

# Separating Key Management from File System Security

1999

David Mazières, Michael Kaminsky, M.  
Frans Kaashoek, Emmet Witchel (MIT)

# The Problem . . .

- No practical, global, secure, decentralized network file system that scales to the size of the Internet.
- Why? Good security is hard:
  - Key management scaling issues
  - Cross administrative domain issues

# The Solution . . .

- Self-certifying pathnames
  - Pathname includes server's public key fingerprint
- Global – /sfs/...
- Secure – authenticate/authorize decoupled
- Decentralized – anyone client; anyone server

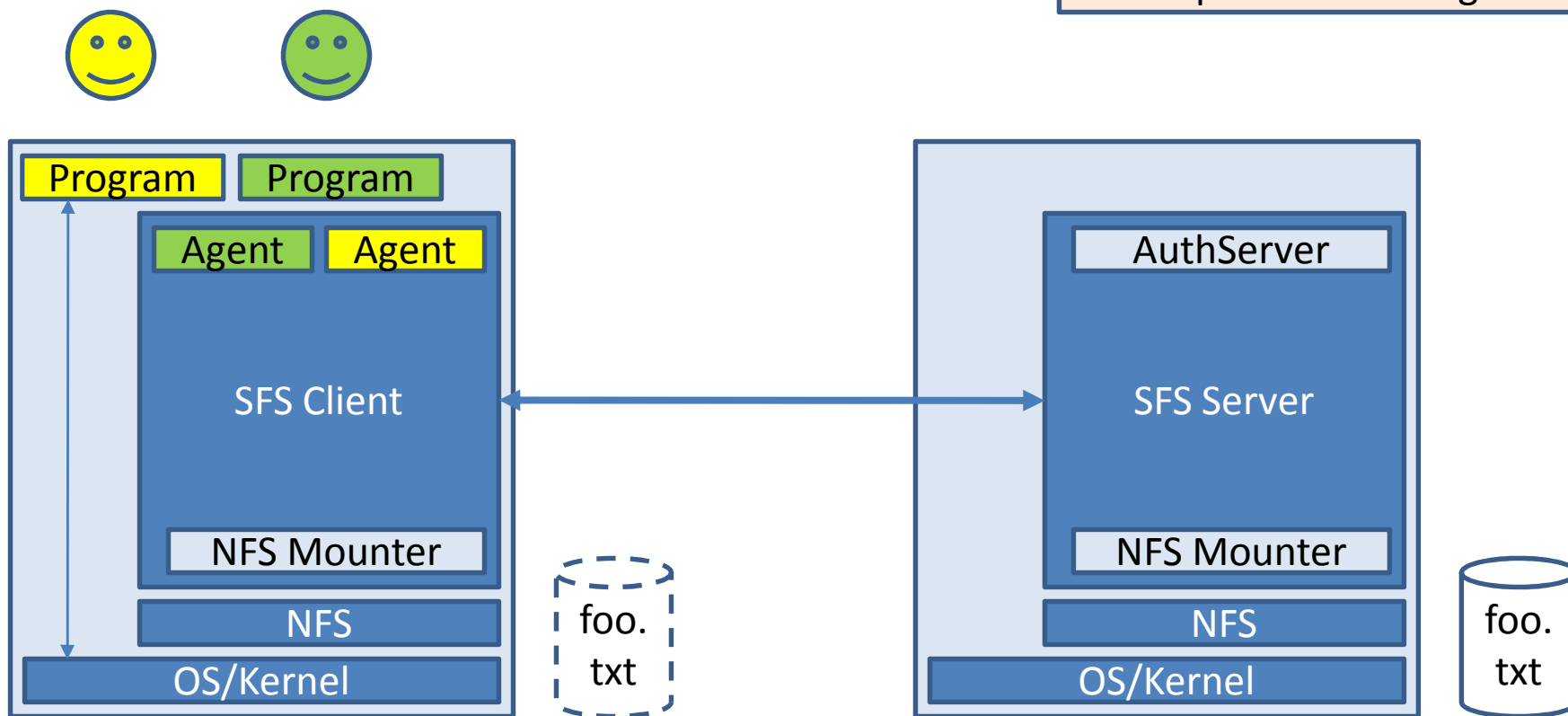
# Pathnames

- /sfs/ + Location + HostID + local path
  - sfs.lcs.mit.edu
  - SHA-1 (“HostInfo”, Location, PublicKey, “HostInfo”, Location, PublicKey)
    - 20 byte output encoded in base 32 (no 0,1,l,o)
    - vefvsv5wd4hz9isc3rb2x648ish742hy
  - /pub/links/sfscvs

/sfs/sfs.lcs.mit.edu:vefvsv5wd4hz9isc3rb2x648ish742hy/pub/links/sfscvs

[/sfs/sfs.lcs.mit.edu:vefvsv5wd4hz9isc3rb2x648ish742hy/pub/links/sfscvs](https://sfs.sfs.lcs.mit.edu:vefvsv5wd4hz9isc3rb2x648ish742hy/pub/links/sfscvs)

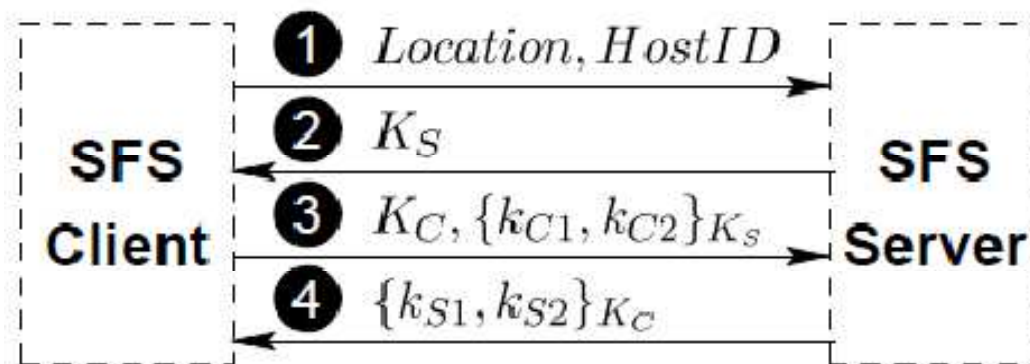
\*Example based on Figure 2



```
> temp-> /sfs/@sfs.lcs.mit.edu:vefvsv5wd4hz9isc3rb2x648ish742hy
>
> cat /sfs/temp/pub/links/sfscvs/foo.txt
Hello World!
>
```

# Key Negotiation Details

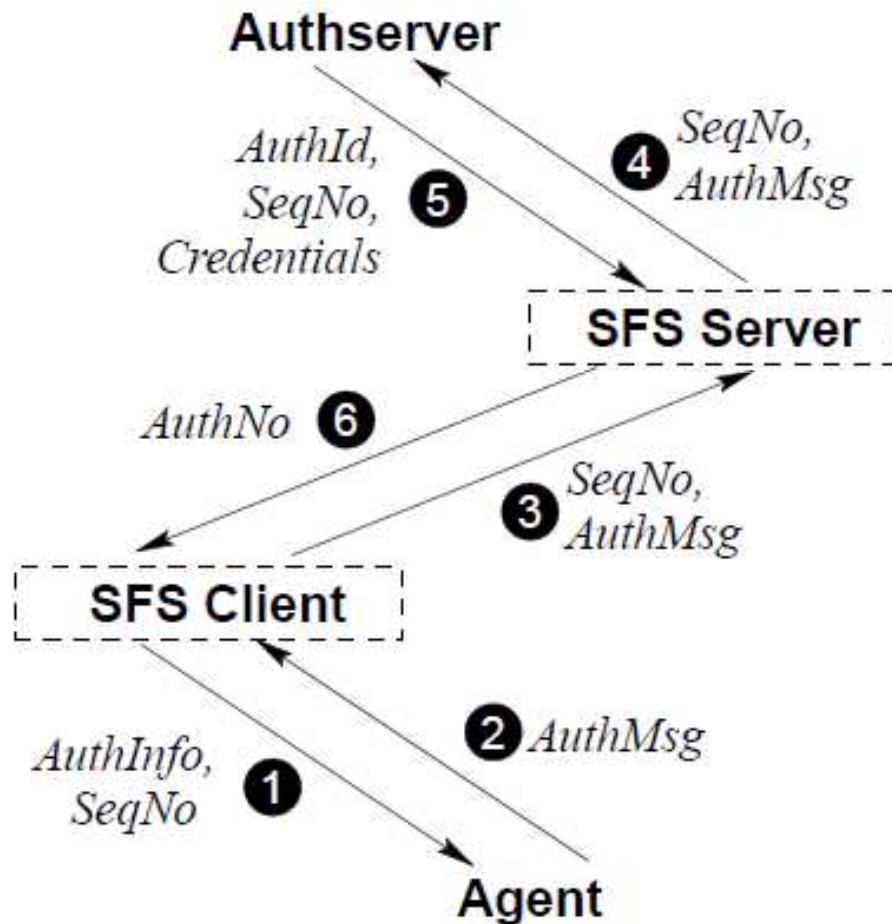
- Shared session keys
- Ensures forward secrecy



$$k_{CS} = \text{SHA-1}(\text{"KCS"}, K_S, k_{S1}, K_C, k_{C1})$$

$$k_{SC} = \text{SHA-1}(\text{"KSC"}, K_S, k_{S2}, K_C, k_{C2})$$

# User Authentication Details\*



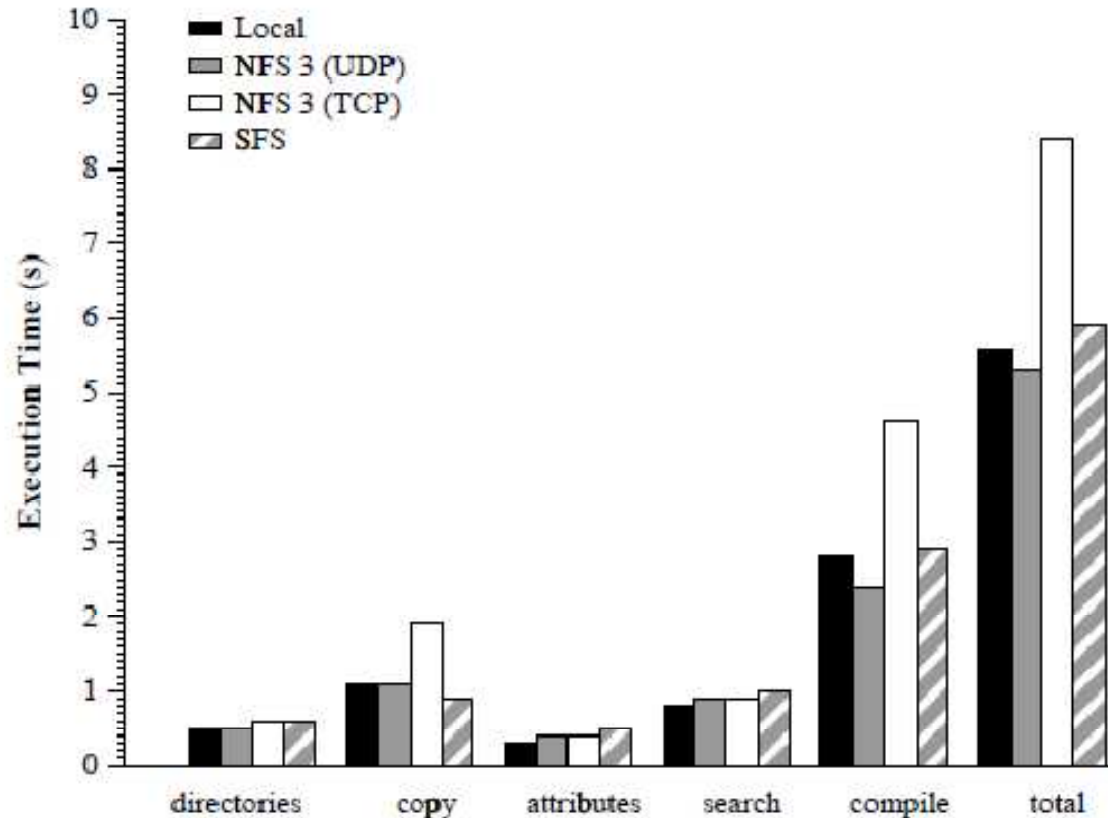
- Protocol separate from SFS file handling
- Can use different methods
- Users trust clients

\*More details in the next paper

# Revocation

- If server private key compromised
  - Key revocation
  - HostID blocking
- Client returns “file not found”

# Performance



**Figure 6:** Wall clock execution time (in seconds) for the different phases of the modified Andrew benchmark, run on different file systems. Local is FreeBSD's local FFS file system on the server.

# Conclusion

- Self-certifying path name is clever
  - Shifts user trust from CA to . . . Aaron?
- Updating keys = new path name
  - Embedded links are tricky
- SFS CAs are interactive (secret keys online)
- Crossing administrative domains is easier to reason about

# Decentralized User Authentication in a Global File System

2003

Kaminsky, George Savvides,  
Mazières, Kaashoek, (MIT)

# The Problem . . .

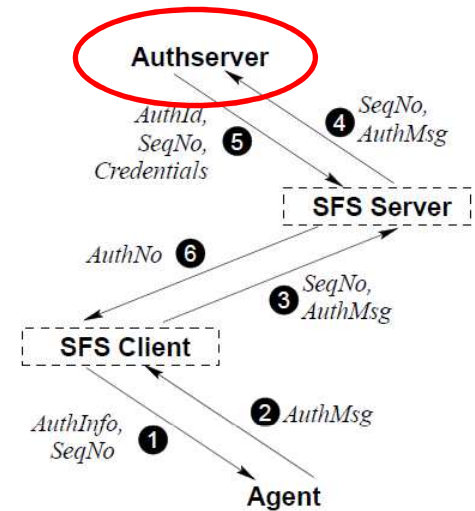
- No “easy,” global, decentralized system for authenticating users across administrative domains.
- Why? Good security is hard:
  - Often requires pre-existing admin coordination
  - Certificates work, but require a lot of overhead

# The Solution . . .

- Allow addition of remote users and groups to local ACLs
  - Aided by . . . self-certifying pathnames
- Local
  - u=aaron
  - g=aaron.syschat
- Remote (must be contained in a local group)
  - u=sanket@ucc.mu.ac.in, fr2eisz3fifttrtvawhnygzk5k5jidiv
  - g=students@ucc.mu.ac.in, zk5k5jidivfttrtvawhnygfr2eisz3fi

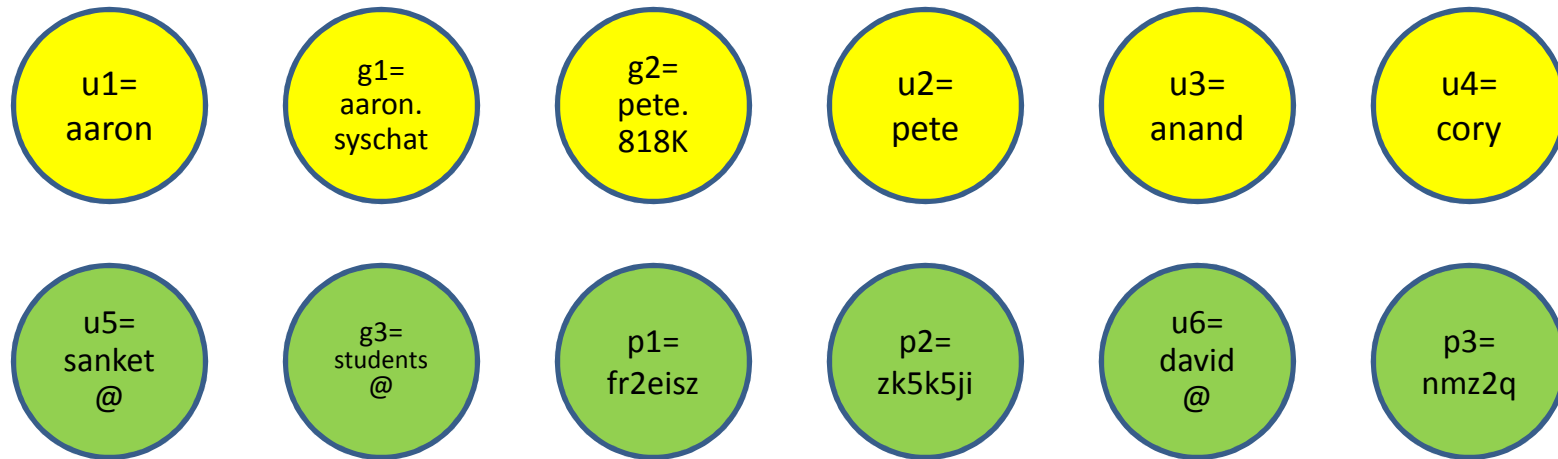
# User Authentication Details\*

- Pick-up at the Authserver
- Two functions
  - Generic user authentication service
  - Management interface for users
- Resolves membership
  - Containment graph -> membership graph
- Issues credentials



\*As promised . . .

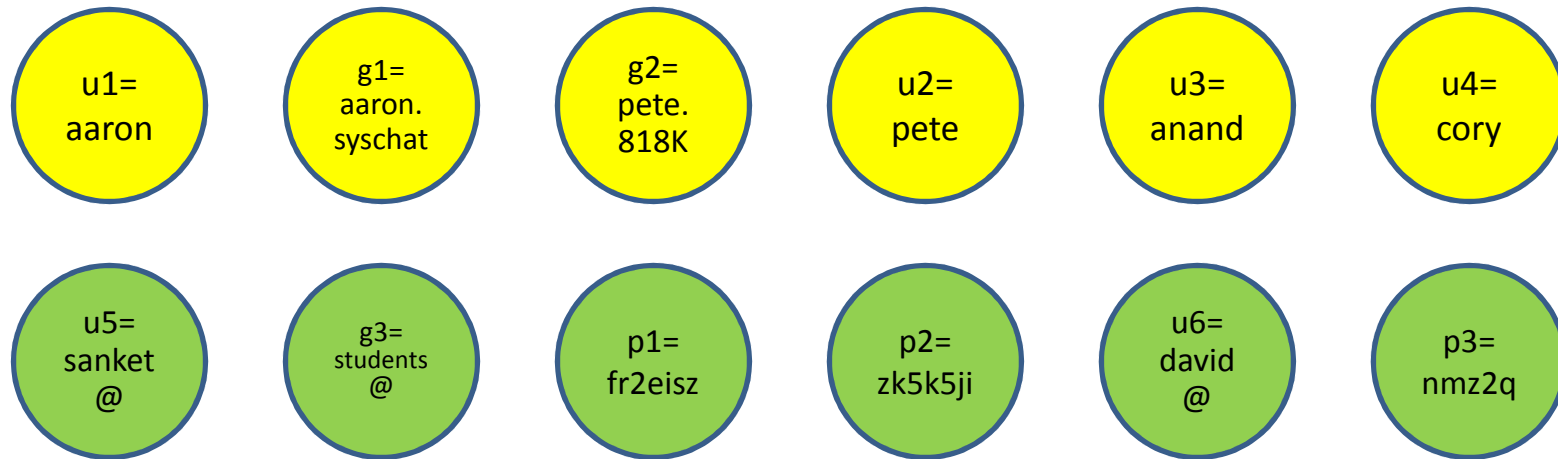
# Containment Graph



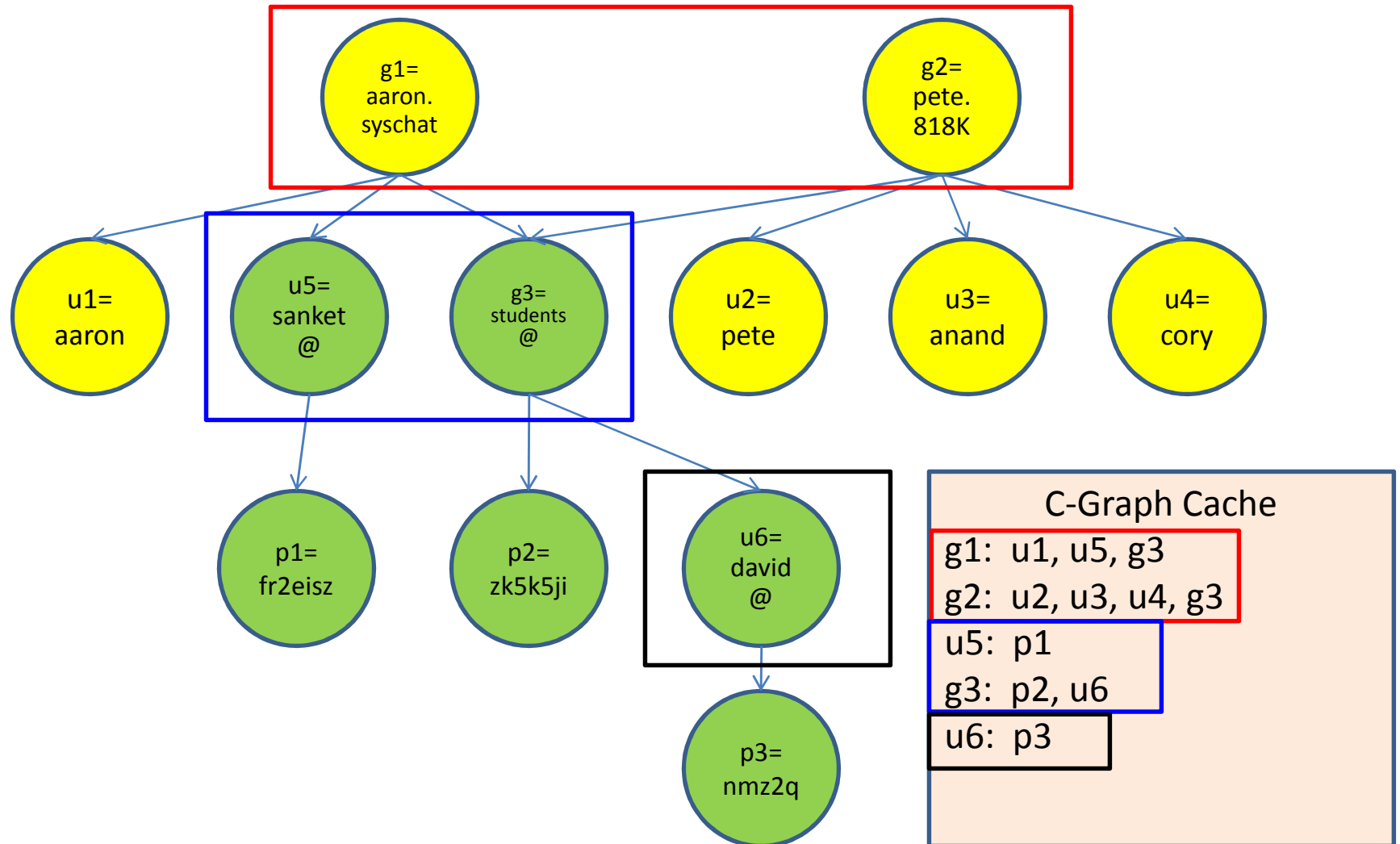
u=aaron  
g=aaron.syschat

u=sanket@ucc.mu.ac.in, fr2eisz3fiftrtvawhnygzk5k5jidiv  
g=students@ucc.mu.ac.in, zk5k5jidivfttrtvawhnygfr2eisz3fi

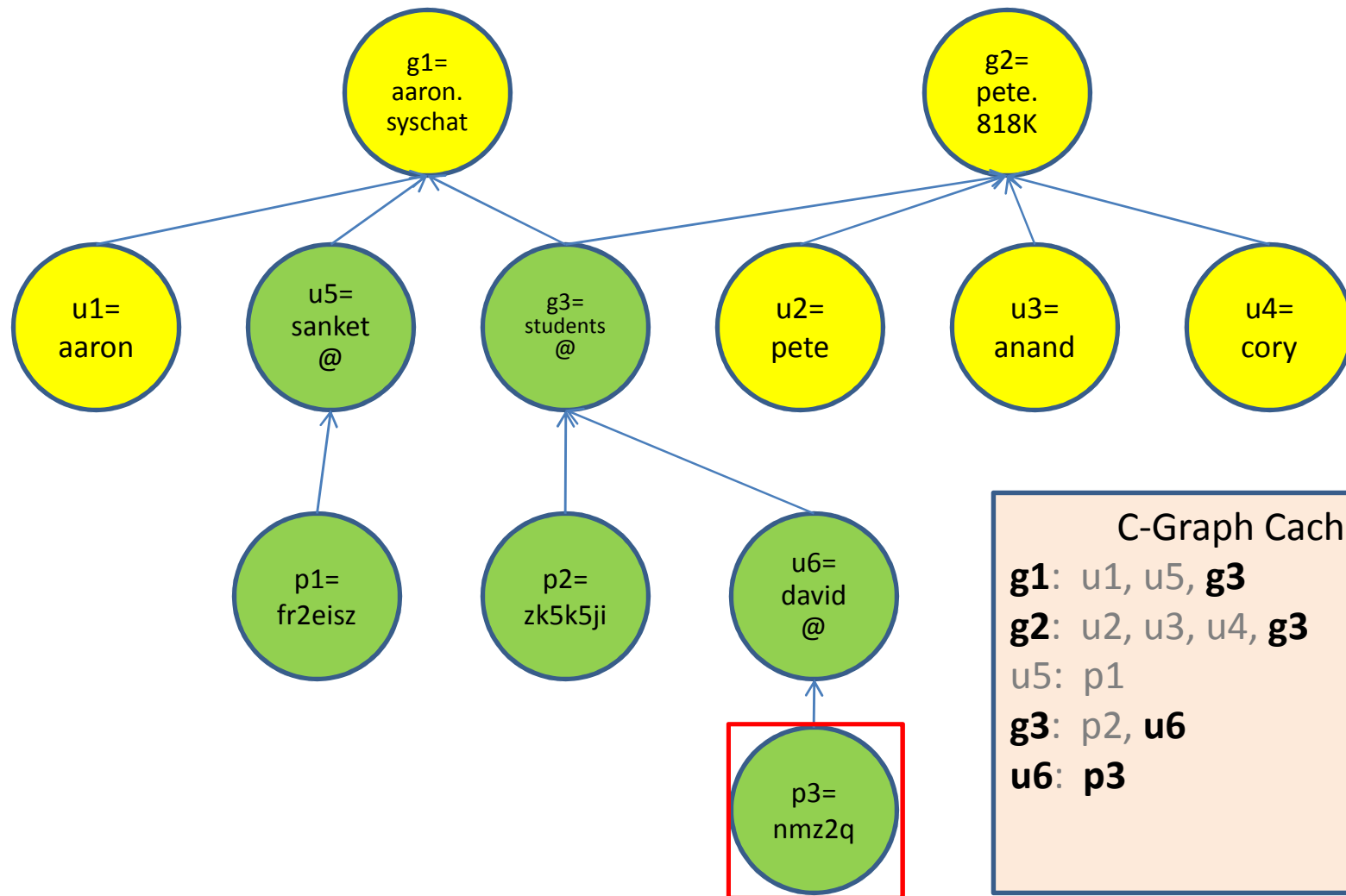
# Containment Graph



# Containment Graph



# Membership Graph



C-Graph Cache

**g1:** u1, u5, **g3**  
**g2:** u2, u3, u4, **g3**  
u5: p1  
**g3:** p2, **u6**  
**u6:** **p3**

# Optimizations

- Eventual Consistency
  - Get remote info periodically
  - Authorize only with local information
- Performance
  - Avoid many public-key operations
  - Transfer changes only
  - If get a public-key, determine user

# Authorization

- Once the authentication server has validated that a given user belongs to a local group (or not), those credentials are handed to SFS
- SFS conducts the mapping of local group names to access rights, thereby separating authentication from authorization

# Performance Evaluation

Phase	Original SFS seconds	ACL SFS with caching seconds (slowdown)	ACL SFS without caching seconds (slowdown)
create	15.9	18.1 (1.14X)	19.3 (1.21X)
read	3.4	3.5 (1.03X)	4.3 (1.26X)
delete	4.8	5.1 (1.06X)	6.0 (1.25X)
Total	24.1	26.7 (1.11X)	29.6 (1.23X)

- Figure 6 – LFS small file benchmark with 1,000 files created, read, and deleted.
  - Best result of 5 trials

# Conclusion

- Self-certifying path name is *still* clever
  - Allows for this local user control
- Extends how we think about user authentication and (*implied*) delegation
- Shows flexibility of SFS' modular design and is itself extensible
- Crossing administrative domains is easier . . .
  - Because Aaron allows it?