

Netzicherheit – Architekturen und Protokolle PGP



1. PGP PKI
2. E-Mail Sicherheit



Netzicherheit – Architekturen und Protokolle PGP



1. PGP PKI
2. E-Mail Sicherheit



PGP: Pretty Good Privacy

- Entwickler: Phil R. Zimmermann
- IETF-Standard: *OpenPGP*
- PGP definiert Formate und Prozeduren für:
 - Digitale Signaturen
 - Verschlüsselung
 - Kompression
 - Radix-64-Umwandlung (binär Daten in druckbare ASCII-Zeichen)
 - Anwendungsspezifische Unterschlüssel



„PGP is for people who prefer to pack their own parachutes“
(P.R. Zimmermann)

3



Vertrauensmodell von PGP

- Vertrauensmodell Anarchie: *Web of Trust*
 - Vollständig dezentral
 - ▶ Verzicht auf zentrale Strukturen
 - Verantwortung beim Benutzer
 - ▶ Widerspiegelung sozialer Netze
 - ▶ Key Signing
 - ▶ Vertrauen auf Sorgfalt anderer Benutzer
 - Jeder Benutzer agiert als CA
 - ▶ *Trust Introducer (TI)* stellen Vertrauen in ID-Zertifikate her
 - ▶ *Meta Trust Introducer (MTI)* stellen transitiv Vertrauen in ID-Zertifikate her

4

- Zertifikate werden in einem **Schlüsselbund** (Public Key Ring) verwaltet
 - öffentlicher Schlüssel
 - User-ID
 - keine, eine oder mehrere Signaturen
- Folgende Felder definieren Vertrauen
 - **Owner Trust:** Vertrauen in einen Benutzer, Zertifikate mit zu unterschreiben
 - **Signature Trust:** Vertrauen in eine Signatur, die eine Zuordnung Public Key↔ID zertifiziert
 - ▶ **Key Legitimacy:** Vertrauen in Zuordnung Public Key↔ID
 - ▶ mehrere Signature Trust Felder zur Berechnung der Key Legitimacy

5

- Struktur

| Feld | Bedeutung |
|---|---|
| Public Key Packet | Primärer Schlüssel |
| [Signature Packet]* | Widerrufssignatur |
| [Signature Packet] | Selbstsignatur des Schlüssels |
| [User-Id Packet [Signature Packet]*]+ | User-IDs plus Signaturen, die diese zertifizieren |
| [Public Subkey Packet Signature Packet [Signature Packet]]* | Unterschlüssel <ul style="list-style-type: none"> • Bindung an primären Schlüssel • Widerrufssignatur |
| Trust Packet | Lokale Vertrauensparameter |

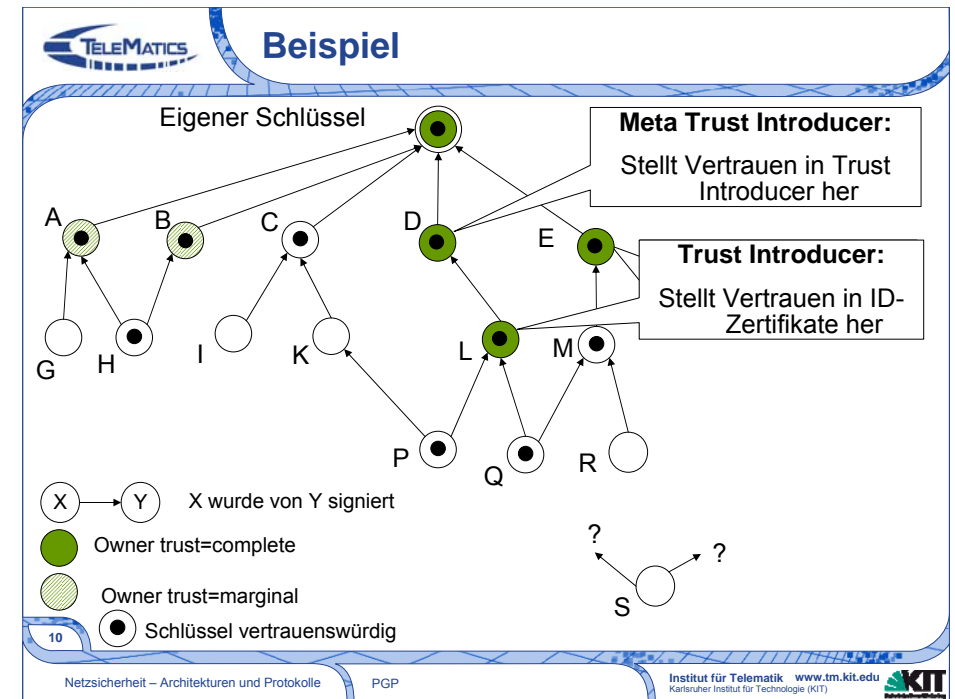
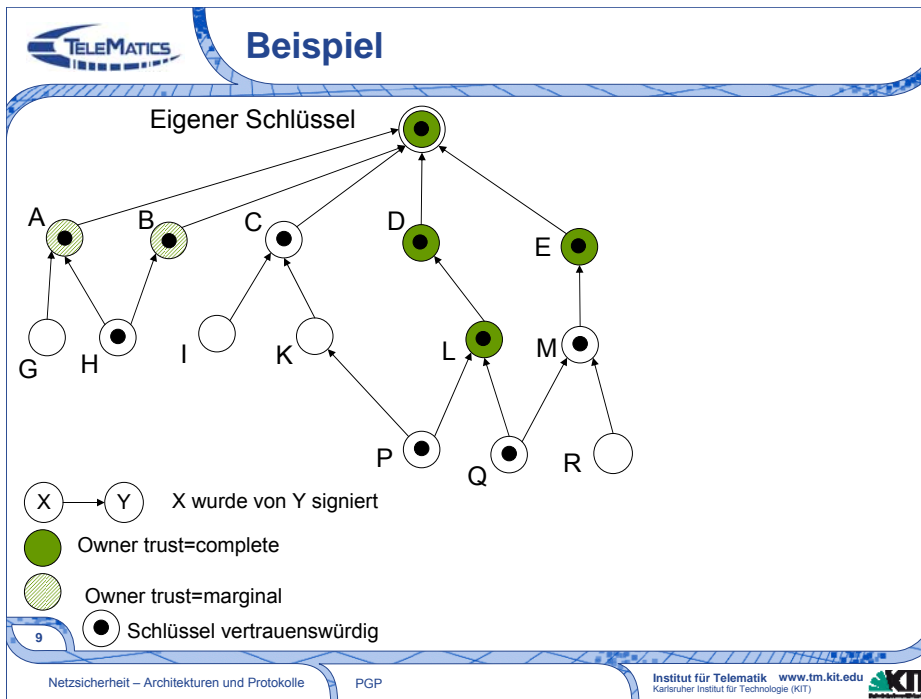
6

- **Owner Trust** und **Signature Trust**
 - unknown
 - untrusted
 - marginal
 - complete
- **Key Legitimacy**
 - valid
 - marginally valid
 - invalid

7

1. Benutzer **legt Owner Trust** fest
2. Für jede Signatur des neuen öffentlichen Schlüssels **durchsuche Public Key Ring**
 - a) falls Signierender bekannt:
Signature Trust=Owner Trust
 - b) falls Signierender nicht bekannt:
Signature Trust=unknown
3. **Berechne Key Legitimacy**
 - a) lokal definierte Skepsis ermitteln, üblicherweise
COMPLETES_NEEDED (Default: 1)
MARGINALS_NEEDED (Default: 2)
 - b) gewichtete Summe über Signaturen ergibt Key Legitimacy
 - c) falls die Summe größer oder gleich 1 ist: Zertifikat wird als vertrauenswürdig angesehen

8



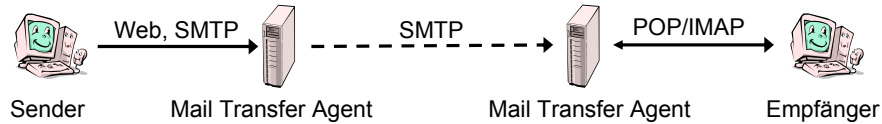
- ## Widerruf bei PGP
- Keine zentrale Instanz: **Widerruf verteilt angelegt**
 - Widerruf einer Signatur durch den Aussteller
 - Widerruf des/der Schlüssel durch Besitzer
 - Probleme
 - Bei Verlust des privaten Schlüssels kann keine Widerrufs-signatur mehr erstellt werden!
 - ▶ Widerruf eines öffentlichen Schlüssels durch Nicht-Besitzer
 - ▶ Besitzer kann über Selbstsignatur des Schlüssels andere Schlüsselpaare spezifizieren, die ebenfalls zum Widerruf berechtigt sind
 - Jedes Zertifikat muss vor Nutzung „aktualisiert“ werden (Schlüsselserver)
- 11
- Netzicherheit – Architekturen und Protokolle PGP Institut für Telematik www.tm.kit.edu Karlsruher Institut für Technologie (KIT)

Netzicherheit – Architekturen und Protokolle PGP

TELEMATICS

1. PGP PKI
2. E-Mail Sicherheit

KIT



- Abholen neuer Mails: **POP oder IMAP**
 - Authentifizierung: Benutzernamen und Passwort im Klartext
 - Ergänzung: kryptographische Authentifizierung
 - ▶ S/Key
 - ▶ Kerberos
 - Sicherung durch SSL/TLS
- Transport vom Mails: **SMTP**
 - Klartext-Protokoll ohne Authentifizierung

13

- RFC 822-konforme Mail darf nur aus ASCII-Zeichen bestehen
- Notwendige Felder im Nachrichten-Kopf
 - **Date:** Wed, 21 Apr 2009 15:45:00 +0200
 - **From:** schoeller@nw.neclab.eu
 - **To:** vorlesung@tm.uka.de
- Optionale Felder im Nachrichten-Kopf
 - **Cc und Bcc:** Kopien
 - **Message-ID:** Zuordnung von Antworten
 - **Reply-To:** Antwort-Adresse
 - **Subject:** Betreff

14

- **Problem:** Nicht-ASCII-Mails und Sprachen mit Sonderzeichen
- **Lösung:** Multipurpose Internet Mail Extensions
 - [RFC 2045 – 2049]
 - Definition von Nachrichtenformat u. syntaktische Regeln
- Unterstützung von Multimedia-E-mails durch zwei Protokoll-Köpfe
 - Art der Nachricht
 - ▶ **Content-Type:** JPEG, MP3, MPEG
 - Codierung der Nachricht
 - ▶ **Content-Transfer-Encoding:** Base64 (B), Quoted-Printable (Q)

15

Client codiert Nachricht, z.B.:

```

From: schoeller@nw.neclab.eu
To: vorlesung@tm.uka.de
Subject: Bild der Vorlesung
Mime-Version: 1.0
Content-Transfer-Encoding:
  base64
Content-Type: image/jpeg
(base64 encoded data ..... )
  
```

Empfänger:

- Decodieren der Nachricht
- Übergabe der Nachricht an spezielles Programm zur Anzeige
- MIME Content-Type Konfiguration:
 - Text/Plain:
 - Image/jpeg: mozilla
 - Audio/mp3: xmms
 - Video/mpeg: xine
 - Application/ms-word: star-office

16

- Angriffe einfach möglich

Typischer Ablauf

HELO

220

Spoofing-Angriffe
leicht möglich

MAIL FROM:
schoeller@nw.necolab.eu

250 OK

RCPT TO: vorlesung@tm.uka.de

250 OK

DATA

...

Manipulation am Inhalt
leicht möglich

Wie viele Studenten waren
heute in der Vorlesung?

.

250 OK

QUIT

17

Welche Anforderungen gibt es an E-Mail Sicherheit?

Welche Design-Ziele gibt es?

Was ist die passende Analogie?

18

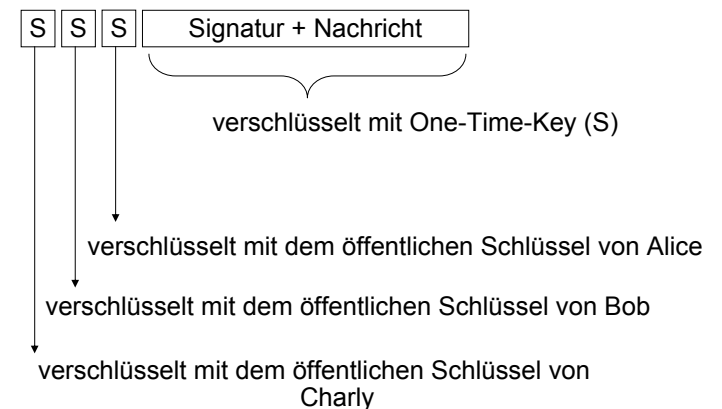
- Einfacher Ansatz

- Asymmetrisches Verfahren mit öffentlichem Schlüssel
- (Symmetrisches Verfahren mit vor-verteiltem Geheimnis)

- Probleme

- Hoher Aufwand
 - ▶ Separate Verschlüsselung der Nachricht für jeden Empfänger
- Skalierungsproblem
 - ▶ Verteilung symmetrischer Schlüssel
- Verringerungen der Sicherheit
 - ▶ Häufige Verwendung langlebiger Schlüssel

19



20

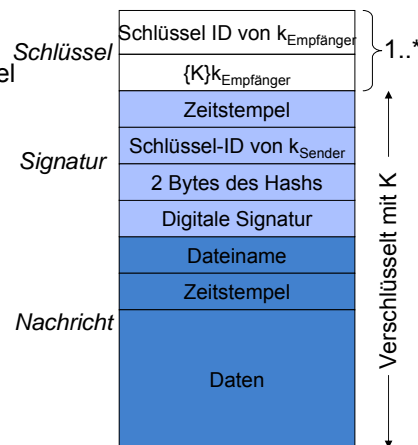
- **Public Key Kryptographie**
 1. Berechnen des Hashwerts der Nachricht
 2. Signieren des Hashs mit dem privaten Schlüssel
 3. Anhängen der Authentifizierungsdaten an die Nachricht
 - Einsetzbar auch bei entfernten Vervielfachungsdiensten
- **Symmetrische Kryptographie**
 1. CBC Restwert des Nachricht (z.B. CBC-MAC)
 2. Message Authentication Code (z.B. HMAC)
 3. Verschlüsselter Hashwert der Nachricht
 - Bei mehreren Empfänger ist die dritte Variante zu bevorzugen
 - Bei entfernten Vervielfachungsdiensten keine direkte Authentifizierung des Senders möglich

21

- PGP als *hybrides Kryptosystem*
 - Dynamisch generierter symmetrischen Schlüssel
 - Verschlüsselung der Nachricht mit Sitzungsschlüssel
 - Verschlüsselung des Sitzungsschlüssels mit öffentlichem Schlüssel des Empfängers
- **Problem:** Schlüssel-Identifizierung
 - Wie erkennt der Empfänger, welcher Public Key verwendet wurde, um den Sitzungsschlüssel zu verschlüsseln?

22

- **Schlüssel-Teil**
 - Schlüssel-ID
 - Verschlüsselter Sitzungsschlüssel
- **Signatur-Teil**
 - Erstellungsdatum der Signatur
 - Schlüssel-ID des öffentlichen Schlüssels
 - 2 Bytes des unverschlüsselten Nachrichten-Hashs
 - Signatur über Zeitstempel und Daten
- **Nachrichten-Teil**
 - Name der PGP-Datei
 - Erstellungsdatum der Datei
 - Daten



23

- **Möglichkeiten**
 - **Ausprobieren:** Empfänger testet jeden seiner Private Keys
 - **Mitschicken:** der Sender hängt den verwendeten Schlüssel bzw. das Zertifikat an
 - **Schlüssel-ID:** Anhängen einer eindeutigen Referenz auf den Public Key
- **Bewertung**
 - Ausprobieren: sehr hohen Rechenaufwand beim Empfänger
 - Mitschicken: erhöht die verbrauchte Bandbreite
 - Schlüssel-ID: Zuweisung, Speicherung und Verwaltung
 - Niederwertigsten 64 Bit vom SHA-1 Hashwert des öffentlichen Schlüssels

24

- PGP-Grundbausteine: *Pakete* (packets):
 - Paket-Bedeutung durch *Tag* identifiziert
 - Verschachtelung von Paketen
 - ▶ Z.B. ein verschlüsseltes Paket kann ein komprimiertes Paket enthalten
- Paketaufbau

| header | | payload |
|---------------|-----------------------|---------|
| tag 1 byte | length 1/2/5 bytes | |

 - Kopf
 - ▶ Tag
 - ▶ Länge des Pakets
 - Nutzdaten
- Beispiel: Public-Key Encrypted Session Key Packet
 - Tag: 1
 - Daten: Radix-64 codiert

25

- Folgende *Tags* werden für *Nachrichten* verwendet
 - 1: Public-Key Encrypted Session Key Packet
 - 2: Signature Packet
 - 3: Symmetric-Key Encrypted Session Key Packet
 - 4: One-Pass Signature Packet
 - 8: Compressed Data Packet
 - 9: Symmetrically Encrypted Data Packet
 - 11: Literal Data Packet
 - ...
- Folgende *Tags* werden für *Zertifikate* verwendet
 - 2: Signature Packet
 - 5[7]: Secret [Sub-]Key Packet
 - 6[14]: Public [Sub-]Key Packet
 - 12: Trust Packet
 - 13: User ID Packet

26

- *Verschlüsselte Daten*
 - Public-Key Encrypted Session Key Packet(s)
 - Symmetrically Encrypted Data Packet
 - ▶ z.B. Literal Data Packet
- *Signierte Daten*
 - Signature Packet
 - Literal Data Packet
- *Signierte, komprimierte und verschlüsselte Nachricht*
 - Public-Key Encrypted Session Key Packet(s)
 - Symmetrically Encrypted Data Packet
 - ▶ Compressed Data Packet
 - ▶ Signature Packet
 - ▶ Literal Data Packet

27

- Unterschiedlichen Semantiken
 - Zertifizierungen von Schlüssel zu Name
 - ▶ Beinhalten auch, wie sicher sich der Zertifizierende bezüglich der Bindung ist
 - Signieren eines Unterschlüssels
 - ▶ Bindet ihn an den primären Schlüssel (Subkey Binding Signature)
 - Signieren eines Schlüssels direkt
 - ▶ Bindet Infos unabhängig von der Identität des Besitzers an den Schlüssel
 - Widerruf-Signaturen
 - ▶ Über primären Schlüssel: Widerruf des gesamten Zertifikates
 - ▶ Über einen Unterschlüssel: Widerruf nur dieses Schlüssels
 - ▶ Über Signaturen (=Zertifizierungen): nur Signatur des Widerrufenden wird ungültig
- *Zeitstempel-Signatur*

28

- Varianten

- *Direct key self-signature*
 - ▶ Bindung von Daten an den gesamten Schlüssel, d.h. primären Schlüssel und Unterschlüssel
- *Certification self-signature*
 - ▶ Zuordnungsdaten für Bindung der User ID an den Schlüssel
- *Subkey self-signature*
 - ▶ Bindung des Unterschlüssels an den primären Schlüssel

- Zusätzliche Informationen zu den Selbstsignaturen

- Anwendungen des Schlüssels: Zertifizierung, Verschlüsselung u.a.
- Bevorzugte Algorithmen: PK, SK, Hash, Kompression
- Schlüssel-Server-Parameter: bevorzugte Server, Flags für den Server selbst

29

- PGP hat folgende Vor- und Nachteile

- 😊 Geringer Startaufwand
- 😊 Dezentrale Sicherheit (Vertrauensbildung über mehrere Signaturen)
- 😊 Benutzer hat direkten Einfluss auf die Vertrauensbildung, daher aber auch hohe Verantwortung
- 😊 Robuster bei Kompromittierung eines Schlüssels
- ☹ Keine einheitliche Sicherheitsrichtlinie (jeder signiert nach eigenem Ermessen)
- ☹ Kettenbildung zwar möglich, Pfadfindung aber nicht trivial
- ☹ großen Anzahl von Zertifikaten(=Signaturen), da dezentral

- Größte Gefahren: alle nicht PGP-spezifisch

- Man-in-the-Middle
- Passwort Guessing
- Trojaner zum Upload des Private Key Rings

30

- Open Source Implementierung

- OpenPGP compliant
- etliche Optionen, um abwärtskompatibel zu sein

- Dokumentation: <http://www.gnupg.org/documentation/>

- Zertifikat-Management:

- Erstellen eines Zertifikates/Schlüsselringes:
`gpg --gen-key`
- Exportieren eines Zertifikates (ASCII-codiert)
`gpg --armor --output <Datei> --export <ID>`
- Ausgabe des Fingerprints
`gpg --fingerprint <ID>`
- Widerrufen des Zertifikates
`gpg --gen-revoke`

31

- Verwalten von fremden Zertifikaten:

- Importieren eines Zertifikates

```
gpg --import <Datei>
gpg --recv-keys <ID>
```

- Auflisten von Zertifikaten

```
gpg --list-keys [<ID>]
gpg --list-sigs [<ID>]
gpg --check-sigs [<ID>]
```

- Manipulieren eines Zertifikates:

- ```
gpg --edit-key <ID>
```
- Signieren: `sign`
  - Erstellen einer Vertrauenssignatur: `tsign` (ab v1.3.1)
  - Einstellen des Vertrauenslevels: `trust`
  - Widerrufen einer Signatur: `revsig`
  - Viele andere Operationen (siehe Manpage der jeweiligen Version)

32



- **Signieren einer Datei:**

```
gpg --sign <Datei>
```

- **Verifizieren einer Datei:**

```
gpg --verify <Datei mit Signatur>
```

- **Verschlüsselung einer Datei:**

```
gpg --encrypt --recipient <ID> <Datei>
```

- **Entschlüsseln einer Datei:**

```
gpg --decrypt <Datei>
```

- **Kombinationen:**

```
gpg --encrypt --recipient <ID> --sign
<Datei>
```

```
gpg --decrypt --verify <Datei>
```

- C. Kaufmann, R. Perlman, M. Speciner; Network Security – Private Communication in a public world; Prentice Hall; 2003
- R. Oppliger; Secure Messaging with PGP and S/MIME; Artech House, 2001
- J. Callas, L. Donnerhacke, H. Finney, R. Thayer; RFC 2440: OpenPGP Message Format, 1998
- W. Stallings; Cryptography and Network Security, Prentice Hall, 2006