

## Anwendung kryptographischer Programme am Beispiel von NetBSD

Chemnitzler Linxtag 2004

Stefan Schumacher, <stefan@net-tex.de>, PGP Key <0xB3FBAE33>

<http://www.net-tex.de/krypt/>

\$Header: /home/daten/cvs/lectures/crypt/fohlen.tex,v 1.16 2004/03/07 08:44:13 stefan Exp \$

## Gliederung

- Motivation
- Prüfsummenverfahren
- symmetrische Verfahren
- asymmetrische Verfahren
- kryptographische Signaturen
- weitere Anwendungen

## Motivation

- Schutz von Geheimnissen
- Verifikation von Daten oder Kommunikation
- Mittel der Kriegsführung
- seit kurzem frei verfügbar

## Prüfsummen

- <http://www.pruetziffernberechnung.de/>
- Basiszahl (Information) und Prüfziffer
- Prüfziffer wird aus Basiszahl berechnet
- beide Werte werden übertragen
- Empfänger berechnet Prüfziffer und vergleicht
- Authentizität (Integrität der Prüfziffer?)

## Anwendung md5 (1)

- **RF1321**, erzeugt 128 Bit Prüfsummen
- Falltürfunktion
- als Programm bei (fast) jedem Unix dabei
- Einsatz u.a. als Passworthash
- Prüfsumme für FTP / pkgsrc

## Anwendung: aide

<http://www.cs.tut.fi/~rammer/aide.html>

- reaktives HIDS / Dateisystemintegritätsprüfer
- Datenbank mit Dateistatus & Prüfsummen
- ermöglicht Vergleich des aktuellen Systems
- deckt Manipulationen an Dateien auf

## Anwendung: verifidedexec (4)

- Ab NetBSD 2.0 (1.6-current)
- proaktives HIDS
- Prüfsummenvergleich im exec() Pfad des Kernel
- Abhängig von Kernel Securitylevel
- verweigert Ausführung bei unpassender Prüfsumme

## symmetrische Verfahren

- verwendet genau einen Schlüssel
- wandelt Klartext mit Schlüssel  $k$  in Code um
- wandelt Code mit Schlüssel  $k$  in Klartext um
- Anwendung: `mcrypt(1)`
- `mcrypt -k GEHEIM Brief.tex`
- `mcrypt -d -k GEHEIM Brief.tex.nc`

## symmetrische Verfahren

- Alice schickt Brief. tex.nc an Bob
- Alice schickt Schlüssel  $k$  an Bob **GEFAHR**
- Schlüssel kann abgefangen werden → sicherer Kanal nötig
- Integrität?
- Gruppenkommunikation:  $n * (n - 1) / 2$  Schlüssel nötig
- 10 Personen → 45 Schlüssel

## Anwendung: cgd(4)

- Ab NetBSD 2.0 (1.6-current)
- verschlüsselt Blöcke auf dem Weg zur Partition
- Daten liegen verschlüsselt auf Platte
- über Pseudodevice ver-/entschlüsselt eingemountet
- kann beim booten neuen Schlüssel generieren

## Anwendung: cfs

- cryptographic filesystem von Matt Blaze
- implementiert im Userland über NFS ähnlichen cfsd
- Daten liegen verschlüsselt im Dateisystem
- bei Zugriff über cfsd auf localhost ver-/entschlüsselt
- verschlüsselte Daten können über Netz gehen und lokal
- verschlüsselt werden
- portabel

## Exkurs: Daten löschen

- rm(1) löscht Daten nicht sondern gibt die Datenblöcke frei
- Daten können rekonstruiert werden
- Curietemperatur oder Degasser, vernichten Datenträger
- Daten müssen mehrfach überschrieben werden
- Wiper (bcwiper, wipe, ncrypt ...)
- `dd if=/dev/urandom of=/dev/wd0d bs=1k`

## asymmetrische Verfahren

- Schlüsselpaar aus öffentlichem und privatem Schlüssel
- privater Schlüssel ist **geheim** zu halten
- Alice und Bob haben je öffentlichen  $(A_o, B_o)$  und privaten  $(A_p, B_p)$  Schlüssel
- Alice verschlüsselt für Bob *schematisch* so:  
 $Chiffrierung(Klartext + A_p + B_o) \rightsquigarrow Code$
- Bob entschlüsselt so:  
 $Dechiffrierung(Code + B_p + A_o) \rightsquigarrow Klartext$

## asymmetrische Verfahren

- Alice schickt *Code* und  $A_o$  an Bob
- Sniffer kann nichts damit anfangen, benötigt  $B_p$
- Bob besitzt  $B_p$  und kann *Code* dechiffrieren
- Bob kann Integrität prüfen, da nur Alice  $A_p$  hat

## kryptographische Signatur

- Sonderfunktion der asymmetrischen Verfahren
- sichert Integrität und Authentizität
- Alice bildet Prüfsumme über Nachricht  $h(N)$
- verschlüsselt Prüfsumme mit  $A_p$ :  $S = \{h(N)\}_{A_p}$
- Bob empfängt Nachricht, Signatur und  $A_o$
- Bob bildet selbst Prüfsumme  $h'(N)$
- Bob entschlüsselt  $S$  mit  $A_o \rightsquigarrow h(N)$
- gilt  $h'(N) = h(N)$ ?

## Beispielsignatur

```

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Von Zeit zu Zeit seh ich den Alten gern,
Und hute mich, mit ihm zu brechen.
Es ist gar hübsch von einem großen Herrn,
So mit dem Teufel selbst zu sprechen.
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.2.4 (NetBSD)
iDEBQFAA8ySEfTEHrP7rjMRaMLxAlJ0ePbmsTfzK4NVSzFMSpvhI2NqACdeuFA
NChp1stKID03nVEDEDD0UmA=
=+1st
-----END PGP SIGNATURE-----

```

<stefan@net-tex.de>

## kryptographische Signatur

- Gehört  $A_0$  wirklich Alice? (0xDEADBEEF)
- Ist Alice überhaupt Alice?
- Jemand signiert  $A_0$
- Prüfe Alice Identität (Ausweis)
- Prüfe  $A_0$  (Name, eMail ...)
- Schicke Signatur verschlüsselt per eMail

kryptographische Signatur

kryptographische Signatur

## kryptographische Signatur

- **Baumhierarchie**
- zentrale Zertifizierungsinstanz
- anfälliger (Kann ich der CA trauen?)
- **Web of Trust**
- Netz, hat daher keine Wurzel
- baut Kette auf (über Signatur der Signierer ...)
- weniger anfällig bei einem Fehler
- keine CA nötig

Bsp.: <http://www.uni-magdeburg.de/steschum/local-web.jpg>

<stefan@net-tex.de>

## Anwendung: PGP/GnuPG

- **P**retty **G**ood **P**rivacy von Phil Zimmermann
- "Killerapplikation" der Kryptographie
- hauptsächlich für eMail entwickelt
- resultierte in OpenPGP (RFC2440)
- GnuPG als Alternative
- Standard auf Unixsystemen

kryptographische Signatur

Ausblick

## Steganographie

- Verstecken von Information
- Bspw. "Geheimtinte"
- Wasserzeichen, "Kopierschutzverfahren"
- nützlich wenn Kryptographie verboten
- Dateisystem: **StegFS**
- Anwendung: steghide