ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

DECENTRALIZED ACCESS CONTROL

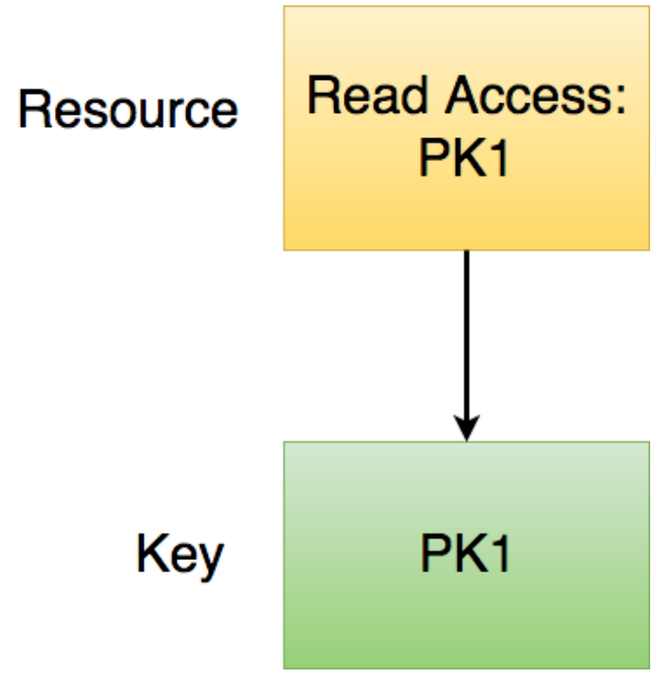
Sandra Siby

EDIC Semester Project (DEDIS)

Supervisors: Eleftherios Kokoris-Kogias, Prof. Bryan Ford

Motivation

- Access Control: *Management of access to a resource*
- Simple access control
  - Static binding of a resource to a key
- Key changes by owner?
- Organization of users into groups?
- Multiple rules or rules with conditions?
- Changes to access control rule?



Project Aim

- Design and implement a system that achieves the following:
    - Creation and management of access control rules
    - Allows users to manage their identities independent of the access control rules
    - Creation and management of groups of users for better organization
    - Evolution of identities and access control with time

Related Work

- At DEDIS
  - Managing Identities Using Blockchains and CoSi [1]
    - CISC (Cisc Identity Skipchains) [2]
    - DARC (Distributed Access Rights Control)
- Blockchain Based Access Control [3]

System Overview

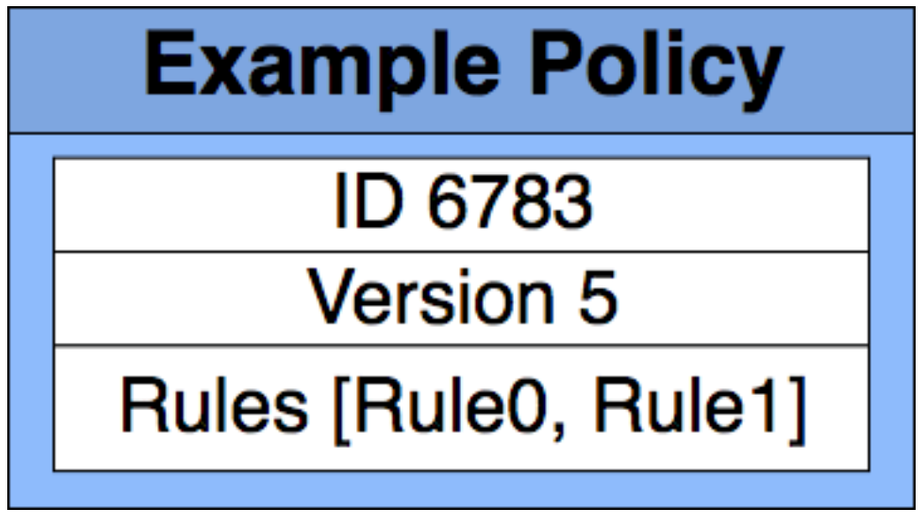
- Several types of access control: DAC, MAC, RBAC
  - ABAC - Attribute Based Access Control
  - Usage of **Policies**
- **JSON** based access control language to express policies
- Design
  - Policy Structure
  - Access Requests
  - Request Verification

Design Overview

- Policy Structure
- Access Requests
- Request Verification

Policy

- Consists of:
  - ID
  - Version
  - List of Rules



JSON based language

```
{
    "ID" :  6783,
    "Version" : 5,
    "Rules" : [Rule0, Rule1]
}
```

Rule

- Consists of:
  - Action (user specified)
  - Subjects
    - Public Key
    - Another Policy ID
  - Expression

Example Rule

Action Read
Expression "{'AND' : [0,1]}"
Subjects [GroupA_ID, Bob_PK]

JSON based language

```
{
    "Action" : "Read",
    "Subjects" : [GroupA_ID, Bob_PK],
    "Expression" : "{'AND' : [0, 1]}”
}
```

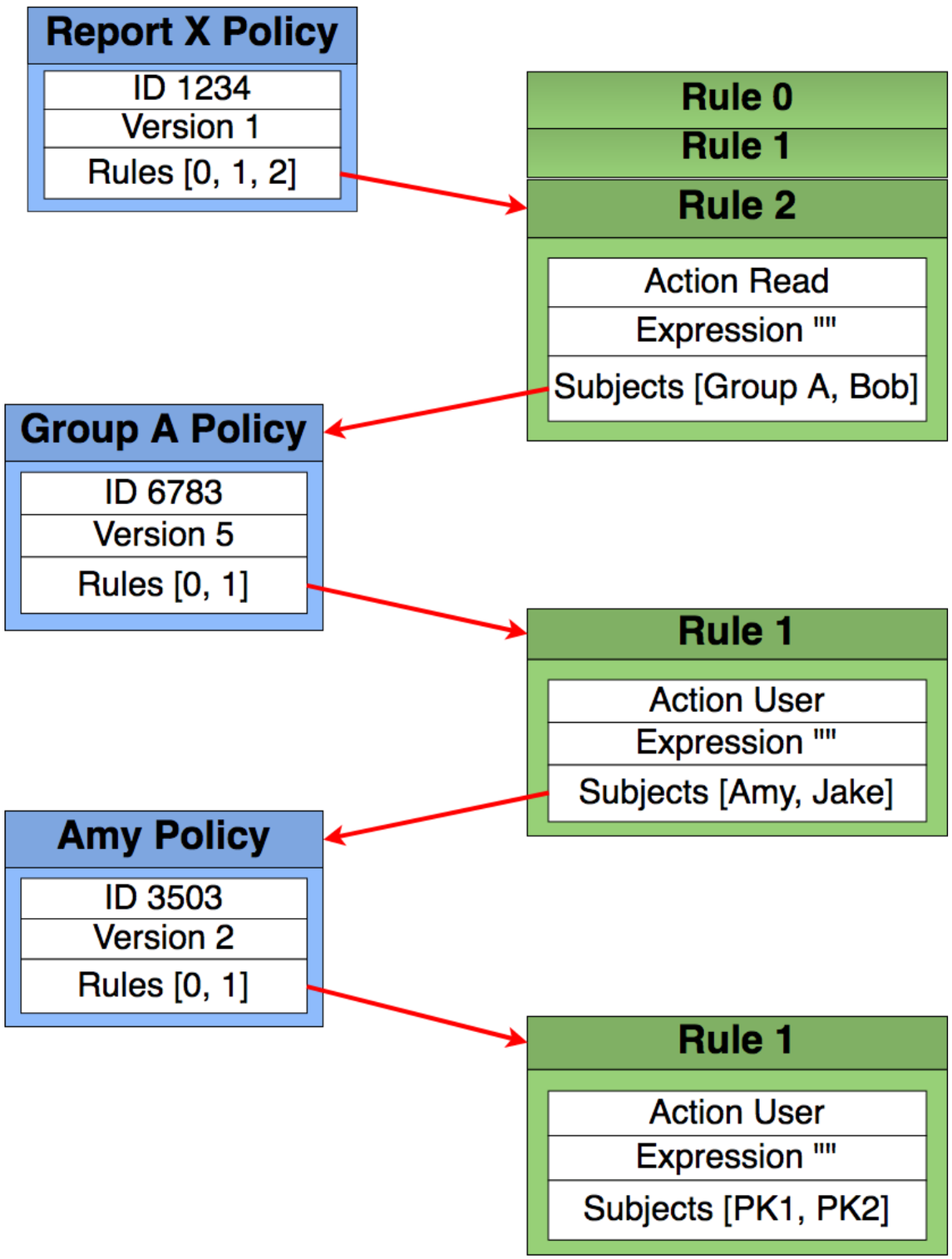
Expressions

- Allows for more sophisticated conditions
- Basic format: {operator : [operand]}
- Operations can be combined to build complex expressions
- Examples:
  - “Need S1 and S2’s approval” -> S1 AND S2 -> {“AND” : [S1, S2]}
  - “Need either S1 and S2 or S3 and S4 to approve” -> (S1 AND S2) OR (S3 AND S4) -> {“OR” : [{“AND” : [S1, S2]}, {“AND” : [S3, S4]}]}
  - Current functionality: Logical operations AND/OR/NOT

Example

- Access Control and Identity Management can be achieved using policies

- Admins for policies



Design Overview

- Policy Structure
- Access Requests
- Request Verification

Access Requests

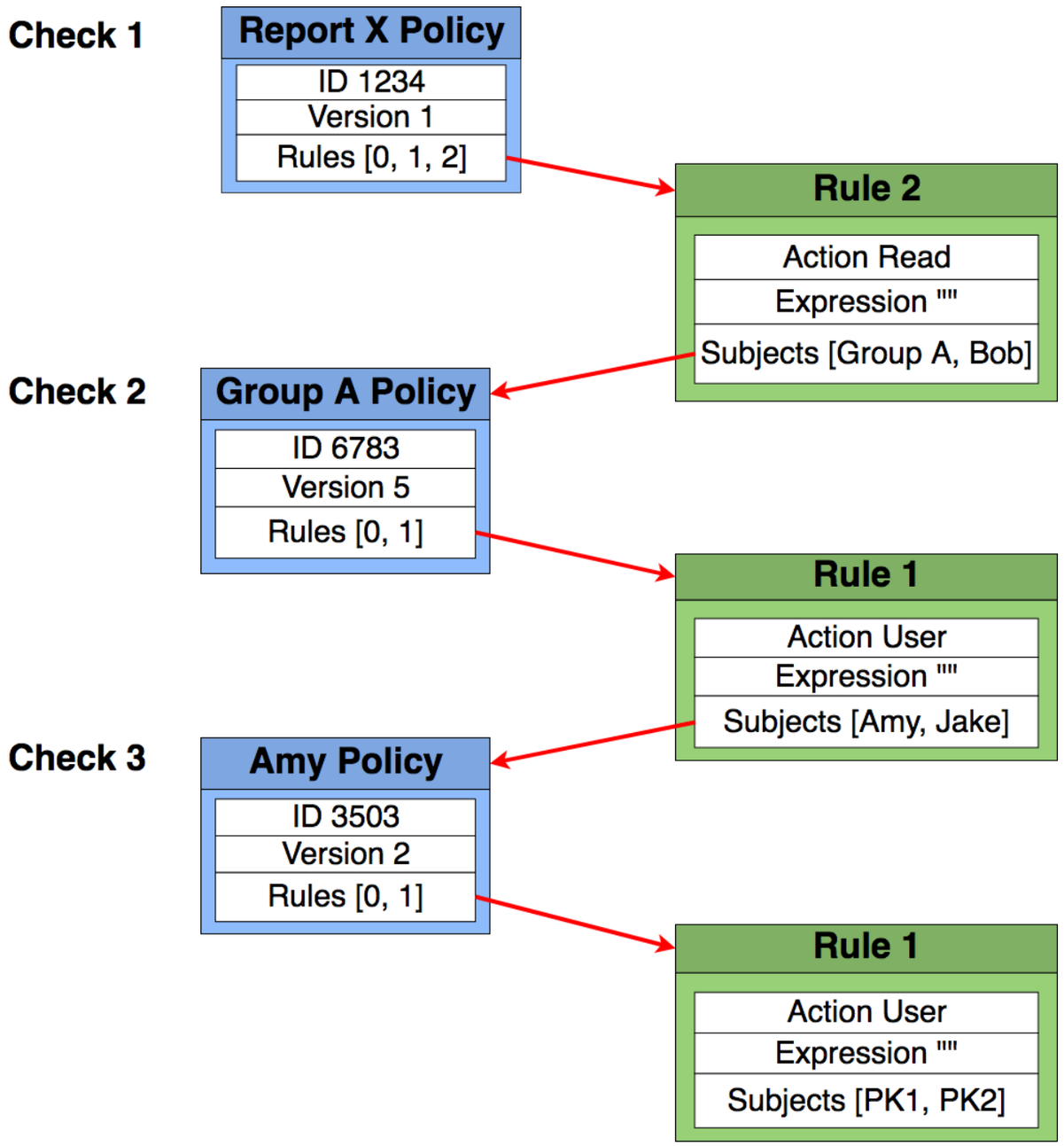
- Request consists of
  - Policy ID - target access policy
  - Rule Index - specific rule indicating access
  - Message - extra relevant information
- Signing
  - Requester signs request with signing key
  - Request Signature consists of signed request and requester’s public key
- Requester sends **Request + Request Signature**

Design Overview

- Policy Structure
- Access Requests
- Request Verification

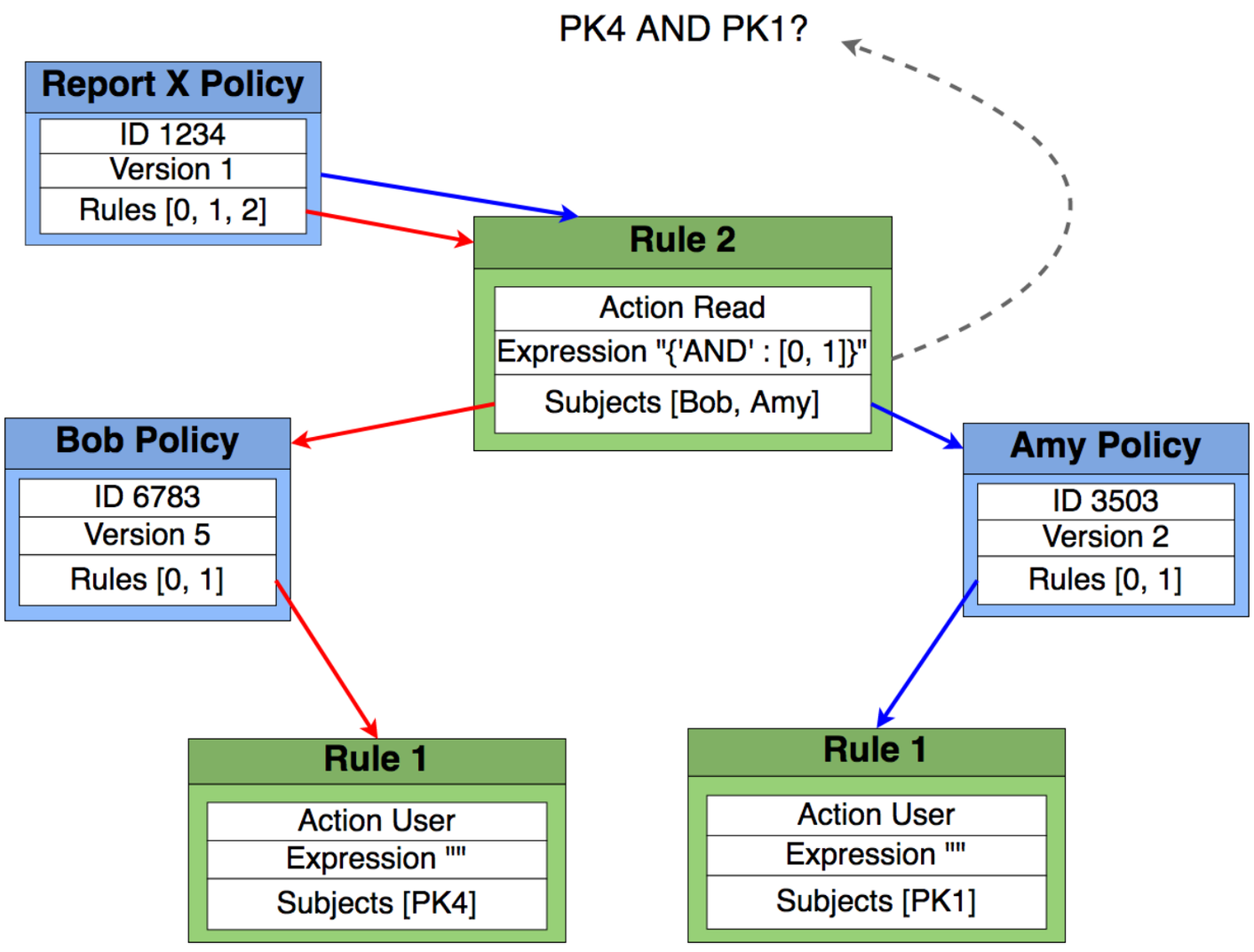
Verification

- Verifier checks signature
- Verifier checks path from access policy to requester



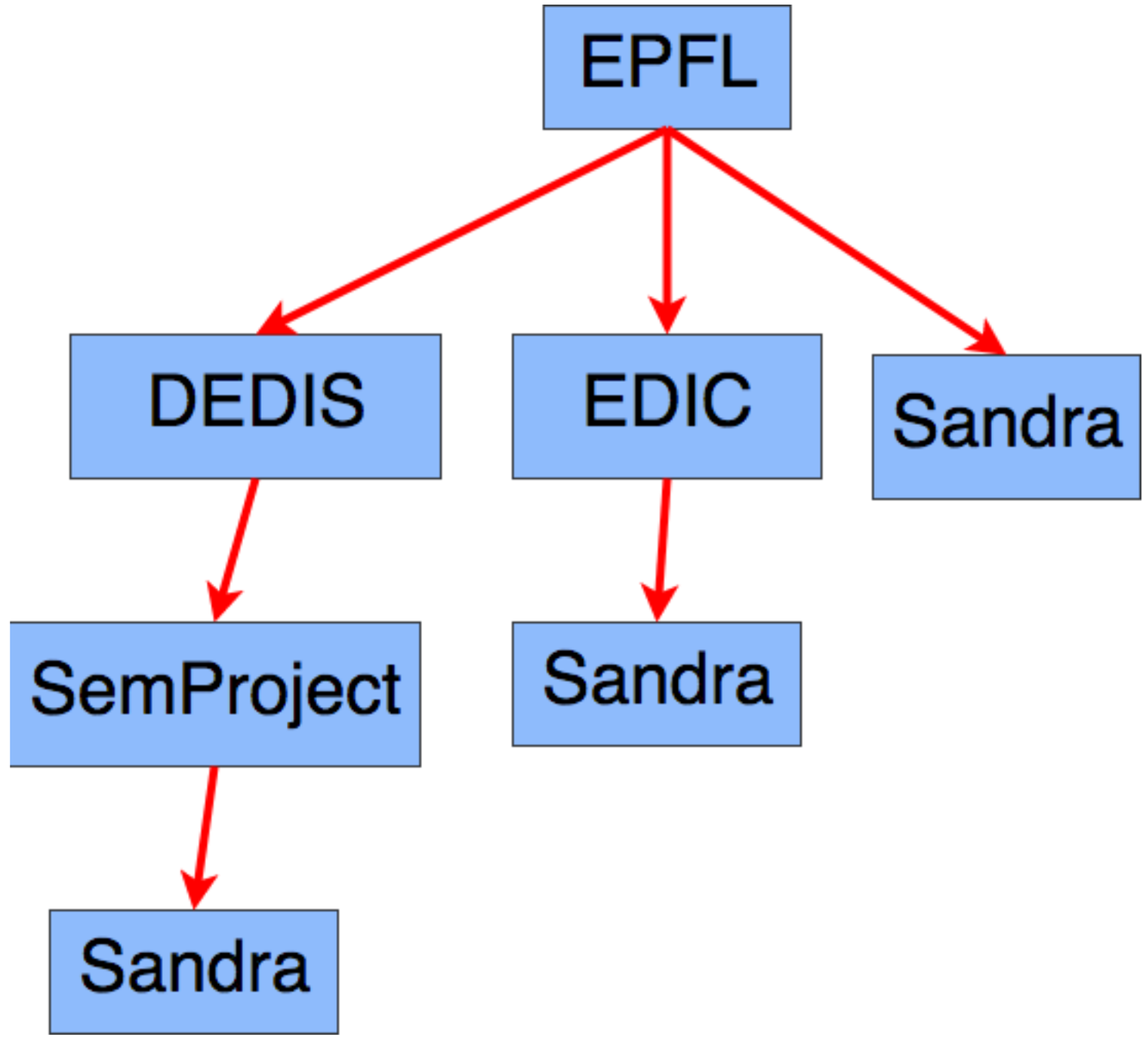
Verification - Multisig

- Requester sends Request + List of signatures
- Verifier checks all signatures
- Verifier checks all paths
- Verifier validates expression



Verification: Multiple paths?

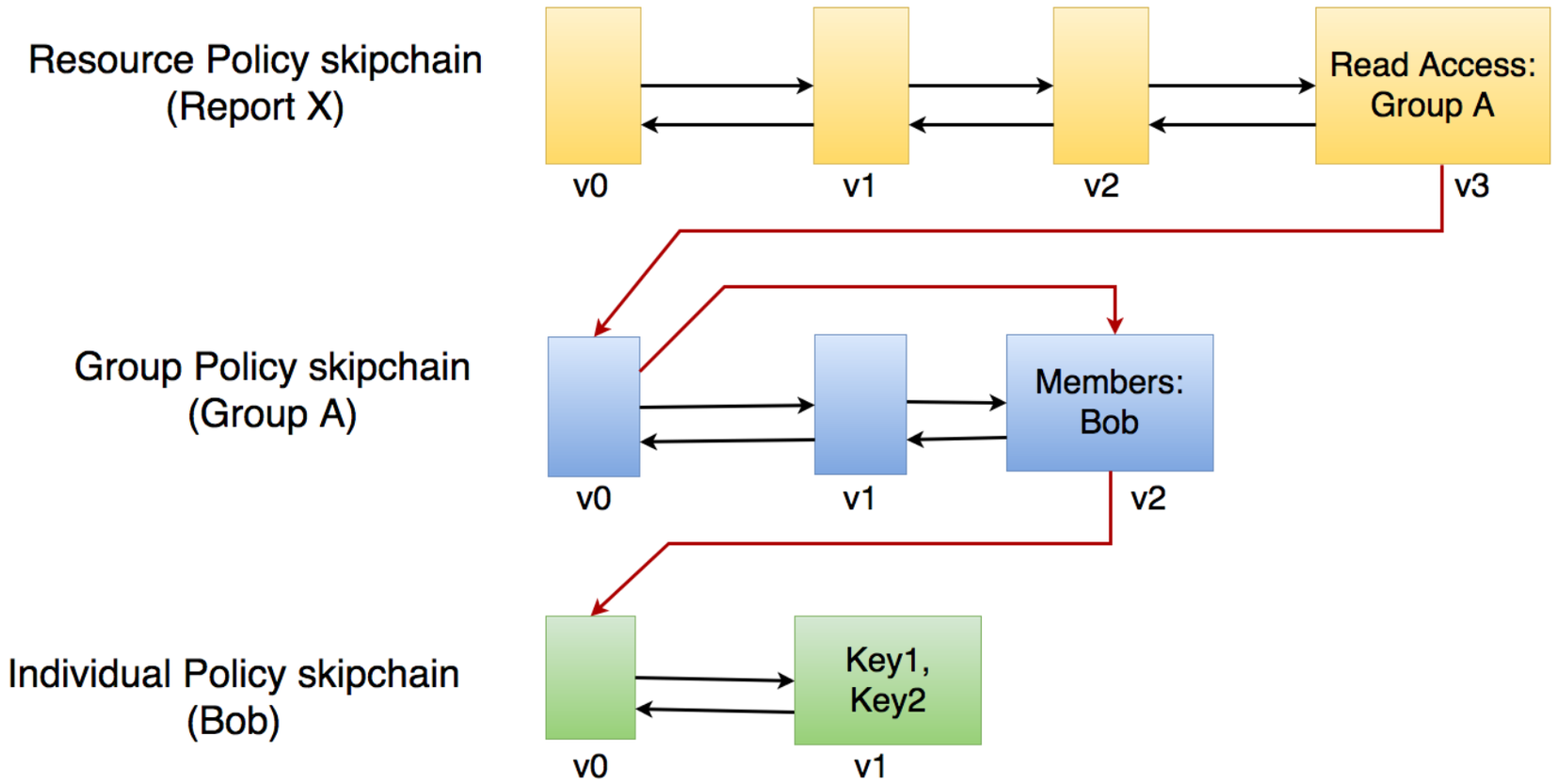
- Example: Request needs signature from member of EDIC
- Mechanism to choose path required
- On verifier or requester side



Path Selection by Requester

- Signing
    - Requester searches for all paths
    - Picks appropriate path
    - Sends path information with request
        - Request Signature consists of signed request + public key + path
- Verification
    - Verifier checks signature
    - Checks path in requester’s message
    - Checks presence of key in path

Evolving Policies



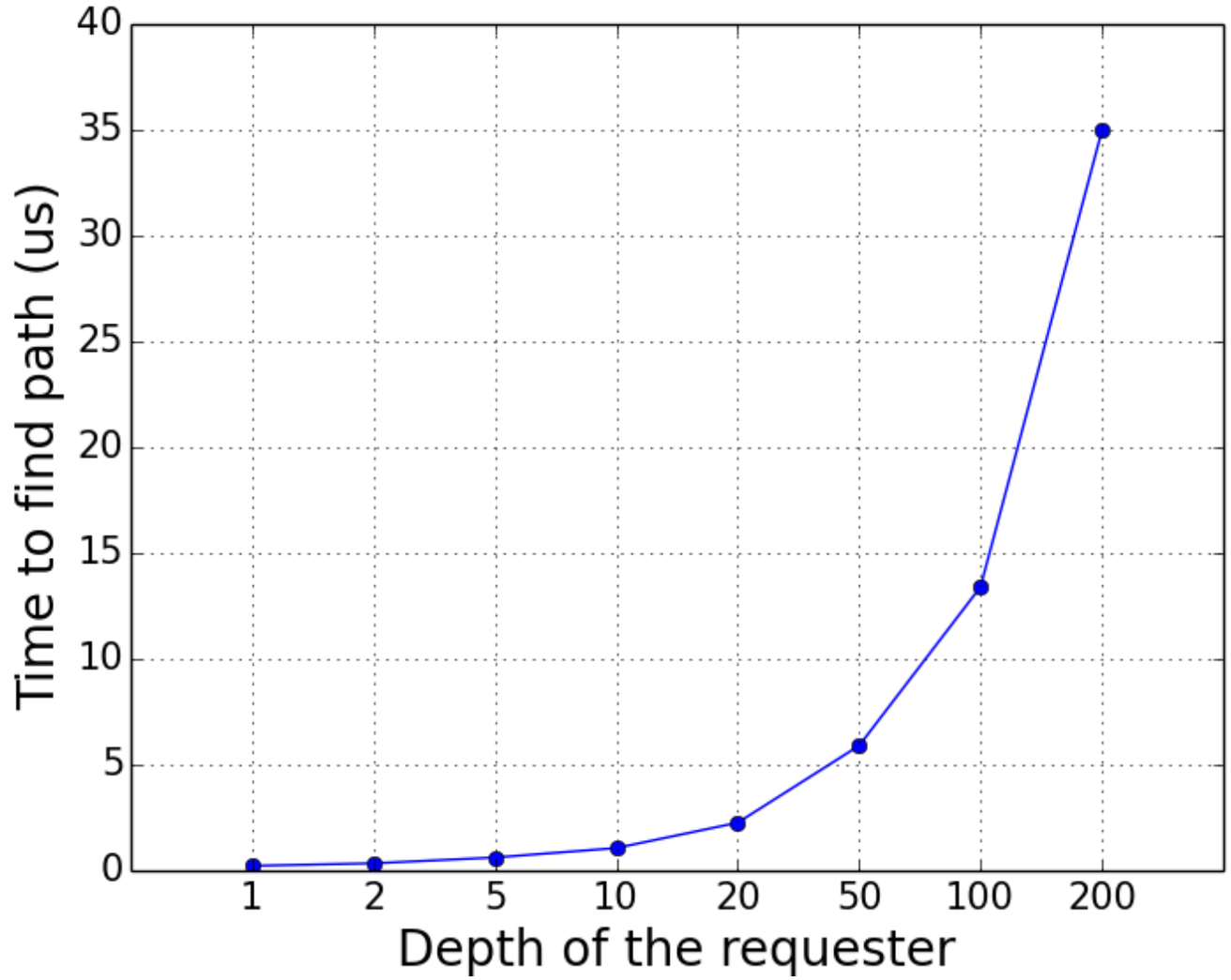
- Skipchain architecture can be used (integration to be done)
- Each policy object gets a skipchain
- Allows for verified record of all policy changes over time
- Skiplinks assist in fast traversal during path search

Evaluation

- Unit tests to check implemented functionality
- Benchmark tests for verification functions
  - Single signature request verification
  - Multi signature request verification
  - Single signature with path selection request signing and verification

Benchmark Results

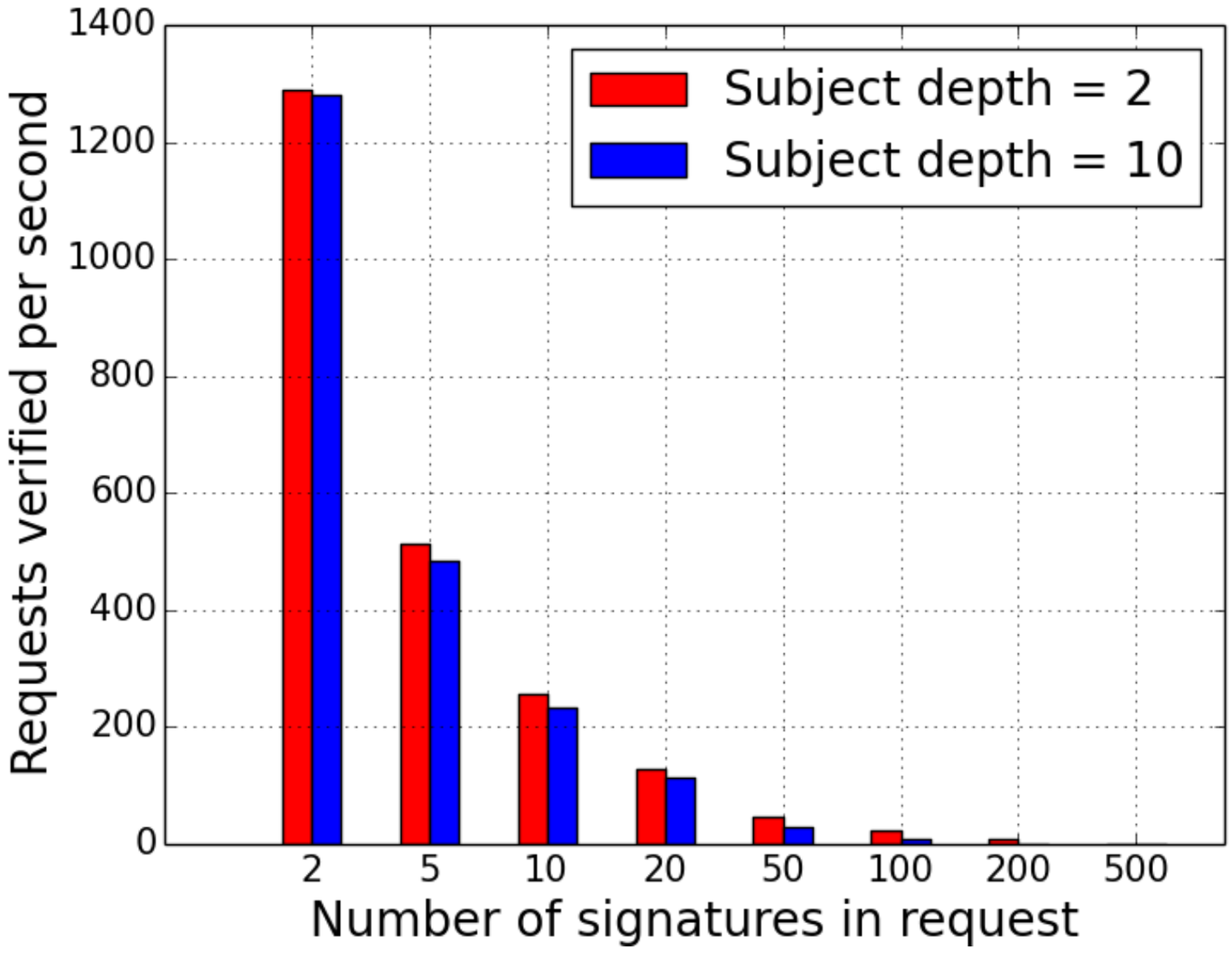
- Total verification time for single signature requests
- Depth of requester is varied (depth = distance between target policy and requester's parent policy)
- Signature verification = ~381 us
- Signature verification accounts for 92.04 - 99.94 % of total verification time



Total verification time = Signature verification time + Path finding time

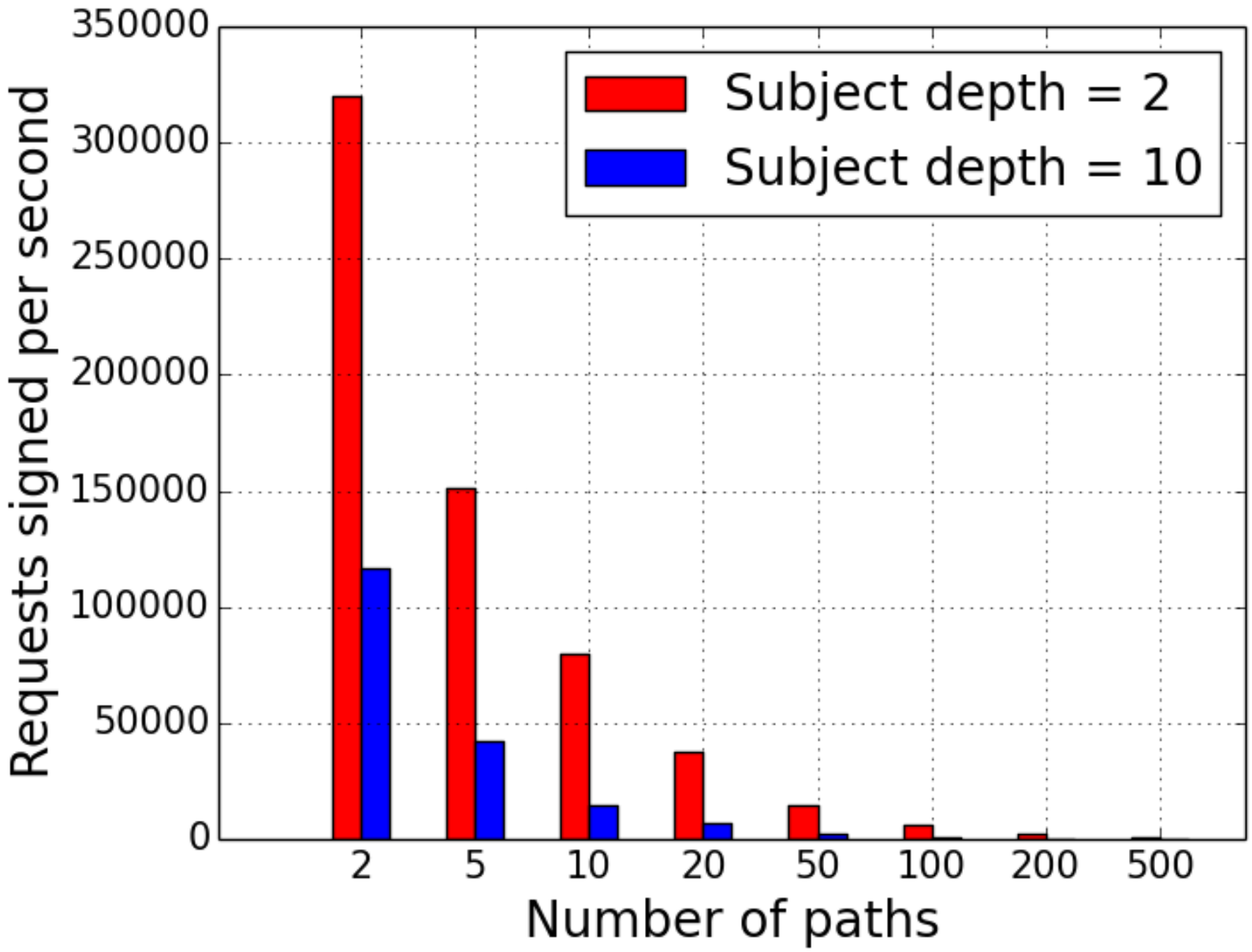
Benchmark Results

- Verification rate for multi-signature requests
- Number of signatures in request is varied
- Requester depth is set to 2 and 10



Benchmark Results

- Signing and verification for multi-path
- Signing rate ~530 at 500 paths, depth 2
- Verification time is dominated by signature verification
- Verification time similar to single signature verification case



Conclusion

- **Achieved**
  - Design, implementation and testing of a policy based access control system
  - Functionality: Policy Creations, Access Requests and Verification
  - API in Google Doc [4], Code on Github [5]
- **Future Work**
  - THR operator + weights in expressions
    - {“THR” : [thr_val, Subject1, weight1, Subject2, weight2…]}
  - Extensions of attributes in the access control model
  - Sub-policies and policy linking
  - Integration with skipchain architecture
  - Alternatives to the ‘one policy per skipchain’ model

References

- [1] Kokoris-Kogias, L., Gasser, L., Khoffi, I., Jovanovic, P., Gailly, N. and Ford, B., 2016. **Managing Identities Using Blockchains and CoSi**. In *9th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs 2016)*

- [2] **CISC Identity SkipChain**: https://github.com/dedis/cothority/tree/master/cisc

- [3] Maesa, D.D.F., Mori, P. and Ricci, L., 2017, June. **Blockchain Based Access Control**. In *IFIP International Conference on Distributed Applications and Interoperable Systems* (pp. 206-220). Springer, Cham.

- [4] **API description**: https://docs.google.com/document/d/1OoH0ecg1EF4xybD1tQAx9Ei7klHj-rYFYk1aRFoSFg0/edit?usp=sharing

- [5] **Implementation code**: https://github.com/sandrasiby/cothority_template - creating_policies branch