

# Kryptographische Dateisysteme im professionellen Umfeld

Stefan Schumacher

`www.sicherheitsforschung-magdeburg.de`  
Magdeburger Institut für Sicherheitsforschung

Chemnitzer Linux-Tage 2019





# Über Mich

- Berater für Finanzinstitute, Regierungen, Sicherheitsbehörden
- Direktor des Magdeburger Instituts für Sicherheitsforschung  
Forschungsprogramme zur Unternehmenssicherheit
- Geek, Nerd, Hacker seit knapp 20 Jahren
- Herausgeber des Magdeburger Journals zur Sicherheitsforschung
- `www.Sicherheitsforschung-Magdeburg.de`
- CLT2004: Kryptographische Dateisysteme für NetBSD
- CLT2011: Kryptographische Dateisysteme genauer betrachtet



# Inhaltsverzeichnis

- 1 Einführung
- 2 Betriebsarten von Algorithmen
- 3 Kryptographische Dateisysteme
- 4 Cloud und Backup



# Warum das alles?

- Kryptographie sichert Vertraulichkeit, Integrität, Authentizität von Daten
- kryptographische Dateisysteme verschlüsseln relativ bequem Dateien
- der einzig brauchbare Weg um Daten vor Verlust zu schützen, Zugriffsrechte im Betriebssystem lassen sich relativ einfach mit Live-System aushebeln
- kryptographische Dateisysteme sind wenig benutzerfreundlich
- Administration lässt zu wünschen übrig
- vor allem in großen Installationen (viele Benutzer)
- Kollaboration in der Cloud

# Bedrohungsszenario

- Festplattenverschlüsselung ist komplex, kryptographische Dateisysteme sind komplex!
- TrueCrypt schützt mich vor Viren! Falsch!
- Threat Model: Wovor will ich mich schützen? Was kann der Angreifer?
- verschlüsselte Daten auf nicht-vertrauenswürdigem System (Angreifer ist root)

	Chiffre	gemountetes Dateisystem
Eve	nur lesen	nur schreiben
Cloud	lesen - schreiben	kein Zugriff
Mallory	lesen - schreiben	lesen - schreiben auf einigen Verzeichnissen

# Bedrohungsszenario

- Festplattenverschlüsselung ist komplex, kryptographische Dateisysteme sind komplex!
- TrueCrypt schützt mich vor Viren! Falsch!
- Threat Model: Wovor will ich mich schützen? Was kann der Angreifer?
- verschlüsselte Daten auf nicht-vertrauenswürdigem System (Angreifer ist `root`)

	<b>Chiffre</b>	<b>gemountetes Dateisystem</b>
<b>Eve</b>	nur lesen	nur schreiben
<b>Cloud</b>	lesen - schreiben	kein Zugriff
<b>Mallory</b>	lesen - schreiben	lesen - schreiben auf einigen Verzeichnissen

# Arten von kFS

- jede Datei einzeln verschlüsseln

```
openssl enc -aes256 -in Backup.txz -out Backup.txz.enc
```

- portabel, funktioniert
- Aufwand



# Arten von kFS

## Vorteile/Nachteile

### Container

- Gebrauch transparent
- Portabilität eingeschränkt (OS ./ . Anwendungsprogramm)
- netzwerktauglich
- geringe Granularität auf Dateiebene und Benutzerebene, feste Größe (!)
- Container kann gesichert werden, aber: Platzbedarf, Passwortdatei nicht vergessen, inkrementelles/differentielles Backup geht nicht ohne eingebundenen Dateien, Datenmenge  $\geq 1$ , idR  $\gg 1$
- keine Metadaten sichtbar!



# Arten von kFS

## Vorteile/Nachteile

### Stacked Filesystem

- Gebrauch transparent
- Portabilität eingeschränkt (OS ./ . Anwendungsprogramm)
- netzwerktauglich
- hohe Granularität auf Dateiebene und Benutzerebene
- verschl. Verz. kann problemlos gesichert werden und ist dann schon geschützt, inkrementelles/differentielles Backup funktioniert (Datenmenge= 1)
- Metadaten idR sichtbar!



# Arten von kFS

## Vorteile/Nachteile

### Blockebene/Dateisebene

- Einrichtungs-Aufwand hoch
- Gebrauch transparent
- Portabilität eingeschränkt (OS ./ . Anwendungsprogramm)
- nicht netzwerktauglich
- null Granularität auf Dateiebene und Benutzerebene: entweder an oder aus
- Device kann mit dd(1) gesichert werden, aber: Platzbedarf, Passwortdatei nicht vergessen, inkrementelles/differentielles Backup geht nicht ohne eingebundenen Dateien, Datenmenge  $\geq 1$ , idR  $\gg 1$
- keine Metadaten sichtbar!



# Swap und Temp

- Auslagerung von RAM bzw. temporäre Inhalte
- d.h. Swap und Temp können eigentlich verschlüsselte Inhalte unverschlüsselt enthalten
- müssen verschlüsselt werden (oder RAM-Disk)
- idR nur auf Blockebene praktikabel

# Inhaltsverzeichnis

- 1 Einführung
- 2 Betriebsarten von Algorithmen**
- 3 Kryptographische Dateisysteme
- 4 Cloud und Backup



Nimm AES-256, das ist sicher!

# Block und Strom

- Stromchiffre: Daten und Schlüssel als Strom, die miteinander verknüpft werden, bspw. Sprachdaten im Telefon
- Blockchiffre: Daten und Schlüssel werden in gleichgroße Blöcke zerlegt und blockweise verknüpft
- Datenträger sind in Blöcken organisiert  $\rightsquigarrow$  Blockmodi einsetzen

Rindflei schettik etierung süberwac hungserg  
PASSWORT PASSWORT PASSWORT PASSWORT PASSWORT  
änzungsg esetz000  
PASSWORT PASSW000



# Betriebsarten

- Electronic Code Book (ECB): selber Klartextblock  $\rightsquigarrow$  selbes Chifftrat
- Cipher Block Chaining (CBC): (Klartextblock  $n$  + Chifftrat  $n - 1$  + Passwort)  
Lawineneffekt, daher Gruppierung, Watermarking-Attacke möglich
- Encrypted Salt-Sector IV (ESSIV): Clemens Frühwirth für LUKS: IV aus der verschlüsselten Sektornummer und einem Hash aus dem Passwort, Passwort ist nicht vorhersagbar
- Galois/Counter Mode (GCM): Authenticated Encryption with Associated Data:  
Verschlüsselung *und* Authentifikation  
wahlfreier Zugriff auf jeden Block möglich, parallelisierbar



- Schlüsselableitungsfunktion
- früher: Passwort als Hash gespeichert (MD5, SHA1, SHA1(MD5(Passwort+Salz)))
- leicht brechbar durch Wörterbuchattacke, auch bei Passwort mit Zahl und Sonderzeichen (`geheim!`, `password1`)
- PBKDF2 (RFC 2898, 2000) und bcrypt (1999, Niels Provos) entwickelt
- anpassbare Rundenzahlen, Zufallswert in jeder Runde (z.B. HMAC), Salt
- erschweren Wörterbuchattacken massiv, aber angreifbar durch Hardware (FPGA, GPU) da geringe Speicheranforderungen (4kb)
- Hash-Runden parallelisierbar
- scrypt 2010 von Colin Percival (FreeBSD) entwickelt
- fordert standardmäßig einen 16MB-RAM-Vektor an
- damit sequenziell Speicher-intensiv

# Inhaltsverzeichnis

- 1 Einführung
- 2 Betriebsarten von Algorithmen
- 3 Kryptographische Dateisysteme**
- 4 Cloud und Backup



## Hardwareverschlüsselung

- bei HDDs/SHDDs/SSDs möglich, Standard bei Samsung SSDs
- Transparent für Nutzer und OS, lediglich Passwort beim booten erforderlich
- aber: fehlerhafte Implementierungen (Self-encrypting deception: weaknesses in the encryption of solid state drives (SSDs) by Meijer and Gastel)

## TrueCrypt/Veracrypt

- Container (ein großer BLOB) oder Blockebene
- Windows, MacOS X, Linux

## LUKS/LVM

- Festplattenvollverschlüsselung möglich
- externe Datenträger (USB-Stick, Festplatte) komplett verschlüsselbar
- Datei als Schlüssel und/oder 2FA mit Yubikey möglich
- Freischaltung über Netzwerk mit Clevis und Tang möglich (entwickelt von Red Hat)  
Dabei wird kein Geheimnis über das Netz übertragen (modifizierter DHKE)

## ext4

- unterstützt Verschlüsselung, selten in Benutzung
- einzelne Verzeichnisse werden verschlüsselt
- portabel auf Linux

- ab Linux-Kernel 2.6.19
- Metadaten *in* der Datei  $\rightsquigarrow$  Portabilität
- stark auf Ubuntu zugeschnitten
- Einrichtung vergleichsweise komplex
- Zukunft in der Schwebe

- FUSE-basiert, entwickelt seit 2003
- unterstützt die im Linux-Kernel vorhandenen Kryptalgorithmen
- kann jede Datei mit eigenem IV verschlüsseln
- 8-Bit-Prüfsumme für jede Datei (Überprüfung kann in .conf abgeschaltet werden)
- Linux, FreeBSD, Mac OS X, Windows
- Prinzipiell für Cloud geeignet
- aber: Sicherheitsbedenken!

Links	Datei	Befehl	Optionen	Rechts																																
<-	....daten.encfs/LeVK4gYRfAvArWw-bl1bnciQ	-	.	> [^]																																
<table border="1"><thead><tr><th>.n</th><th>Name</th><th>Größe</th><th>Modifikations</th></tr></thead><tbody><tr><td>/..</td><td>ÜBERVZ.</td><td>02. Mär 19:13</td><td></td></tr><tr><td>/Bx-vdG0hP281uV8e41vY6NXX</td><td>4096</td><td>04. Aug 2014</td><td></td></tr><tr><td>AjqCXVnNXFDsp3qPp9QPRpGp</td><td>641</td><td>11. Feb 2017</td><td></td></tr><tr><td>FJIZH70epW0YzN4UDCKI7pR,</td><td>433</td><td>11. Feb 2017</td><td></td></tr><tr><td>Pv8-mKVqCJy-w~LMtWpG1JI8BM1</td><td>10336</td><td>06. Mär 18:50</td><td></td></tr><tr><td>iP0p4IJQ,LCB7JGtTXWIigfZ</td><td>688</td><td>11. Feb 2017</td><td></td></tr><tr><td>oqptlR9MsTGijUYdCio0lQnr</td><td>1699</td><td>11. Feb 2017</td><td></td></tr></tbody></table>					.n	Name	Größe	Modifikations	/..	ÜBERVZ.	02. Mär 19:13		/Bx-vdG0hP281uV8e41vY6NXX	4096	04. Aug 2014		AjqCXVnNXFDsp3qPp9QPRpGp	641	11. Feb 2017		FJIZH70epW0YzN4UDCKI7pR,	433	11. Feb 2017		Pv8-mKVqCJy-w~LMtWpG1JI8BM1	10336	06. Mär 18:50		iP0p4IJQ,LCB7JGtTXWIigfZ	688	11. Feb 2017		oqptlR9MsTGijUYdCio0lQnr	1699	11. Feb 2017	
.n	Name	Größe	Modifikations																																	
/..	ÜBERVZ.	02. Mär 19:13																																		
/Bx-vdG0hP281uV8e41vY6NXX	4096	04. Aug 2014																																		
AjqCXVnNXFDsp3qPp9QPRpGp	641	11. Feb 2017																																		
FJIZH70epW0YzN4UDCKI7pR,	433	11. Feb 2017																																		
Pv8-mKVqCJy-w~LMtWpG1JI8BM1	10336	06. Mär 18:50																																		
iP0p4IJQ,LCB7JGtTXWIigfZ	688	11. Feb 2017																																		
oqptlR9MsTGijUYdCio0lQnr	1699	11. Feb 2017																																		
<table border="1"><thead><tr><th>.n</th><th>Name</th><th>Größe</th><th>Modifikations</th></tr></thead><tbody><tr><td>/..</td><td>ÜBERVZ.</td><td>02. Mär 19:13</td><td></td></tr><tr><td>/schumacher-ssh</td><td>4096</td><td>04. Aug 2014</td><td></td></tr><tr><td>id_dsa</td><td>672</td><td>11. Feb 2017</td><td></td></tr><tr><td>id_dsa.pub</td><td>625</td><td>11. Feb 2017</td><td></td></tr><tr><td>id_rsa</td><td>1675</td><td>11. Feb 2017</td><td></td></tr><tr><td>id_rsa.pub</td><td>417</td><td>11. Feb 2017</td><td></td></tr><tr><td><b>schumacher-ssh.tar</b></td><td><b>10240</b></td><td><b>06. Mär 18:50</b></td><td></td></tr></tbody></table>					.n	Name	Größe	Modifikations	/..	ÜBERVZ.	02. Mär 19:13		/schumacher-ssh	4096	04. Aug 2014		id_dsa	672	11. Feb 2017		id_dsa.pub	625	11. Feb 2017		id_rsa	1675	11. Feb 2017		id_rsa.pub	417	11. Feb 2017		<b>schumacher-ssh.tar</b>	<b>10240</b>	<b>06. Mär 18:50</b>	
.n	Name	Größe	Modifikations																																	
/..	ÜBERVZ.	02. Mär 19:13																																		
/schumacher-ssh	4096	04. Aug 2014																																		
id_dsa	672	11. Feb 2017																																		
id_dsa.pub	625	11. Feb 2017																																		
id_rsa	1675	11. Feb 2017																																		
id_rsa.pub	417	11. Feb 2017																																		
<b>schumacher-ssh.tar</b>	<b>10240</b>	<b>06. Mär 18:50</b>																																		
iP0p4IJQ,LCB7JGtTXWIigfZ																																				
schumacher-ssh.tar																																				
99G/223G (44%)																																				

Metadaten sichtbar  
Nutzbar z.B. für rsync, duplicity etc.

- EncFS 1.7.4 Security Audit by Taylor Hornby
- 7/10 Schwachstellen noch offen
- *EncFS is not safe if the adversary has the opportunity to see two or more snapshots of the ciphertext at different times.*
- Angriff möglich, wenn unterschiedliche Versionen einer verschlüsselten Datei vorliegen
- Nicht sicher im Cloud-Betrieb!
- Version 2.0 eher unwahrscheinlich

- Master-Arbeit von Sebastian Messmer 2015 am KIT
- Stacked Filesystem (FUSE)
- versteckt Metadaten wie Dateigröße, [mca]time und komplette Verzeichnisstruktur
- Nutzt aes-256-gcm (Galois/Counter Mode), ein authentifizierter Verschlüsselungsmodus
- Dateien werden in gleichgroße Blöcke zerlegt und in einer Baumstruktur verteilt
- kein Rückschluss auf Original-Dateistruktur möglich
- Speicher-Overhead kann vergleichsweise groß werden
- Verschlüsselungs-Schlüssel zufallsgeneriert und mit scrypt-abgeleitetem Benutzerpasswort verschlüsselt, `cryfs.config` kann in Cloud liegen

# CryFS

.CloudArchivCryFS/000  
.CloudArchivCryFS/000/1B7E011BB56CDBCE2A03867205475  
.CloudArchivCryFS/000/8187B021FCDC4111D75F1B0DD7866  
.CloudArchivCryFS/000/83774EE59D5E374F24EF759357CC1  
.CloudArchivCryFS/000/869FA7754CED7A200D118783942FD  
.CloudArchivCryFS/000/BBE33C35117E0E9A1192B071CDF3C  
.CloudArchivCryFS/000/E32BE40536C427D26C97C03FF7100  
.CloudArchivCryFS/000/EB51F900C9B806E9F8E50918AB984  
.CloudArchivCryFS/001  
.CloudArchivCryFS/001/77F6A4A61EFA462050214C908F496  
.CloudArchivCryFS/001/7BE814DEB0899DF1D316BBBD9877F  
.CloudArchivCryFS/001/856C8D10FC5E90D016FBF262DBF85  
.CloudArchivCryFS/001/A1FAEE159930F0CB7810FCBE2AB03  
.CloudArchivCryFS/001/A349527B295D2A2D5FA02C4FADEAD  
.CloudArchivCryFS/001/F0868ED109E739885694F7F7656F6  
.CloudArchivCryFS/002

```
find CloudArchiv -type f | wc -l  
726
```

```
find .CloudArchivCryFS -type f | wc -l  
29125
```

```
du -sm CloudArchiv .CloudArchivCryFS  
437      CloudArchiv  
472      .CloudArchivCryFS
```

- Stacked Filesystem, FUSE
- Implementiert in Go
- Gilt als Nachfolger von EncFS
- Security Audit von Taylor Hornby von 2017 (2 Tage)
- nutzt bessere kryptographische Primitiven als EncFS

- scrypt für Schlüsselableitung des KEK
- EME-Verschlüsselung des Dateinamens
- AES-GCM-Verschlüsselung des Dateiinhalts
- Dateien in 4KiB-Blöcke zerlegt, jeder Block mit eigenem IV
- AES-256-EME-Verschlüsselung der Dateinamen, IV per Verzeichnis
- Dateinamen bis 255 Zeichen möglich

unverschlüsseltes Verzeichniss wird in ein verschlüsseltes Verzeichnis synchronisiert

- 1 `gocryptfs -init -reverse /home/alice`
- 2 `gocryptfs -reverse /home/alice /tmp/reverse`
- 3 `date > /home/alice/date.txt`
- 4 `ls /tmp/reverse/6g7UuJbwo8n1oEh4SZ1DHw`
- 5 `sshfs alice@raspberrypi:/home/alicebackup /mnt/alicebackup`
- 6 `cp -r /tmp/reverse/ /mnt/alicebackup`
- 7 `fusermount -u /tmp/reverse`

# Benchmark

<b>FS</b>	<b>tar xpf</b>	<b>rm -rf</b>
ohne	0m10,862s	0m1,458s
encfs p	1m6,941s	0m21,877s
encfs	0m55,878s	0m17,176s
gocryptfs	0m38,431s	0m14,556s
cryfs	0m54,010s	0m39,007s

time tar xpf linux-4.20.14.tar.xz 925MB 66638 Dateien



- **eCryptFS**: komplizierte Benutzung, Zukunft ungewiss, hätte Mehrbenutzerpotenzial
- **encfs**: veraltend, Sicherheitsprobleme in der Cloud, 2.0 sehr unwahrscheinlich
- **GoCryptFS**: modernere Methoden, Integritätssicherung, schnell, leaked Metadaten
- **CryFS**: modernere Methoden, Integritätssicherung, verschleiert Metadaten komplett, langsam bei Benutzung und u.U. beim synchronisieren
- Abwägung Geschwindigkeit/Platz vs. Metadaten-Leak
- Prüfen wo Bottleneck liegt, ob Geschwindigkeitseinschränkung überhaupt zum Tragen kommt

# Inhaltsverzeichnis

- 1 Einführung
- 2 Betriebsarten von Algorithmen
- 3 Kryptographische Dateisysteme
- 4 Cloud und Backup**



# einfaches Backup

- `tar cpf /mnt/usb/daten-`date -I`.tar /.daten.encfs`
- `scp /.daten.encfs stefan@raspi:/home/backup`
- **Verschlüsselung vs Kompression**



# Duplicity

- unterstützt inkrementelle Backups via rsync-Algorithmus
- erzeugt tarballs die mit GnuPG verschlüsselt werden
- kann auf entfernte Ziele schreiben (ssh, sftp, webdav, Amazon S3, Dropbox)



- »rsync für Clouddienste«
- Konsole, screen
- FTP, SFTP, HTTP, DAV, local, Google, Dropbox, Nextcloud, Owncloud, Amazon ...
- MD5/SHA1 Hashes, Check modus, Update, Sync, FUSE-Mount
- Verschlüsselungsmodus
- `rclone sync /Backup GoogleDrive:Backup/ -progress -v`

# Mein Backup

- duplicity von gocryptfs-verschlüsseltem Verzeichnis von SSD nach HDD
- rsync von HDD auf Raspberry Pi
- des nächstens: rclone vom Raspberry Pi nach NextCloud im Büro, Google, Dropbox
- rsync von HDD auf externe HDD



- Mittwoch früh: Praktische Security Awareness am Beispiel von Passwörtern und 2-Faktor-Authentifikation  
keine Technik notwendig
- Heinlein Akademie ab Oktober 2019: Social Engineering / Security Awareness Workshop (3d)

- `sicherheitsforschung-magdeburg.de`
- `stefan.schumacher@sicherheitsforschung-magdeburg.de`
- `sicherheitsforschung-magdeburg.de/publikationen/journal.html`



- `youtube.de/Sicherheitsforschung`
- Twitter: 0xKaishakunin
- LinkedIn/Xing: Stefan Schumacher
- ZRTP: 0xKaishakunin@ostel.co

- `sicherheitsforschung-magdeburg.de`
- `stefan.schumacher@sicherheitsforschung-magdeburg.de`
- `sicherheitsforschung-magdeburg.de/publikationen/journal.html`



- `youtube.de/Sicherheitsforschung`
- Twitter: 0xKaishakunin
- LinkedIn/Xing: Stefan Schumacher
- ZRTP: 0xKaishakunin@ostel.co