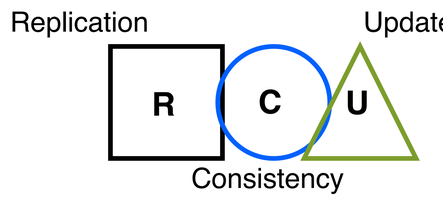
Don't Trust Your Roommate, or, Access Control and Replication Protocols in "Home" Environments

Vassilios Lekakis, Yunus Basagalar, Pete Keleher
{lex,yunusb,pete}@cs.umd.edu

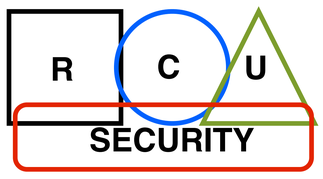
INFORMATION LEAKAGE



- Current systems combine R,C,U elements to:
  "Keep a user's data in sync across multiple devices"
- Security is left to the application layer
- Multi-user environments +  lead to:

Information Leakage

THESIS & CONTRIBUTIONS

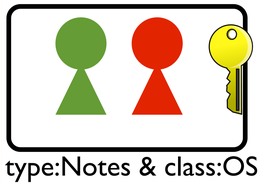


- Security ***not orthogonal*** with  but should be integrated
- Eliminate **information leakage** by:
  - Role-based access control
  - Role-based consistency
  - Object Forking

ACCESS CONTROL ELEMENTS

Principals

Role



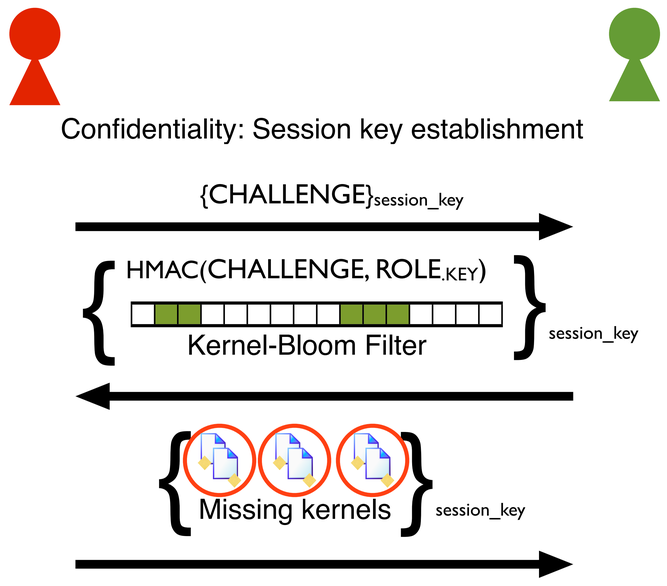
type:Notes & class:OS

- Users are the principals
- Users organize their contacts in roles
- Roles consist of:
  - Principals
  - Label predicate describing the role access rights
  - Role key, for encryption
- Augment files' metadata to include label-value pairs
- Devices replicate collections at the level of roles

LEAKAGE-FREE ANTI-ENTROPY

What we consider as information leakage?
- Data access outside the realm of the roles played by a replica
- Replicas should not reveal their roles

Goals:
- Eliminate information leakage
- Maintain the flexibility of a topology independent update mechanism



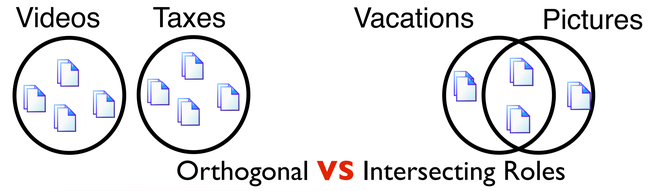
Kernels:
- Have a random GUID (global unique identifier)
- Cryptographic grouping of role updates after a time period t
- Encrypted with the appropriate role key

Bloom filters:
- Populated by the kernel GUIDs
- Summarize the kernels stored in each replica

HMAC exchange:
- Find out the role of other replicas without revealing the locally replicated roles

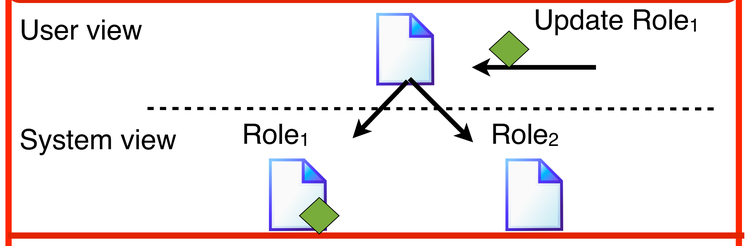
Kernel Exchange:
- Pass encrypted consistency information to replicas with different roles
- Act as caches

DOUBLE HATTED REPLICAS



- Orthogonal roles are ideal but:
  - Users lack expertise
  - Don't always fit users' preferences
- Intersecting roles may lead to information leakage
  - Consistency schemes can mitigate this
- Two consistency schemes:
  - Object-based: Stronger guarantees, but leaks information
  - Role-based: Inspired from fork-consistency, avoids leaking

OBJECT FORKING



- No information leakage between intersecting roles
- Storage overhead due to multiple role versions
- Local reads need the "role-context" to be implemented correctly

CONCLUSIONS

- Current protocols, in a multiuser setting, leak information
- Security should be a major building block of personal data management systems
- Elimination of leakage through:
  - Role-based access control
  - Object forking
  - Role-based consistency