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
Building Blocks

Replication
Consistency
Update mechanism
Building Blocks

Replication

Update Mechanism

Consistency

Keep all devices synchronized
Personal Spaces

collaboration everywhere
“Leaking” Spaces

Information Leakage

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

Situation: Information Leakage everywhere
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

Charlie  Bob  Alice

File 1  File 2

Monday, June 18, 12
Leakage in action

File

Beach

Dorm
Leakage in action

B
File_2
Beach

A
File_2

C
Dorm

Monday, June 18, 12
Leakage in action

File\textsubscript{2}  

Beach

File\textsubscript{2}  

Dorm
Leakage in action

File 2

Beach

File 2

File 2

Dorm
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

Monday, June 18, 12
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

Alice.OS-Notes

type:Project & class:OS
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

{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

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

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

\[
\text{CMP} \left\{ \begin{align*}
\text{HMAC(} \text{OS-Project.\text{CHALLENGE}, OS-Notes.\text{KEY}}) \\
\text{HMAC(} \text{OS-Project.\text{CHALLENGE}, OS-Project.\text{KEY}})
\end{align*} \right. \]

OS-Project Kernels

VS

Alice’s Bloom Filter
Leakage-free update

\{ \text{session\_key} \}
Double hatted replicas

Orthogonal VS Intersecting Roles

Videos

Taxes

Vacations

Financial Planning
Consistency & information leakage

Bob.OS-Project

Alice.OS-Notes

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

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

OR

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

**type:** Notes & **class:** OS
Consistency & information leakage

Bob.OS-Project

File_3

Alice.OS-Notes

C

File_3

B

A
Consistency & information leakage

Bob.OS-Project  File$_3$  Alice.OS-Notes

C  

A  ✗

B  🟢

Kernel  File$_3$
Object-based consistency

Bob.OS-Project

Alice.OS-Notes

File

Do nothing
Updates appear to intersecting roles

Object consistency VS Information leakage

Monday, June 18, 12
Role-based consistency & forking objects

Bob.OS-Project → File3

User → System
Role-based consistency & forking objects
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