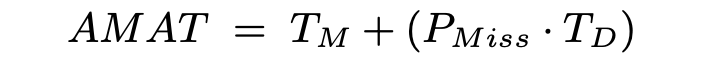
**Page replacement**

Average Memory Access Time (AMAT)



**Tm = time require to access memory**

**Td - time require to access disk**

**Pm = Probability of page miss**

For example with 10 page access, 9 hit and 1 miss, which is equivalent to a hit rate of 90%, therefore **Pm** = 10% or 0.1. In the meantime **Tm** = 100 ns and **Td** = 10 ms.

AMAT = 100 ns + 0.1 x 10 ms = 100ns + 1ms = 1.0001ms

Hit rate



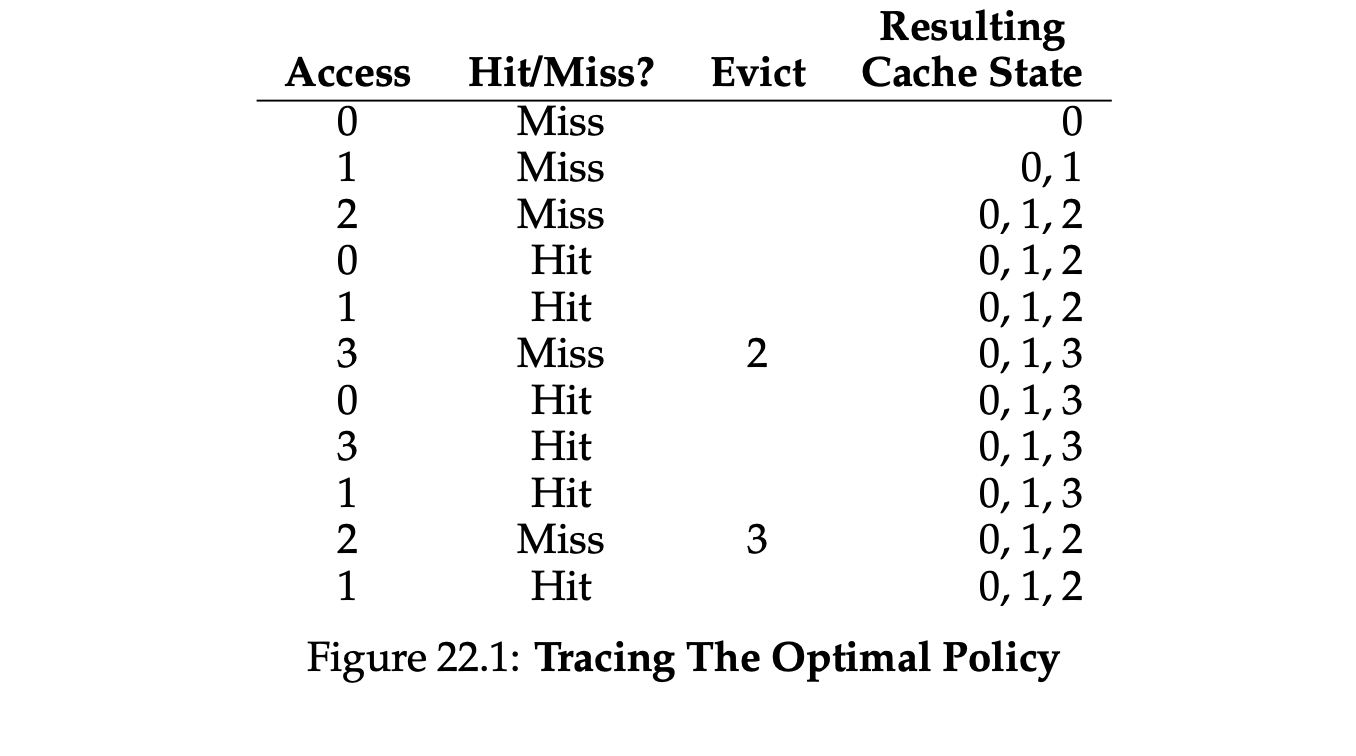
计算方法很简单，就是命中次数除以总次数，同时还有另一种计算方法是忽略compulsory miss（就是每个新task第一次出现时的miss，因为必定会miss）

如optimal，一共11次access，其中4次（0，1，2，3）是必定miss的，所以剔除，也就是6/7 = 85.7%

**Different policy**

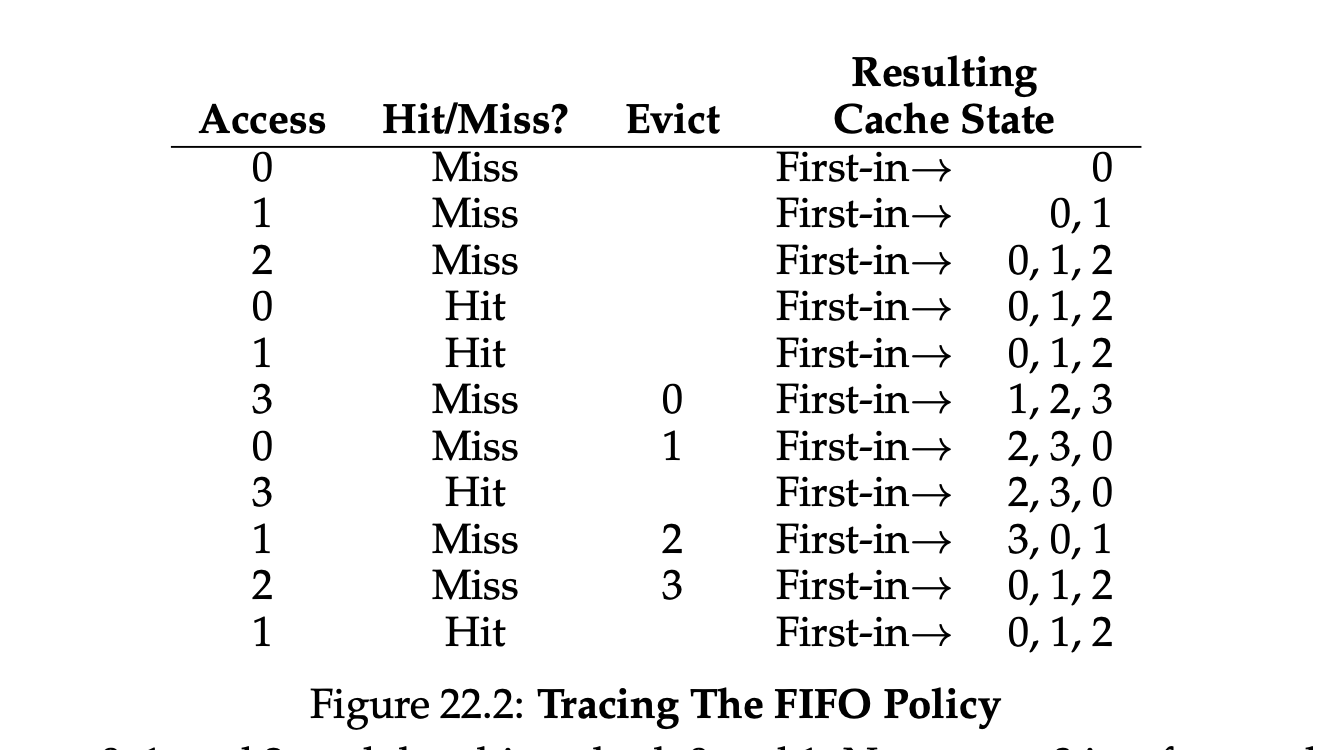
1. Optimal policy

这是理论上最好的policy，原理是假设系统知道所有他即将进行的task，那么在page frame有限的情况下，系统可以选择最远的task来剔除。比如frame里有0，1，2，此时frame已满同时task3即将到来，因此我们需要踢出去一个。这时候我们看到3之后的下一个task马上就是0且1紧随其后，但是2在好几个task之外，这时我们就可以把2踢出去替换成3，以此类推。



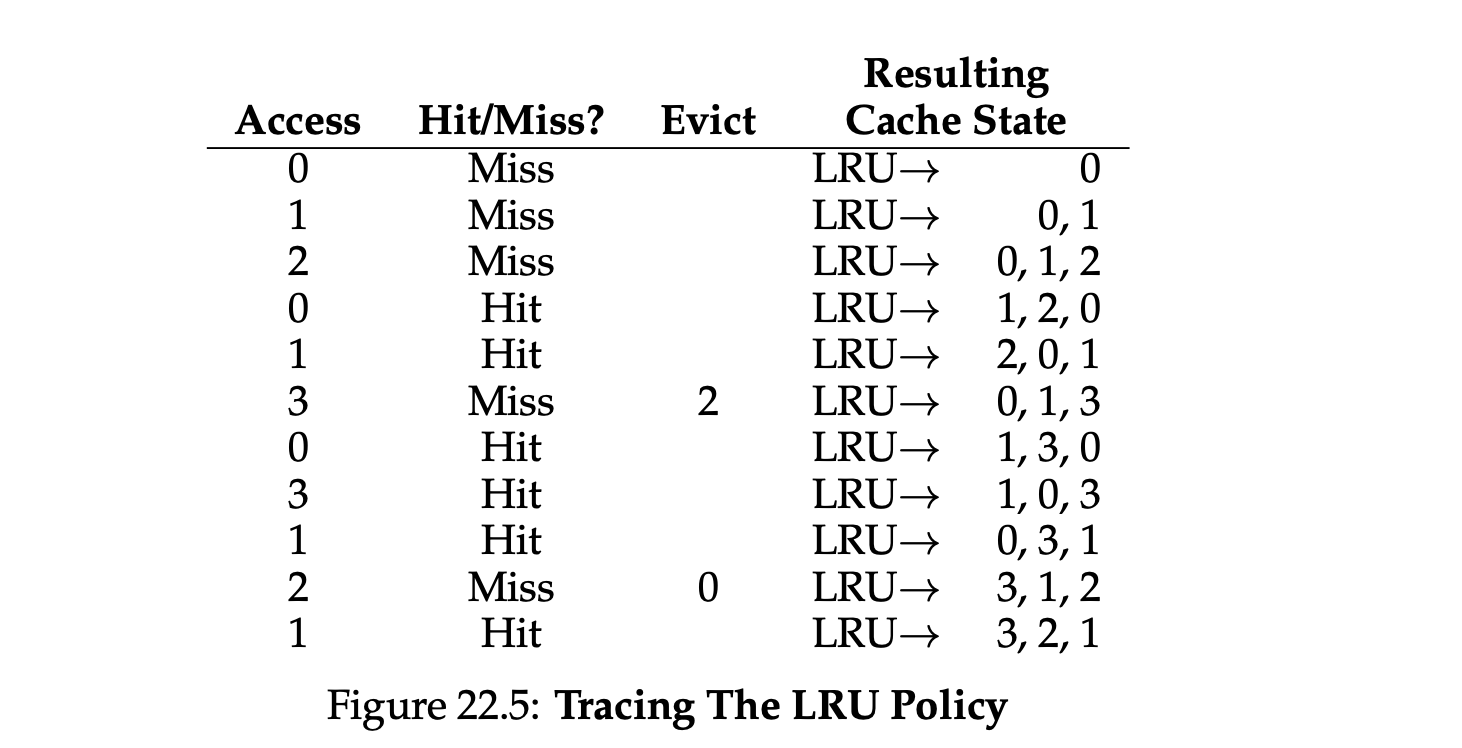
2. FIFO (First In First Out)

原理很简单，先进先出



3. Random

原理很简单，就是随便选一个



4. LRU (Least Recent Used)

就是选择最久没有hit过的task踢出去，还有一种LFU (Least Frequently Used) 踢用得最少的

这里可能会出现的一种用法是clock algorithm，就是像钟表一样一个一个扫过去，但是具体选择踢哪个就需要used bit (referenced bit) 以及dirty bit的帮助。

Used bit就是在记录了在一定时间内这个page是否被read 或者 wirte过，如果有就是1如果没就是0，同时used bit每隔一段时间都会清零，不然有些过时的page本可以踢出去却还是留在memory里。

Dirty bits 用于记录这个page是否被修改过，系统倾向于踢出没修改过的page，因为修改过的page还需要把他们记录回disk里，这需要很多时间，但是没修改过的page直接删除TLB里的内容就行，disk里依然存有与他一模一样的page。

系统优先踢出没用过也不是脏的page，如果不存在则踢出没用过但是是脏的page，因为用过的肯定更重要一点

Thrashing:

指进程因为分配的memory不够因此需要不断进行IO从disk里面调用东西进而导致CPU工作效率变得很低。比如给所有进程均匀分配memory，就会导致有些需要更多memory的process容量不够需要已知IO，有些很小的的进程会剩余很多memory。

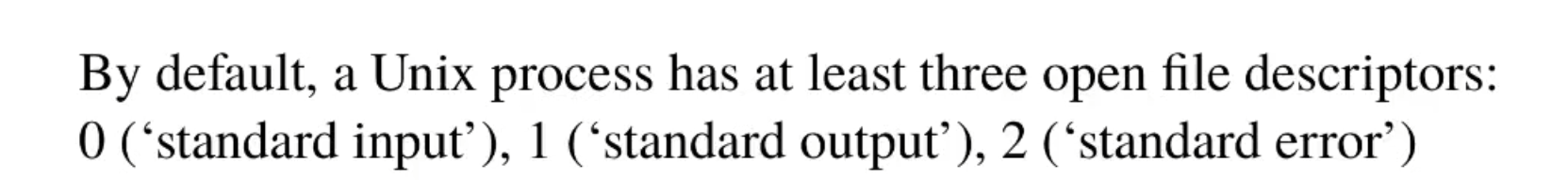
Malloc is highly optimize concurrent codebase

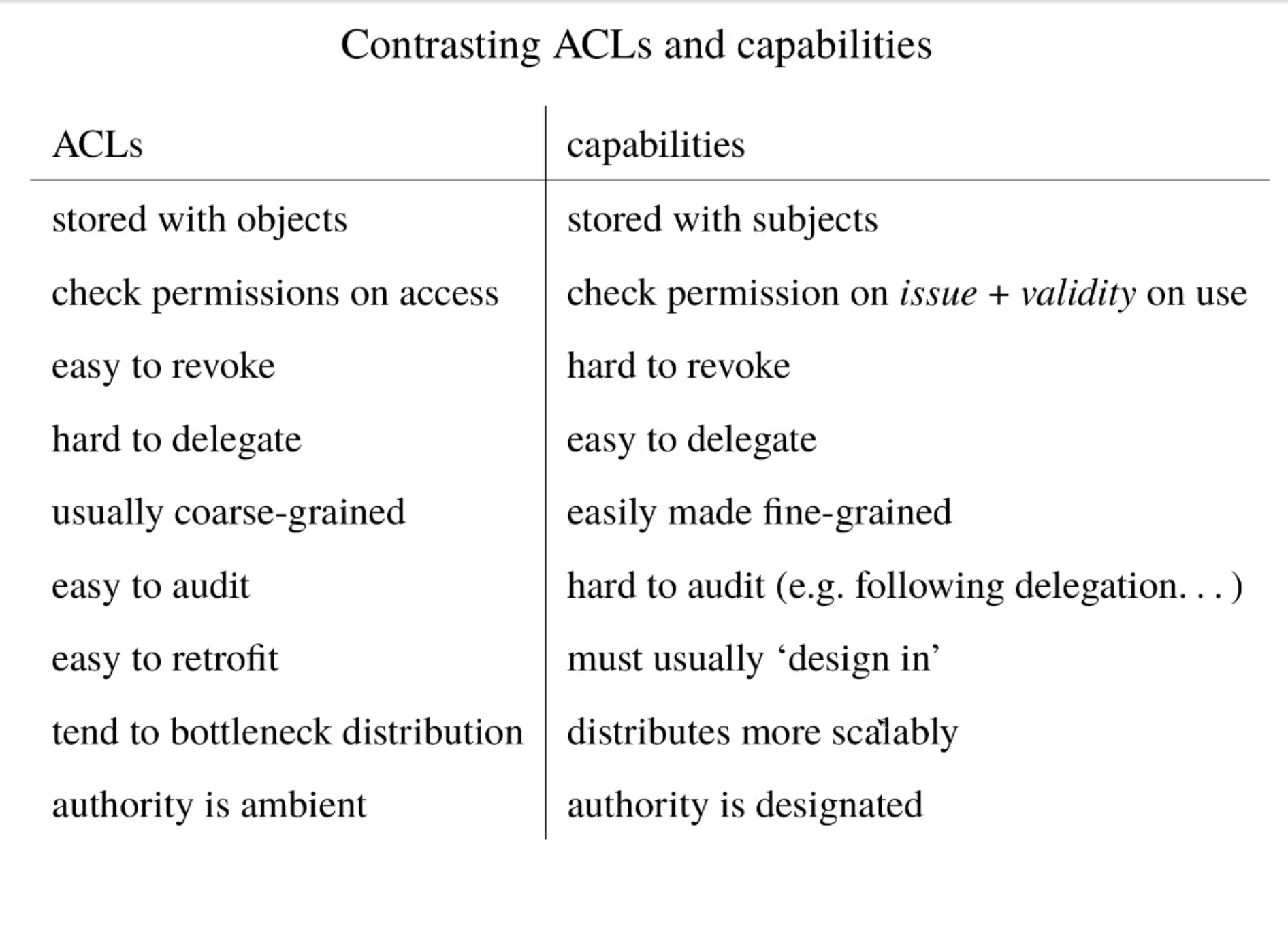
Access Control

Subject: A subject is a entity that want to perform the access, e.g user or a process

Object: Things that subject want to access, e.g file or device

Access

\



Here's a detailed explanation of each point contrasting ACLs (Access Control Lists) and capabilities:

Stored with objects vs. stored with subjects:

1. ACLs are stored with objects (resources), meaning the permissions are associated with the resources themselves. Capabilities, on the other hand, are stored with subjects (users or processes), meaning the permissions are associated with the user or process that holds the capability.

Check permissions on access vs. check permission on issue + validity on use:

1. With ACLs, permissions are checked when the resource is accessed. Capabilities involve checking permissions when the capability is issued (i.e., granted to the user or process) and ensuring its validity when it is used.

Easy to revoke vs. hard to revoke:

1. Revoking permissions is relatively easy with ACLs since the permissions are stored with the object. Revoking capabilities is more challenging because they are stored with subjects, and tracking all the subjects with a specific capability can be difficult.

Hard to delegate vs. easy to delegate:

1. ACLs can be hard to delegate due to their complexity and lack of flexibility, as mentioned in a previous answer. Capabilities make delegation easier because they can be passed from one subject to another, granting the receiving subject the necessary permissions.

Usually coarse-grained vs. easily made fine-grained:

1. ACLs typically provide a coarse-grained level of control over permissions, while capabilities can be made fine-grained, allowing for more precise control over permissions.

Easy to audit vs. hard to audit:

1. Auditing ACLs is relatively straightforward since the permissions are stored with the object. Auditing capabilities is more difficult because they are stored with subjects, and following the chain of delegation can be complex.

Easy to retrofit vs. must usually 'design in':

1. ACLs can be added to existing systems more easily, whereas capabilities typically need to be designed into the system from the beginning.

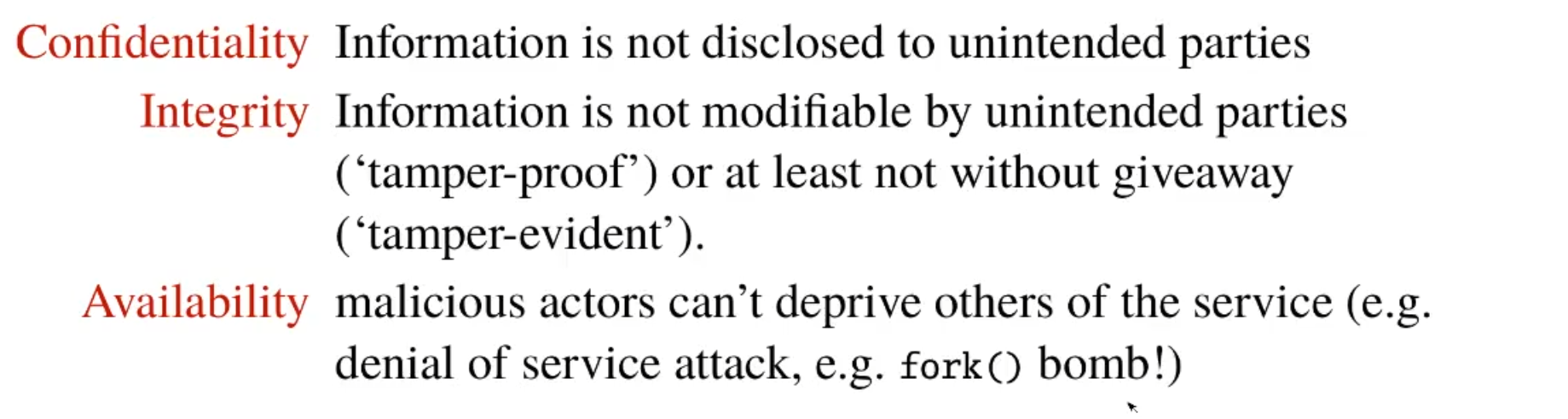
Tend to bottleneck distribution vs. distributes more scalably:

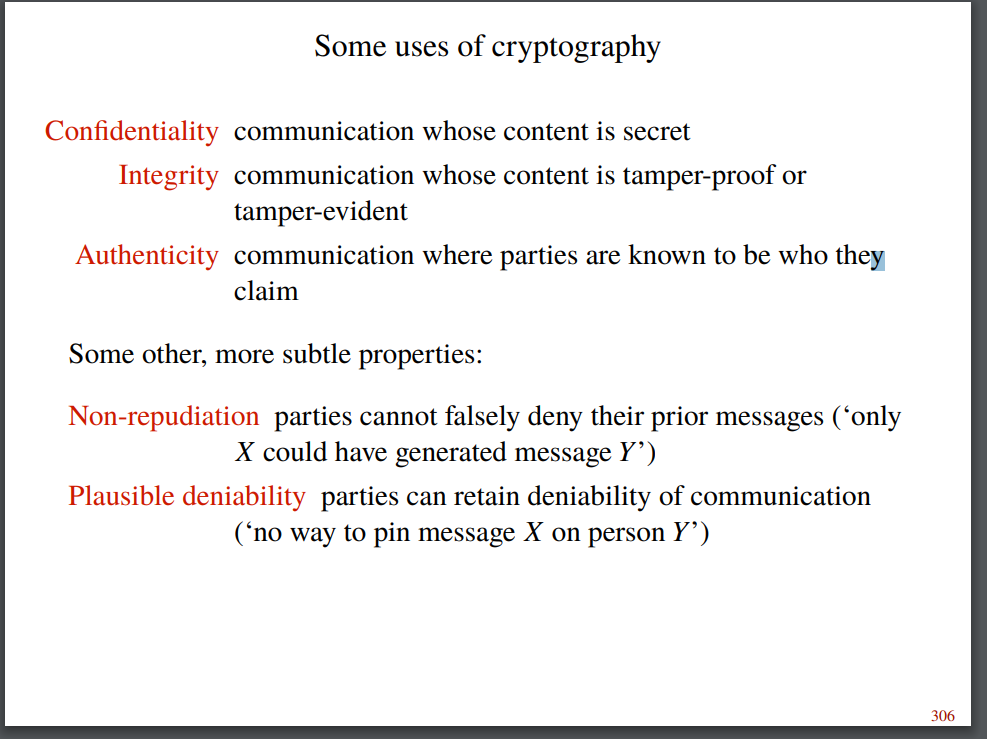
1. ACLs can create bottlenecks in distributed systems, as all access requests must go through the ACL mechanism. Capabilities, stored with subjects, can scale more effectively in distributed systems because the permissions check can be performed locally by the subject holding the capability.

Authority is ambient vs. authority is designated:

1. In ACL-based systems, authority is ambient, meaning it is inherent in the user or process attempting to access the resource. In capability-based systems, authority is designated, meaning it is explicitly granted to a user or process via the capability.

Security property





Confidentiality: 信息不能别其他人看到

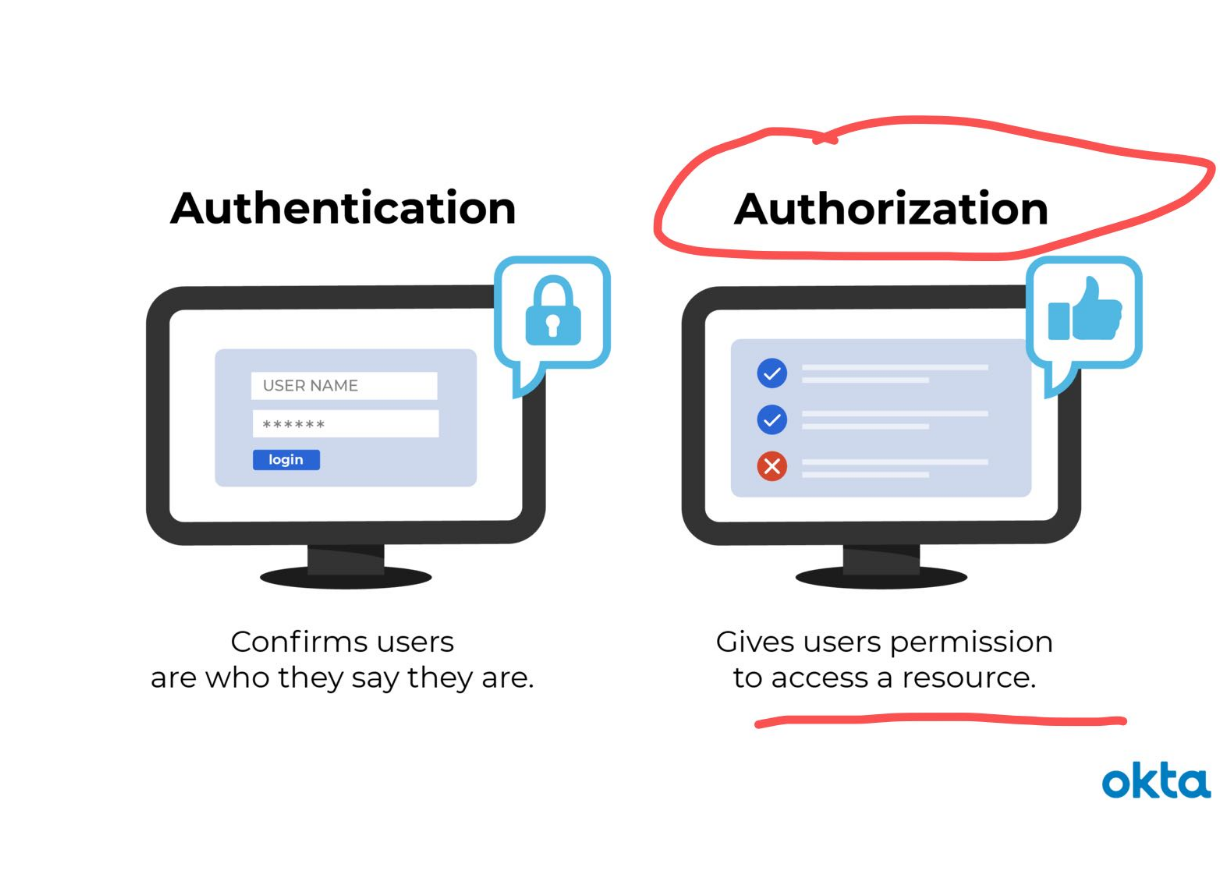
Integrity: 信息不能被其他人更改

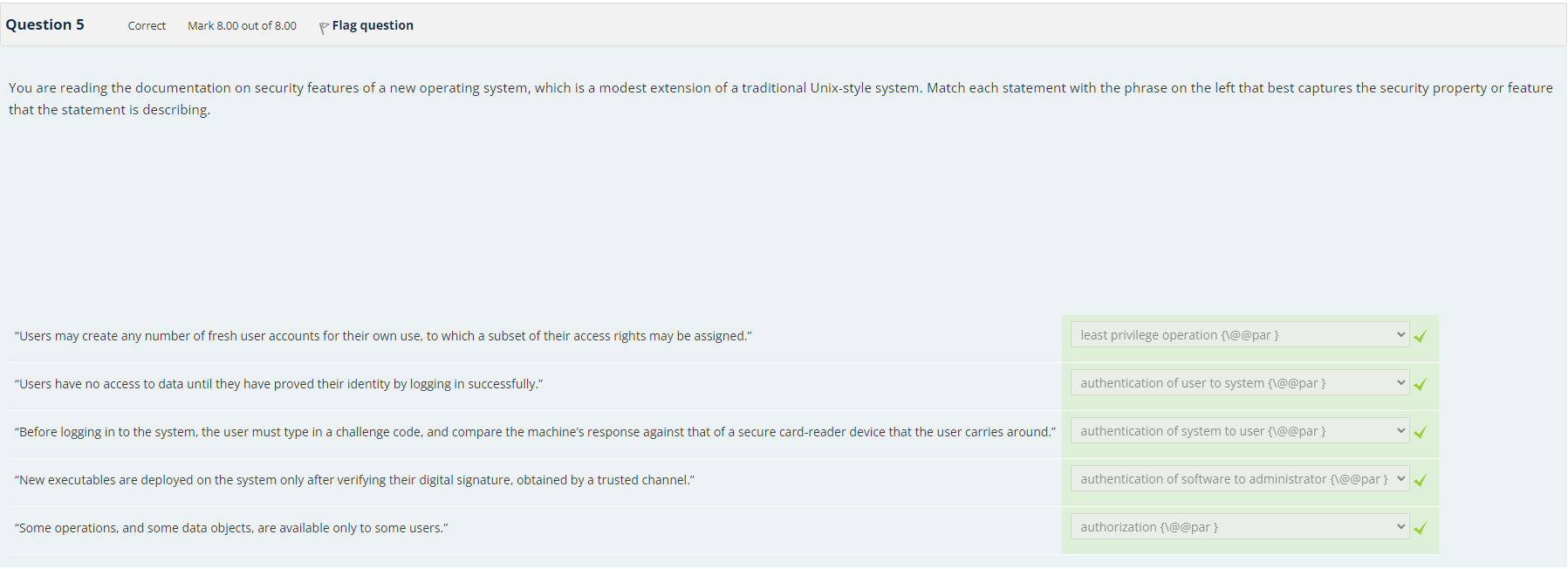
Availablity：危险行为不能影响他人的服务，比如炸房挂就违反了

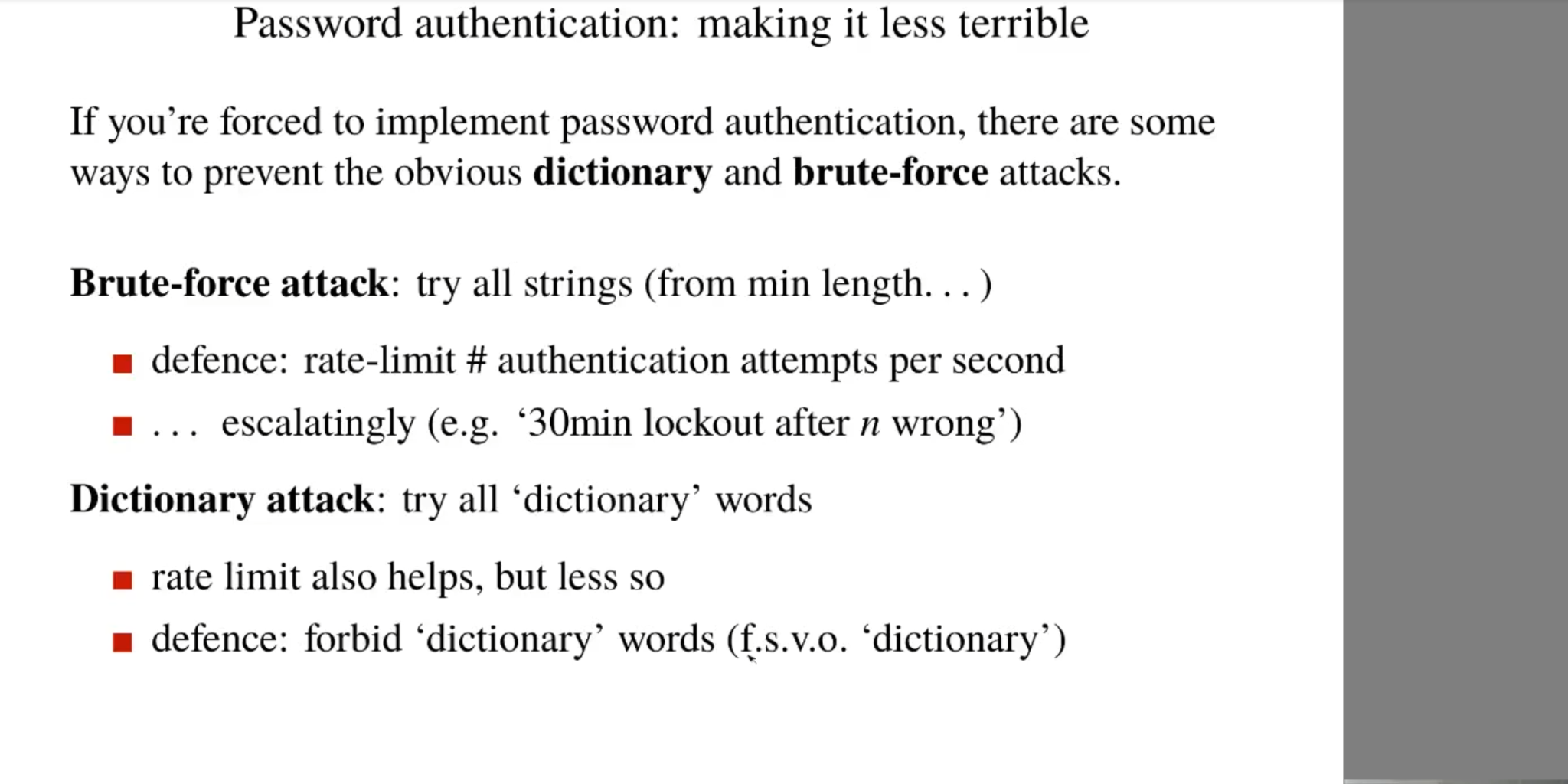
Non-repudiation（不可否认）：你发了就是发了不能说没发,且能被确认

Plausible deniability：但也不是完全认死了就是你发的，你可以通过手段证明没发

Authenticity（真实性）:传达的信息是它声称的内容，包括沟通方是他们声称的人







实际上电脑上不存密码，而是存散列，类似加密后的密码，通过特定的function才能generate，所以就算泄露了也没问题，其他人不可能知道。