Sichere Datenhaltung und Backup in der Cloud

Stefan Schumacher

Chemnitzer Linux-Tage 2024

$Id: CLT2024-SichereDatenhaltung.tex,v 1.10 2024/03/16 06:23:39 stefan Exp $

Über mich



Über mich

- CLT2004: Kryptographische Dateisysteme für NetBSD
- CCC2004: Einführung in die Kryptographie
- CLT2006: Methoden zur Datensicherung
- CLT2011: Kryptographische Dateisysteme genauer betrachtet
- CLT2016: Zwei-Faktor-Authentifizierung mit Yubikeys
- CLT2019: Kryptographische Dateisysteme im professionellen Umfeld

Literatur I

- Schumacher, S. (2004). Einführung in kryptographische Methoden. Verfügbar 19. April 2004 unter http://www.cryptomancer.de/21c3/21c3-kryptograhie-paper.pdf
- Schumacher, S. (2006a). Methoden zur Datensicherung – Strategien und Techniken für NetBSD. In German Unix User Group (Hrsg.), *Proceedings des GUUG Frühjahrsfachgespräches 2006* (S. 79–98). Universität Osnabrück. Berlin: Lehmanns Media.
- Schumacher, S. (2006b). Methoden zur Datensicherung – Strategien und Techniken für NetBSD. net-tex.de. Verfügbar unter http://www.net-tex.de/backup.pdf
- Schumacher, S. (2006c). Verschlüsselte Dateisysteme für NetBSD. *UpTimes*, 25–31. Verfügbar 7. November 2009 unter http://kaishakunin.com/publ/guug-uptimes-cgd_cfs.pdf

Literatur II

Schumacher, S. (2011). Kryptographische Dateisysteme im Detail. In Team der Chemnitzer Linux-Tage (Hrsg.), *Chemnitzer Linux-Tage 2011: Tagungsband* (S. 39–46). Technische Universität Chemnitz. Chemnitz: Universitätsverlag. Verfügbar unter https://monarch.qucosa.de/api/qucosa%3A19466/attachment/ATT-0/

Schumacher, S. (2012). Wenn der GAU kommt: Datensicherungsstrategie erarbeiten und umsetzen. *UpTimes*, 21–29. Verfügbar 7. Juli 2013 unter https://guug.de/wp-content/uploads/2020/11/uptimes_2012-03.pdf

Schumacher, S. (2013). Wenn der Mini-GAU kommt: Kleine Programme zur Sicherung einzelner Rechner. *UpTimes*, 17–21. Verfügbar 7. Juli 2013 unter https://guug.de/wp-content/uploads/2020/11/uptimes_2012-02.pdf

Schumacher, S. (2018). Zwei-Faktor-Authentifizierung mit Yubikey-Token: Ein kostengünstiges Verfahren. *UpTimes*, 16–27. Verfügbar 1. Februar 2019 unter https://www.guug.de/uptimes/2018-2/uptimes_2018-02.pdf

Inhaltsverzeichnis

1 Einführung

2 Kryptographische Dateisysteme

3 Die Cloud mounten

4 Cloud und Backup

Warum das alles?

- Die Cloud ist anderer Leute Computer.
- Cloud-Anbieter haben Zugriff auf meine Daten
  - können meine Daten lesen!
  - können meine Daten schreiben!
- Meine Daten sind bares Geld wert
- Wenn etwas umsonst ist, bist du das Produkt
- Clouds sind praktisch zur Datensicherung und Kollaboration
- Meine Daten liegen sicher (Zugang, Zutritt, Brandschutz ...) an einem anderen Ort
- Self Hosting macht Spaß, Datensicherheit muss trotzdem gewährleistet sein, was wenn der RZ-Betreiber einfach meine SSD ausbaut und anschaut? Verliert?

Warum das alles?

- Kryptographie sichert Vertraulichkeit, Integrität, Authentizität von Daten
- *Nicht* Verfügbarkeit!
- Backup *auch* in der Cloud, aber nicht *nur* in der Cloud
- Kryptographische Dateisysteme verschlüsseln relativ bequem Dateien
- Der einzig brauchbare Weg um Daten vor Verlust zu schützen,
- Kryptographische Dateisysteme sind wenig benutzerfreundlich
- Administration lässt zu wünschen übrig
- Vor allem in großen Installationen (viele Benutzer)

Inhaltsverzeichnis

1 Einführung

2 Kryptographische Dateisysteme

3 Die Cloud mounten

4 Cloud und Backup

Arten von kFS

Vorteile/Nachteile in der Cloud

- Full Disk Encryption (LUKS, ext4, ZFS) : nützlich u.U. auf einem Server, VM oder Container, Schützt nicht, wenn eingemountet oder vor Serverbetreiber!
- Container (Veracyrypt, LUKS): möglich
  - Nachteil: muss aber i.d.R. immer voll übertragen werden,
  - Vorteil: Versteckt Metadaten der Nutzlast komplett
- Schlüsselmanagement komplex (siehe mein Vortrag morgen)
- Kollaboration in der Cloud umständlich

Arten von kFS

Vorteile/Nachteile in der Cloud

Stacked Filesystem mit FUSE (Filesystem in User Space)

- Gebrauch transparent
- Portabilität eingeschränkt (OS ./. Anwendungsprogramm)
- Netzwerktauglich
- Hohe Granularität auf Dateiebene und Benutzerebene
- Verzeichnis kann problemlos gesichert werden und ist dann schon geschützt
- Inkrementelles/differentielles Backup funktioniert (Datenmenge == 1)
- Metadaten idR sichtbar!

Dateisystemsnapshots

ZFS und BTRFS

- Verschiedene Dateisysteme unterstützen Snapshots
- Dateisystemzustand wird eingefroren und kann gesichert werden
- Ideal zur Sicherung der OS-Platte
- ZFS & BTRFS: inkrementelle Snapshots und send/receive
- ZFS: eingebaute Verschlüsselung
- (Verschlüsselte) Snapshots können übers Netz gesendet werden
- Zielsystem muss auch ZFS bzw. BTRFS unterstützen

```
zfs send -i tank/dana@snap1 tank/dana@snap2 | \
     ssh host2 zfs recv newtank/dana

btrfs send -c /.snapshot/23 | \
      ssh root@ip "btrfs receive /mnt/backups/23"
```

kFS Kurzüberblick

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
- Einzelne Verzeichnisse können verschlüsselt werden
- Portabel auf Linux

verschlüsselnde FUSE FS

- cfs, encfs, ecryptfs: Nicht mehr verwenden, obsolet!
- GoCryptFS und CryFS empfehlenswert
- Aktuelle kryptographische Primitiven, Schlüssellableitungen und Betriebsmodi
- Beide bieten Authenticated Encryption with Associated Data durch GCM $\rightsquigarrow$ sichert Authentizität und Integrität weiterer Daten (z.B. Header und Metadaten)

CryFS ./. GoCryptFS

CryFS

- Versteckt Metadaten wie Dateigröße, [mca]time und komplette Verzeichnisstruktur
- Dateien werden in gleichgroße Blöcke zerlegt und in einer Baumstruktur verteilt
- Kein Rückschluss auf Original-Dateistruktur möglich
- Speicher-Overhead und Zugriffsgeschwindigkeit kann vergleichsweise groß werden
- Blockgröße von CryFS *und* Dateisystem anpassen

CryFS ./. GoCryptFS

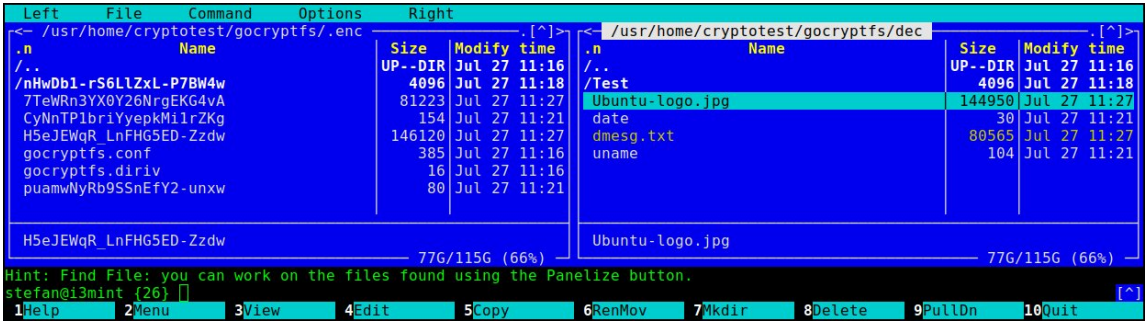
GoCryptFS

- Metadaten werden nicht versteckt, Dateinamen können verschlüsselt werden
- Sehr geringer Speicher-Overhead, kaum Geschwindigkeitsverluste
- FIDO2 Token (z.B. Yubikey ab 5) möglich
- Mehrbenutzer-Hack `cp gocryptfs.conf gocryptfs.conf.bob && gocryptfs -passwd -config gocryptfs.conf.bob ./`

»Kryptographische Sicherheit« für die meisten Einsatzzwecke gleichwertig

GoCryptFS

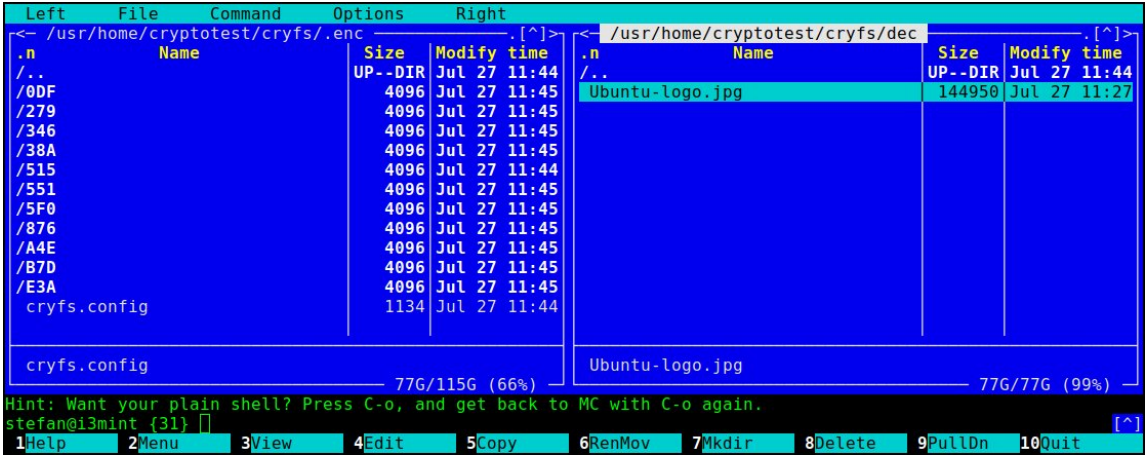
Im Dateisystem



Metadaten (Größe, Datum) sichtbar

CryFS

Im Dateisystem



Metadaten (Größe, Datum) *nicht* sichtbar

GoCryptFS

Reverse Mode

Unverschlüsseltes Verzeichniss wird in ein verschlüsseltes Verzeichnis synchronisiert ohne weiteren Platzbedarf, ideal für verschlüsselte Backups

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@raspberry:/home/alicebackup /mnt/alicebackup`
6. `cp -r /tmp/reverse/ /mnt/alicebackup`
7. `fusermount -u /tmp/reverse`

Inhaltsverzeichnis

1 Einführung

2 Kryptographische Dateisysteme

3 Die Cloud mounten

4 Cloud und Backup

rclone

- »rsync für Clouddienste« auf Konsole, screen
- Synchronisiert Verzeichnisstrukturen, überträgt nur geänderte Daten,
- FTP, SFTP, HTTP, DAV, local, Google, Dropbox, Nextcloud, Owncloud, Amazon, Pcloud ...
- MD5/SHA1 Hashes, Check modus, Update, Copy, Sync, Bisync, FUSE-Mount, Union
- Verschlüsselungsmodus
- Passwörter obfuscated in der `.rclone.conf`, Masterpasswort möglich
- Automatisierte Konfigurationsabfrage
- `rclone sync  /Backup GoogleDrive:Backup/ -progress -v`

Inhaltsverzeichnis

1 Einführung

2 Kryptographische Dateisysteme

3 Die Cloud mounten

4 Cloud und Backup

Weise Worte

Kristian Köhntopp zitiert einen Vertriebler

Niemand will Backup, alle wollen Restore!

http://www.iks-jena.de/mitarb/lutz/usenet/Fachbegriffe.der.Informatik.html

- Testen!
- Testen!
- Regelmäßig Testen!

Snapshots

- Rsnapshot und Rdiff-Backup erzeugen Snapshots via Rsync
- Rsnapshot via Hardlinks, Rdiff-Backup via Deltas/Increments
- Rsnapshots: alle Snapshots direkt im Dateisystem zugänglich, Hardlinks müssen unterstützt werden!
- Rdiff-Backup: nur aktueller Zustand direkt zugänglich
- Platzsparende Archivierungsmethode
- Automatisch per Anacron bei jedem Startvorgang

Duplicity

- Unterstützt inkrementelle Backups via rsync-Algorithmus
- Erzeugt tarballs die mit GnuPG verschlüsselt werden
- Metadaten der Quellen versteckt
- Kann auf entfernte Ziele schreiben (ssh, sftp, webdav, Amazon S3, Dropbox ...)

Alternativen

- Borg: Kryptographie vor 2.0 suboptimal, muss auf Zielserver laufen
- Restic: speicherintensiv bei Sicherungen >20GB
- git-annex: Dateisynchronisierung - nicht Backup, große Dateibäume synchronisieren und teilen, Metadatenverwaltung in git, Dateien selbst nicht im git
  - Archivverwaltung: dutzende von Festplatten $\rightsquigarrow$ wo ist welche Datei?
  - Synchronisation: Alice und Bob synchronisieren große Dateibäume via git-annex

Authentizität von Daten

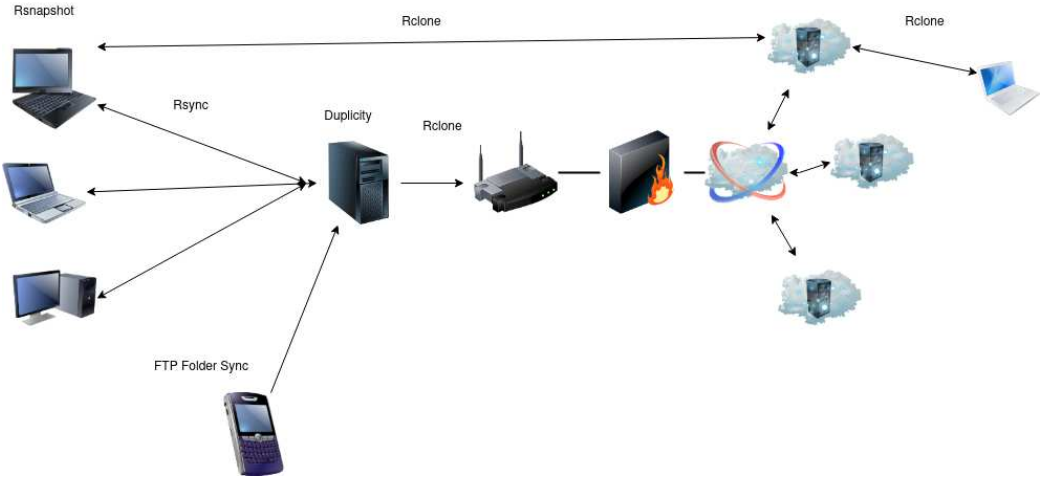
- Wurden meine Daten verändert? ⇝AEAD hilft – bei/nach der Entschlüsselung
- Vor der Entschlüsselung oder ohne Entschlüsselung?
- Signaturen (Hashes) erstellen und prüfen

```
find . -type f -exec sha256sum {} + > SHA256.txt
gpg -a --output SHA256.sig --detach-sign SHA256.txt
gpg --verify SHA256.sig && sha256sum --quiet -c SHA256.txt
./SHA256.txt: FEHLSCHLAG
sha256sum: WARNUNG: 1 berechnete Pruefsumme passte NICHT
```

Authentizität von Daten

Alternativen zur Handarbeit

- Dateisystemintegritätsprüfer zum automatisieren
- mtree, aide, tripwire
- Erzeugen Signaturdatei mit unterschiedlichen Attributen
- Eine Prüfsumme (SHA3 oder SHA2) reicht in der Regel aus
- Signaturdatei sicher aufbewahren oder mit GnuPG signieren
- Muss bei jedem Schreibzugriff aktualisiert werden


Rsnapshot
Rclone
Rclone
Rsync
Duplicity
Rclone
FTP Folder Sync

Fragen?

- stefan.schumacher2 <at> dataport.de