Lecture 10:

Access Control II:

Authorization

4th Year Course- CCSIT, UoA

Aim of Lecture

- To study authorization as the part of access control concerned with restrictions on the actions of authenticated users.

- To explore CAPTCHAs

Authentication vs. Authorization

- **Authentication** — Are you who you say you are?
  - Restrictions on who (or what) can access system
- **Authorization** — Are you allowed to do that?
  - Restrictions on actions of authenticated users
- In its most basic form, authorization deals with the situation where we've already authenticated Alice and we want to enforce restrictions on what she is allowed to do.
- Note that while authentication is binary (either a user is authenticated or not), authorization can be a much more fine grained process

Lampson's Access Control Matrix (1)

- This matrix contains all of the relevant information needed by an operating system to make decisions about which users are allowed to do what with the various system resources.
- We'll define a ***subject*** as a user of a system (not necessarily a human user) and an ***object*** as a system resource.
- **Two fundamental constructs** in the field of authorization are access control lists, or **ACLs**, and capabilities, or **C-lists**.
- Both ACLs and C-lists are derived from Lampson's access control matrix, which has a row for every subject and a column for every object.
- Sensibly enough, the access allowed by subject *S* to object *O* is stored at the intersection of the row indexed by *S* and the column indexed by *O*.
- An example of an access control matrix appears in the next slide, where *x*, *r*, and *w* stand for execute, read, and write privileges, respectively.

Lampson's Access Control Matrix (2)

- **Subjects** (users) index the rows
- **Objects** (resources) index the columns

	OS	Accounting program	Accounting data	Insurance data	Payroll data
Bob	rx	rx	r	—	—
Alice	rx	rx	r	rw	rw
Sam	rwx	rwx	r	rw	rw
Accounting program	rx	rx	rw	rw	rw

Performance for authorization operations (1)

- Since all subjects and all objects appear in the access control matrix, it contains all of the relevant information on which authorization decisions can be based. However, there is a practical issue in managing a large access control matrix.
- Access control matrix has all relevant information. But this could be (e.g.) 100's of users, 10,000's of resources. Then matrix has 1,000,000's of entries
- How to manage such a large matrix?
- *Note: We need to check this matrix before access to any resource by any user*
- How to make this more efficient/practical?

Performance for authorization operations (2)

- The solution is that the access control matrix can be partitioned into more manageable pieces. There are **two** obvious ways for this:

1. We could split the matrix into its columns and store each column with its corresponding object. Then, whenever an object is accessed, its column of the access control matrix would be consulted to see whether the operation is allowed. These columns are known as access control lists, or **ACL**s.
2. We could store the access control matrix by row, where each row is stored with its corresponding subject.. This approach is know as capabilities, or **C-lists**.

Access Control Lists (ACLs)

- ACL: store access control matrix by **column**
- Example: ACL for **insurance data** is in **blue**

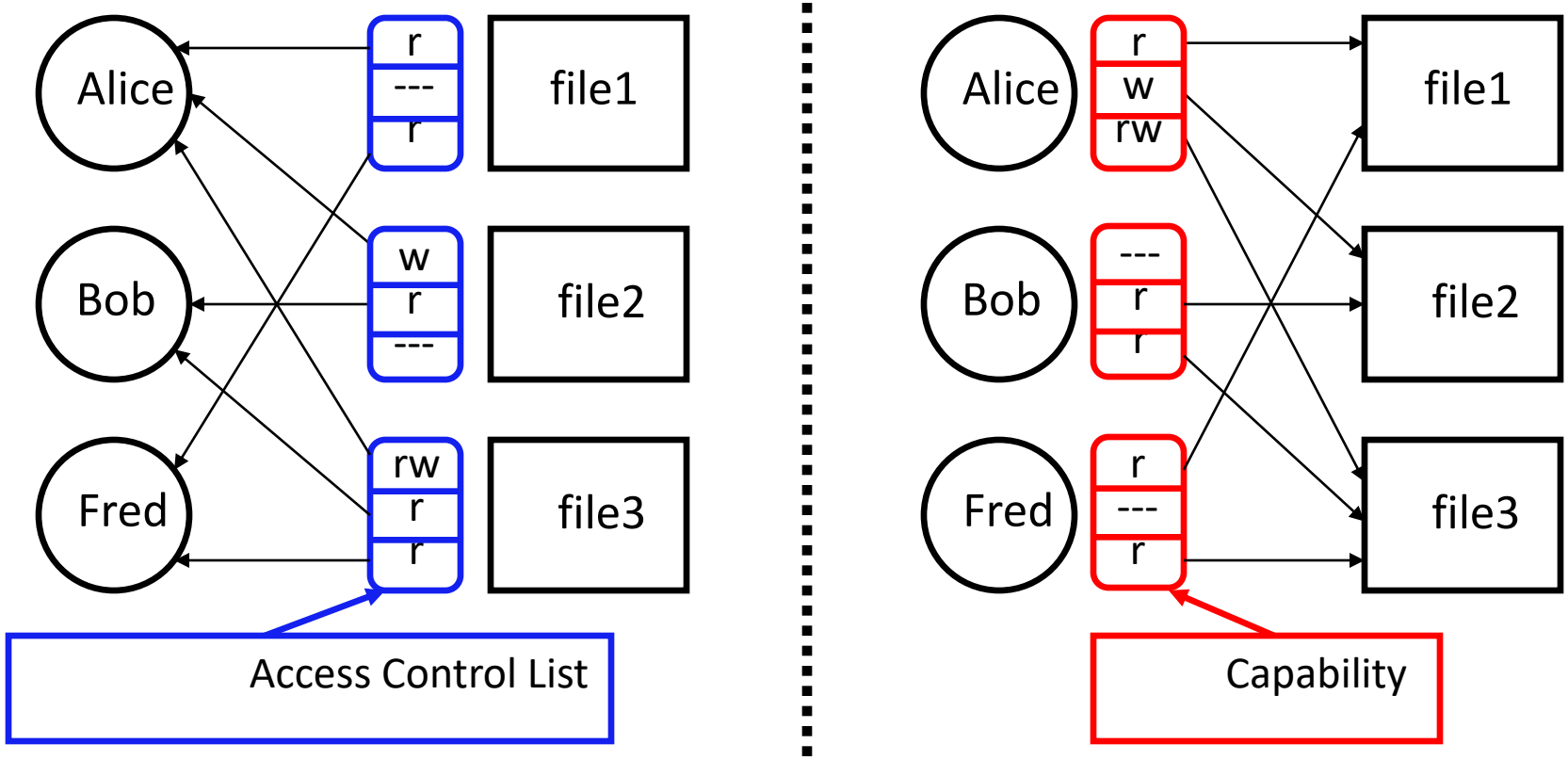
	OS	Accounting program	Accounting data	**Insurance data**	Payroll data
Bob	rx	rx	r	—	—
Alice	rx	rx	r	rw	rw
Sam	rwx	rwx	r	rw	rw
Accounting program	rx	rx	rw	rw	rw

Capabilities (or C-Lists)

- Store access control matrix by **row**
- Example: Capability for **Alice** is in **red**

	OS	Accounting program	Accounting data	Insurance data	Payroll data
Bob	rx	rx	r	—	—
Alice	rx	rx	r	rw	rw
Sam	rwx	rwx	r	rw	rw
Accounting program	rx	rx	rw	rw	rw

ACLs vs. Capabilities (1)



- Note that arrows point in opposite directions…
- with capabilities, the association between users and files is built into the system,
- while for an ACL-based system, a separate method for associating users to files is required

ACLs vs. Capabilities (2)

- Capabilities have several security advantages over ACLs and, for this reason.
- One potential security advantage of capabilities over ACLs is the so called ***confused deputy*** problem. To illustrate this problem, consider:

1. A system with two resources, a compiler and a file named BILL that contains critical billing information, and one user, Alice.
2. The compiler can write to any file.
3. Alice can invoke the compiler and she can provide a filename where debugging information will be written.
4. However, Alice is not allowed to write to the file BILL, since she might corrupt the billing information.

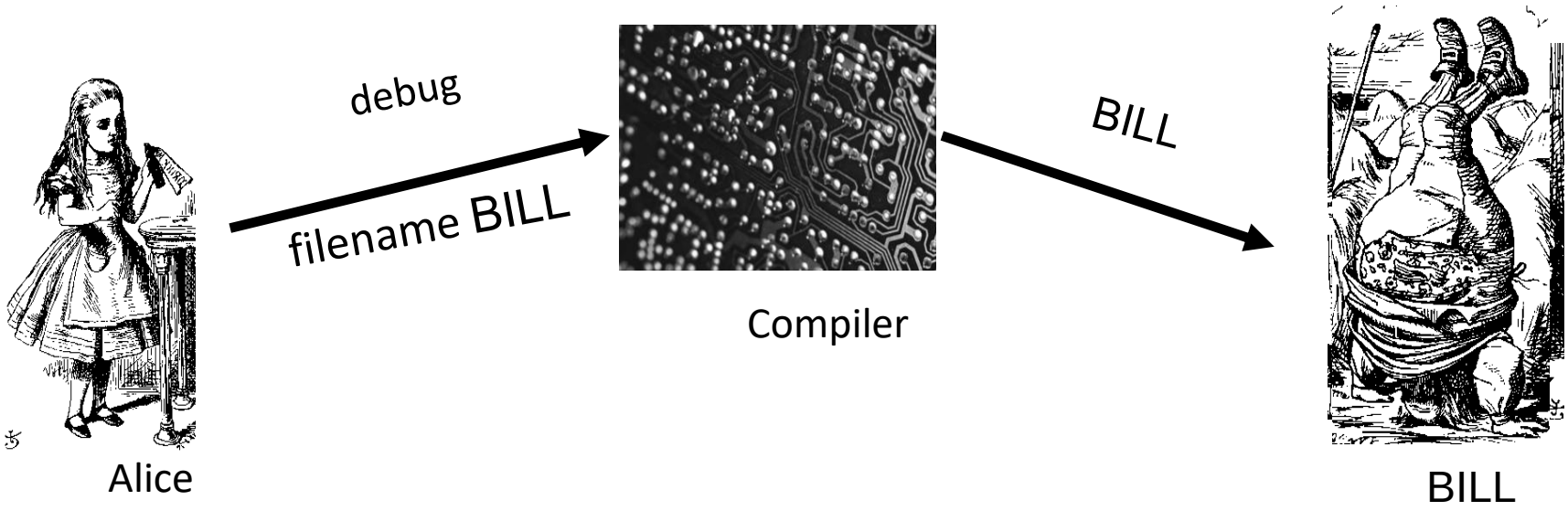
Confused Deputy

- Two resources: Compiler and BILL file (billing info)
- One user: Alica
- Compiler can write file BILL
- Alice can invoke compiler with a debug filename
- Alice not allowed to write to BILL

- Access control matrix

	Compiler	BILL
Alice	x	—
Compiler	rx	rw

ACL's and Confused Deputy



- Compiler is **deputy** acting on behalf of Alice
- Compiler is **confused:** Alice is not allowed to write BILL
- Compiler has confused its rights with Alice's