Universität zu Köln

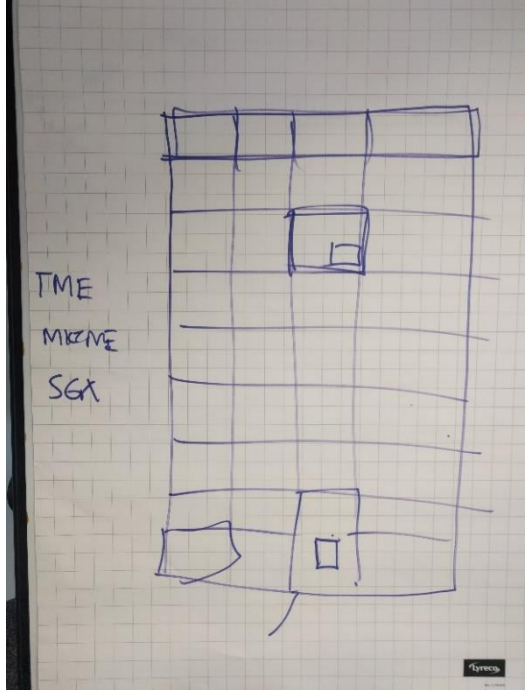
Verschlüsselung mit GPFS Advanced Edition

Regionales Rechenzentrum der Universität zu Köln | Abteilung HPC und Visualisierung | Dr. Roland Pabel

Motivation Verschlüsselung

- Sensible Daten vor unbefugtem Zugriff schützen
  - Life Sciences (Genomsequenzierungen)
- Klassische Angriffsszenarien und Lösungen:
  - Abhören: Transportverschlüsselung, Abschottung (RAM)
  - Diebstahl: SED HDDs, TME
  - Privilege Escalation: 2FA, SELinux, Kerberos
- ParFS mit Verschlüsselung auf Dateiebene
  - Schützt idealerweise gegen alle 3 Angriffsszenarien
  - Dateiebene flexibler und feingranularer konfigurierbar
    - Abschottung der User untereinander
    - Root entmachten

Verschlüsselung in CPUs: Intel



- TME: Total Memory Encryption
  - Verhindert Cold Boot Attacks
  - Komplett transparent
  - … was ist mit RDMA?
- MKTME: Multi-Key TME
  - VMs auf gleichem Rechner abschotten
  - Transparent, wenn in HyperVisor integriert (Docker? CGroups? Leider nicht auf der Roadmap…)
- SGX: Software Guard Extensions
  - Geschützte Enklaven im Arbeitsspeicher
  - Z.B. um Passwörter im RAM zu schützen
  - Benötigt Änderungen am Code (Key Management)

Verschlüsselung löst Probleme

- Angreifer hat physischen Zugang zum System:
  - SED: Festplatten mit sensiblen Daten wertlos (Austausch, Entsorgung, Diebstahl)
  - TME: Cold Boot Attacks sinnlos
- Angreifer ist User/root auf dem System
  - MKTME/SGX: Speicherbereich nicht sinnvoll auslesbar (CPU/RAM Exploits)
  - Encryption at Rest: Storage nicht sinnvoll auslesbar (dd)
  - Transportverschlüsselung (tcpdump, wireshark, …)
  - SELinux, Kerberos: Root darf nicht automatisch Rechte eines Users (und damit Zugang zu seinen Daten) bekommen können

Verschlüsselung macht Probleme

- Hauptproblem: Key Management
  - auf Hardwareebene sollte es transparent, aber konfigurierbar (skriptbar) sein
  - auf Softwareebene wird es idealerweise dem User aufgebürdet (Accountability)

- Ansonsten: Höherer Stromverbrauch, geringere Performance
  - Linderung durch Verschlüsselung in dedizierter Hardware (AMD/Intel CPUs)

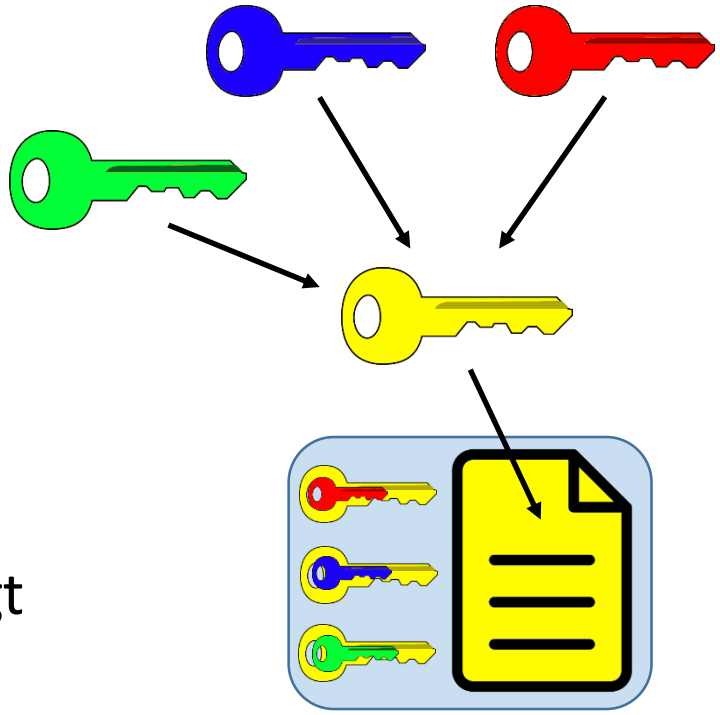
Verschlüsselung mit GPFS: Einkaufsliste

- IBM Spectrum Scale Advanced Edition
  - Server Lizenzen
    - Lizensierung pro TiB (DME – Data Management Edition)
  - Keine Lizensierung für Clients nötig
- Externe Key Server
  - IBM Security Key Lifecycle Manager (SKLM) Lizenzen
    - Key Management Interoperability Protocol (KMIP)
    - Auch von anderen Herstellern und Open Source (OpenKMIP)
    - Hardwareanforderungen minimal (2 Kerne, 4GB RAM), VMs ausreichend
    - Auch benutzbar für SED Passwörter, u.ä.
  - Mindestens HA Pärchen, bis zu 5 Mirror möglich

Verschlüsselung mit GPFS: The Basics

- Nur Daten werden verschlüsselt, nicht Metadaten
  - Kleine Dateien dürfen nicht in MD gespeichert werden („Data in Inode“, außer mit SED HDDs)
  - Jeder Client sieht gesamte Verzeichnisstruktur, Dateinamen, etc …
- Entschlüsselung auf Client, nicht Server
  - Encryption At Rest (LROC/HAWC nur mit SED HDDs)
  - Transportverschlüsselung RPCs: `mmauth cipher`
- Key Management
  - Master Encryption Key (MEK)
  - File Encryption Key (FEK)

GPFS: Key Management



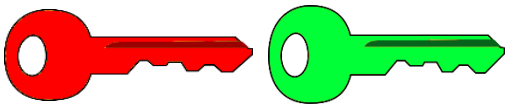
- FEK verschlüsselt Datei
- MEKs verschlüsseln („wrap“) FEK
- FEK kann von verschiedenen MEKs verschlüsselt werden (wFEKs)
- wFEKs werden in EAs der Datei abgelegt
- wFEKs werden automatisch erzeugt
  - max. 8 Encryption Policies pro Datei, 1 MEK pro Policy

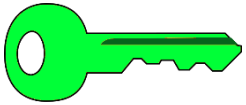
- Im Key Server
  - „Clients“ werden den MEKs zugeordnet
    - Key Management erlaubt Veränderungen an den Keys ohne die Daten neu verschlüsseln zu müssen
  - MEKs werden nur auf dem Key Server gespeichert
    - Keys werden eine Zeit lang auf Clients vorgehalten
    - Authentifizierung per Zertifikat/Passwort

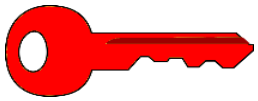
Verschlüsselung in GPFS: Ablauf

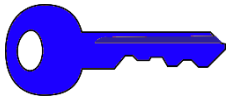
- Wie läuft die Verschlüsselung ab?
  - Eine Datei wird mit einem zufälligen Schlüssel (FEK) verschlüsselt
  - Der FEK wird mit ausgewählten MEKs verschlüsselt („wrapped“) und alle verschlüsselten wFEKs in den Extended Attributes der Datei gespeichert
- Wie läuft die Entschlüsselung ab?
  - Hat ein Client einen passenden MEK, kann er einen* der verschlüsselten FEKs entschlüsseln (*) Vielleicht auch mehrere möglich?
  - Mit dem FEK kann dann die Datei entschlüsselt werden
- Leider: Clients == Rechner, nicht User!
  - Mit dedizierten Knoten ist eine Separierung auf Userbasis möglich?

Partitionierung per MEKs

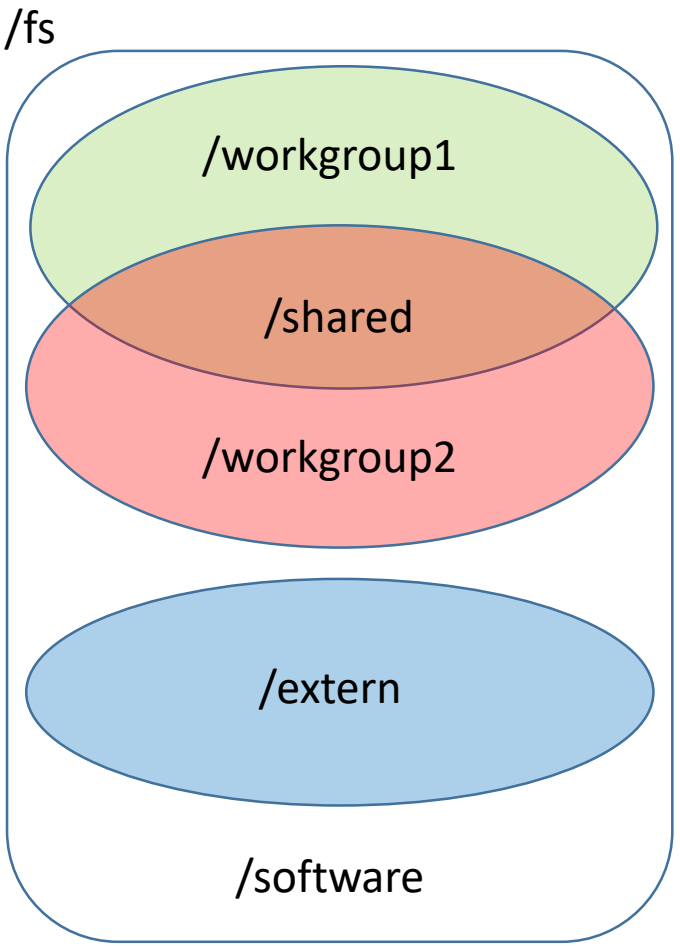
/fs/shared 

/fs/workgroup1 

/fs/workgroup2 

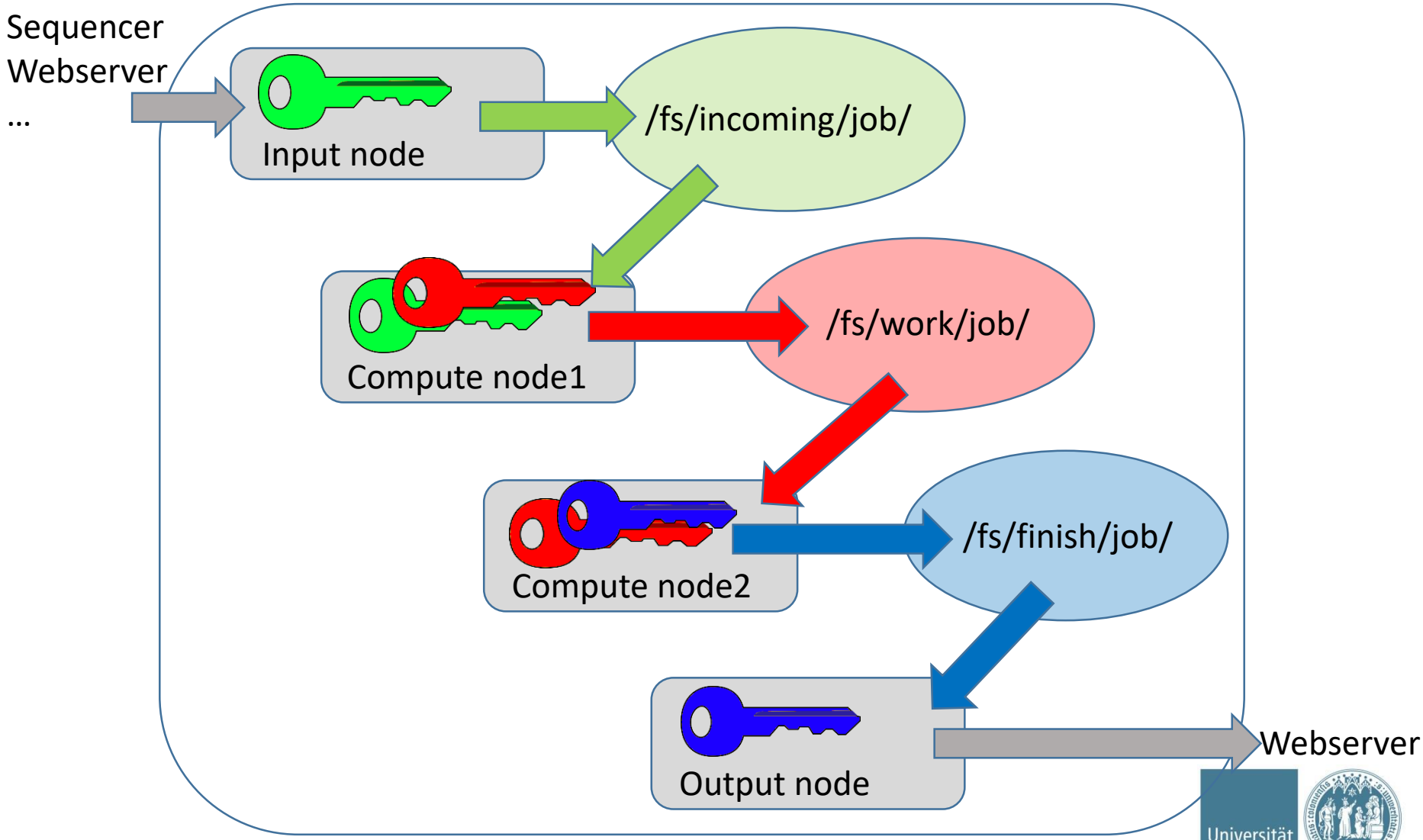
/fs/extern 

/fs/software



- MEKs flexibel per Policies zuweisbar (mmchpolicy), abhängig von Kriterien wie Pfad, Name, Fileset, …

MEK Pipelining



Technische Details

- Verschlüsselungsalgorithmen
  - AES 128/256 XTS, HMAC SHA512 (empfohlen)
  - AES 128/192/256 CBC, HMAC SHA512
- Administration Key Server
  - IBM SLKM Webserver GUI
    - Tenant: SKLM Group holding encryption keys
  - mmkeyserv command
    - `mmkeyserv tenant add t1`
    - `mmkeyserv client register c1 --tenant t1`
    - `mmkeyserv key create --tenant t1 --count 3`
- Konfiguration RKM.conf
  - Lokal: /var/mmfs/etc/, global: /var/mmfs/ssl/keyServ/
  - RKM Stanzas: Keyserver URI, TentantName, Server-zertifikat und Passwort

Zusammenfassung

- GPFS Adv.Edt.: „Transportverschlüsselung und SED“
  - Ein Baustein im Sicherheitskonzept (mit (MK-)TME u.ä.)
  - Zu diesem Zeitpunkt ist die GPFS Verschlüsselung eine effektive Methode den Cluster zu partitionieren
  - Dedizierte Knoten emulieren Partitionierung auf Userbasis
- Key Management
  - Erlaubt Zugang zu Datei zu ändern, ohne die Daten neu verschlüsseln zu müssen (mmchpolicy)
  - User ohne einen MEK sehen den gesamten Verzeichnisbaum (Metadaten), aber der Inhalt ist unlesbar
  - Funktioniert nur innerhalb von GPFS
    - Nicht mit TSM (Spectrum Protect), aber HSM (ILM, Cloud)

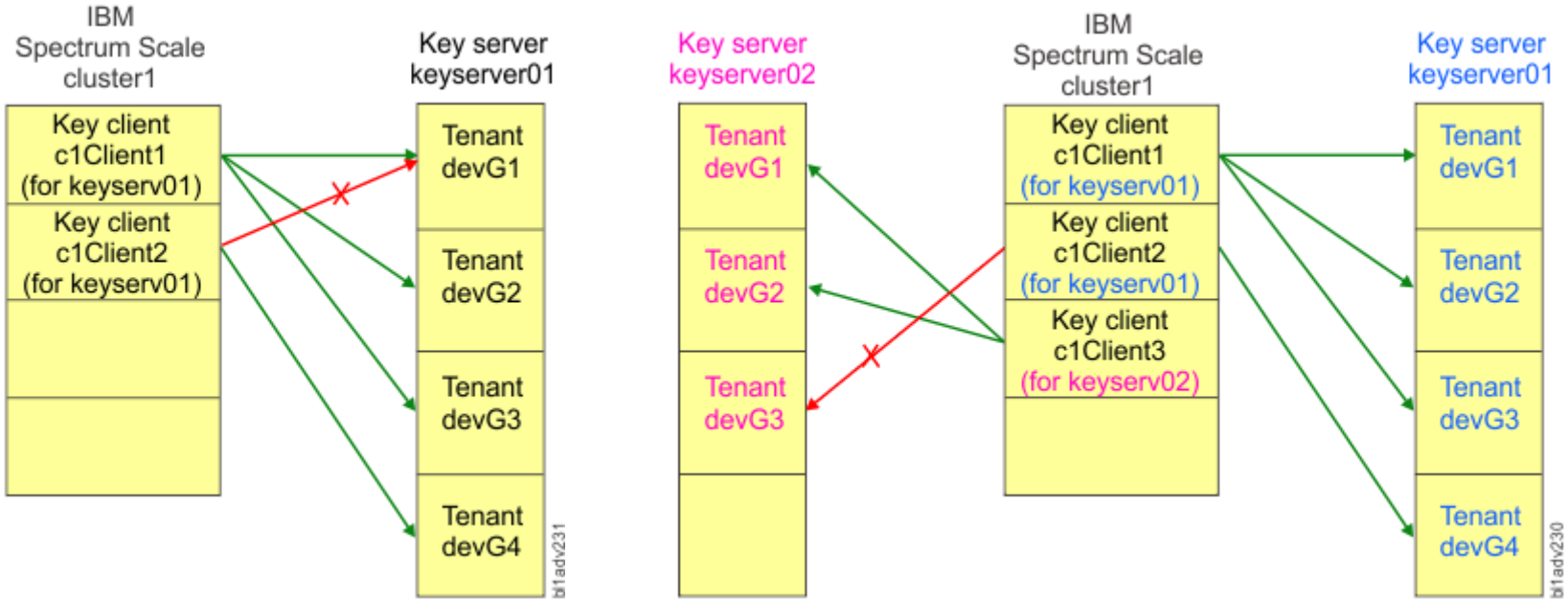
Ausblick

- Die Konkurrenz
  - Multitenancy Lustre (DDN): Network Isolation
  - Key Management in Xeons: Intel Key Protection Technology (KTP), Quick Assist Technology (QAT)

- Nächstes Mal (voraussichtlich):
  - Beispiele, Tests
  - Performance

KeyServer Tenants: Mapping

- KeyServer Mirroring (max 5 Mirrors)
- 1 Cluster, 1 KeyServer
- 1 Cluster, 2 KeyServer



Verschlüsselung in Festplatten

- SED: Self-Encrypting Drives
  - Praktisch kostenlos (minimal erhöhter Anschaffungspreis/ Stromkosten), kaum/kein Performanceverlust
  - Nutzlos, wenn der Key nicht geändert wird
  - Praktisch um kaputte Festplatten einfach austauschen/entsorgen zu können …
  - … außer der Hersteller kennt den Key, weil man ihn im Storagesystem speichern muss.