

Maxoid: Transparently Confining Mobile Applications with Custom Views of State

Yuanzhong Xu and Emmett Witchel
University of Texas at Austin

4/24/2015
Bordeaux, France

Focus of this work

- *Security problems*: when different apps ***interact*** with each other, secrecy/integrity of data is often compromised
- *Cause*: insufficient support from the platform

Mobile platform: app-centric security



Mobile platform: app-centric security

Principals: apps from different developers

- User may not trust them
- They may not trust each other

Platform:

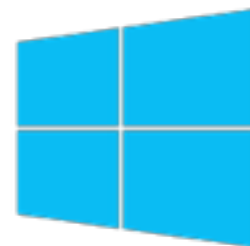
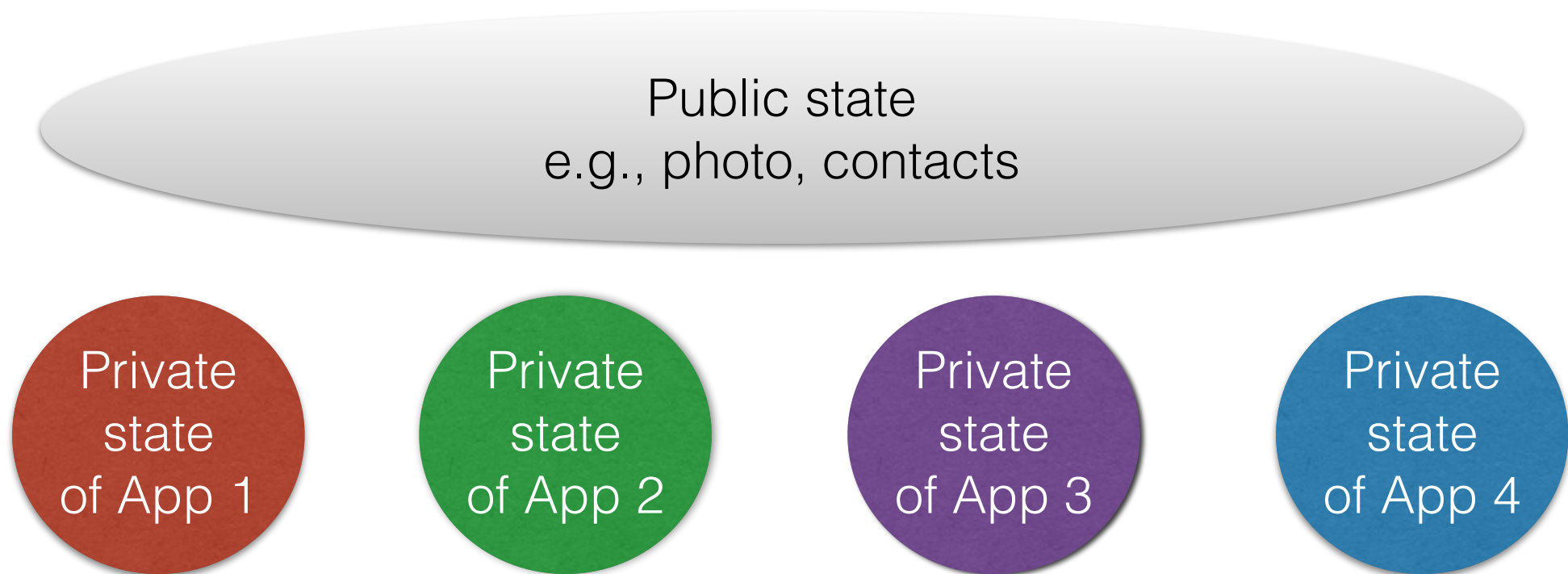
- Minimize apps' privilege
- Protect apps from each other



many apps

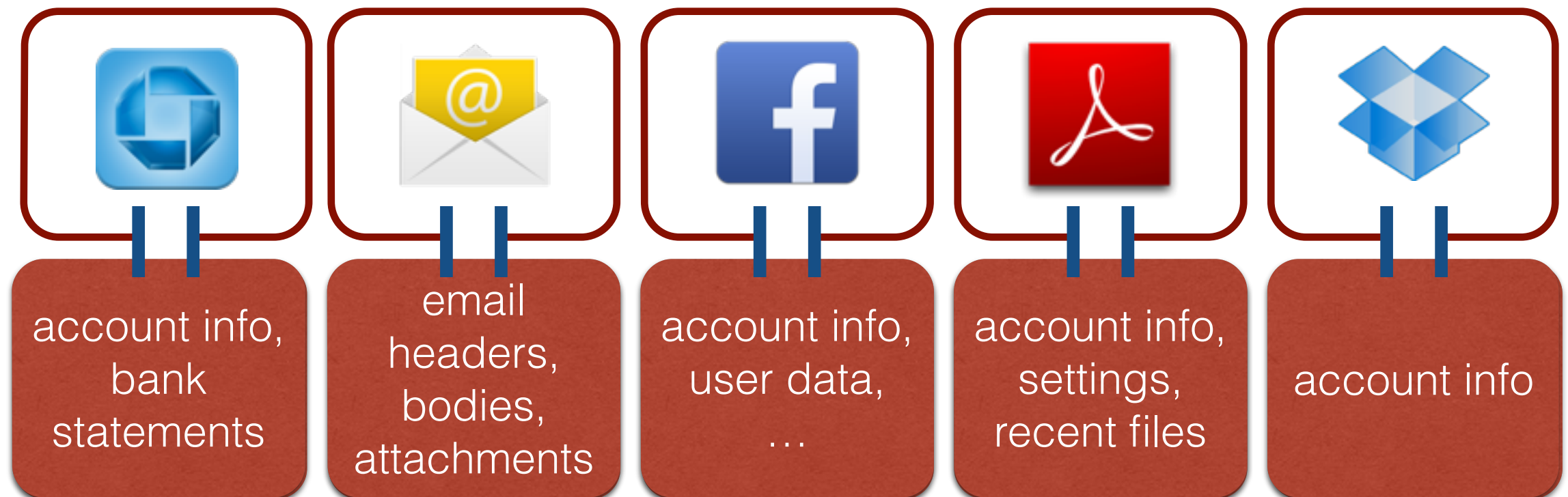
Public and private state of mobile apps

*Principals are the **apps**, not users*



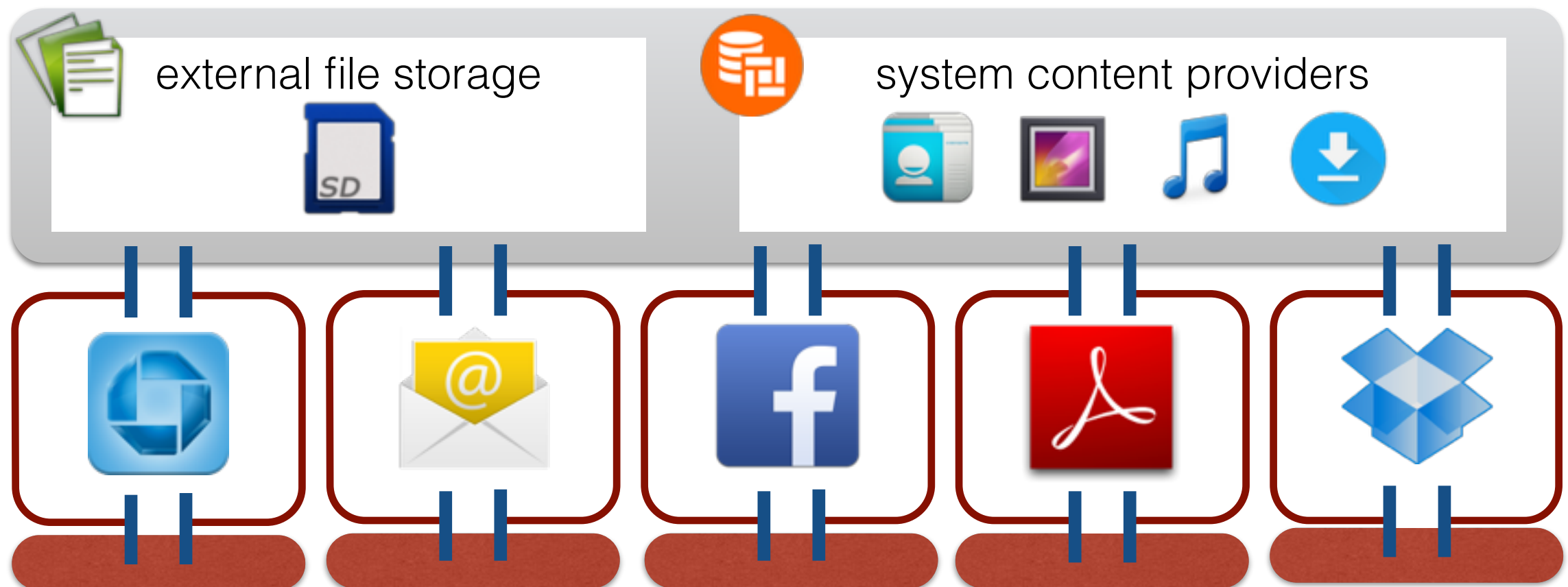
Android apps' private data

- Each app has its own UNIX UID (app sandbox)
 - Private files owned by the app
 - Higher-level APIs: database, key-value store, etc. implemented as libraries



Android apps' public data

- Public data shared by apps
 - Files in external storage (e.g., SD card)
 - Structured data in *system content providers*
 - Contacts, Media, Downloads, User Dictionary, etc.

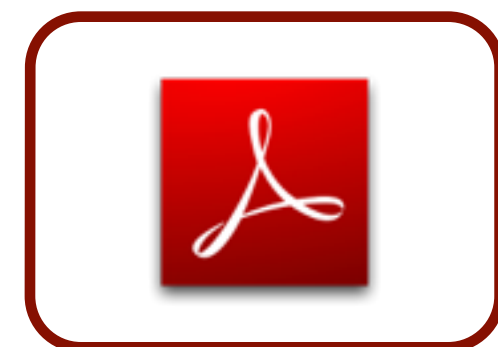


Problem

Sometimes an app needs to share its private data with another app

“initiator”: holding private data

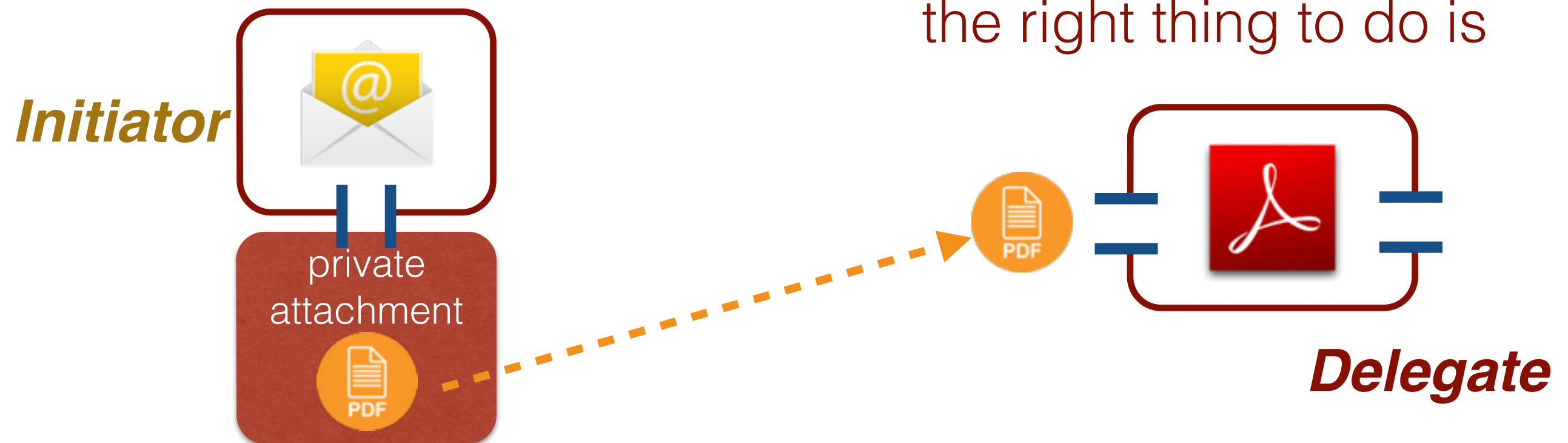
“delegate”: processing initiator’s private data



Disclose data to delegate

Sometimes an app needs to share its private data with another app

not aware that
this is private, or what
the right thing to do is



Initiators fail to protect data



Email

- No confinement on document viewers



Dropbox

- Stores all files in public SD card to allow other apps to open
- No confinement on document viewers



Browser's incognito mode

- Even in incognito mode, downloaded files are public
- No confinement on document viewers

Our contributions — Maxoid

- Security: confining delegates
 - Coarse-grained information flow control (IFC)
- Usability: support legacy apps
 - Multiple versions of data organized in custom views of state for apps

Legacy delegates leak data

Popular document viewers, scanners, cameras, media players, etc. leak data in the public state about the input



Naive IFC #1

Delegates are disallowed to leak data to public state or network



Naive IFC #1

Mobile apps typically tolerate *temporary* network disruptions

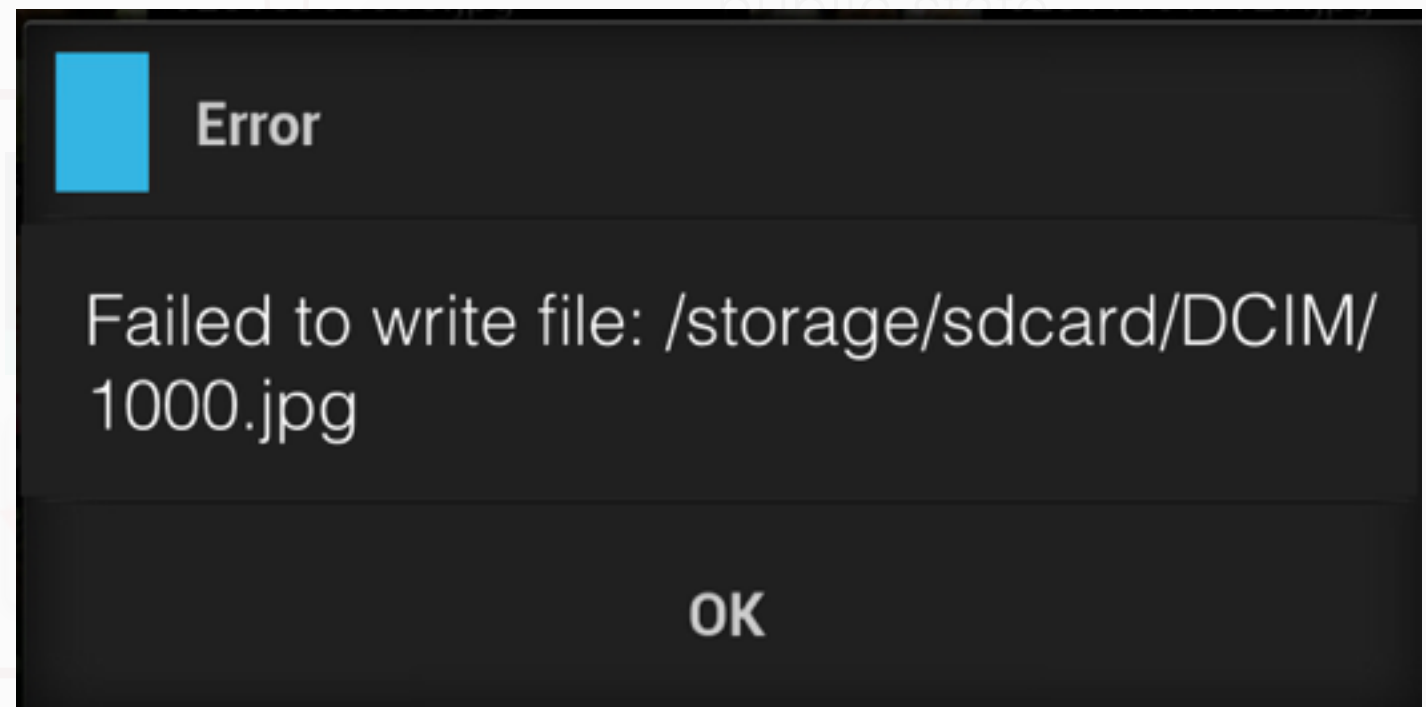
Naively blocking data flows

would break legacy apps!

- Unexpected permission errors

Too restrictive for legacy apps

input from
initiator



delegates

Naive IFC #2

- Taint tracking on input data
- Delegates can write tainted data to public state
- Control disclosure of tainted data to network



Naive IFC #2

- Taint tracking on input data
- Delegates can write tainted data to public state
- Control disclosure of tainted data to network



Naive IFC #2

- Taint tracking
- Delegates can
- Control disclosure

Too permissive about public state:

Uncontrolled taint propagation cripples unrelated apps on the device



Dilemma in naive IFC

- Allow or disallow delegates to write tainted data out?
 - Allow: uncontrolled taint propagation
 - Disallow: breaking legacy apps

Maxoid

- Addressing the dilemma:
 - Allow delegate apps to write “out” tainted data
 - Controls taint propagation transparently
- Key technique: maintain multiple versions of data when necessary
 - Isolate tainted versions from untainted versions
 - Present different **views** of state for different app instances

Two execution modes

An app can run in one of two *modes*:

- Running on behalf of itself (normal mode):

— as an initiator



- Running on behalf of another app:

— as a delegate

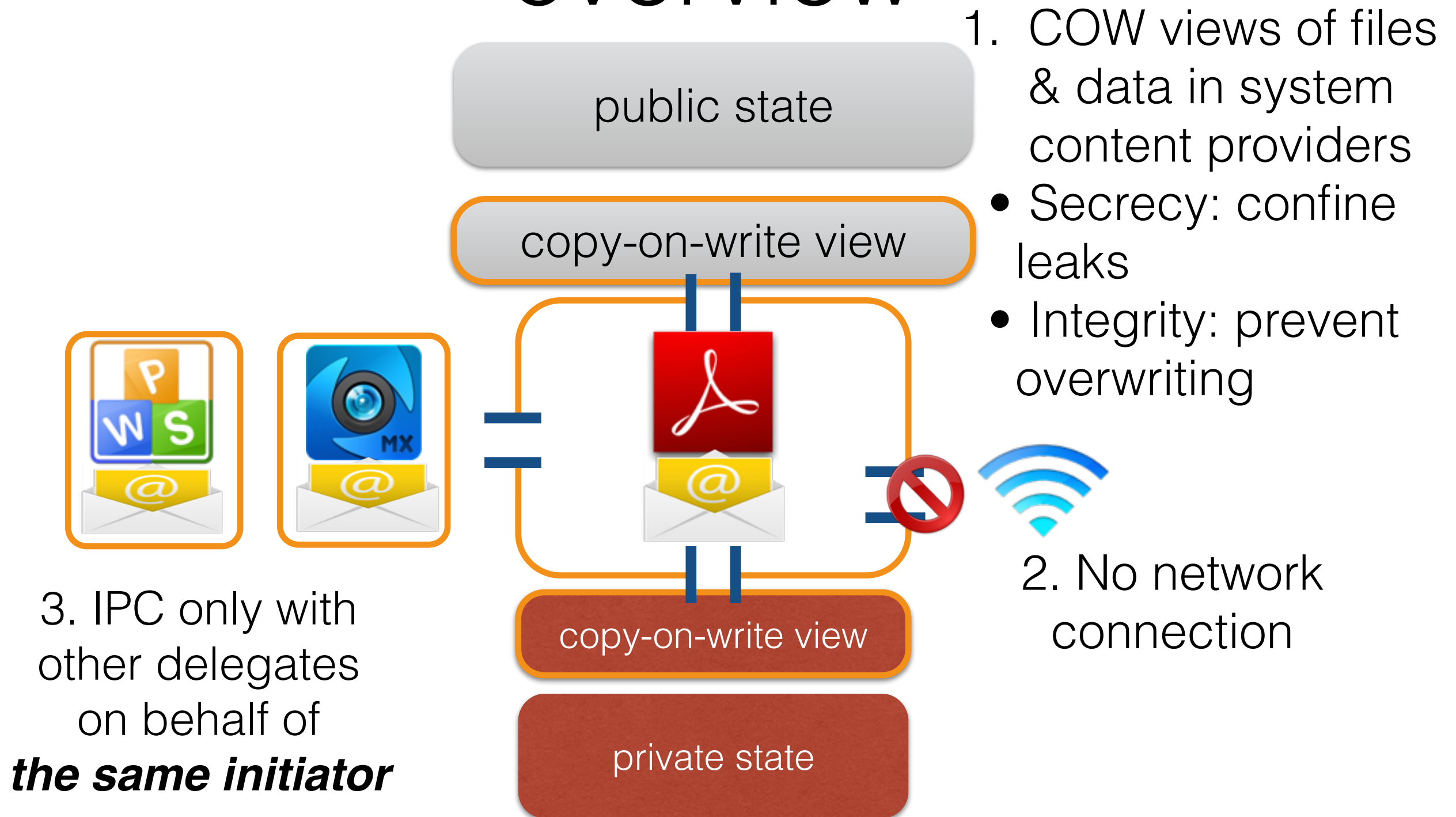
e.g. Adobe Reader on behalf of Email







Coarse-grained taint tracking made usable

- Coarse-grained taint tracking
All output of the delegate is considered tainted by the private input from the initiator
 - Conservative, strong security
 - False positives — safe data flows recognized as unsafe ones
 - Maxoid makes it usable

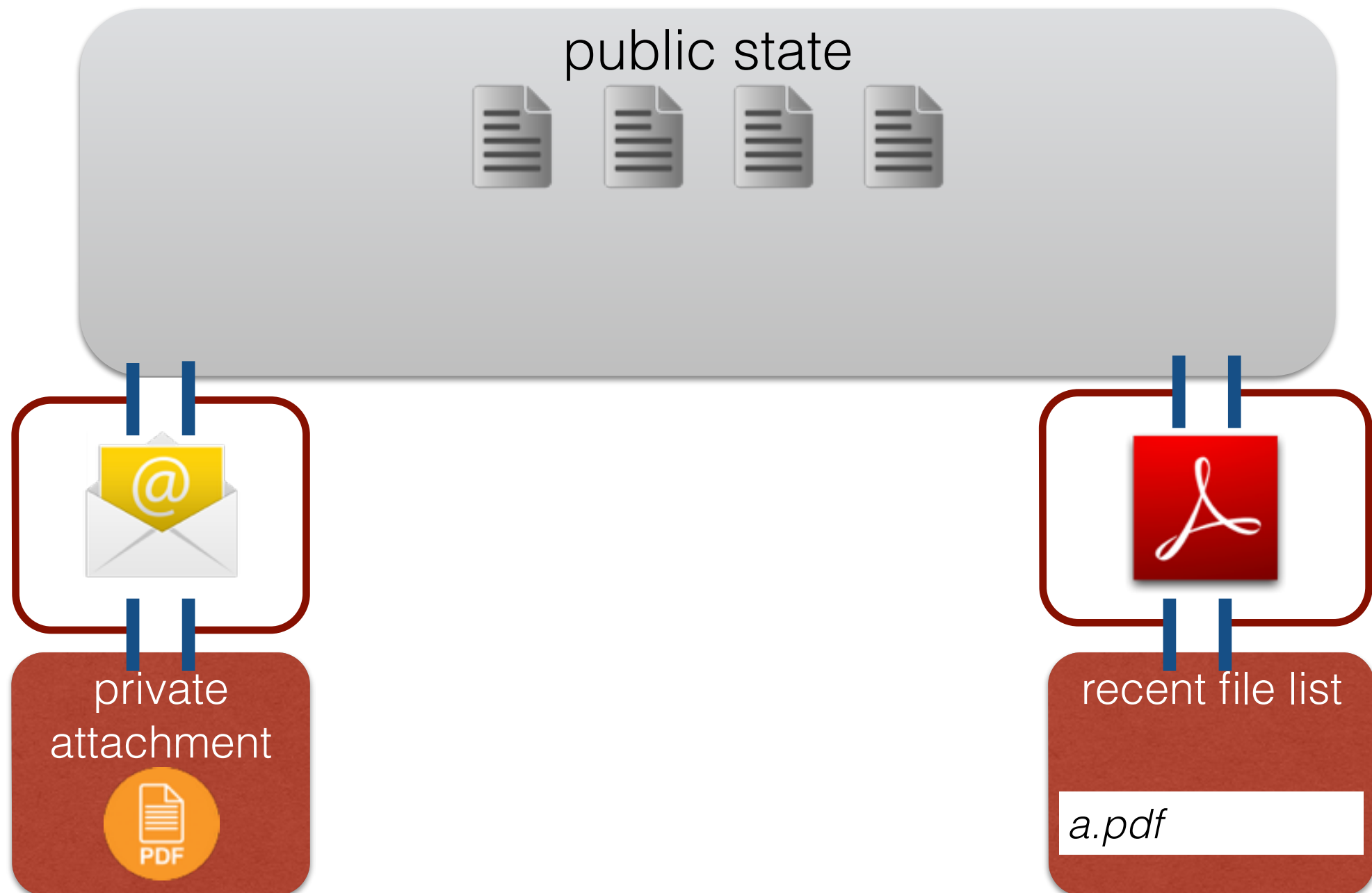
Delegate confinement overview



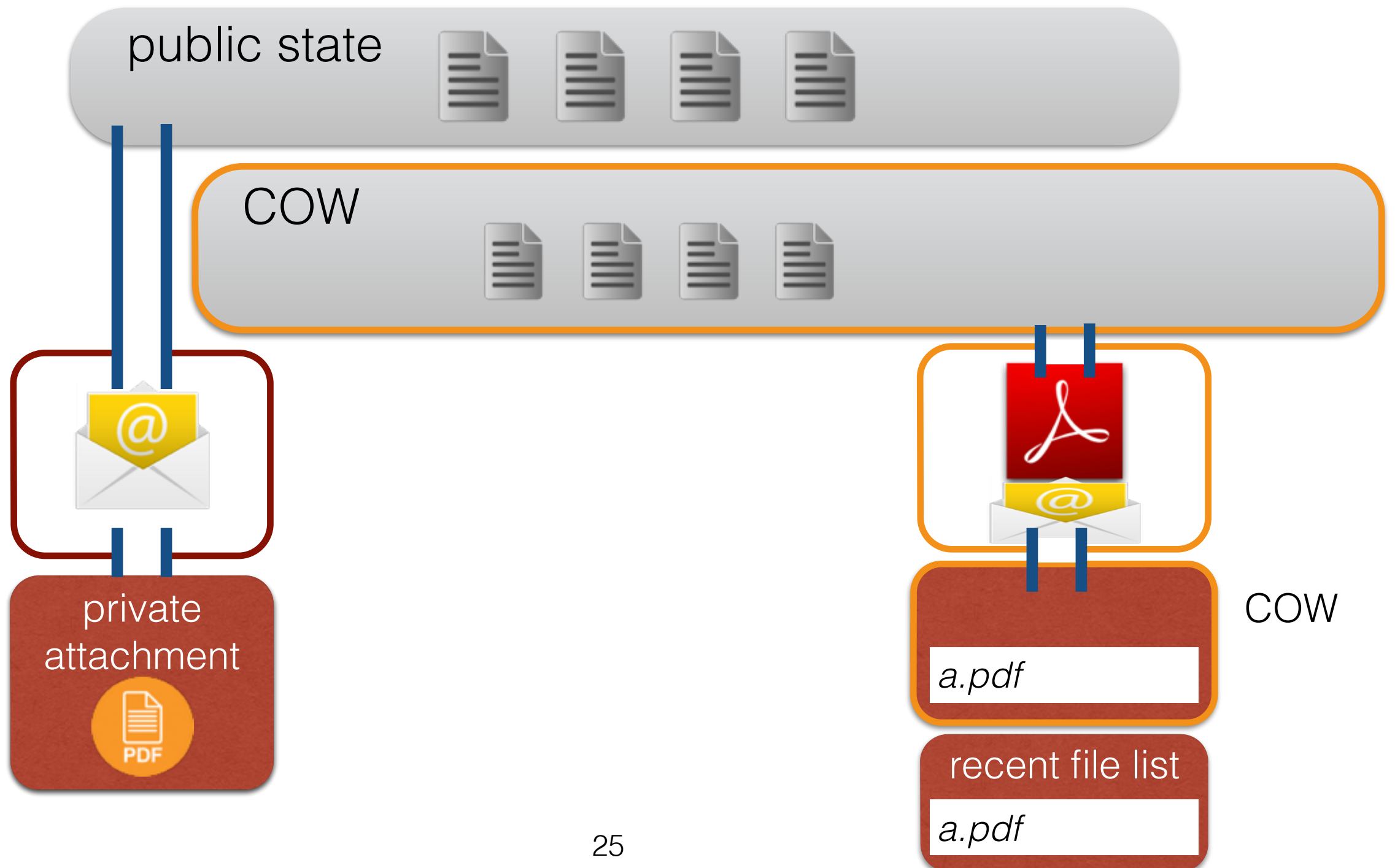
Use cases

Initiator app	Current	Changes	Maxoid Improvement
Email 	<ul style="list-style-type: none"> No confinement on doc viewers 	Config.	<ul style="list-style-type: none"> Invoke <i>unmodified</i> delegates to open attachments
Dropbox 	<ul style="list-style-type: none"> Store public files No confinement on doc viewers 	Config.	<ul style="list-style-type: none"> Store private files Invoke <i>unmodified</i> delegates to open files
Browser's incognito mode  	<ul style="list-style-type: none"> Downloaded files are public No confinement on doc viewers 	1 line of source code	<ul style="list-style-type: none"> Private downloads for incognito mode are only accessible to delegates ◆ Extend incognito mode to the whole device

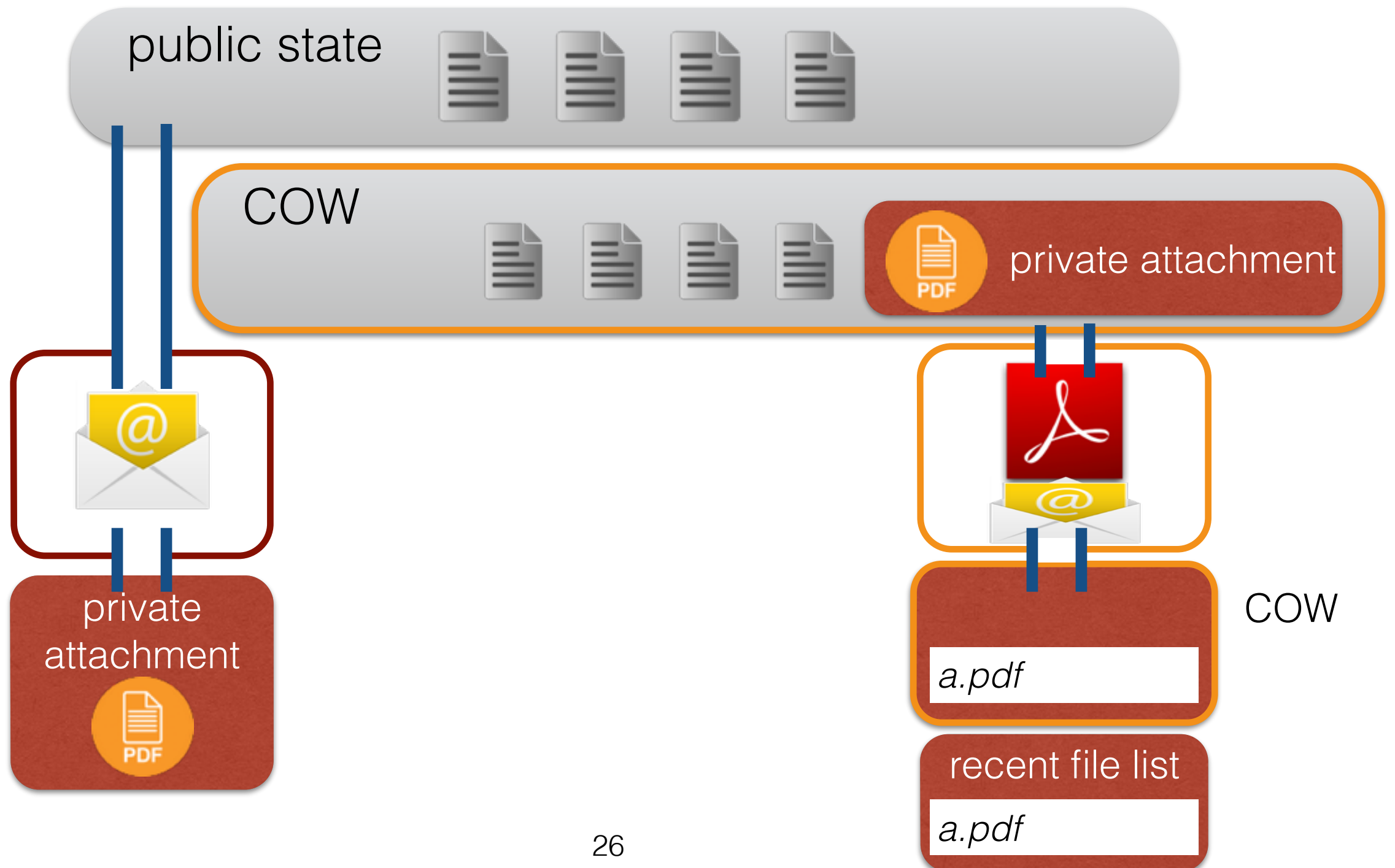
Views for initiators are identical to Android's data model



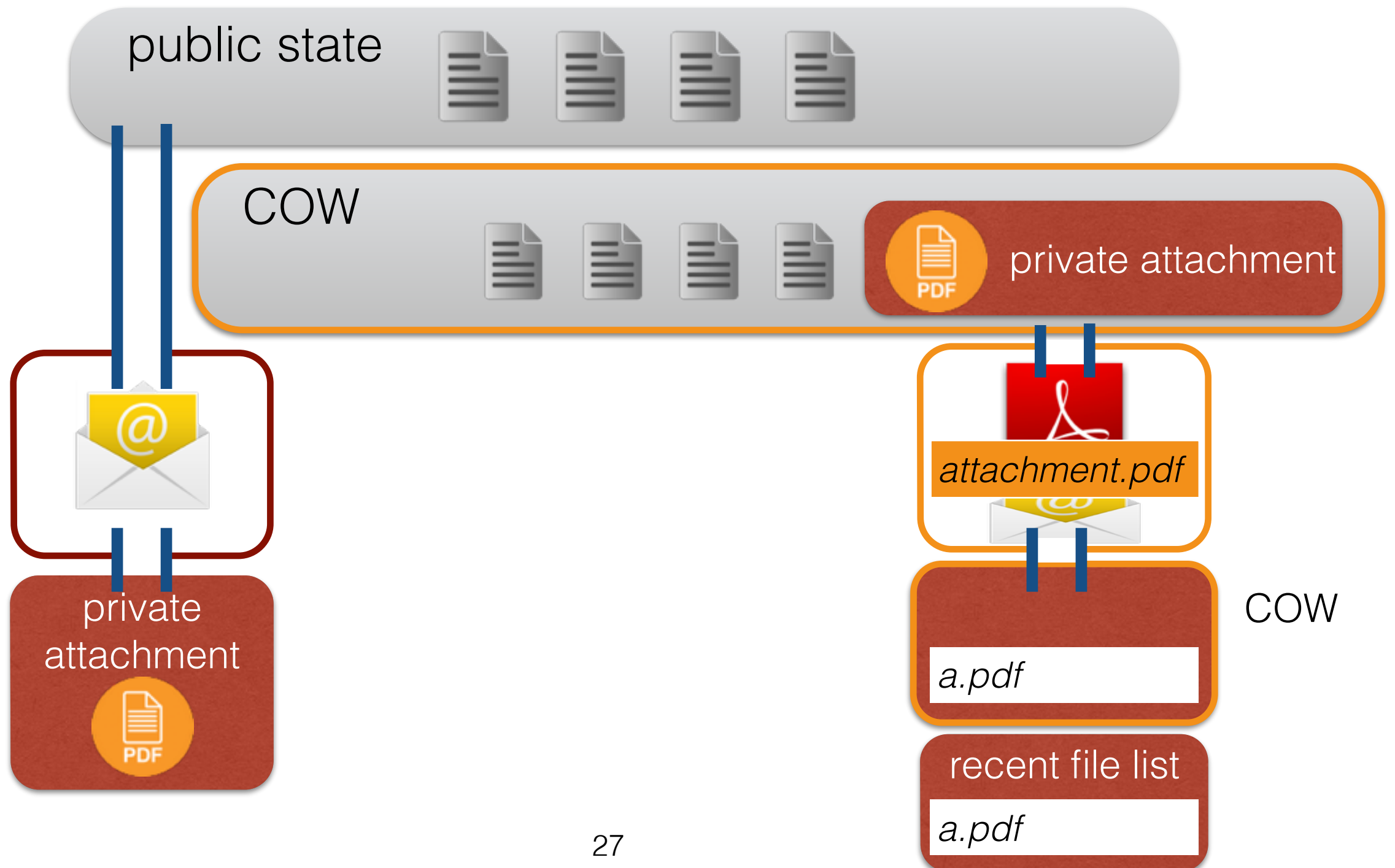
Copy-on-write (COW) views for delegates



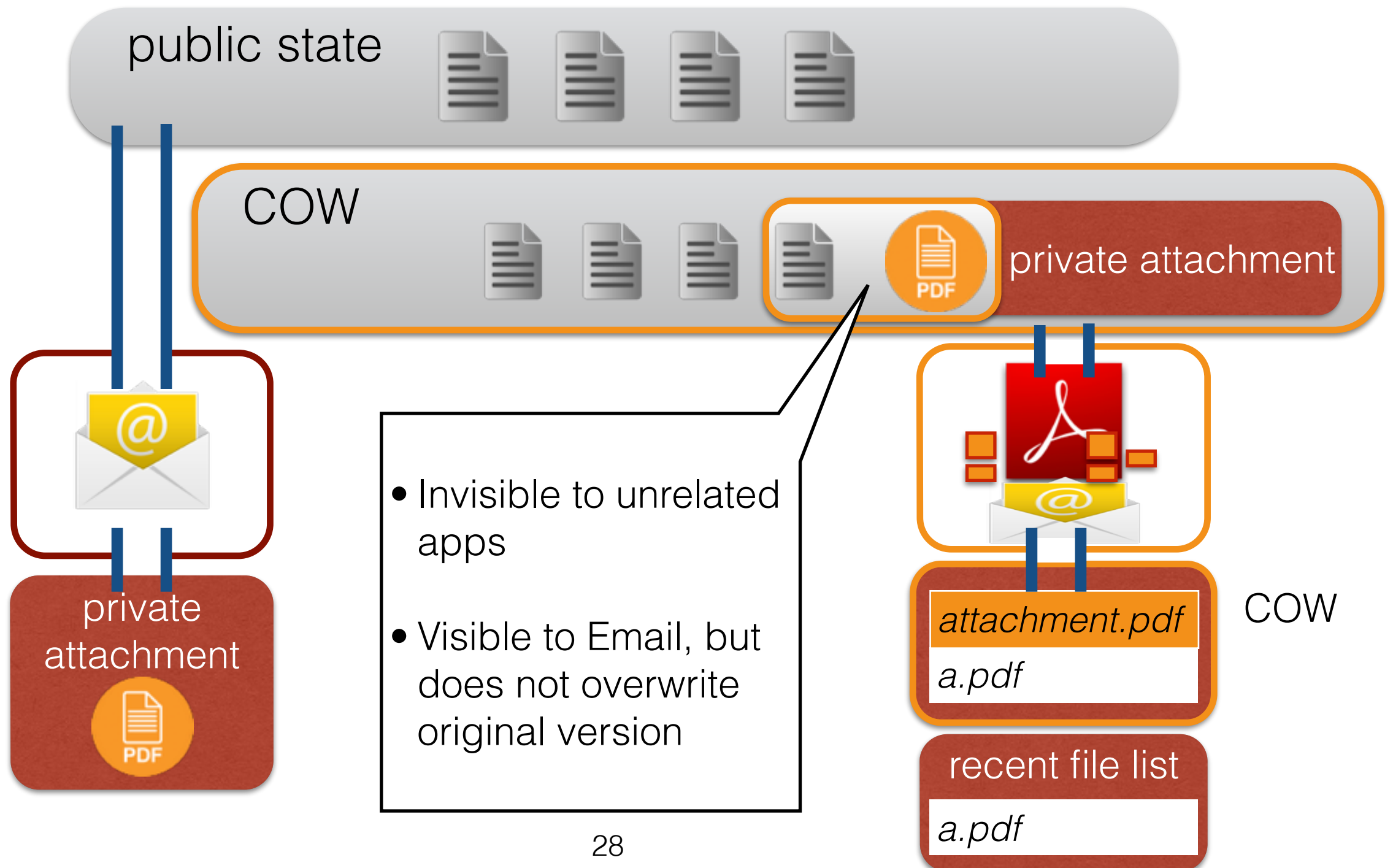
Delegate's view of public state contains initiator's private data



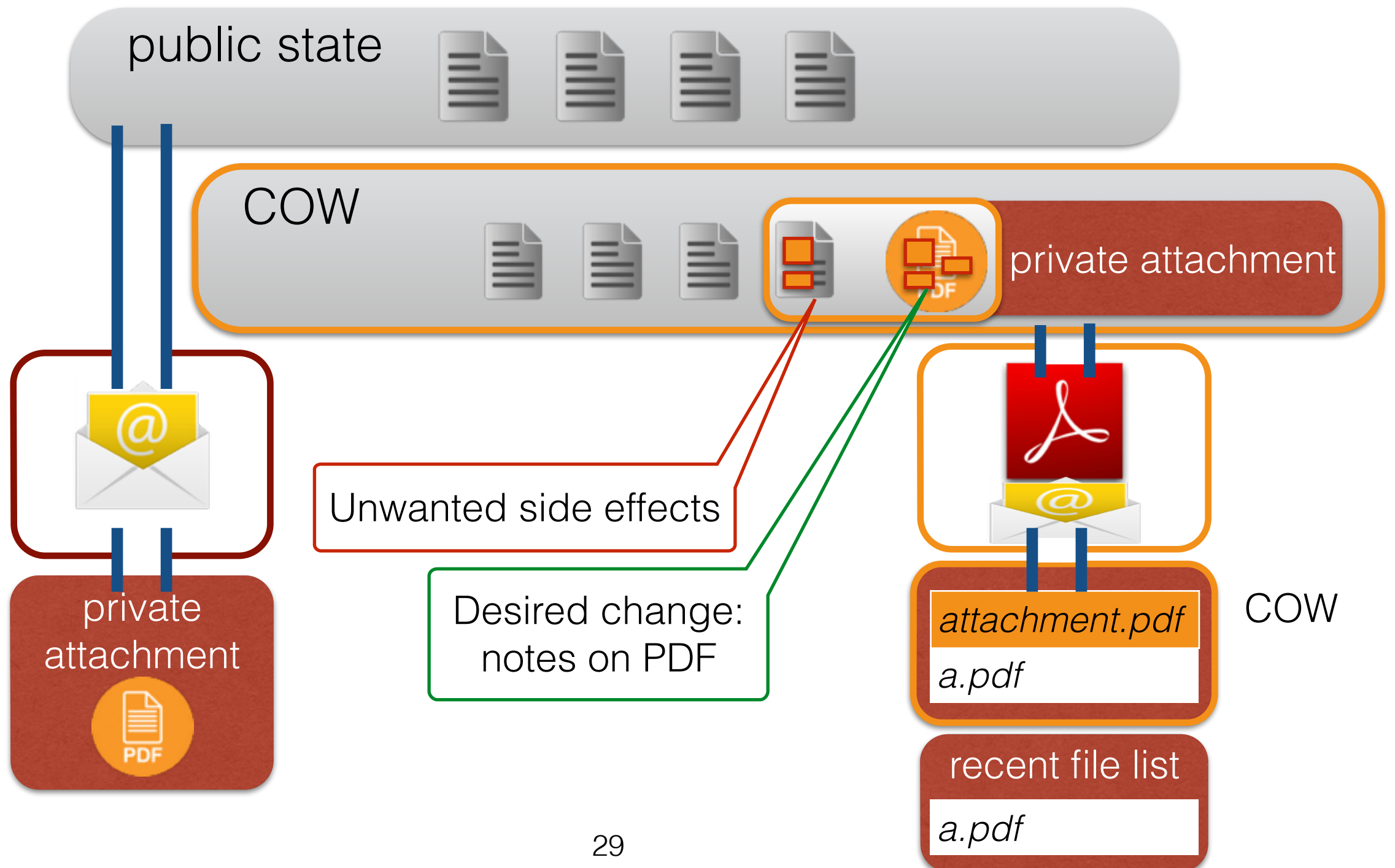
Delegate writes to its private state are confined by COW



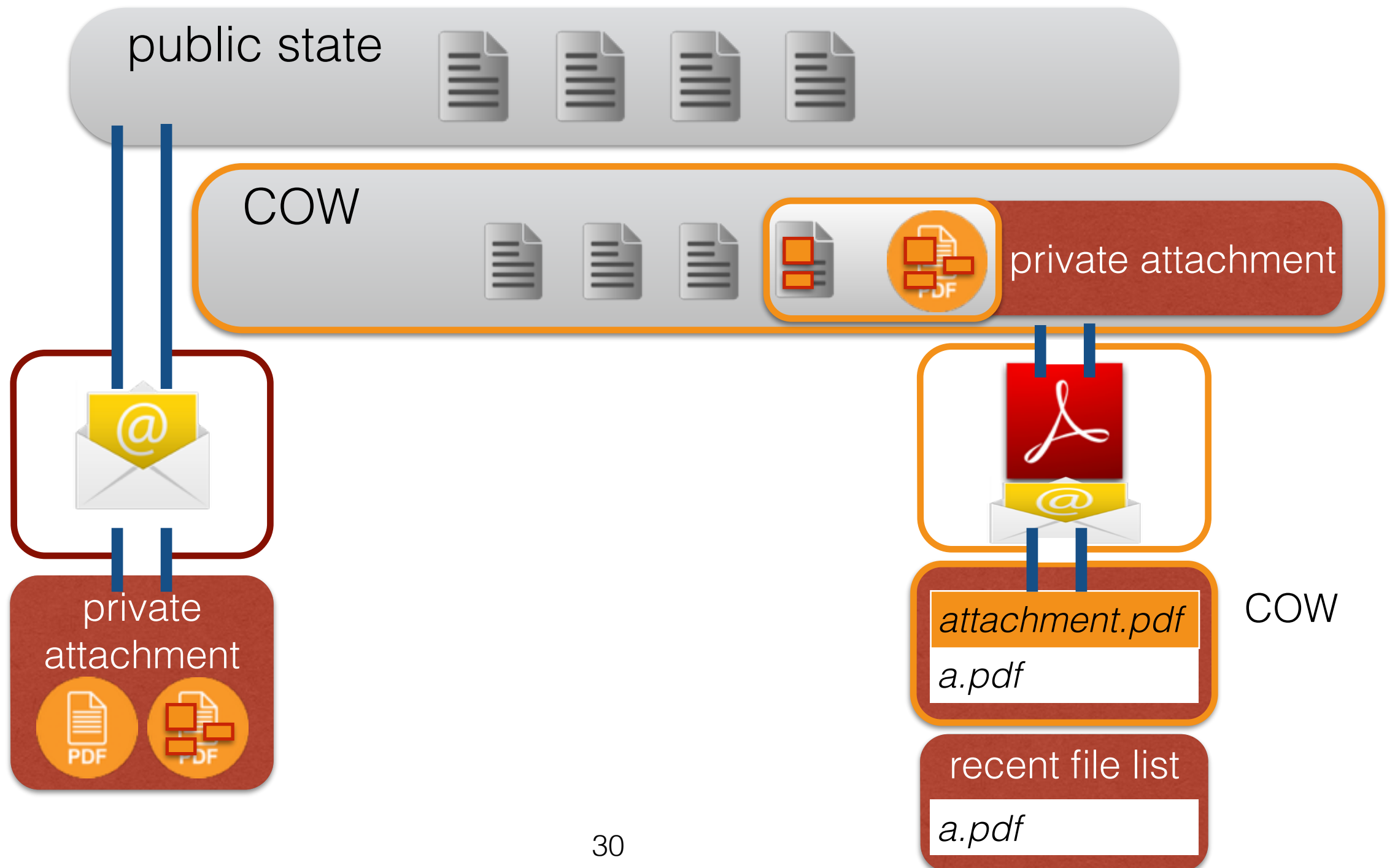
Delegate writes to its view of public state are confined by COW



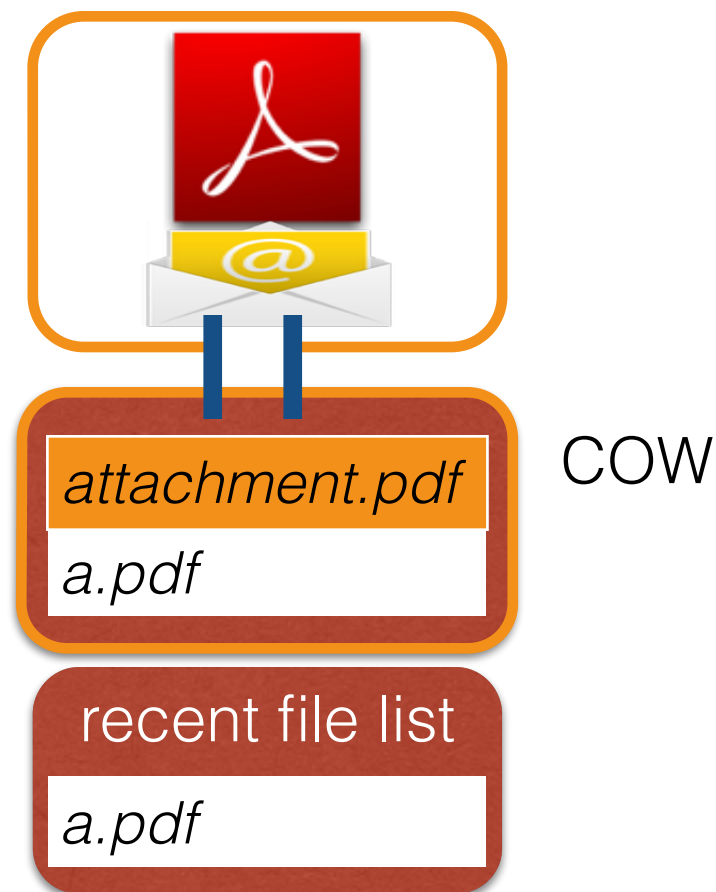
Save desired change, erase side effects



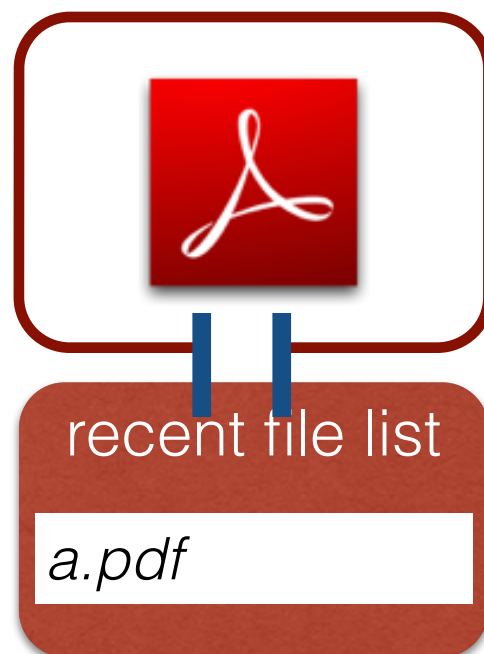
Save desired change, erase side effects



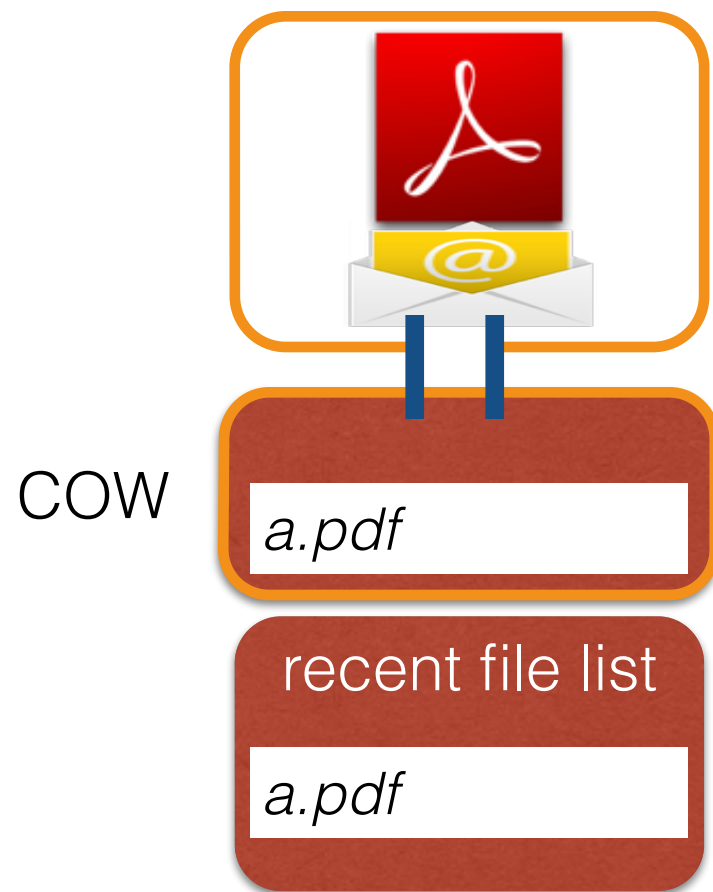
Delegate's changes to private state will be discarded



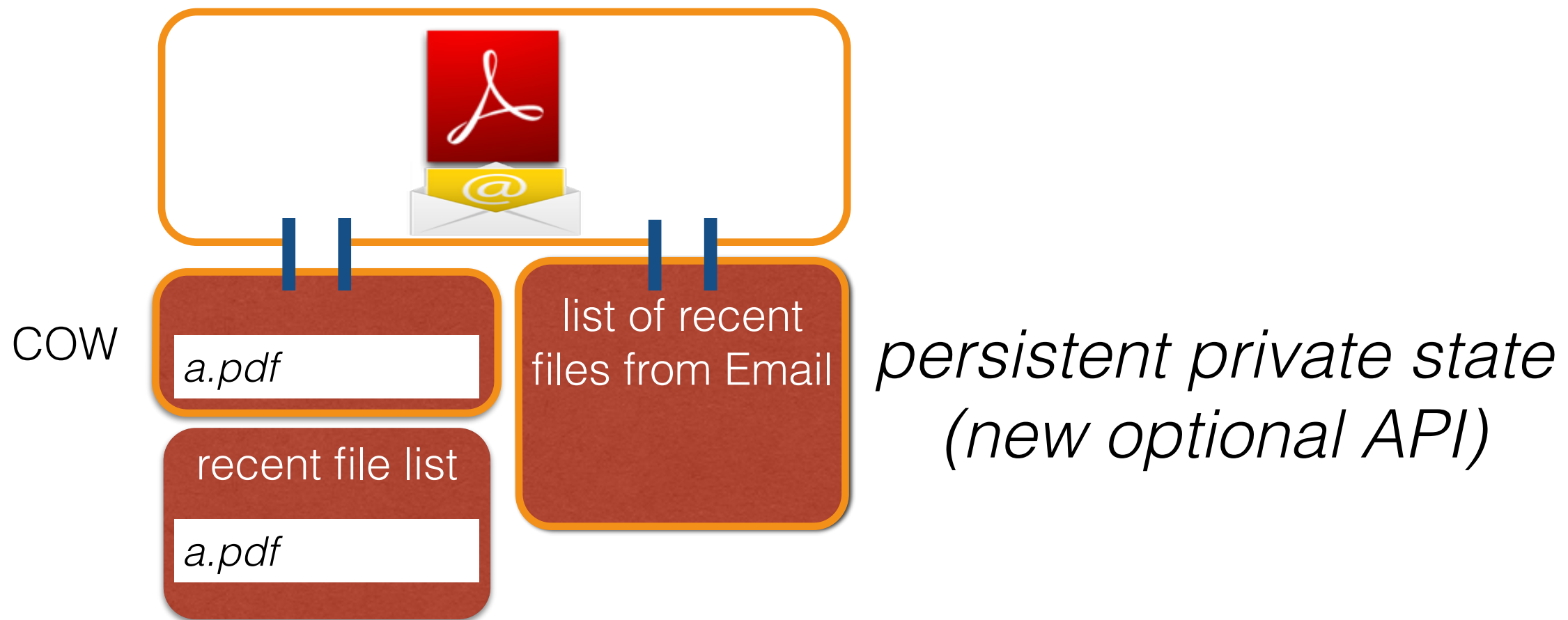
Adobe restarts as an initiator



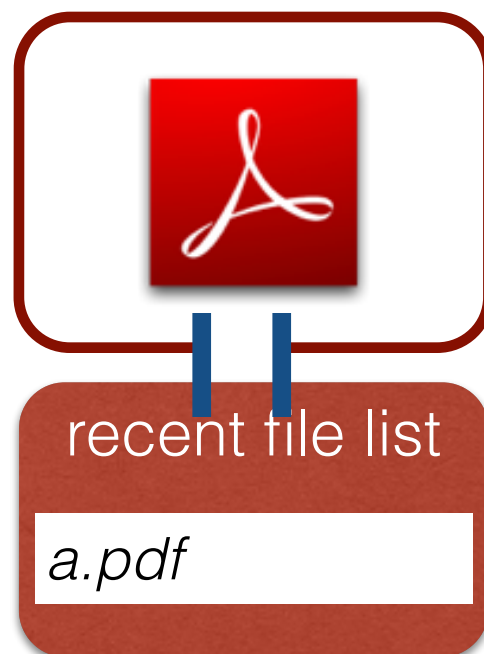
Adobe restarts on behalf of Email again



Keep persistent changes to delegate's private state

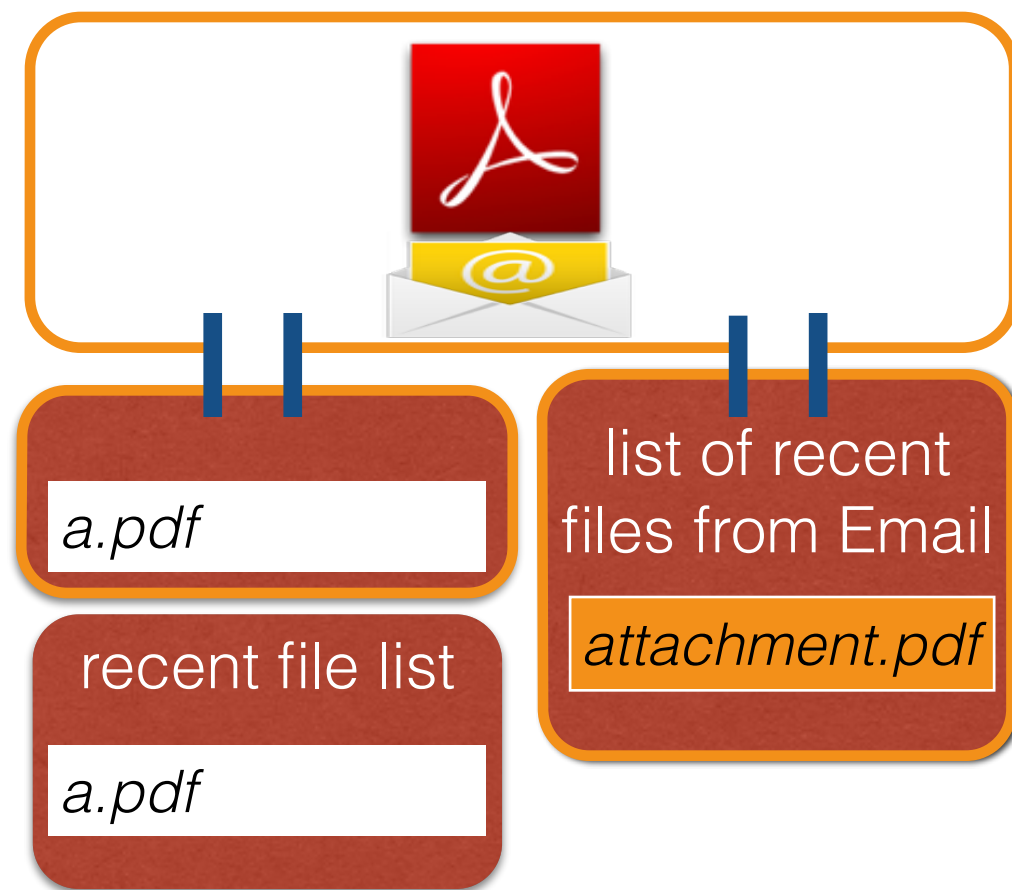


Adobe restarts as an initiator



*persistent private state
(new optional API)*

Adobe restarts on behalf of Email again



*persistent private state
(new optional API)*

Usability properties

- Transparent confinement for delegates
- Tainted data will not affect unrelated apps
- Initiator can get result back from delegate

Implementing Maxoid views

File system

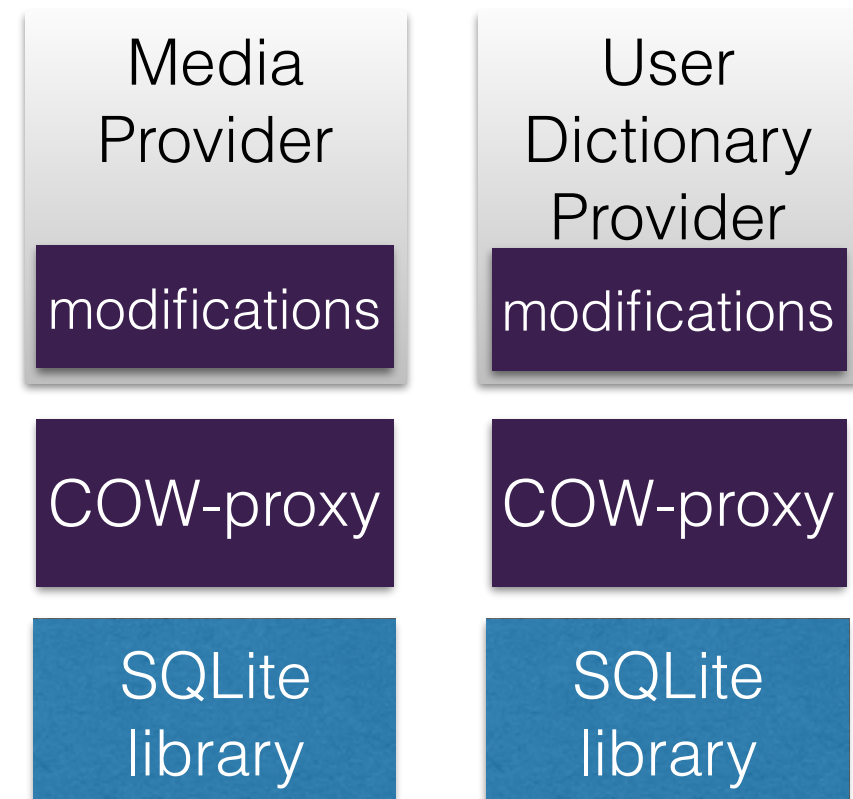
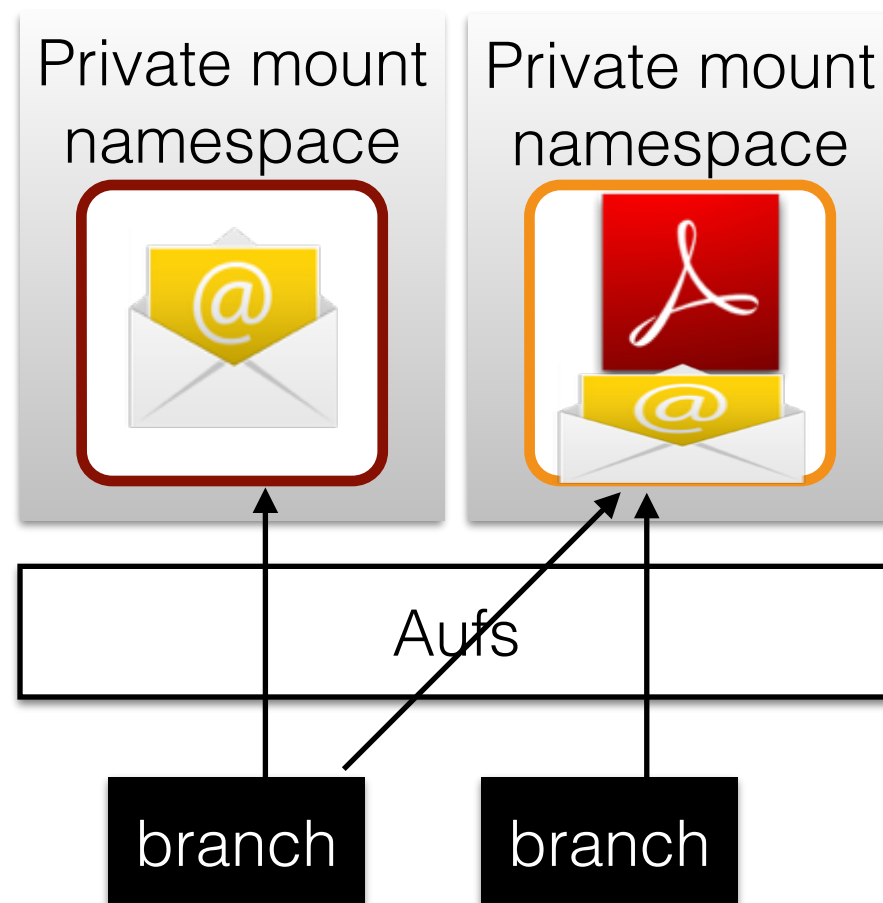
per-file copy-on-write

- private mount namespace
- Aufs — a union file system

System content providers

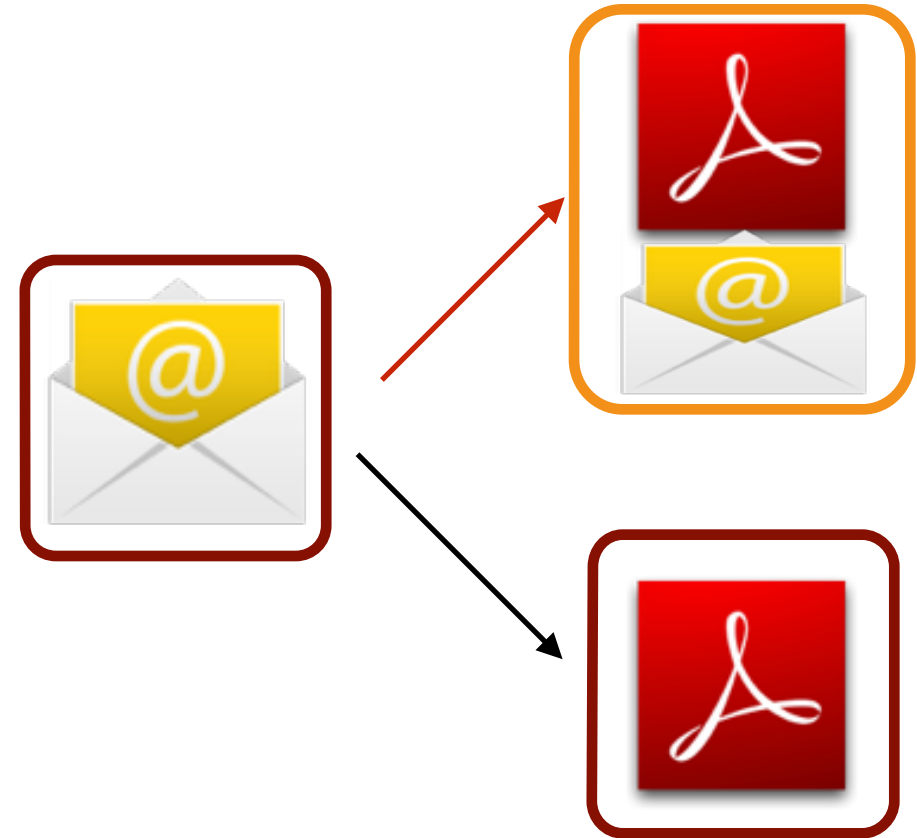
per-row copy-on-write

- Copy-on-write proxy for SQLite
- Provider-specific modifications



Confined & unconfined invocations from initiators

- Invoke another app
 - As a delegate
 - In the initiator mode
- Specify invocation type
 - Statically: with config. file, no code change
 - Dynamically: with new API, requires code change



Performance

- CPU-bound workload: no overhead
- File system/system content providers:
 - Initiators: negligible
 - Delegates:
 - Microbenchmarks
 - Worst-case: the *first* modification to a large file. Overhead depends on the file size. (Copy the entire file to private branch.)
 - 0%~31% in other cases
 - Macrobenchmarks:
 - Negligible user-perceived latencies in real-world apps (e.g., Adobe Reader, CameraMX, CamScanner)

Conclusion

- Information flow control can prevent data leakage in mobile platforms
- Maxoid provides coarse-grained, conservative information flow control
- Maxoid uses per-app-instance custom views of state to make the confinement transparent