der sogenannten Shares (Teilgeheimnisse) erfolgt von einer vertrauenswürdigen Instanz, die auch Verteiler oder Dealer genannt wird.
- Die Rekonstruktion wird von einem Zusammensetzer oder Combiner ausgeführt, der im Namen der Teilgruppe das Geheimnis berechnet und allen t Gruppenmitgliedern mitteilt.
- Die Gruppe aller Teilnehmer sei durch $\{P_1, \dots, P_n\}$, $n \in \mathbf{N}$ gegeben.

(t, t)-Schwellenwertverfahren:

Geheimnis $k \in \mathbf{N}$ soll auf Teilnehmer $\{P_1, \dots, P_t\}$, $t \in \mathbf{N}$ verteilt werden.

Verteiler:

1. Der Verteiler wählt den Modulus $m \in \mathbf{N}$ mit $m > k$.
2. Der Verteiler wählt zufällig $t - 1$ Elemente $s_1, \dots, s_{t-1} \in \mathbf{Z}_m$ als Shares für die Teilnehmer P_1, \dots, P_{t-1} .
3. Der Verteiler berechnet dann das Share für den Teilnehmer P_t mit Hilfe von

$$s_t = (k - \sum_{i=1}^{t-1} s_i) \bmod m$$

4. Der Verteiler verteilt die Shares sicher an die Teilnehmer P_1, \dots, P_t .
-

Combiner:

5. Der Combiner erhält auf sicheren Wege die Shares s_1, \dots, s_t von den jeweiligen Teilnehmern P_1, \dots, P_t der Gruppe.
6. Der Combiner berechnet das Geheimnis k mit der Vorschrift

$$k = \left(\sum_{i=1}^t s_i \right) \bmod m$$

7. Der Combiner teilt das Geheimnis k allen Teilnehmern P_1, \dots, P_t mit.

Mit $t - 1$ oder weniger Teilnehmern kann k nicht berechnet werden, da für die fehlende s_i jede Zahl aus \mathbf{Z}_m denkbar ist. Das Verfahren ist somit **perfekt**.

- Secret-Splitting ist das Zerteilen einer Bitfolge (Nachricht, Dokument) in zwei oder ggf. mehrere Teile, die alle für sich allein betrachtet wertlos sind und keine Information über die Nachricht M enthalten.
- Fügt man die einzelnen Teile (sagen wir M_1 und M_2) aber zusammen, so ist die Rekonstruktion der Nachricht M möglich.
- Hat M die Länge n , so nimmt man eine n Bit lange Zufallszahl r und berechnet:

$$M_1 = r \oplus M \quad \text{und} \quad M_2 = r$$

- Ist r echt zufällig, so ist die Aufteilung absolut sicher, genau wie das **One-Time-Pad** und es gilt:

$$M = M_1 \oplus M_2$$

- Damit ein Spezialfall des allgemeinen (t, n) -Schwellenwertproblems.

Kap. 8: Kryptographische Protokolle und Anwendungen

Teil 4: Zero-Knowledge-Protokolle

- Challenge-and-Response-Verfahren
- Das **Fiat-Shamir**-Protokoll

Die Idee des Challenge-and-Response-Verfahrens:

- Das Protokoll des **Herausforderns und Antwortens** dient der **Benutzerauthentifikation**, die gewöhnlich aus einer **Identifikation** und einer sich anschließenden **Verifikation** besteht.
- Dabei wird eine zufällige **Anfrage** (die Challenge) durch eine zugehörige **Response** beantwortet, welche ein Geheimnis benutzt, ohne jedoch nur ein Bit an Information über das Geheimnis preiszugeben.
- Wir gehen davon aus, dass zwei Benutzer **A** und **B** einen gemeinsamen **geheimen** Schlüssel **k** besitzen und setzen voraus, dass sonst niemand diesen Schlüssel kennt.

Zur **Authentifizierung** von **B** gegenüber **A** dient dann folgendes Protokoll:

Zero-Knowledge-Protokoll

Challenge-and-Response

Benutzer A

A wählt zwei gleich lange
m-Bit Zufallszahlen s_1 und s_2
 $s := s_1 \parallel s_2 \in \{0, 1\}^{2m}$
und verschlüsselt s mit k .
 $r = E(s, k)$

unsicherer Kanal

übertragen
 $\rightarrow ID_A, r \rightarrow$

Benutzer B

B sucht in Datenbank zu
ID A gehörigen Schlüssel k
und entschlüsselt r mit k .
 $s = s_1 \parallel s_2 = D(r, k)$
B zerlegt s in gleich lange
 s_1 und s_2 .



A entschlüsselt R mit k
und erhält hieraus
 $(s_1 \oplus s_2) \parallel t = D(R, k)$.

A prüft den Wert $s_1 \oplus s_2$
auf Korrektheit.

übertragen
 $\leftarrow R \leftarrow$

B wählt eine m Bit
lange Zufallszahl $t \in \{0, 1\}^m$
und verschlüsselt
 $(s_1 \oplus s_2) \parallel t$ mit k .
 $R = E((s_1 \oplus s_2) \parallel t, k)$

Das Fiat-Shamir-Authentifikationsprotokoll:

Dieses von **A. Fiat**, **A. Shamir** und **U. Feige** entwickelte **Authentifikationsprotokoll** benutzt, wie bei der Signatur mit einem Public-Key-Verfahren, einen **geheimen** Schlüssel **s**.

Benutzer **A** mit dem **geheimen** Schlüssel **s** möchte **B** seine Identität beweisen, ohne jedoch **s** preisgeben zu müssen.

Vorbereitung:

- Ein vertrauenswürdiger Vermittlungsrechner bestimmt zwei zufällige Primzahlen p und q , deren Produkt den Modul n ergibt.
- Der **geheime** Schlüssel **s** von **A** wird zufällig gewählt.
- Der **öffentliche** Schlüssel **v** von **A** wird berechnet nach der Vorschrift $v = s^2 \bmod n$ und öffentlich bekannt gegeben.

Nun läuft folgendes **Protokoll** ab:

1. Benutzer **A** wählt Zufallszahl r und berechnet $x = r^2 \bmod n$ und schickt x an Benutzer **B**.
2. Benutzer **B** wählt zufällig ein Bit $b \in \{0, 1\}$ und schickt dies zum Benutzer **A**.
3. Falls $b = 1$ berechnet **A** den Wert $y = r \cdot s \bmod n$ und falls $b = 0$ den Wert $y = r \bmod n$. **A** sendet nun den Wert y an **B**.
4. Falls $b = 1$ verifiziert **B**, ob $y^2 \bmod n = x \cdot v \bmod n$ und falls $b = 0$ den Wert $y^2 \bmod n = x$.

Die **Sicherheit** des Verfahrens basiert auf der Schwierigkeit der Berechnung **modularer Quadratwurzeln**, was gleich schwierig ist wie die **Primfaktorzerlegung** von n .

Kap. 8: Kryptographische Protokolle und Anwendungen

Teil 5: Public-Key-Systeme

- PGP
- IPSec

Benutzer **A** unsicherer Kanal Benutzer **B**

A wählt zufälligen

Sitzungsschlüssel k ,

verschlüsselt mit k die
Nachricht M und

verschlüsselt mit K_{PB} den
Sitzungsschlüssel k .

$$C = E(M, k)$$

$$C_k = E(k, K_{PB})$$

übertragen

$$\rightarrow (C_k, C) \rightarrow$$

B entschlüsselt

$$k = D(C_k, K_{SB})$$

$$M = D(C, k)$$

Pretty good privacy

- **PGP** wurde 1994 von Phil Zimmermann entwickelt und ist ein verbreitetes Programm zur sicheren Kommunikation per E-Mail.
- Dateien können mit PGP verschlüsselt oder signiert und das Resultat dann per E-Mail verschickt werden.
- Die Versionen 2.6.x benutzen **IDEA** zum Verschlüsseln, RSA zum Schlüsseltausch und **MD5** als Einweg-Hash-Funktion.
- Digitale Signaturen werden mit **RSA** und **MD5** erzeugt.
- Ab Version 5.X.X wird der Schlüsselaustausch mit dem **Diffie-Hellman**-Verfahren realisiert und der **Digital Signature Algorithm (DSA)** zum Signieren verwendet.

IP Security:

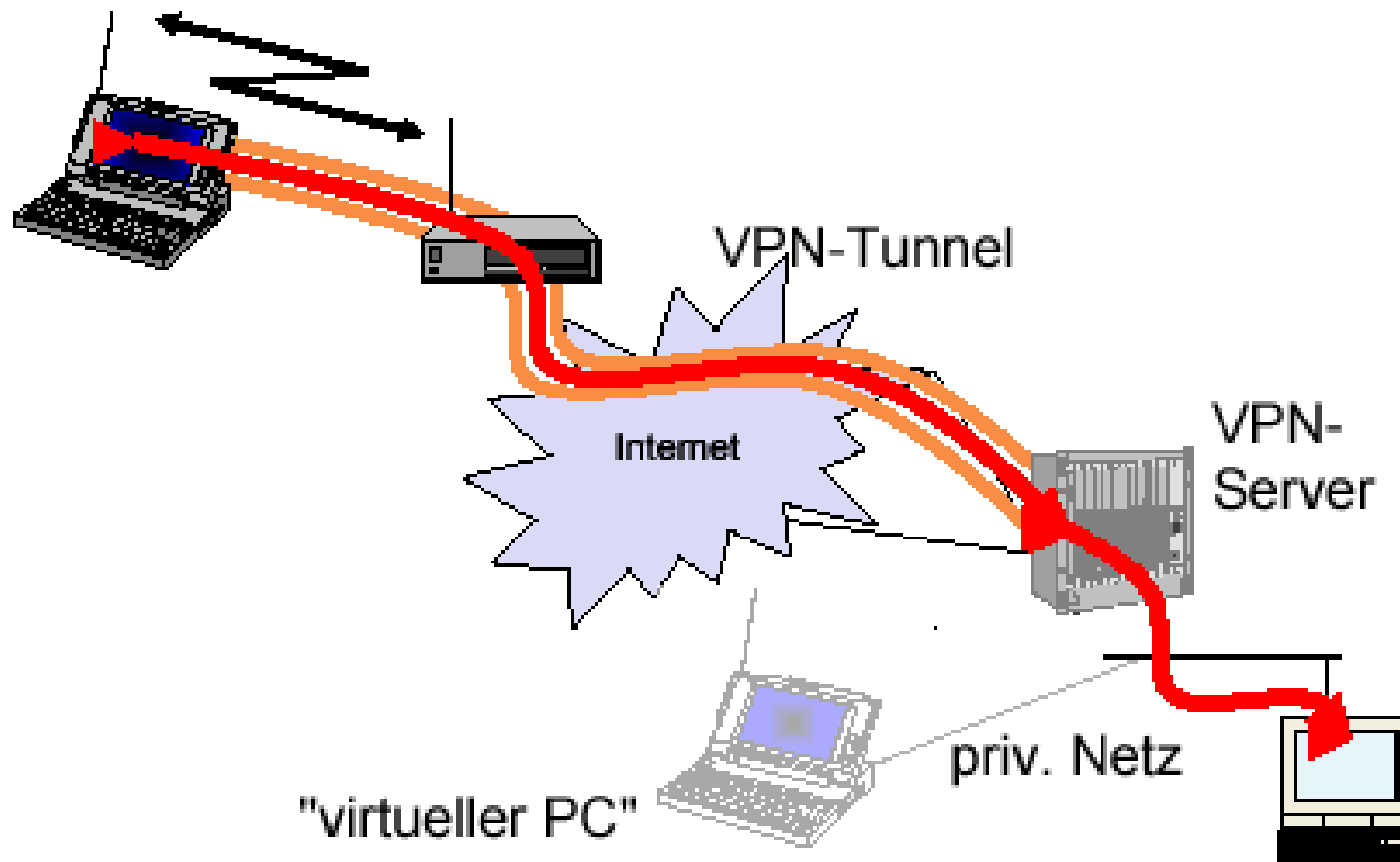
Für die Realisierung von **VPNs** und anderen sicheren Verbindungen wurde 1994 vom **Internet Architecture Board** mit **IP Security (IPSec)** ein mächtiges Protokoll zur Absicherung von **IP-Paketen** initiiert und bis heute weiterentwickelt. Die Verwendung von IPSec auf einer **Firewall** bietet folgende Vorteile:

- Starke kryptographische Sicherheit für den Verkehr nach draußen.
- IPSec auf einer Firewall kann nicht umgangen werden.
- Die Anwendersoftware muss nicht angepasst werden.
- Für den Benutzer ist die Verwendung von IPSec transparent.
- Verwendete Kryptoverfahren sind: DES und Triple-DES, CBC-Mode, RC5, IDEA, Blowfish, MD5, SHA-1 und Diffie-Hellmann.

Kap. 8: Kryptographische Protokolle und Anwendungen

Teil 6: Virtual Private Networking

- Realisierung einer Übertragungssicherheit
- SSH und SSL



Secure Shell (SSH):

Die wichtigste Funktionalität ab Version SSH2 ist:

- Sichere gegenseitige Authentifikation von Client und Server.
 - Sicheres Login durch Authentifikation der Benutzer auf dem Server.
 - Mittels RSA oder DSA wird die Identität des Benutzers geprüft.
 - Jede Kommunikation ist verschlüsselt.
 - Mit dem RSA-Algorithmus wird ein Sitzungsschlüssel vereinbart.
 - Mit dem Diffie-Hellmann-Verfahren erfolgt die Schlüsselvereinbarung.
 - Der DSA dient zur Erstellung von Signaturen.
 - Unter Verwendung des Sitzungsschlüssels erfolgt eine symmetrische Verschlüsselung mittels IDEA, Blowfish oder Triple-DES.
-

Secure socket layer (SSL):

Das **SSL-Protokoll** ist im OSI-Schichtenmodell zwischen der Transportschicht und der Anwendungsschicht eingebettet und dient als Grundlage für die Spezifikation der **Transport Layer Security (TLS)**.

SSL-Handshake-Protokoll	SSL-Change-Cipher-Protokoll	SSL-Alert-Protokoll	HTTP
SSL-Record-Protokoll			
TCP			
IP			

SSL-Record-Protokoll:

Kryptoalgorithmen wie MD5, SHA-1, RSA, DH, IDEA, DES und RC4.

SSL-Alert-Protokoll:

Warn- und Fehlermeldungen.

SSL-Change-Cipher-Protokoll:

Initialisierung der ausgewählten Algorithmen.

SSL-Handshake-Protokoll:

Schlüsselaustausch und Festlegung der kryptographischen Algorithmen.