Don’t Trust Your Roommate, or,
Access Control and Replication in “Home” Environments

HotStorage 2012

Vassilios Lekakis, Yunus Basagalar, Pete Keleher
University of Maryland
One picture is worth a thousand words

Anzere, Bayou, BlueFS, Cimbiosys, EnsemBlue, EYO, HomeViews, Perspective, PodBase, PRACTI, UIA, ZZFS
Building Blocks

Replication  Update Mechanism

Consistency

Keep all devices synchronized
Personal Spaces

collaboration everywhere

gym
lab
coffee shops
classroom
events
home
dorms
commute
“Leaking” Spaces

Information Leakage

- lab
- gym
- events
- dorms
- commute
- classroom
- events

Information Leakage

Friday, June 15, 12
The fourth element

Keep all devices synchronized

Eliminate information leakage
Contributions

Security, as a building block

No information leakage
• Bloom filters
• Role-based consistency
• Object forking
Leakage in action
Leakage in action

File_2

Beach

Dorm
Leakage in action

File\_2 File\_2

Beach Dorm
Leakage in action

File_2

Beach

File_2

Dorm
Leakage in action

B

File

Beach

A

File

Dorm

File

File

Friday, June 15, 12
Leakage in action

Information leakage: Alice learns about File₂
Can we do better?

Eliminate information leakage
Maintain a flexible update mechanism
Access control elements

Principals

Alice.OS-Notes

type: Notes & class: OS
What we consider leakage?

Any data access outside the realm of a role
Replicas should not reveal their roles to other replicas
Leakage-free update

Bob.OS-Project

type:Project & class:OS

Alice.OS-Notes

type:Notes & class:OS
Leakage-free update

Bob.OS-Project

type:Project && class:OS
Leakage-free update

Bob.OS-Project

type: Project && class: OS
Leakage-free update

Bob.OS-Project

type: Project && class: OS

Bloom filter

Kernel
Why use a bloom filter?

Learn how to update peers without leaking information
Leakage-free update

Confidentiality:
Session Key Establishment
Leakage-free update

\{\text{OS-Project-Challenge}\}_{\text{session_key}}
Leakage-free update

\[
\{ \text{OS-Project-Challenge}\}_{\text{session_key}}
\]

\[
\{ \text{HMAC(OS-Project-Challenge, OS-Notes.KEY)}\}_{\text{session_key}}
\]
Why the HMAC step?

To decide when to use a pure log exchange approach without leaking information
Leakage-free update

\[
\text{HMAC} \left( \text{OS-Project}.\text{CHALLENGE}, \text{OS-Notes}.\text{KEY} \right)
\]

\[
\text{HMAC} \left( \text{OS-Project}.\text{CHALLENGE}, \text{OS-Project}.\text{KEY} \right)
\]

CMP

OS-Project Kernels

VS

Alice’s Bloom Filter
Leakage-free update
Double hatted replicas

Vacations

Financial Planning

Taxes

Orthogonal VS Intersecting Roles

Friday, June 15, 12
Consistency & information leakage

Bob.OS-Project

Alice.OS-Notes

**type:** Project & **class:** OS

**type:** Notes & **class:** OS

**type:** Project & **class:** OS

OR

**type:** Notes & **date:** June 14

Friday, June 15, 12
Consistency & information leakage

Bob.OS-Project ⊆ File₃ ⊆ Alice.OS-Notes
Consistency & information leakage

Bob.OS-Project

File3

Alice.OS-Notes

Kernel

File3

C

A

B

✓
Object-based consistency

Bob.OS-Project  ❋  Alice.OS-Notes

File₃

Do nothing
Updates appear to intersecting roles

Object consistency VS Information leakage
Role-based consistency & forking objects

Bob.OS-Project → File

System

User
Role-based consistency & forking objects

User

System

File3

File3

File3
Role-based consistency discussion

- No Information leakage between roles
- Storage overhead
- Programming complexity
- Local accesses become an issue
Takeaway

‣ Current protocols in a multi-user setting leak information

‣ Thesis: Security should become a major building block of personal data management systems

‣ Elimination of leakage through
  ‣ Role-based access control
  ‣ Object forking
  ‣ Role-based consistency