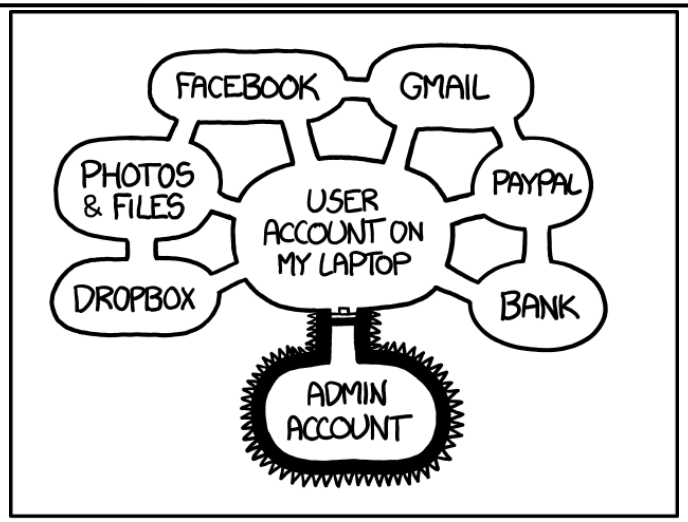
Lecture 17: Capabilities

CS 138

Spring 2026



IF SOMEONE STEALS MY LAPTOP WHILE I'M LOGGED IN, THEY CAN READ MY EMAIL, TAKE MY MONEY, AND IMPERSONATE ME TO MY FRIENDS,

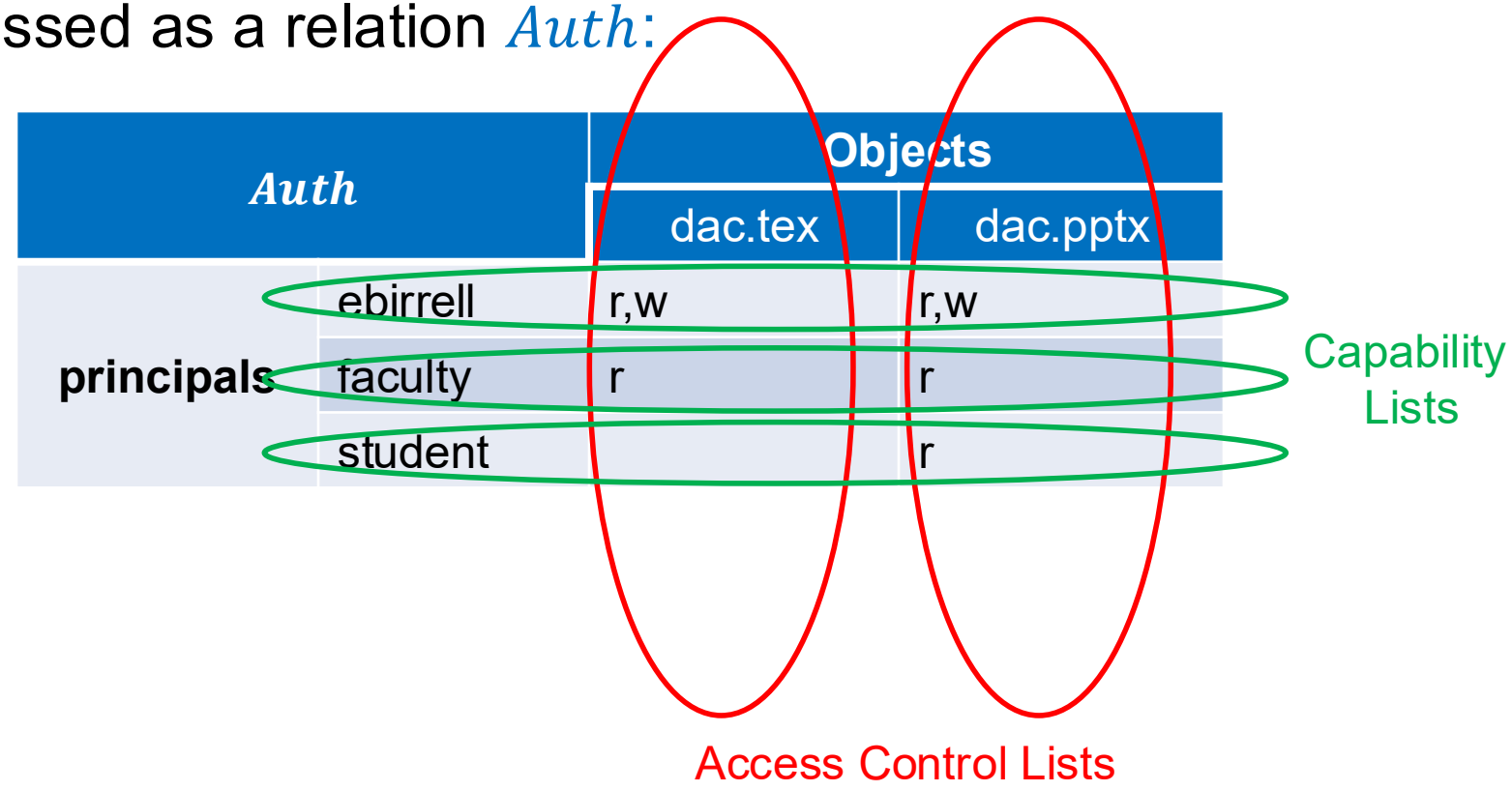
BUT AT LEAST THEY CAN'T INSTALL DRIVERS WITHOUT MY PERMISSION.

Where we were…

- **Authentication:** mechanisms that bind principals to actions
- **Authorization:** mechanisms that govern whether actions are permitted
  - Discretionary Access Control
  - Mandatory Access Control

Access Control Policy

- An **access control policy** specifies which of the **operations** associated with any given **object** each **principal** is authorized to perform
- Expressed as a relation $Auth$:



Capability Lists

- The capability list for a principal P is a list

$$\langle O_1, Privs_1 \rangle, \langle O_2, Privs_2 \rangle, \ldots, \langle O_n, Privs_n \rangle$$

  - e.g., ⟨dac.tex, {r,w}⟩ ⟨dac.pptx, {r,w}⟩
- **Capabilities** carry privileges.
  1) **Authorization:** Performing operation op on object O_i requires a principal P to hold a capability $C_i = \langle O_i, Privs_i \rangle$ such that $op \in Privs_i$
  2) **Unforgeability:** Capabilities cannot be counterfeited or corrupted.
- Note: Capabilities are (typically) transferable

Capabilities

- Advantages:

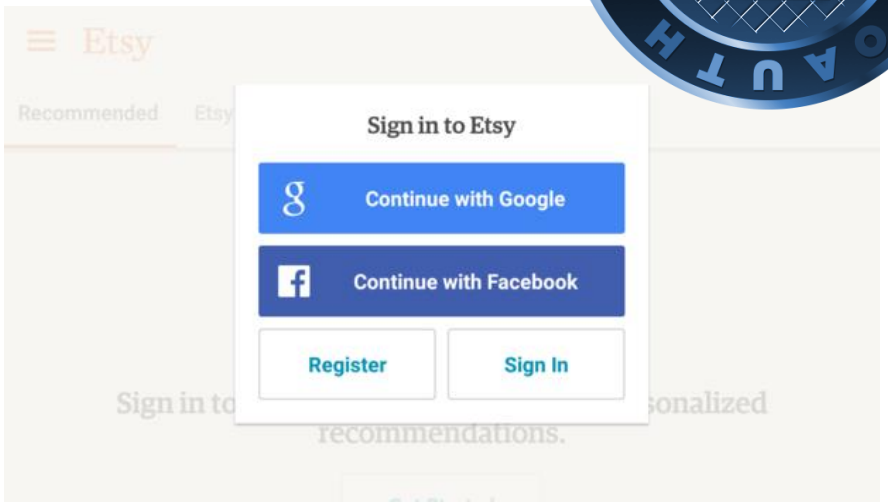
- Disadvantages:

Exercise 1: Capabilities

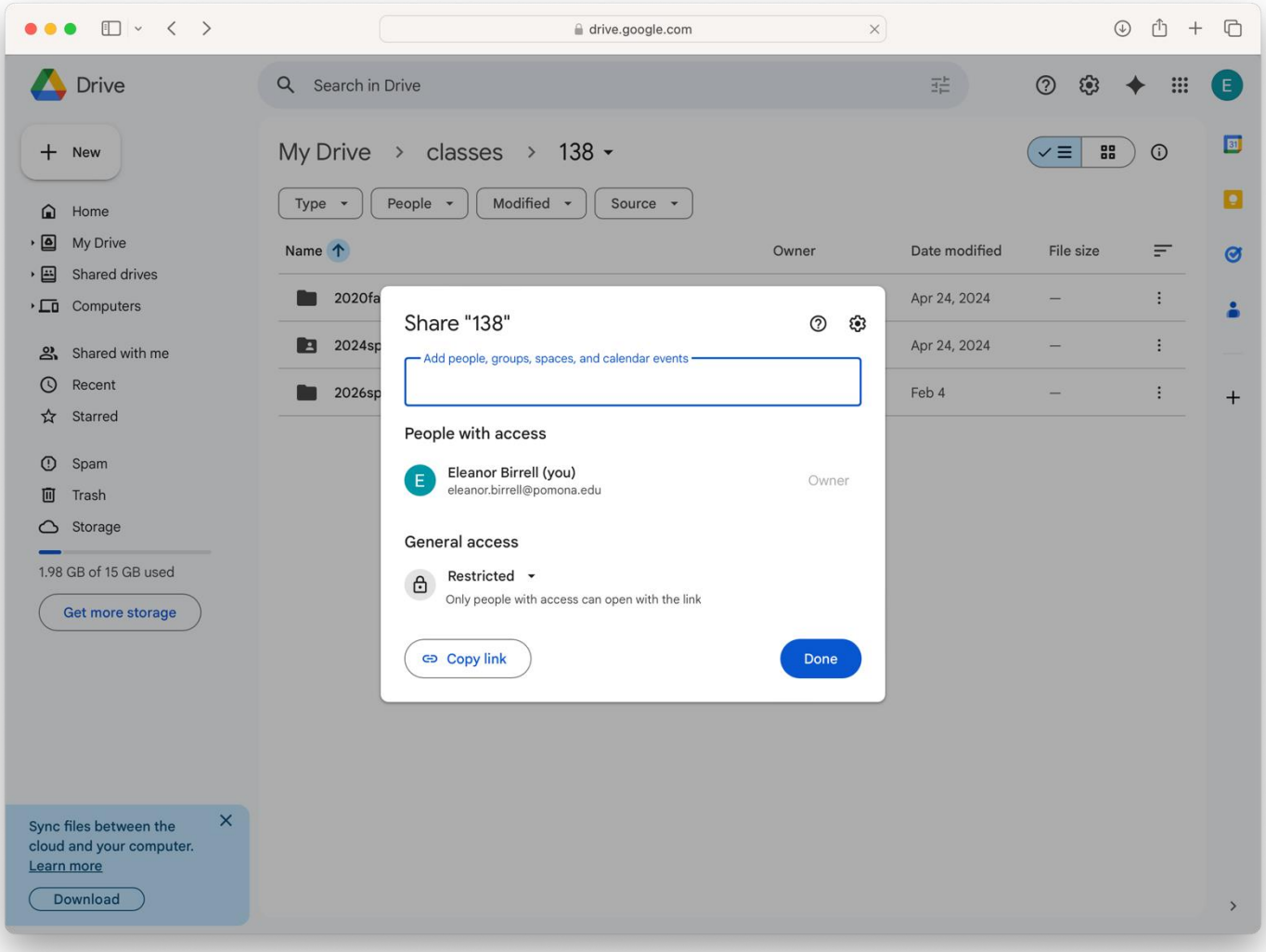
- Consider the following proposal: capabilities will be represented using a pair $\langle Name(Obj), Privs \rangle$, where $Name(Obj)$ is a random 128-bit string and $Privs$ is the set of privileges conferred by the capability. The function $Name$, if it exists at all, is kept secret. What functionality expected for capabilities does this alternative support and where (if at all) does it fall short?

Example: OAuth2

- Industry standard authorization protocol
- Used for single sign-on by major IDPs
  - Facebook, Google
- A **bearer token** contains a unique identifier



Example: Google Drive Links



Example: Authentication Cookies

HTTP

- Hypertext Transfer Protocol (HTTP) is an application protocol for distributed information systems
- Stateless request-response protocol

```
GET / HTTP/1.1
Host: developer.mozilla.org
Accept-Language: fr
```

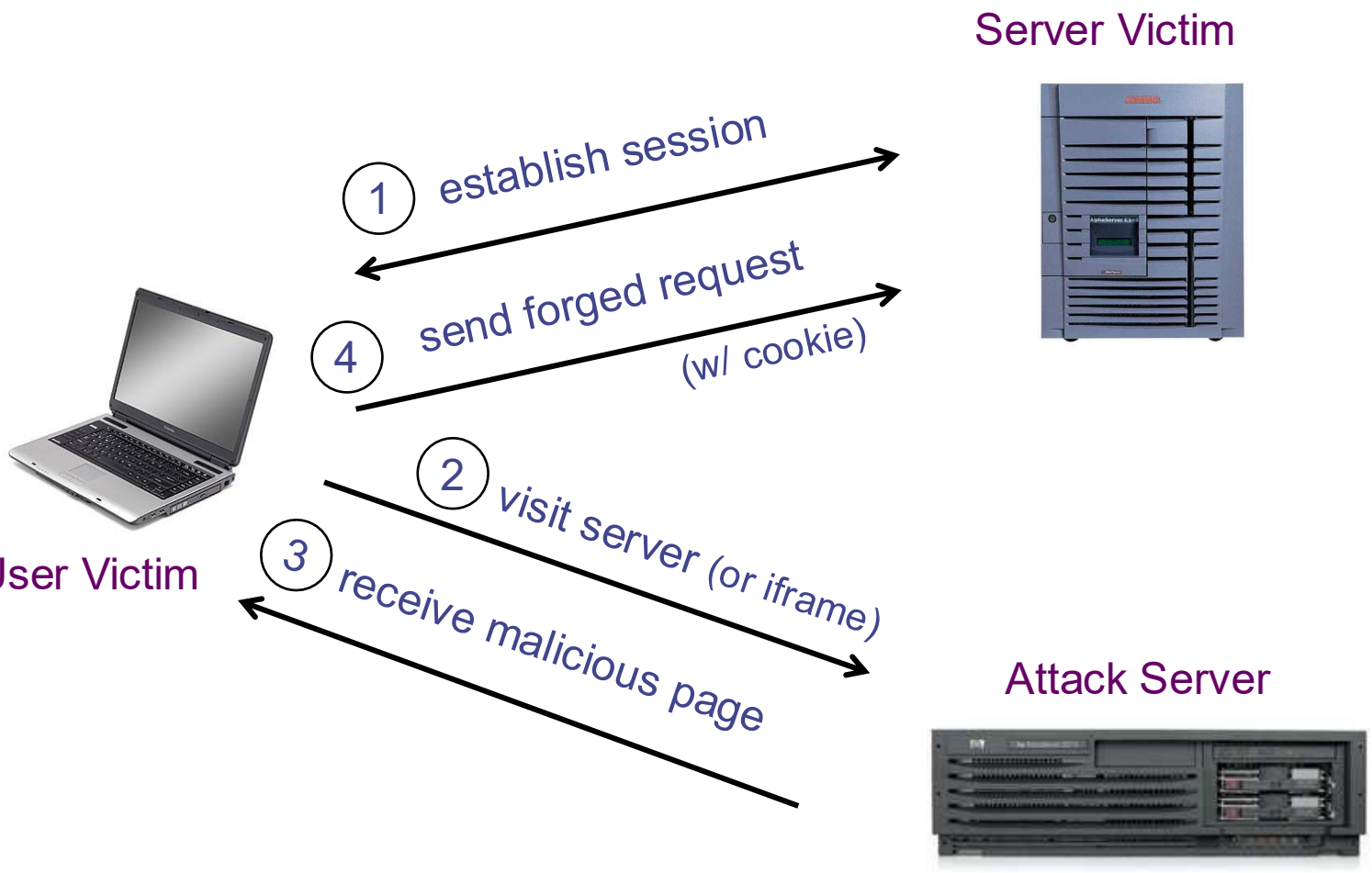
```
HTTP/1.1 200 OK
Date: Sat, 09 Oct 2010 14:28:02 GMT
Server: Apache
Last-Modified: Tue, 01 Dec 2009 20:18:22 GMT
ETag: "51142bc1-7449-479b075b2891b"
Accept-Ranges: bytes
Content-Length: 29769
Content-Type: text/html

<!DOCTYPE html... (here comes the 29769 bytes of the reques
```

Cookies

- Cookies are small blocks of data stored locally by the web browser
- Cookie is sent with every request to that domain
- Can be used to keep track of whether a user has authenticated (as which user)
- And also other things…
- Can be set by third parties

Cross-Site Request Forgery (CSRF)



CSRF Defenses

- Secret Validation Token:


```
<input type=hidden value=23a3af01b>
```

- Referrer Validation:


```
Referrer: http://www.facebook.com/home.php
```

- Custom HTTP Header:

```
X-Requested-By: XMLHttpRequest
```

- User Interaction (e.g., CAPTCHA)

Authenticity: Tagged Memory

- Example: IBM System 38
- tag = 0: normal memory
- tag = 1: this word + next are a capability
- In user mode, cannot modify tag bit or modify word with tag = 1
  - Exception: can copy capabilities
- pass capabilities in function calls

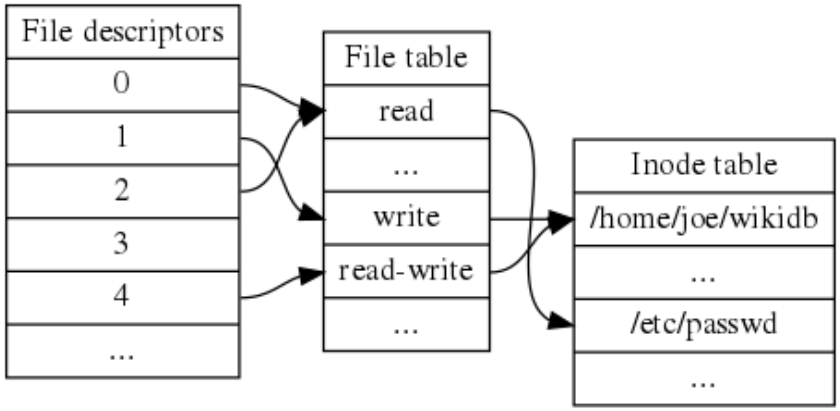
Authenticity: Protected Address Space

- General idea: store capabilities in region of memory we know how to protect
  - Option 1: protected kernel memory
  - Option 2: protected memory segment
- Note: OS must be trusted

- Store list of capabilities in process control block
- Capabilities referenced by index into c-list

Example: File Descriptor Table

- In Unix etc, a file descriptor is a handle used to reference files and I/O resources
- File descriptors have modes (read, write) and are stored in per-process file descriptor table
- File descriptors can be passed between processes using sendmsg()



Cryptographically-protected capabilities

- Object owner creates capabilities using a digital signature scheme
- Capabilities are triples $C = \langle O, Privs, \mathrm{Sig}(O, Privs; k_O) \rangle$
- **Authorization:** P is permitted to perform op on O if P produces a capability for O with $op \in Privs$ and a valid signature
- **Unforgeability:** digital signatures are unforgeable to adversaries who don't know private key k_O

- Note: assumes PKI

Restricted Delegation

- $C_0 = \langle O, Privs_0, pk_1, \sigma_0 \rangle$
  - where $\sigma_0 = \text{Sig}(O, Privs_0, pk_1; sk_0)$
- $C_1 = \langle O, Privs_1, pk_2, (Privs_0, pk_1, \sigma_0), \sigma_1 \rangle$
  - Where $\sigma_1 = \text{Sig}(O, Privs_1, pk_2, (Privs_o, pk_1, \sigma_0); k_1)$

To Authorize op with C_0:

1. Verify σ_0 is a valid signature of $(O, Privs_0, pk_1)$
2. Check that $op \in Privs_0$

To Authorize op with C_1:

1. Verify σ_0 is a valid signature of $(O, Privs_0, pk_1)$
2. Verify σ_1 is a valid signature of $(O, Privs_1, pk_2, (Privs_o, pk_1, \sigma_0))$
3. Check that $Privs_1 \subset Privs_0$
4. Check that $op \in Privs_1$

Exercise 2: Restricted Delegation

- Assume you have a credential

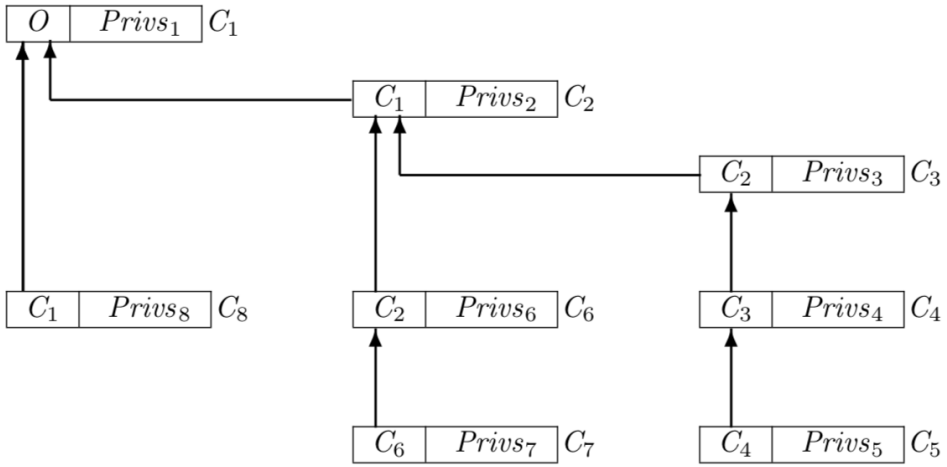
$$C_1 = \langle dac.pptx, \{r, w\}, pk_2, (\{r, w, x\}, pk_1, \sigma_0), \sigma_1 \rangle$$

1. Generate a credential C_2 that would authorized the holder to read (but not write) dac.pptx

2. Define the sequence of steps that should be taken to authorize op with C_2

Revocation

- Revocation Tags
  - Capabilities are tuples $C = \langle O, Privs, rt_c, \mathrm{Sig}(O, Privs, rt_c; k) \rangle$
  - Access to object O is guarded by a reference monitor; monitor maintains a list of revoked tags rt_c
- Capability Chains
  - Objects can be other capabilities!
  - P is authorized to perform op on O if P holds a capability C_i and $op \in Privs_k$ holds for every capability C_k in the chain from C_i to C_1



Keys as capabilities

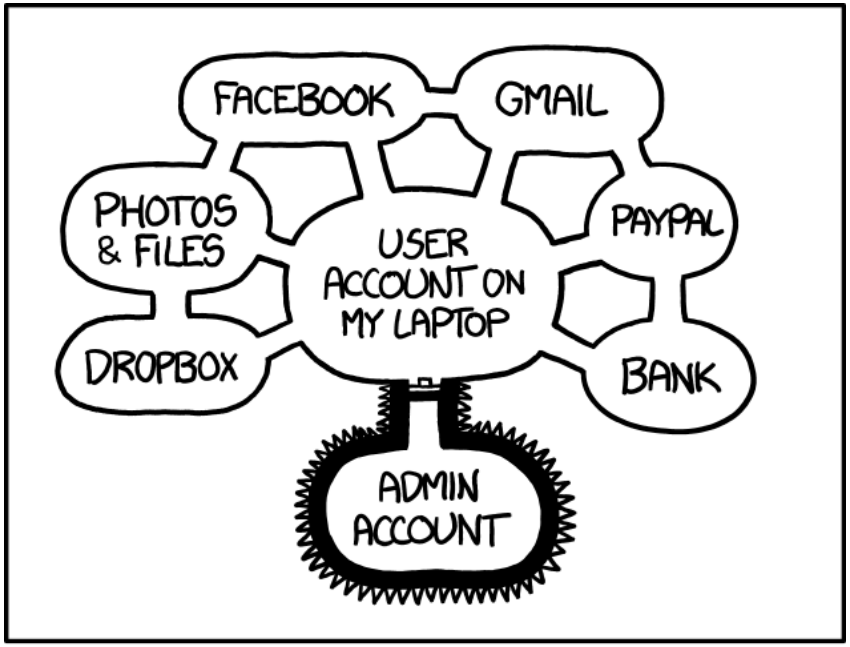
- Encrypt object
- Decryption method functions as reference monitor:
  - **Authorization:** correct key will decrypt object -> allow access
  - **Unforgeability:** incorrect key will not decrypt
- Note: no notion of separate privileges

Example: Mac keychains

- OSX/iOS password manager
- uses password-based encryption (AES-256) to store username/password credentials
- supports multiple keychains

What about privacy?

Capabilities



IF SOMEONE STEALS MY LAPTOP WHILE I'M LOGGED IN, THEY CAN READ MY EMAIL, TAKE MY MONEY, AND IMPERSONATE ME TO MY FRIENDS,

BUT AT LEAST THEY CAN'T INSTALL DRIVERS WITHOUT MY PERMISSION.