

Data Protection and Access-accounting with Trusted Storage

Deepak Garg

Joint work with: Anjo Vahldiek, Eslam Elnikety, Aastha Mehta,
Peter Druschel (**MPI-SWS**)

With contributions from: Ansley Post (**Google**), Rodrigo Rodrigues
(**Universidade Nova de Lisboa**), Johannes Gehrke (**Cornell University**)

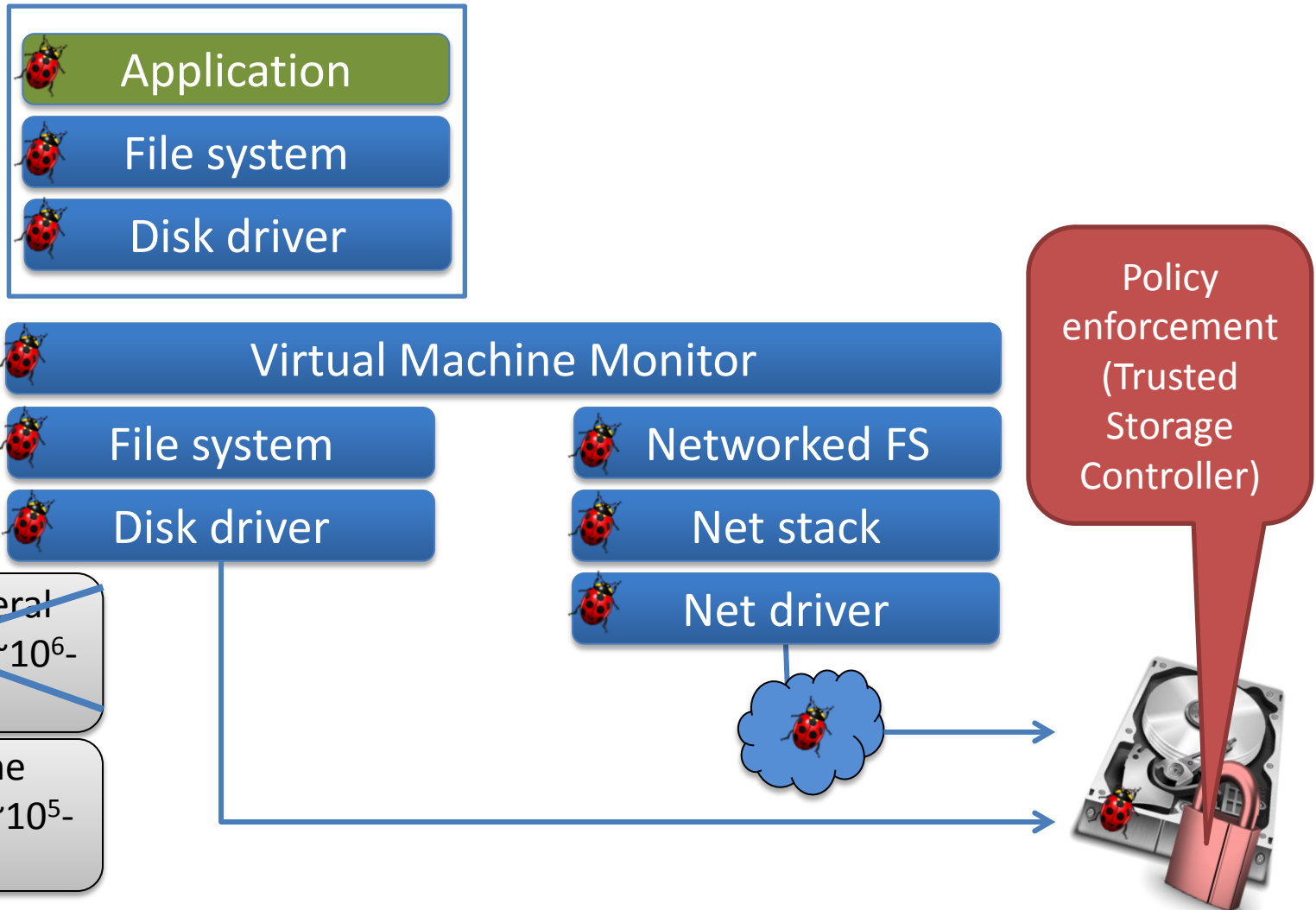


Max
Planck
Institute
for
Software Systems

Motivation: Data Protection and Accountability in the Cloud

- Growing data in third-party environments (e.g., Clouds)
- Data protection and access-accounting necessary
 - Legal and organizational mandates
 - Loss of confidentiality, integrity, access accounts costly
 - (E.g., electronic medical records, financial, corporate data)
- **Increasingly complex storage infrastructure, administration**
- **Many threats, despite best intentions** of Cloud provider
 - Hardware failures, defects
 - Software bugs, vulnerabilities
 - Misbehaving apps
 - Human errors (e.g., misconfigurations)
- Goal: Provide data confidentiality, integrity and access accounting, with minimal trust in storage infrastructure.
- Scope: Persistently stored data. Currently, confidentiality, integrity, verifiability, access-accounting (privacy → next step)

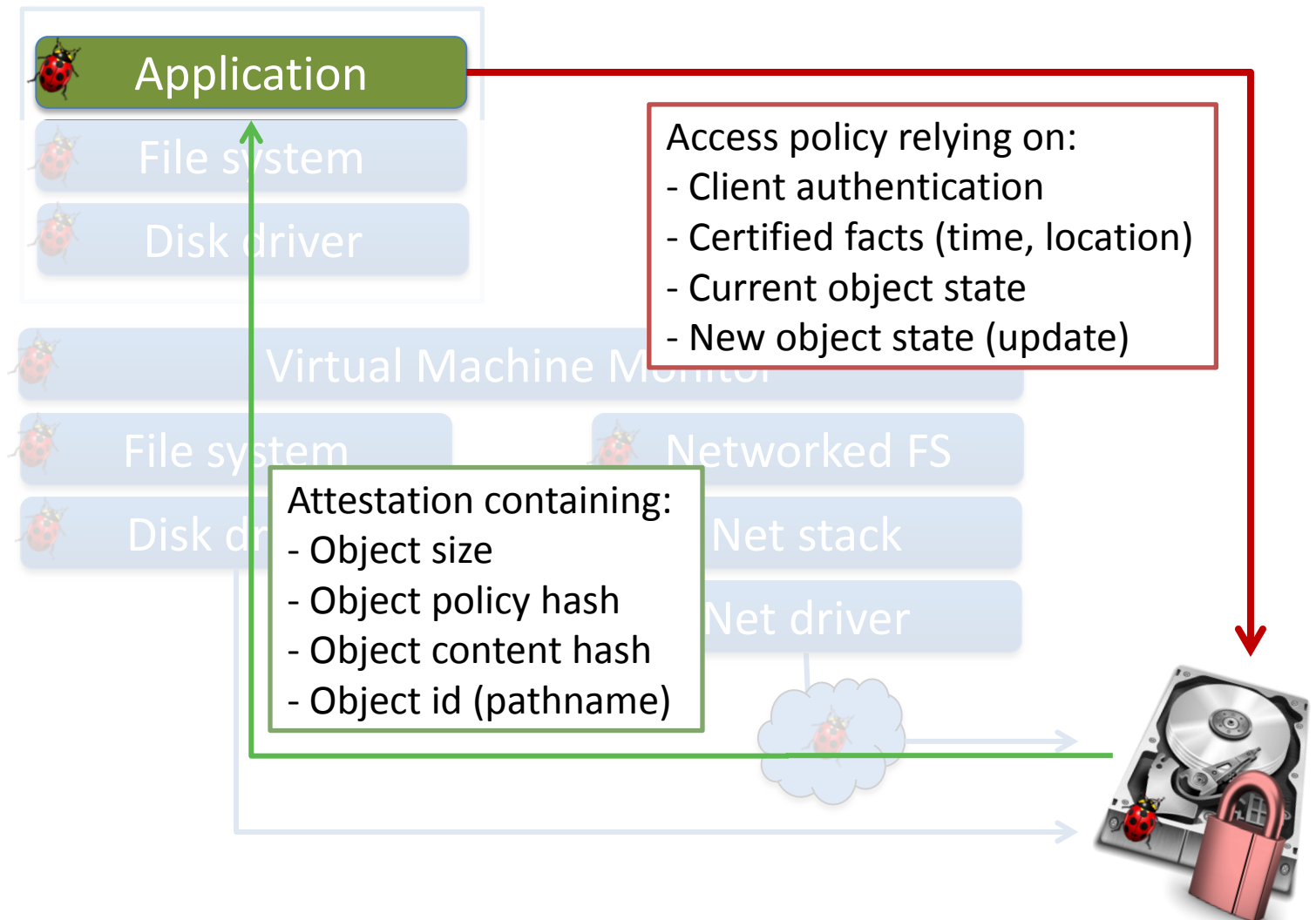
Trusted Storage



Trusted Storage Controller (TSC)

- **An interpreter for very rich policies**
- Policies are provided in a declarative language, by applications
- Policies are applied to **all** reads and writes by TSC (confidentiality and integrity)
- TSC can **attest state of stored data** with embedded private key (verifiability)
- Additionally, TSC implements:
 - An **object-level API** for stored data
 - A **transaction semantics** on individual objects (facilitates integrity checks)
 - Authentication protocols, **cryptographic channels**
 - Verification of **third-party policy certificates** (e.g., x.509 or XACML)
 - Access to object content during policy evaluation
 - Primitives for **opaquely migrating** stored objects between TS devices
- How is additional functionality available to higher layers? (IOCTL?)
- Applications, Cloud orchestrate data operations (as usual), but data security relies on TSC

Trusted Storage Use



Threat Model

- Threat model: Honest, but curious or buggy provider
 - Bugs in storage infrastructure
 - Misconfigurations
- Must trust:
 - Trusted storage controller
 - External dependencies of policy (e.g., time server)
 - No physical attacks on TS enclosure
- Guarantee: All access complies with policy
- TS provides data confidentiality, integrity, verifiability and access accounting (*not* availability)

Example 1: Backup File Integrity

- Threat: Software bug, virus or operator error corrupts backup data stored in the Cloud
- Policy: No update before backup expiration date

update :- key_is(K, "TimeServer") \wedge
K signs time(T) \wedge T \geq expT

External policy
dependency

read :- <no constraints>

Example 2: Append-only Log

- Threat: Accidental or malicious truncation of system log file
- Policy: Allow only appends, except to a trusted system administrator

update :- (old_extents_are(Oext) \wedge
new_extents_are(Next) \wedge
is_prefix(Oext, Next)) \vee session_is(k_{ad})

Refers to both
old and new
structure

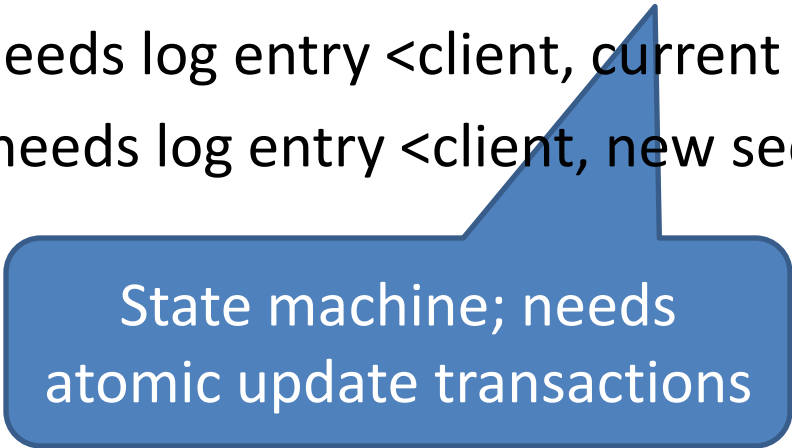
read :- <no constraints>

Example 3: Mandatory Access Logging

- Threat: Unaccounted access to data (e.g., medical records, pay-per-view content)
- Policy: Access allowed if descriptive entry is added to a designated append-only log file

Content file has a seq#, forcibly incremented during update

- Read needs log entry <client, current seq#, locus>
- Write needs log entry <client, new seq#, locus, content hash>



State machine; needs atomic update transactions

Evaluation

- Prototype in an iSCSI Enterprise Target (IET) SAN server, with small Flash memory for metadata
- Microbenchmarks:
 - Small throughput overhead (<0.75%)
 - Low (<0.5%) latency cost, except sequential read/write (2%/7%)
- Applications:
 - Webserver: TS makes log files append-only, protects content from unauthorized modification
 - Secure migration by storage provider: TS prevents third-party provider from reading data, but allows migration; forces N (>1) replicas at all times
 - Mandatory access logging: TS forces logs on all reads and writes, ensures consistency during synchronization of independently modified replicas (in progress)

Conclusion

- TS enforces application-defined policies on stored objects, attests objects
- Rich policies, enable confidentiality, integrity, access-accounting
- Relies only on TSC and policy dependencies
- Efficiently implementable
- Next work: TSC implemented in a VMM, information flow control in the Cloud (goal: privacy)
- Open questions (for future revisions)
 - A comprehensive study of security and privacy requirements for Cloud apps?
 - What do legal requirements entail for security primitives needed in the Cloud?