Running ZooKeeper Coordination Services in Untrusted Clouds

HotDep'14

Stefan Brenner Colin Wulf Rüdiger Kapitza

Institute for Operating Systems and Computer Networks
TU Braunschweig

Oct 5th, 2014

Technische Universität Braunschweig

Lack of Trust in Cloud Providers

- Privacy concerns slow down adoption of cloud technology
  - No trust in providers (especially sensitive applications)

> *"Snowden leaks aren't driving companies away from the cloud; but [...] made them a lot more careful"* [1]

> *"Most IT pros do not trust cloud services with sensitive data"* [2]

- Challenge: Privacy in cloud computing
  - Encryption of sensitive data
  - Trusted execution
  - Verification of software stack



[1] http://www.computerworld.com/article/2487123

[2] http://www.computerweekly.com/news/2240174444/Most-IT-pros-do-not-trust-cloud-services-with-sensitive-data

Technische Universität Braunschweig

Trusted Cloud

Establish Trust in the Cloud

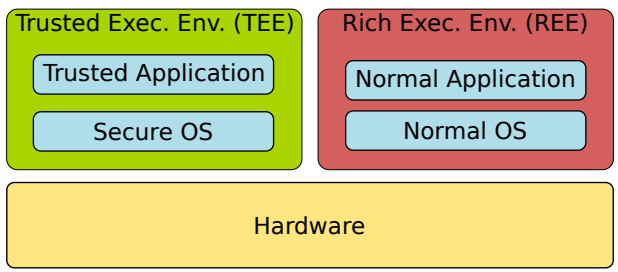
- Homomorphic encryption
- TPM-based trusted software stack
- Hardware-isolated environments



Trusted Execution Environment (TEE)

- Isolation enforced by hardware
  - Intel SGX, ARM TrustZone
- Application logic split into trusted and untrusted part
  - Process sensitive data only in TEEs
  - Verification of trusted software stack
  - Minimizing trusted codebase

$\rightarrow$ Fewer bugs reduce attack surface



Technische Universität Braunschweig

Trusted Coordination

- TEEs coming to cloud environments
  - Trusted distributed applications in the cloud
    $\rightarrow$ Trusted coordination

Trusted Coordination

- TEEs coming to cloud environments
  - Trusted distributed applications in the cloud
    $\rightarrow$ Trusted coordination
- Apache ZooKeeper – coord. service
  - Configuration management for distr. app.
  - Leader election



Technische Universität Braunschweig

Trusted Coordination

- TEEs coming to cloud environments
  - Trusted distributed applications in the cloud
    $\rightarrow$ Trusted coordination
- Apache ZooKeeper – coord. service
  - Configuration management for distr. app.
  - Leader election
- **Trusted ZooKeeper**
  - Transparent encryption layer to ZooKeeper
  - All functionality of ZooKeeper preserved



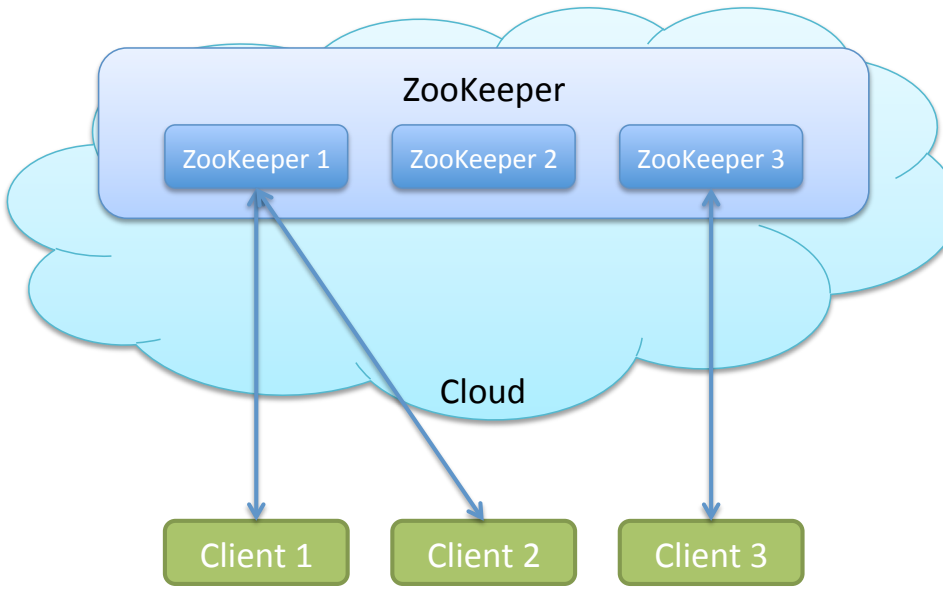
Technische Universität Braunschweig

ZooKeeper Privacy Proxy

System Architecture

ZooKeeper

- Data items: *znode*
- Callback mechanism
  - Update notification
- "Sequential znodes"
  - Leader election
  - Fairness for distr. lock



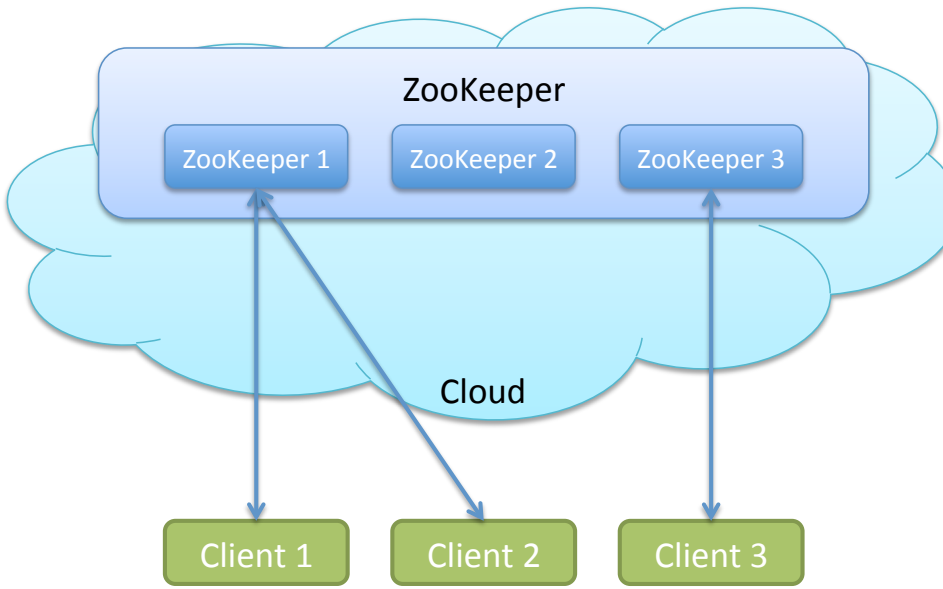
Technische Universität Braunschweig

ZooKeeper Privacy Proxy

System Architecture

Threat

- Provider/Attacker can eavesdrop all storage and transmitted messages



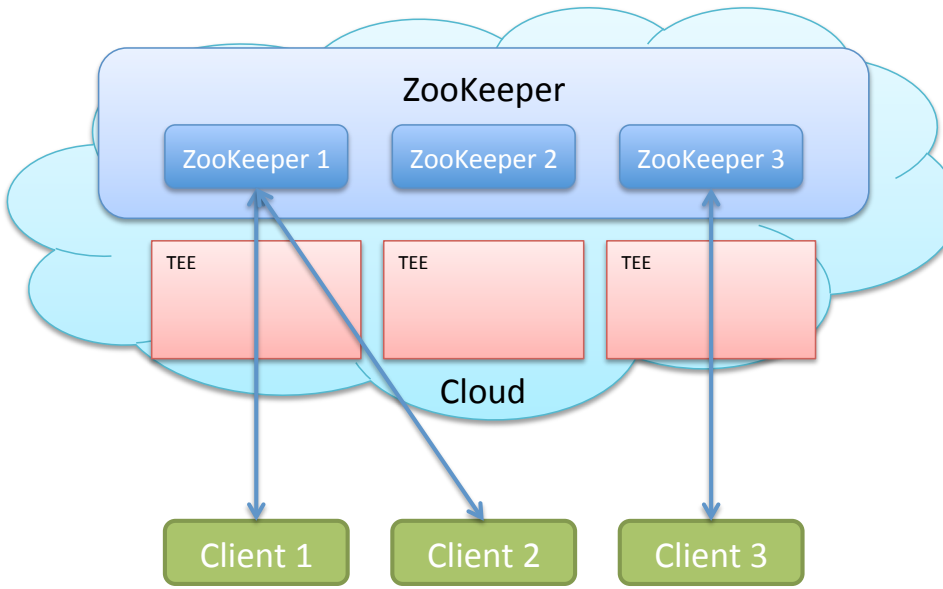
Technische Universität Braunschweig

ZooKeeper Privacy Proxy

System Architecture

Threat

- Provider/Attacker can eavesdrop all storage and transmitted messages
- Provider/Attacker can **not** look inside TEEs



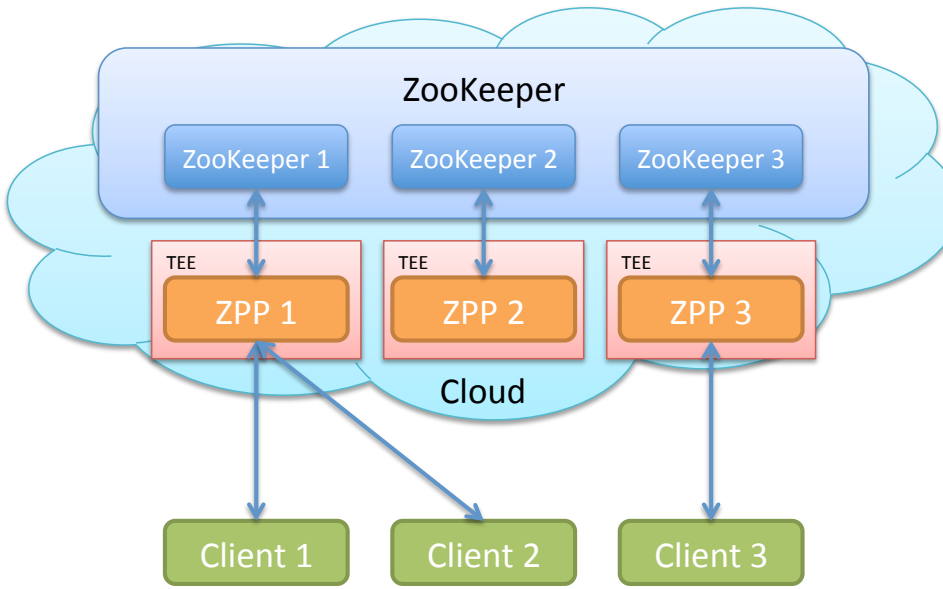
Technische Universität Braunschweig

ZooKeeper Privacy Proxy

System Architecture

Our approach:

- ZooKeeper Privacy Proxy (ZPP)
- Running inside a TEE
- Encryption of:
  - Payload of znodes
  - Names of znodes

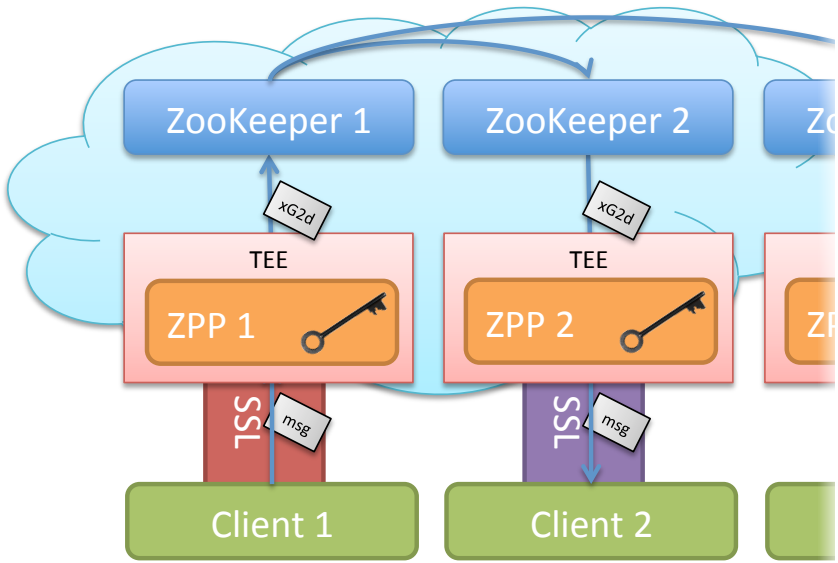


Technische Universität Braunschweig

ZooKeeper Privacy Proxy

Link Encryption

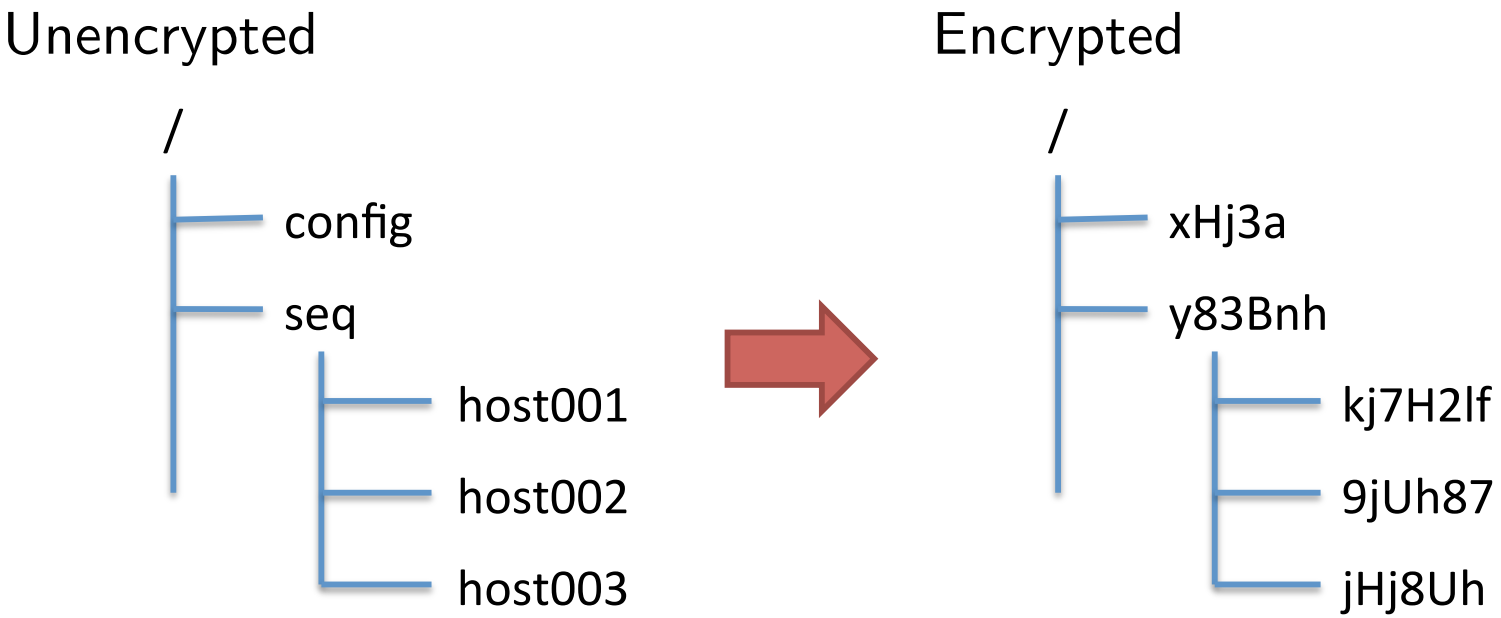
- Client to ZPP
  - SSL-encrypted transport between Client and ZPP
- ZPP to ZooKeeper
  - Shared key-encrypted payload and node names between ZPP and ZooKeeper



Technische Universität Braunschweig

ZooKeeper Privacy Proxy

znode Name Encryption



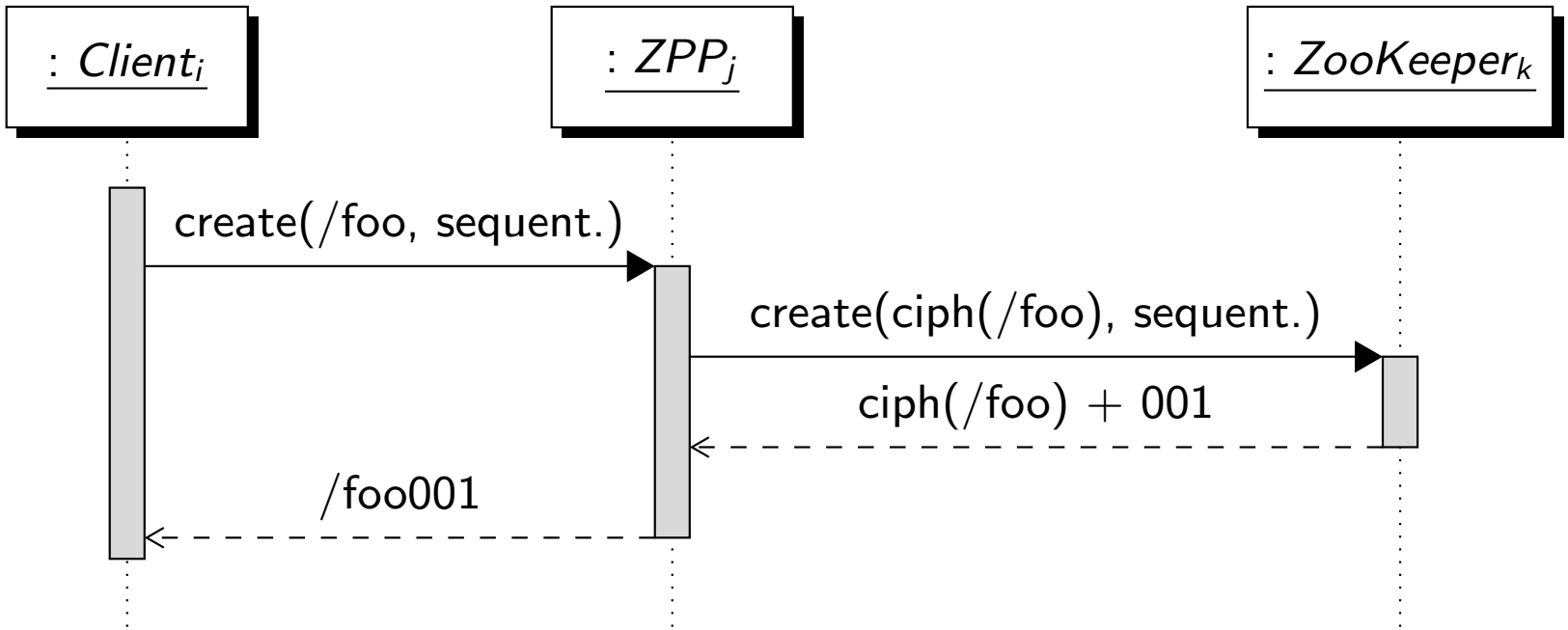
- Encryption of znode names
  - Encryption of individual parts of paths
  - Hierarchy of znodes preserved

  $\Rightarrow$ Works for most operations

Technische Universität Braunschweig

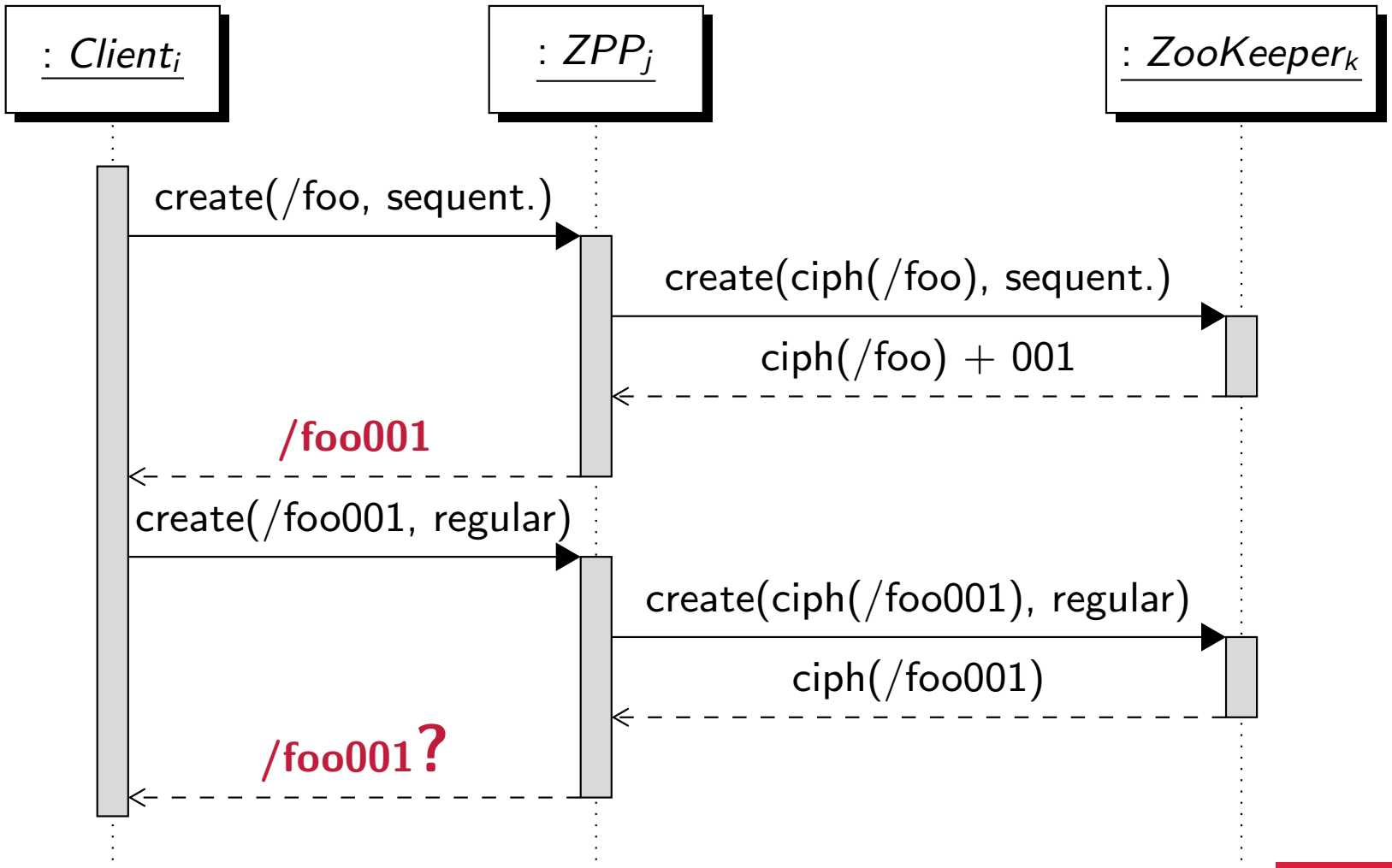
ZooKeeper Privacy Proxy

Name Clash Sequence Numbers



ZooKeeper Privacy Proxy

Name Clash Sequence Numbers

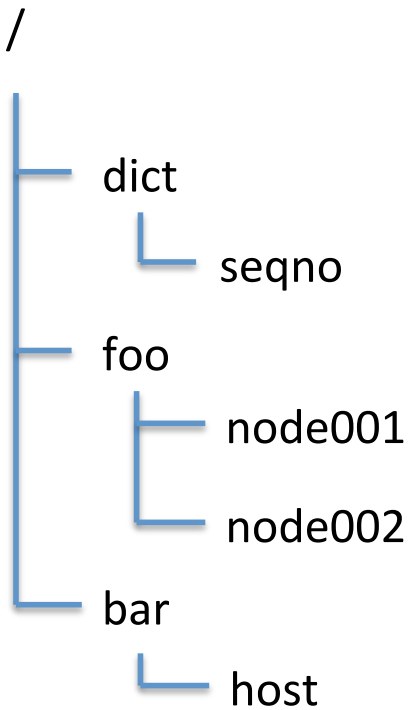


ZooKeeper Privacy Proxy

Supporting Sequential znodes

- "Dictionaries" (Helper znodes)
  - Additional ZPP-maintained data
  - Encrypted payload of helper znode
  - Stored inside ZooKeeper
  - Hidden from clients
- Sequence numbers
  - Mapping of znode name to next free sequence number
  - Example: Payload of `/dict/seqno`:

    `{ "dict": { "/foo": 3, "/bar": 2 } }`



Technische Universität Braunschweig

ZooKeeper Privacy Proxy

Dictionary Nodes

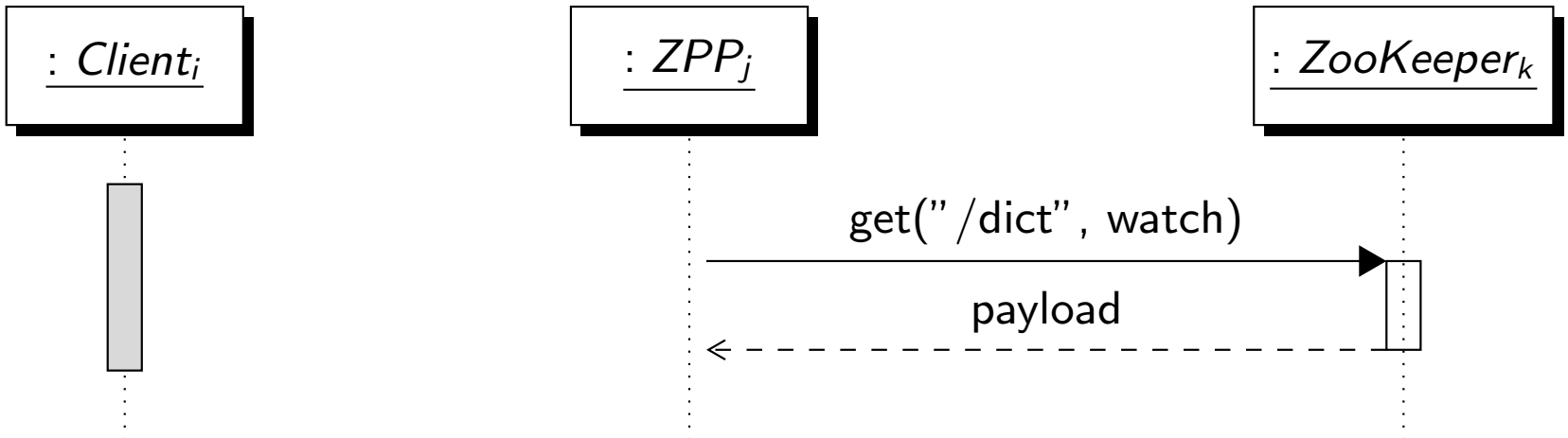
Dictionary Maintenance

- Dictionary in-memory at all ZPPs
- Monitoring dictionary causes callback on change
- **Example**: *create*() sequential node
  1. Sequence number appended to znode name
  2. Sequence number of parent znode incremented
  3. Atomically create znode and update the dictionary

Technische Universität Braunschweig

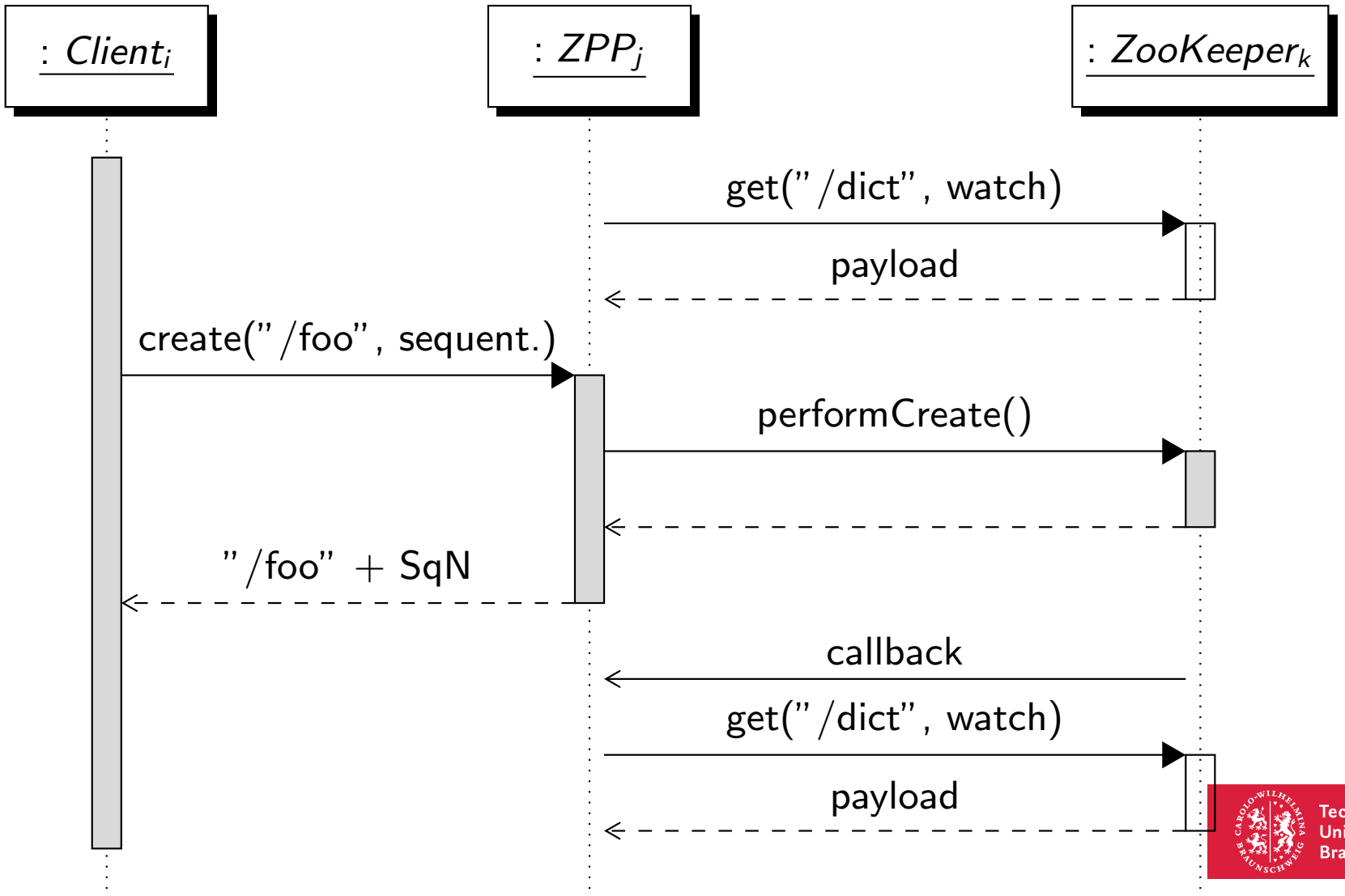
ZooKeeper Privacy Proxy

Dictionary Nodes (Overview)



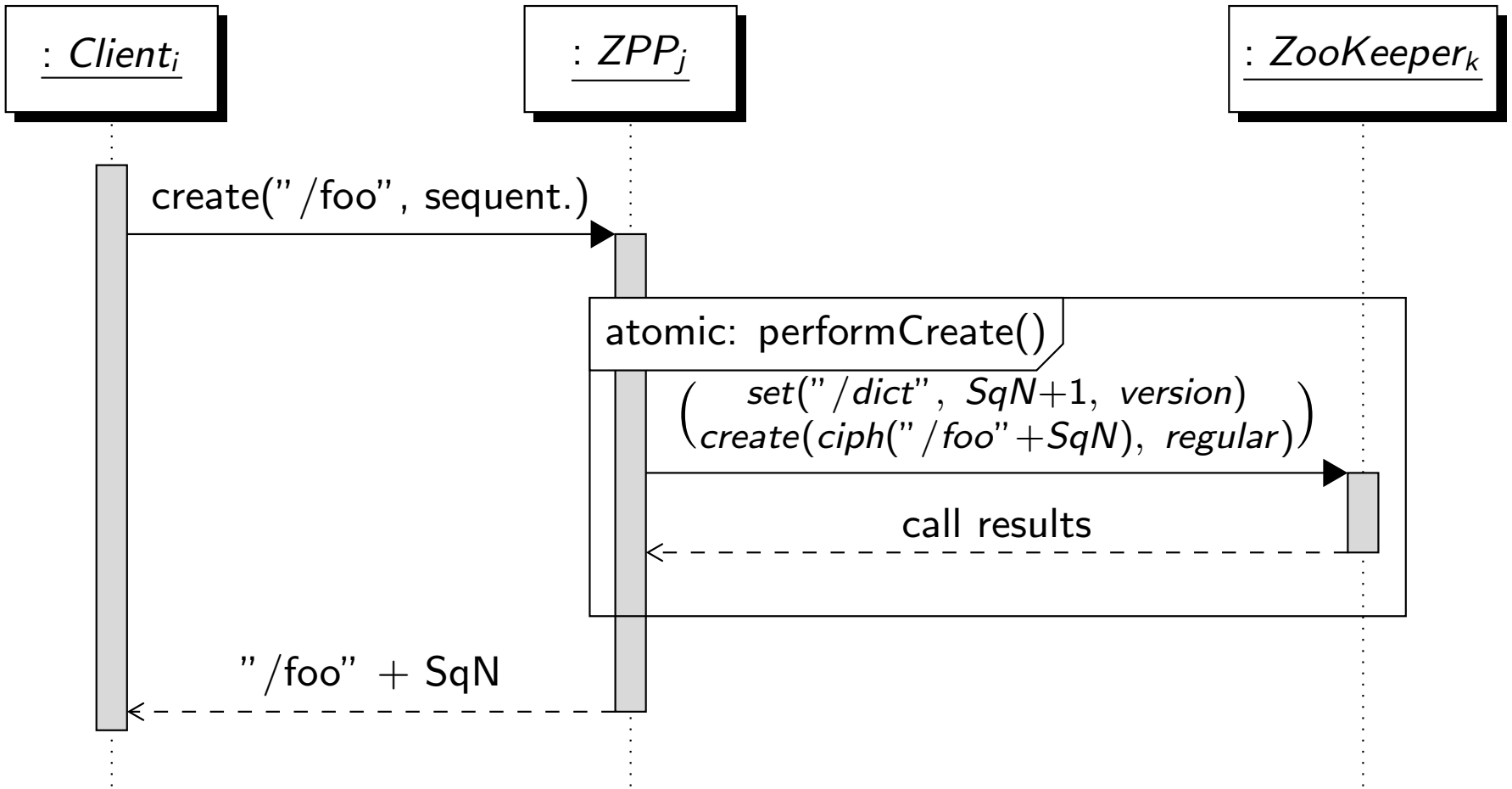
ZooKeeper Privacy Proxy

Dictionary Nodes (Overview)



ZooKeeper Privacy Proxy

Dictionary Nodes (*create*() detailed)



Related Work

Untrusted storage

- Venus: Verification for untrusted cloud storage (CCSW'10)
- Fail-aware untrusted storage (DSN'09)

Full SQL interface

- TrustedDB: A Trusted Hardware Based Database with Privacy and Data Confidentiality. (TKDE'13)
- The Blind Stone Tablet: Outsourcing Durability to Untrusted Parties. (NDSS'09)

Evaluation

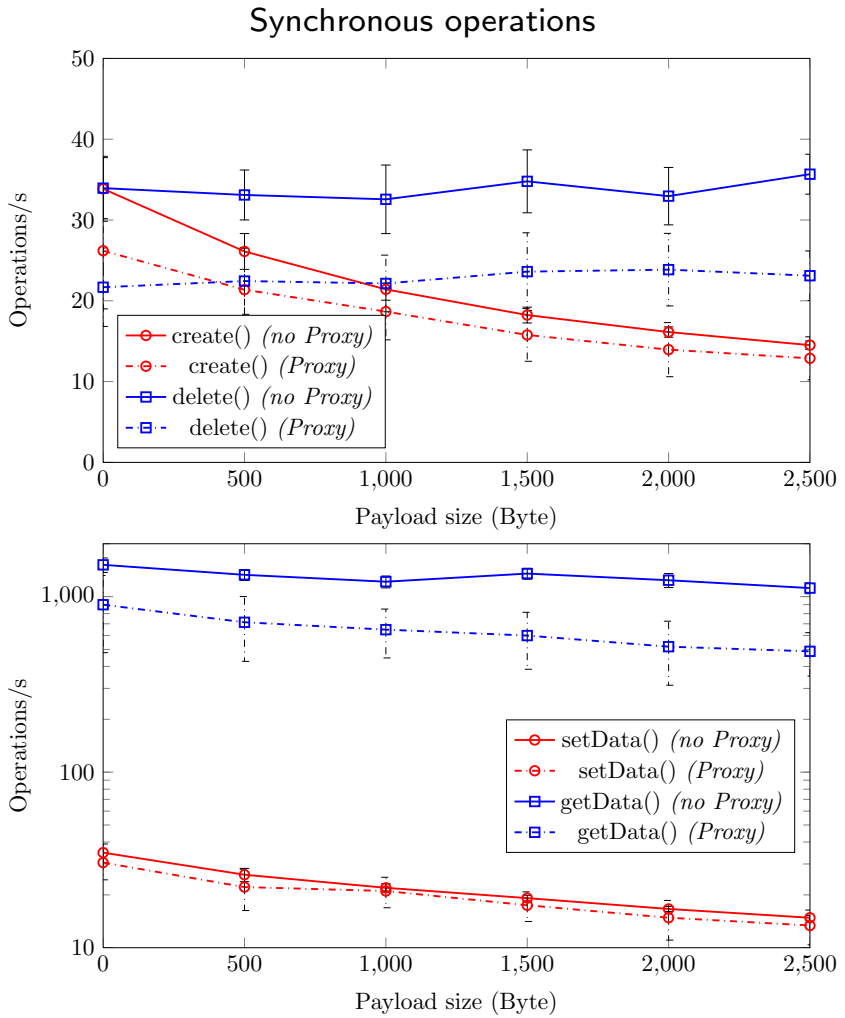
Experimental Setup

- OpenStack virtual machines
- 5 ZooKeeper replicas
- 1 ZooKeeper Privacy Proxy
- 1 Client VM

Experiments

- Batch of 500+ operations
- Various payload sizes

$\Rightarrow$ moderate throughput impact



Technische Universität Braunschweig

Summary

- No trust in cloud providers
  - Sensitive applications conflict untrusted providers
- ZooKeeper deployment in untrusted clouds
  - **ZooKeeper Privacy Proxy**
  - Application logic split in trusted and untrusted part
  - Encryption of all data stored inside ZooKeeper
  - All ZooKeeper functionality and semantics preserved
- SERECA: Upcoming EU project
  - Trusted execution of distributed applications in the cloud
  - TEEs: ARM TrustZone and Intel SGX

Technische Universität Braunschweig