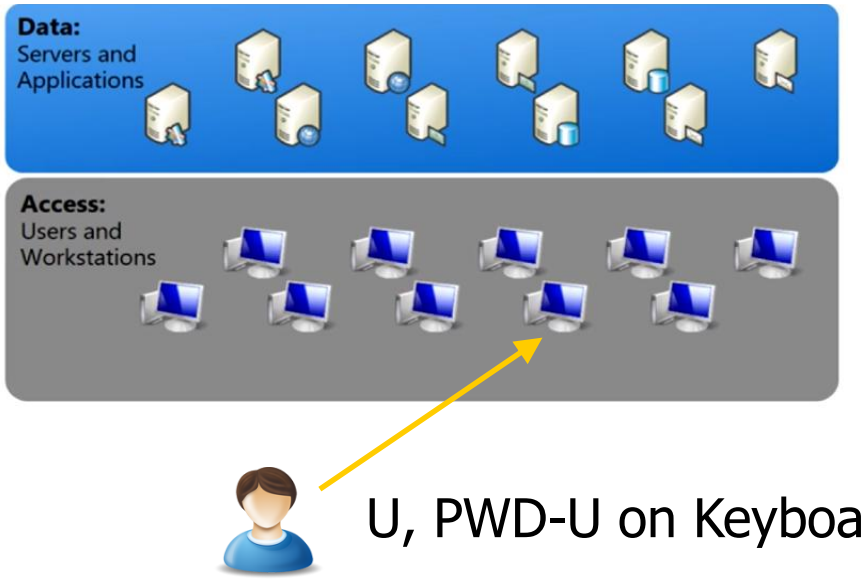
Interactive Logon vs Network Logon

Interactive Logon: Functionality

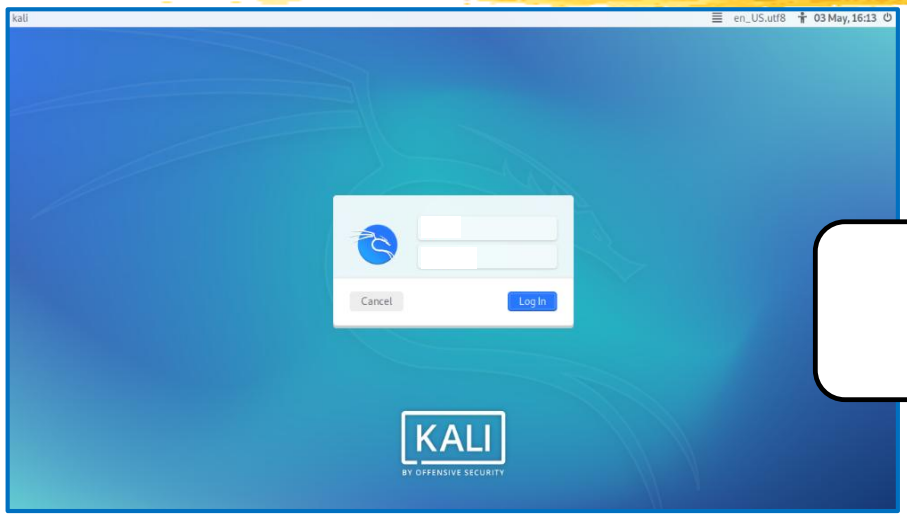


Interactive logon of account U

Workstation needs to **authenticate** U

U, PWD-U on Keyboard+Screen

Example



logon process

`root` / `SYSTEM`

1. Wait for credentials
2. Validate credentials (authenticate account AX)
3. Spawn GUI process that changes account to AX

Authentication DB: Local (I)

Account	Password
...	
Paolo	**pwd-Paolo**
...	

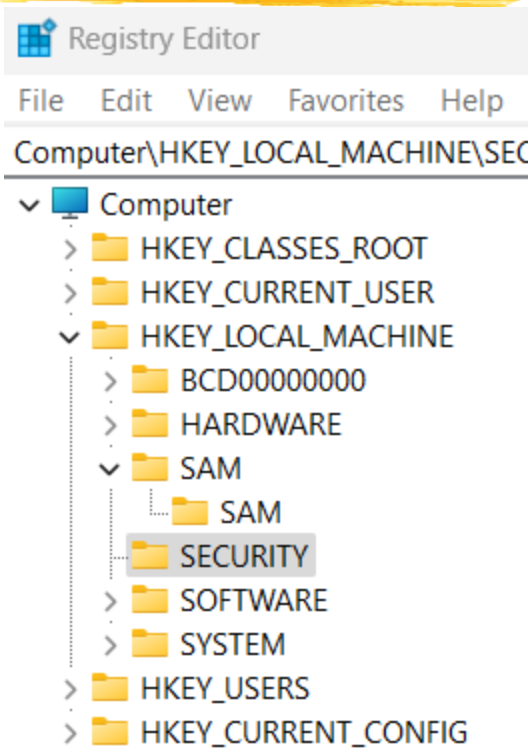
One row for each Account

AuthDB

- Impersonating an account requires proving knowledge of a certain **secret** (password)
- AuthDB usually managed by the **operating system** (a certain file, at a certain location)

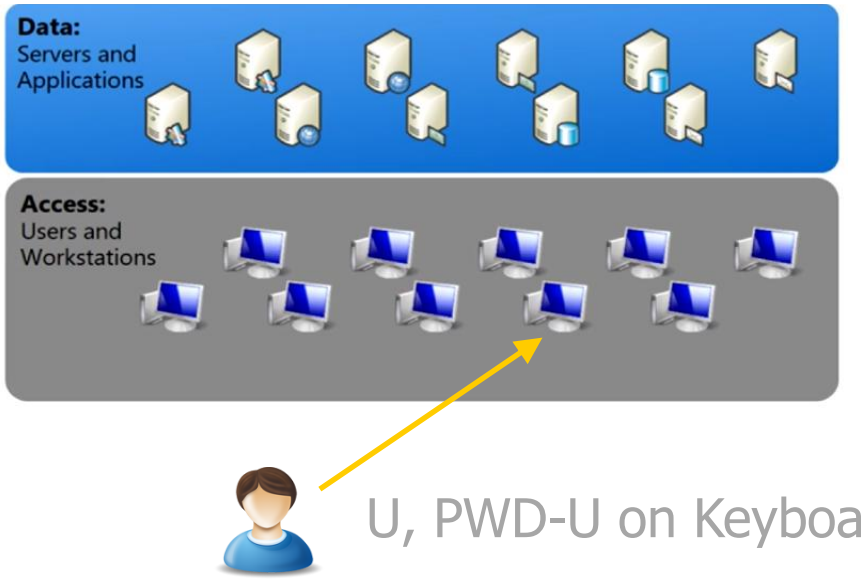
Authentication DB: Local (II)

- Windows
  - **SAM** (Security Account Manager)
  - Account definition + Passwords



- Linux
  - `/etc/passwd` Account definitions
  - `/etc/shadow` Passwords

Interactive Logon: Implementation outline



Interactive logon of account U

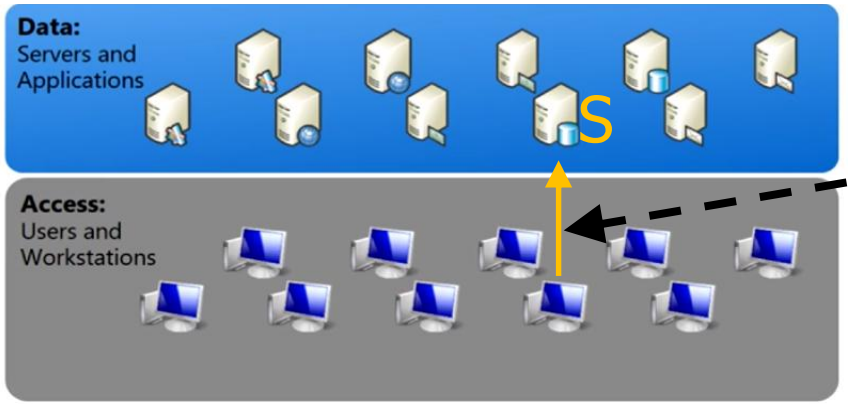
Workstation needs to **authenticate** U

Check credentials in AuthDB

U, PWD-U on Keyboard+Screen

Network Logon (I)

- Service S manages **local resources**
- Client connects to S through the **network**

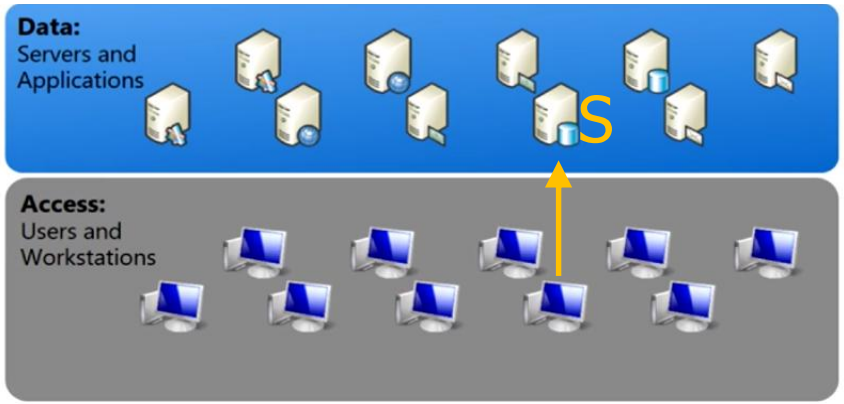


- Application protocol over TCP
  - Mail server
  - DB server
  - Web server
  - Shell server
  - Remote Desktop Server
  - ...

- NB: Client is a **process** (associated with an account)

Network Logon (II-a)

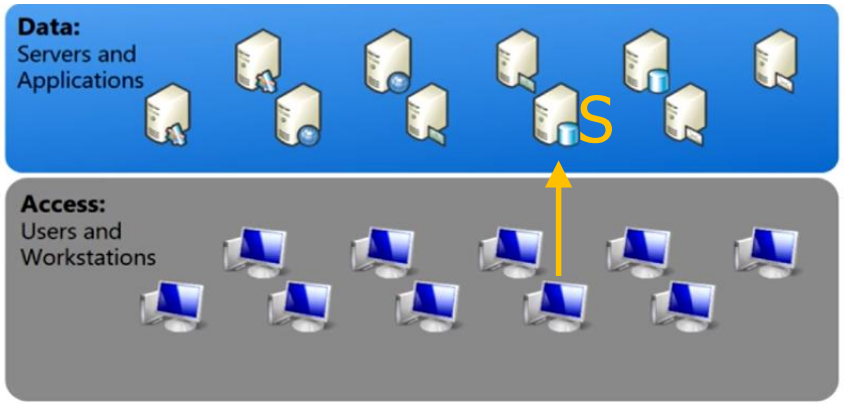
- Service S manages **local resources**
- Client connects to S through the **network**



- S needs to:
  1. **Authenticate** connecting account U
  2. Make sure U has **access rights** for requested resources

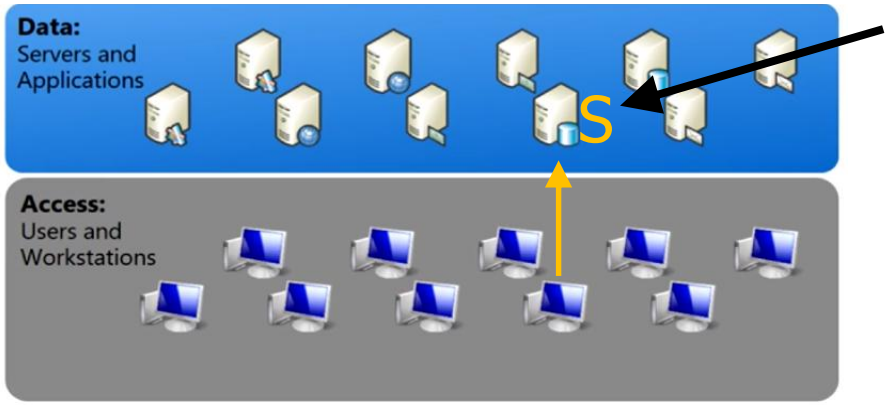
Network Logon (II-b)

- Service S manages **local resources**
- Client connects to S through the **network**



- S needs to:
  1. **Authenticate** connecting account U
     - Authentication protocol embedded in application protocol
     - Check credentials in AuthDB
  2. Make sure U has **access rights** for requested resources
     - ACL of resources are local

Network Logon: AuthDB



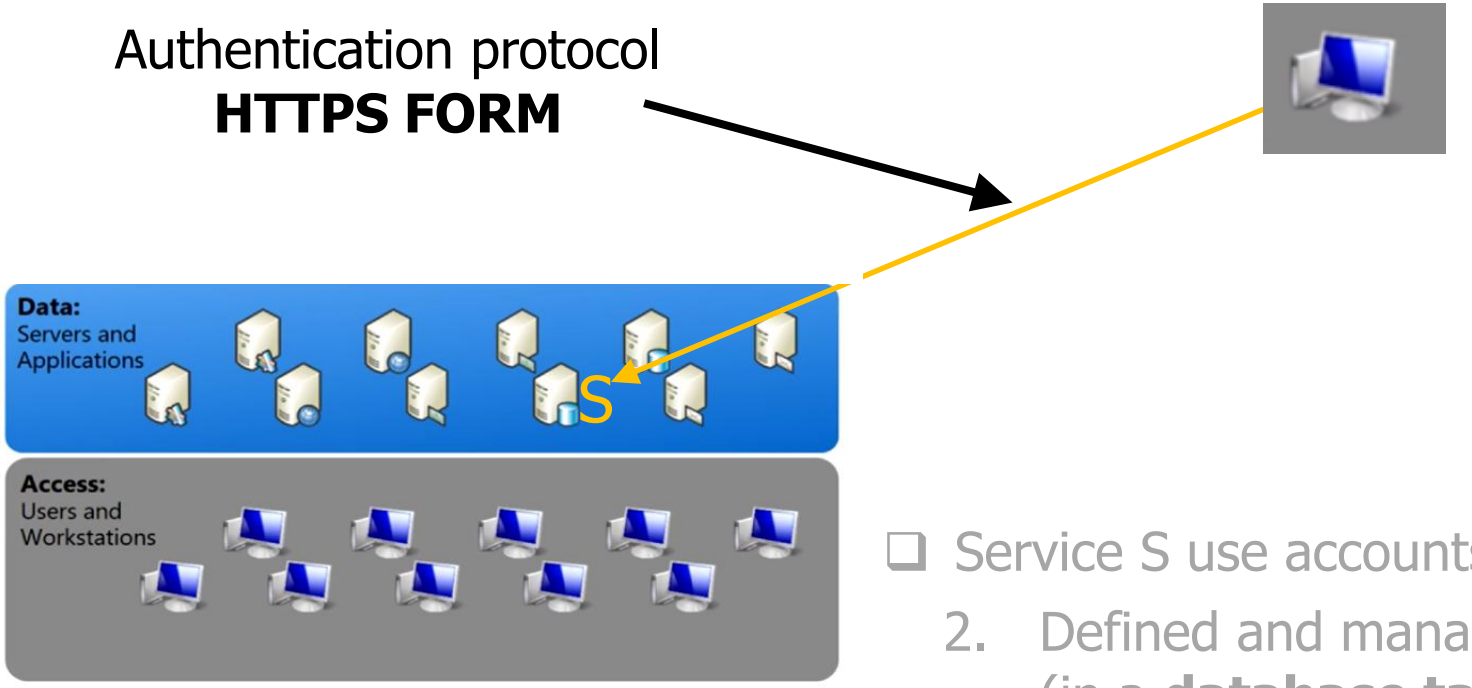
- Service S may be structured to use accounts:
  1. Local O.S.

  or

  2. Defined and managed by S itself (usually in a **database table**)

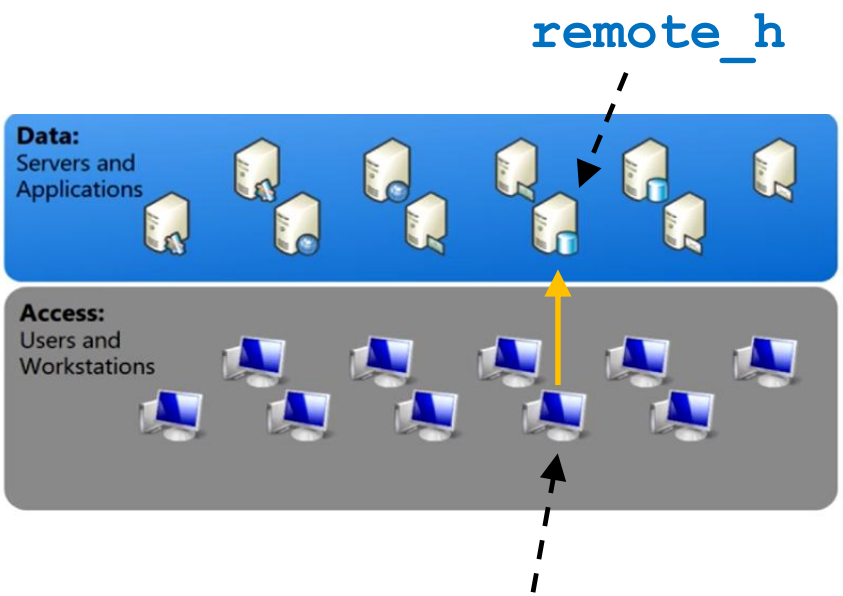
Example: Web Server

Authentication protocol
HTTPS FORM



- Service S use accounts:
  2. Defined and managed by S itself (in a **database table**)

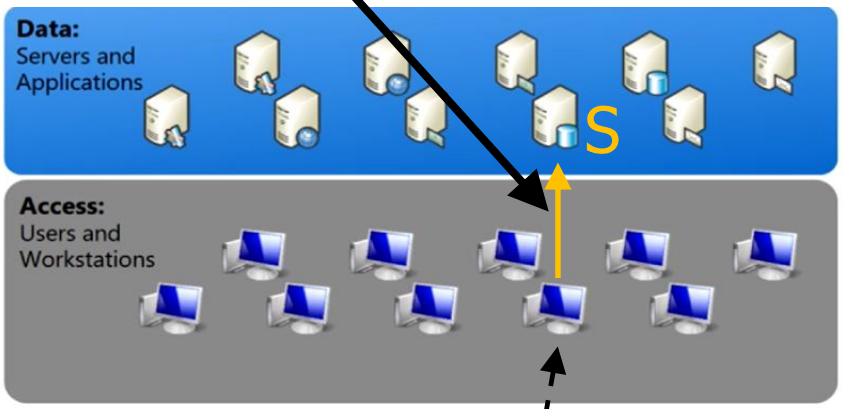
Example: Windows Services (I)



- **net use** G: **remote_h**\remote_f /user:user_x pass_x
  - Mount remote folder exported by **remote_h**

Example: Windows Services (II)

Authentication protocol
NTLM



- Service S may be structured to use accounts:
  1. Local O.S.

Valid on **remote_h**

- `net use G: \\remote_h\remote_f /user:user_x pass_x`
  - SMB protocol (port 445)

Windows Example (I) (Just to have an idea)

- **Mount network printer** `printer_name` from `print_srv`
  - Credentials `u_x` / `p_x` (on `remote_h`)

- `net use LPT1: \\print_srv\printer_name /user:u_x p_x`
  - Protocol SMB (port 445)

- MANY different ways for doing the same thing...

Windows Example (II-a) (Just to have an idea)

- **Execute command** `YourCmd` on `remote_h`
  - Credentials `user_x` / `pwd_x` (on `remote_h`)

- **`psexec`** `\\remote_h -u user_x -p pwd_x cmd /c "YourCmd"`
  - Protocol SMB (port 445)

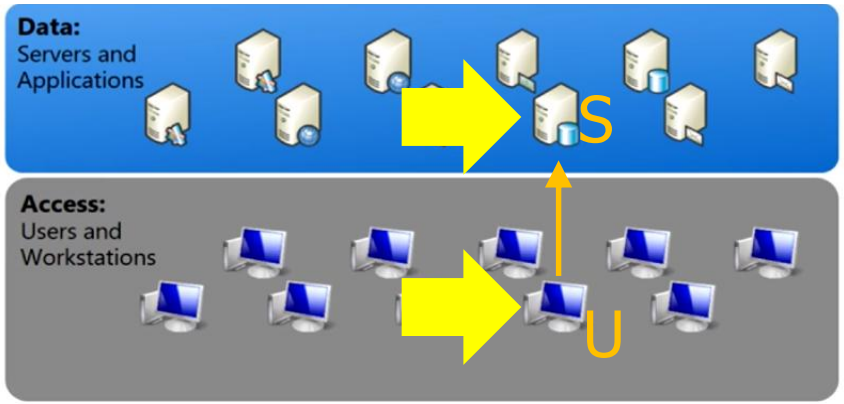
- MANY different ways for doing the same thing…

Windows Example (II-b) (Just to have an idea)

- **Execute command** `YourCmd` on `remote_h`
  - Credentials `user_x` / `pwd_x` (on `remote_h`)
- Powershell script
-
```
$remoteComputer = "remote_h"
$credential = New-Object
-TypeName System.Management.Automation.PSCredential
-ArgumentList @("user_x",
                (ConvertTo-SecureString -String "pass_x"
                -AsPlainText -Force)
                )
Invoke-Command -ComputerName $remoteComputer
                -Credential $credential
                -ScriptBlock { cmd /c "YourCmd" }
```

- Protocol WinRM over HTTPS (port 5986)

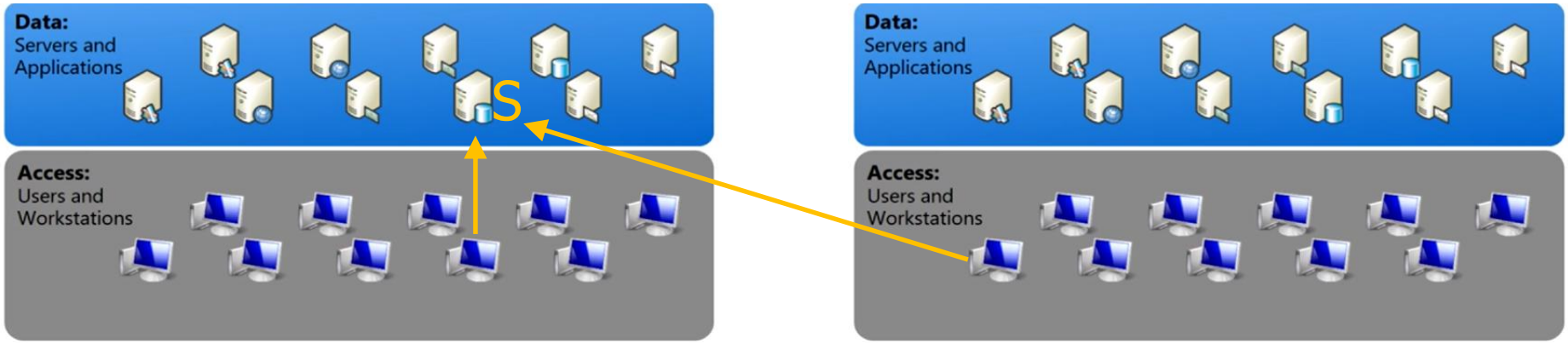
Network Logon: Keep in mind



- AuthDB **nothing** to do with each other
- Client account **nothing** to do with Server account
- Names might be identical but the accounts are in **different** machines
- ACL on Client machine **nothing** to do with ACL on Server machine

Remark

- Client machine and Server machine may be either in the **same** or in **different** organizations
- Our focus: **same** organization



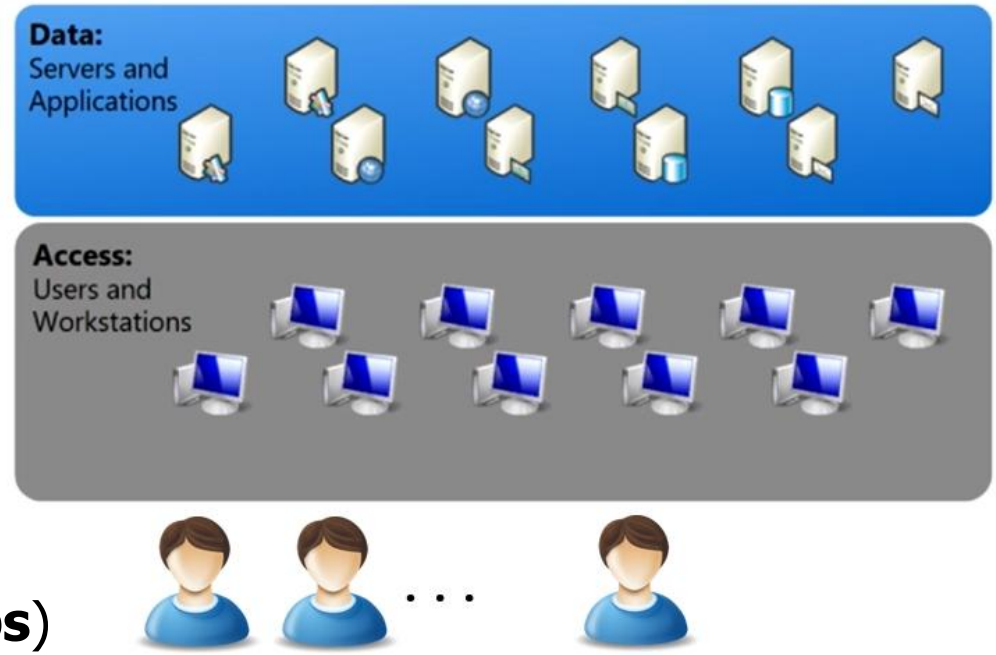
Access Control in Organizations

Large Organizations (I-a)

Tens/Hundreds of **Servers**
(storing **Files**, **Databases**)

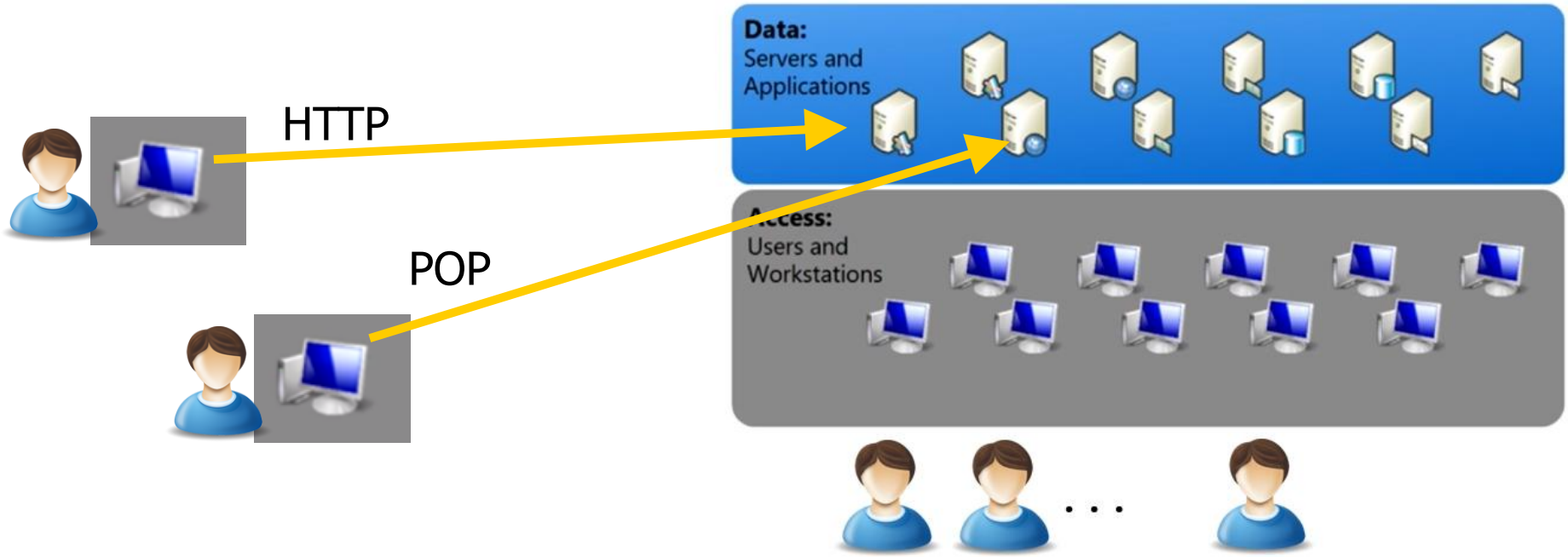
Thousands of
Workstations / Notebooks
(either private or shared)

Thousands of **Accounts**
(**tens** of partially overlapping **Groups**)



Large Organizations (I-b)

Some Servers may be accessed from the **outside**



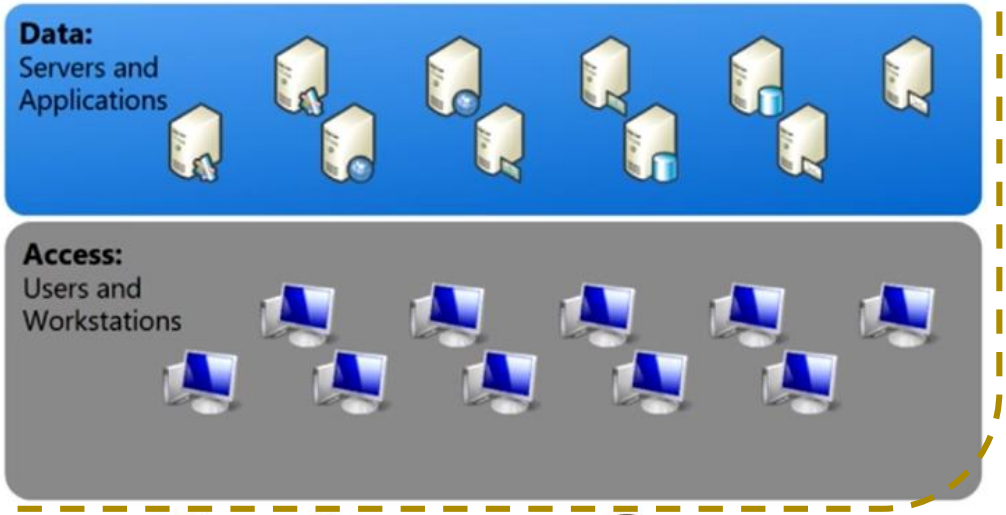
Large Organizations (II)

Resources

Routers, Firewalls, Switches, Networks,...

Servers
(storing **Files**, **Databases**)

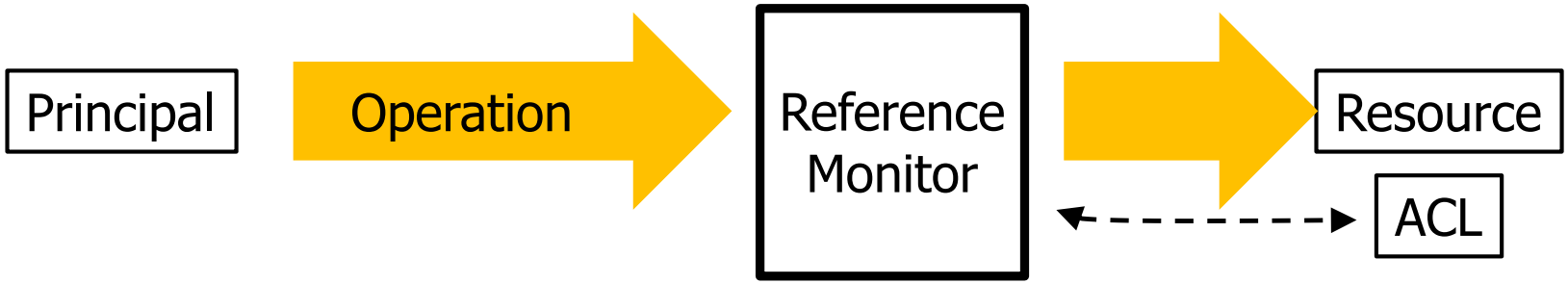
Workstations / Notebooks
(either private or shared)



Accounts
(partially overlapping **Groups**)

Identities

Access Control



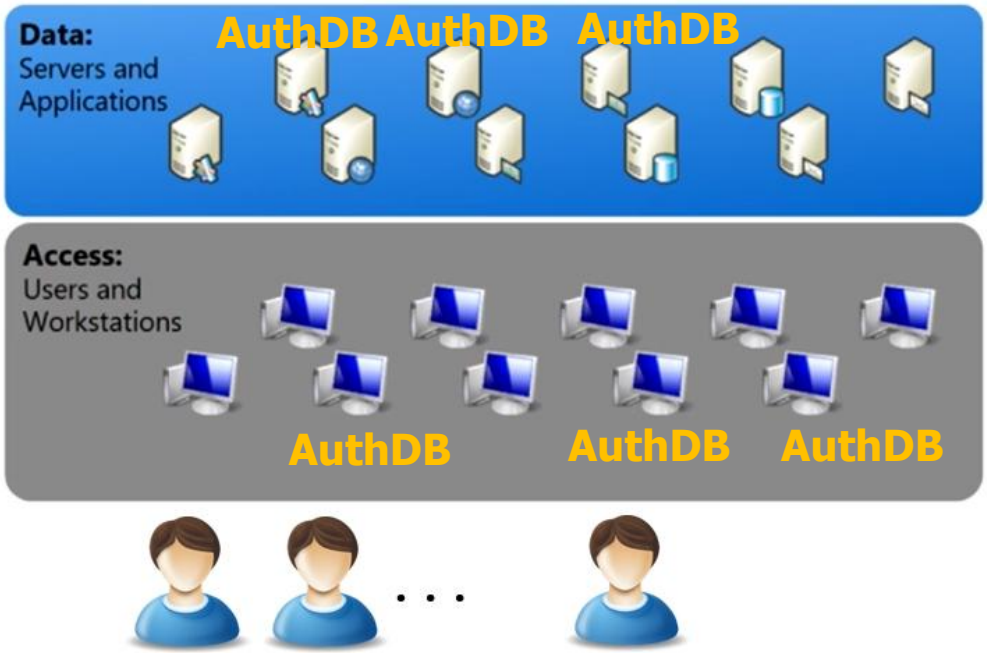
- **Every resource access must follow this framework**
  - Application level (e.g., access to a remote server)
  - O.S. level (e.g., shell / GUI)

- Pre-requisite: Authentication

Authentication: Key practical requirement

We do **not** want a **separate** AuthDB on **each** Reference Monitor

(identity management would be a nightmare)



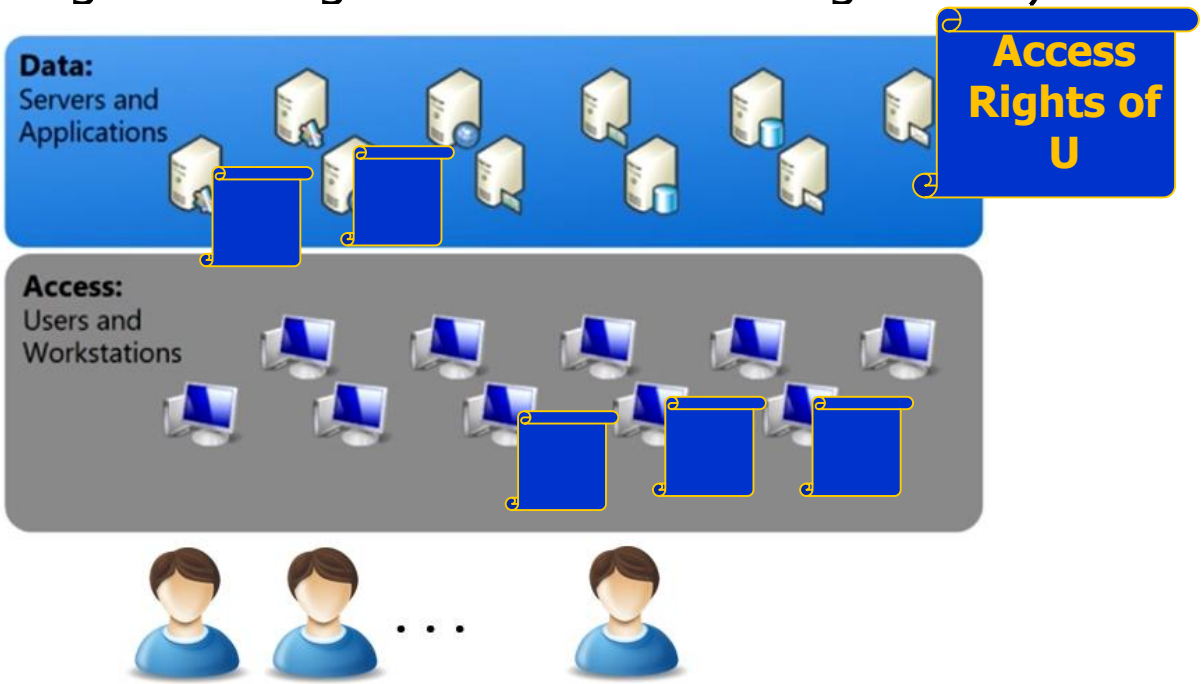
Key Practical Problems

- Can account U modify file F?
- Can account U read database D?
- Can account U logon on computer C?
- …
- Can account U at computer C access server S?
- …
- Can computer C connect to network N?
- Can computer C access server S?
- …

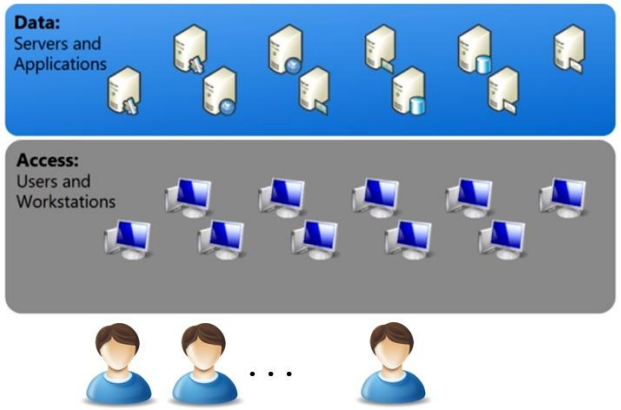
Authorization: Key practical requirement

We do **not** want to specify ACLs **separately** on **each** resource

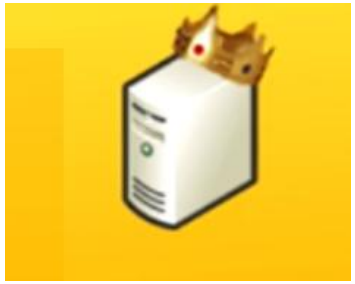
(access rights management would be a nightmare)



Directory Service



DIRECTORY SERVICE

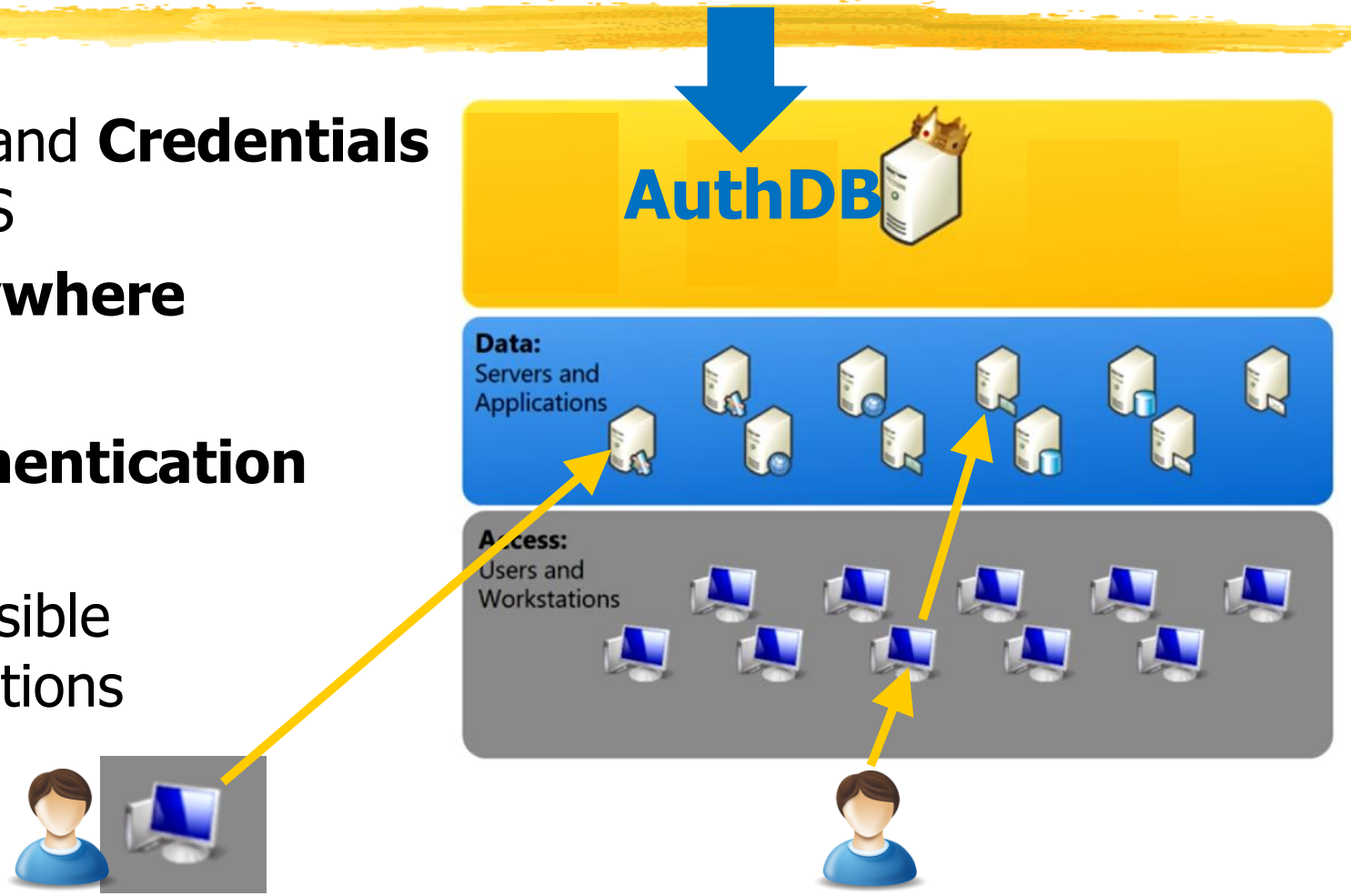


- **Centralized** repository (**Directory Service**) describes:
  - All **accounts** (including their **credentials**)
  - All **resources**
  - All **access rights** of accounts to resources (ACLs)

Single Sign On (SSO)

- **Accounts** and **Credentials** stored in DS
- Valid **everywhere**

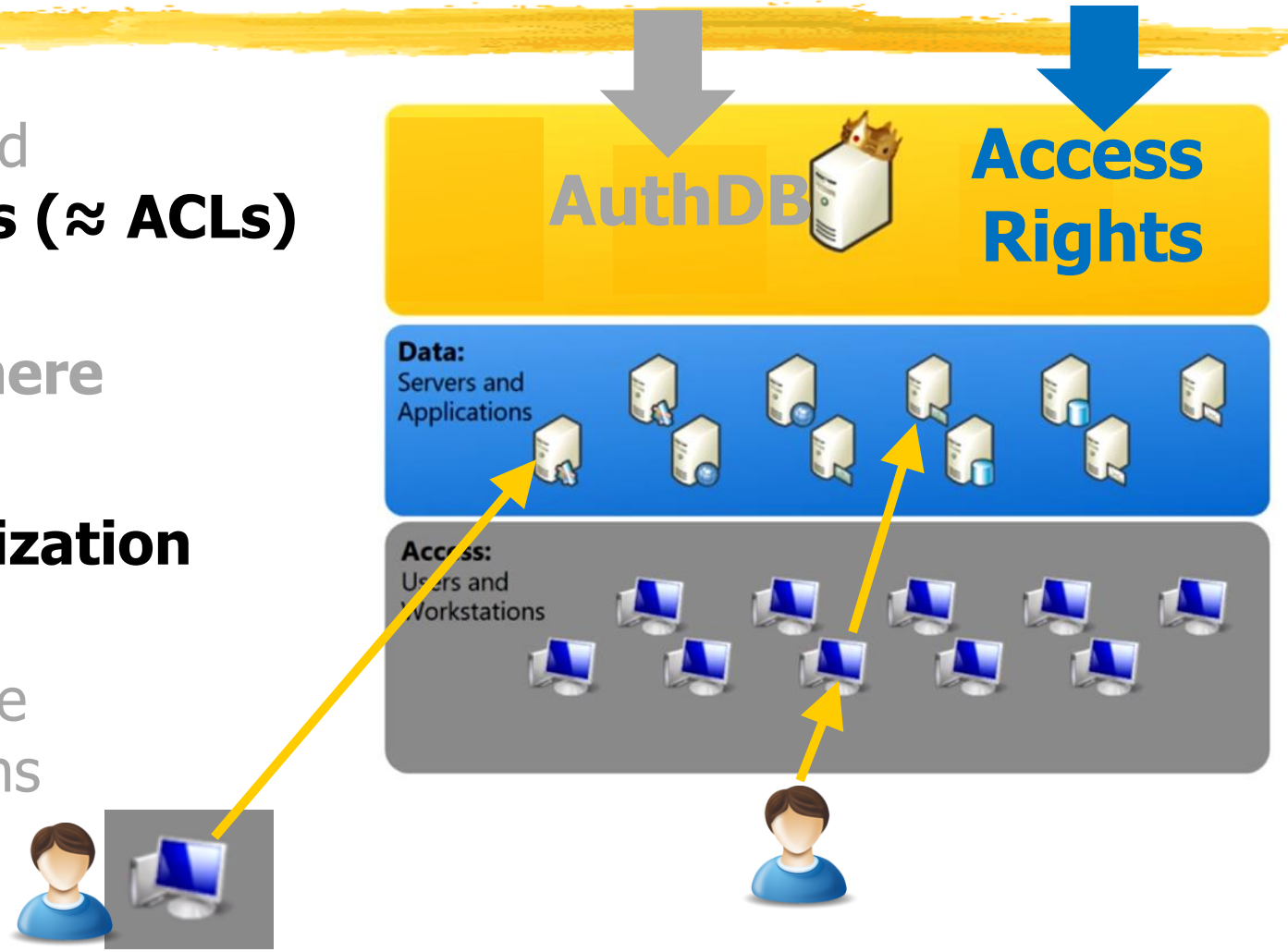
- **Every authentication** involves DS
- Several possible implementations



SSO + Centralized Authorization

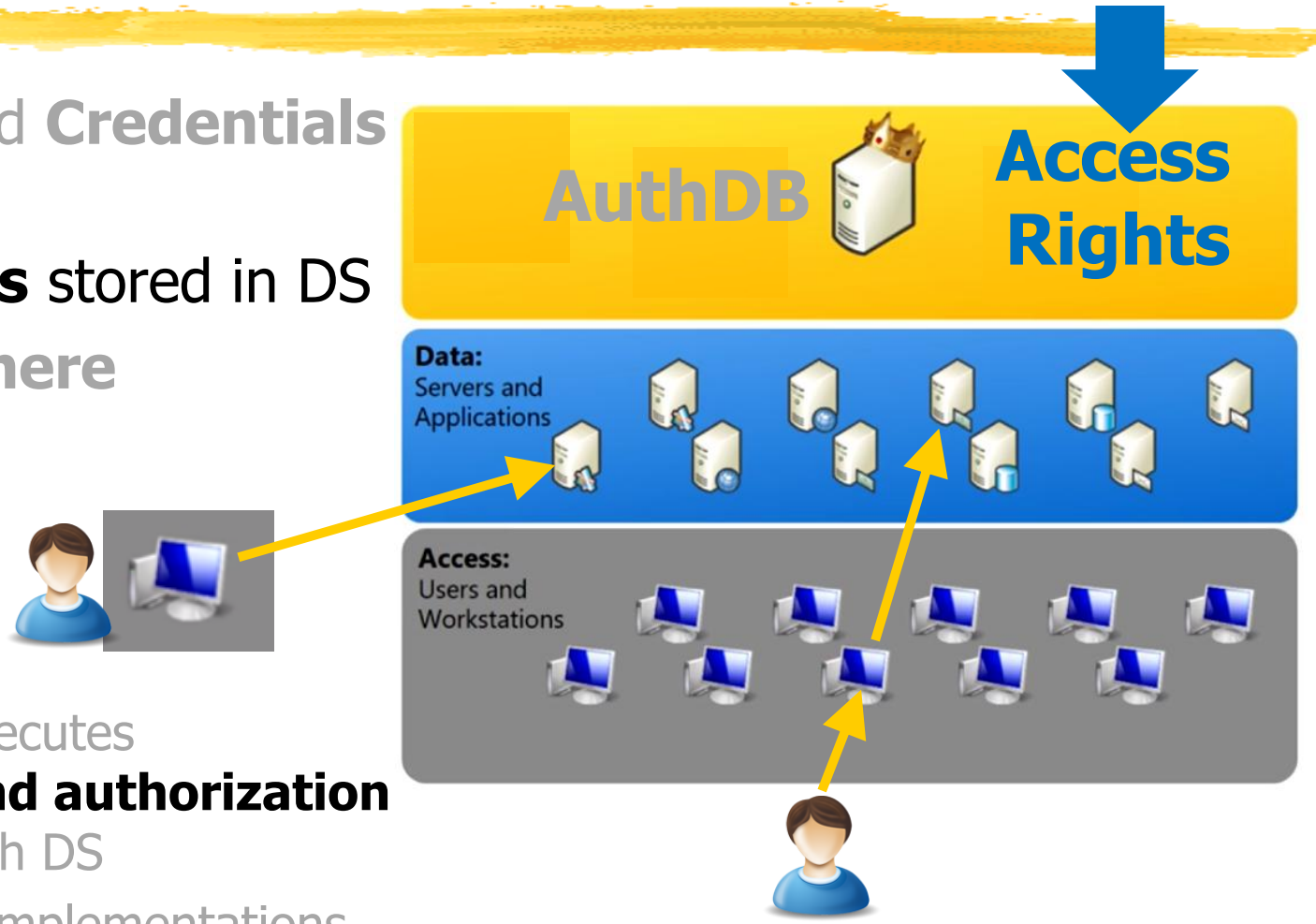
- **Resources** and **Access Rights (≈ ACLs)** stored in DS
- Valid **everywhere**

- **Every authorization** involves DS
- Several possible implementations



SSO + Centralized Authorization

- **Identities** and **Credentials** stored in DS
- **Access Rights** stored in DS
- Valid **everywhere**



- Each resource executes authentication **and authorization** by interacting with DS
- Several possible implementations

Identity and Access Management (IAM)

- **Procedures** and **technologies** for management of individual **identities**, their **authentication**, **authorization**, and **access rights**
- **within** or **across** **enterprise** boundaries

Our focus

- Our focus is **within** enterprise boundaries
  - Account and resource in the same organization
- Widely prevalent technology:
  - **Windows Active Directory**
  - **Domain** ≈ All IT entities in an organization
  - **Domain Controller** ≈ Directory Service

- Technologies **across** enterprise boundaries
  - OAuth, SAML (SPID)
  - Kerberos realms

Keep in mind

- Our focus is **within** enterprise boundaries
  - Account and resource in the same organization
- Widely prevalent technology:
  - **Windows Active Directory**
  - **Domain** ≈ All IT entities in an organization
  - **Domain Controller** ≈ Directory Service

- Technologies **across** enterprise boundaries
  - OAuth, SAML (SPID)
  - Kerberos realms

Our learning path

- Every authentication and every authorization involves DS
- Several possible implementations

- …
- Windows Active Directory (outline)
- Hacking lab / …
- Passwords
- Windows AD implementation (outline)
  - NTLM
  - Kerberos

Windows Active Directory (Outline)

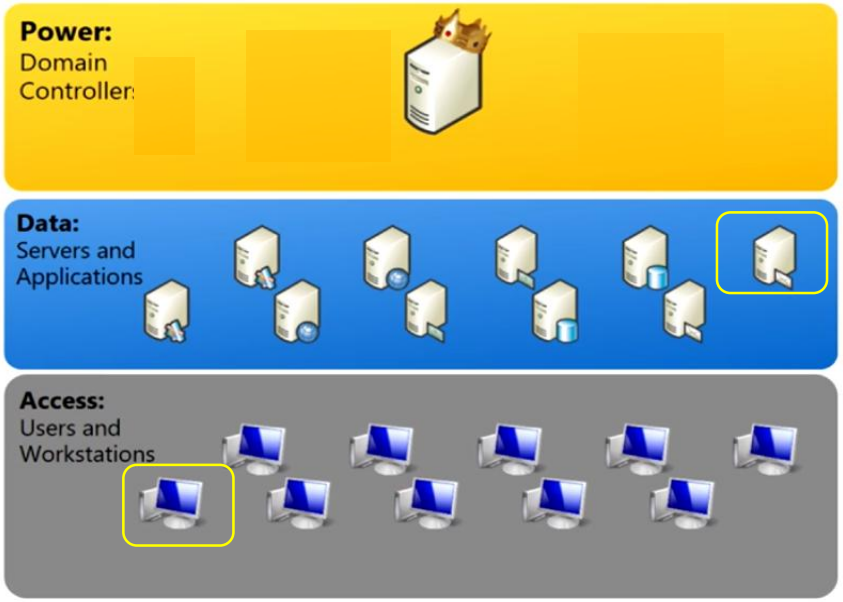
Windows Active Directory (Outline)

Local Accounts vs Domain Accounts (I)

- **Local** accounts:
  - Defined in **SAM Database** (Security Accounts Manager)
  - Authorization data (ACL) for local accounts stored on the machine

- **Domain** accounts:
  - Defined in **Domain Controller** (`c:\Windows\NTDS\NTDS.dit`)
  - Authorization data (ACL) for domain accounts stored on the Domain Controller

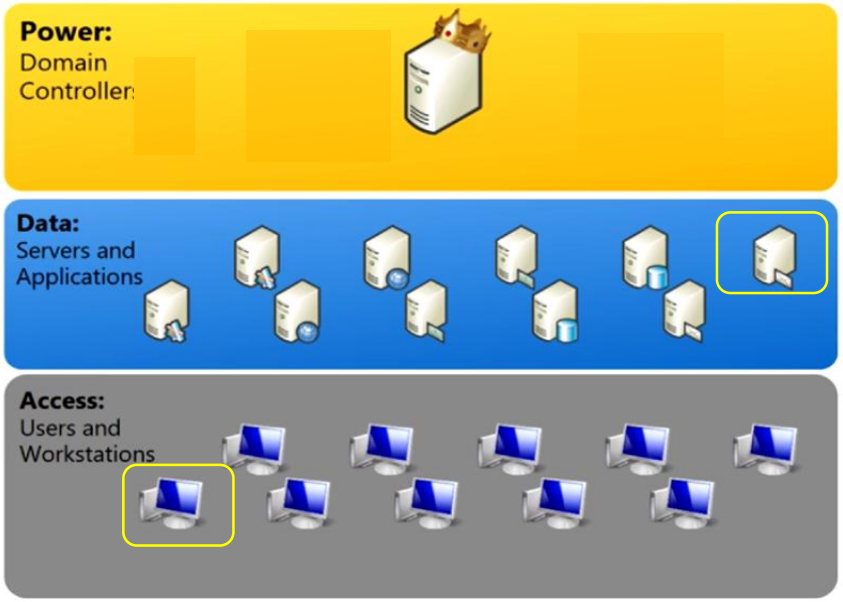
Local Accounts vs Domain Accounts (II)



- Network access to service S:
  - All **local** accounts of this computer
  - All **domain** accounts (with access rights on S)

- Logon allowed to:
  - All **local** accounts of this computer
  - All **domain** accounts (with logon access rights on this computer)

Hhmmm…



- Network access to service S:
  - All **local** accounts of this computer
  - All **domain** accounts (with access rights on S)

*How can you tell whether a given U is a **local** account or a **domain** account?*

- Logon allowed to:
  - All **local** accounts of this computer
  - All **domain** accounts (with logon access rights on this computer)

Local Accounts vs Domain Accounts (III)

- **Local** accounts:
  - Defined in **SAM Database** (Security Accounts Manager)
- **Domain** accounts:
  - Defined in **Domain Controller** (`c:\Windows\NTDS\NTDS.dit`)

- Names of **Local** accounts:
  - `WORKGROUP\`*`name`* (or just *`name`*)
  - `NT AUTHORITY\SYSTEM`
- Names of **Domain** accounts
  - `domain_name\`*`name`*

Keep in mind

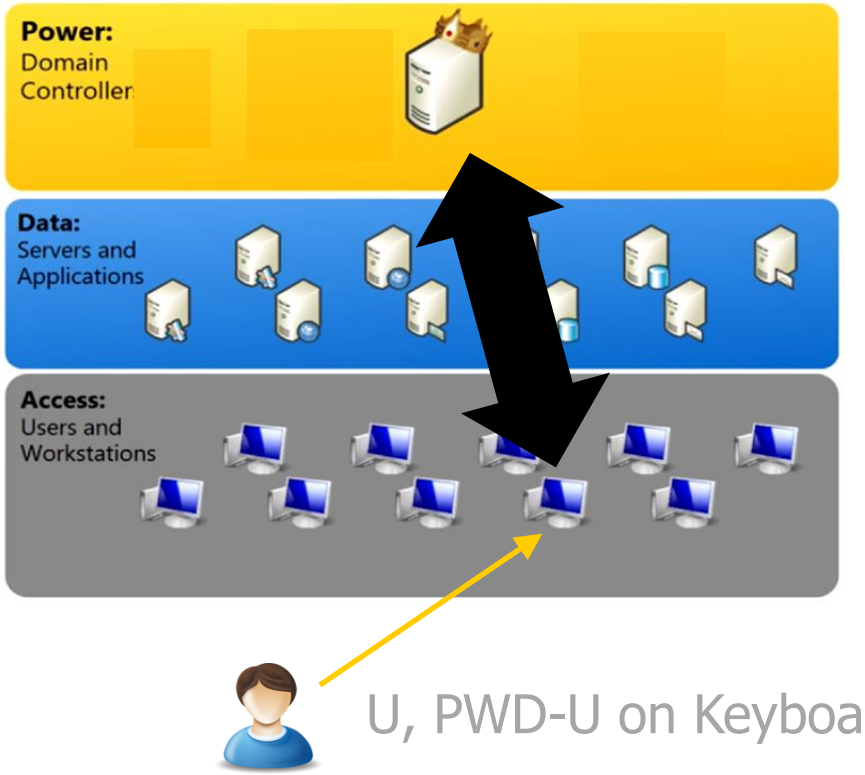
- **Local** accounts:
  - Valid **only** on that machine
  - Credentials (SAM) and Authorization data (ACL) stored **in each machine**

- **Domain** accounts:
  - Valid "**everywhere**"
  - Credentials (NTDS) and Authorization data (ACL) stored **in Domain Controller**

Remark

- From now on, **only domain accounts**
- Unless stated otherwise

Interactive Logon:
Implementation DOMAIN account



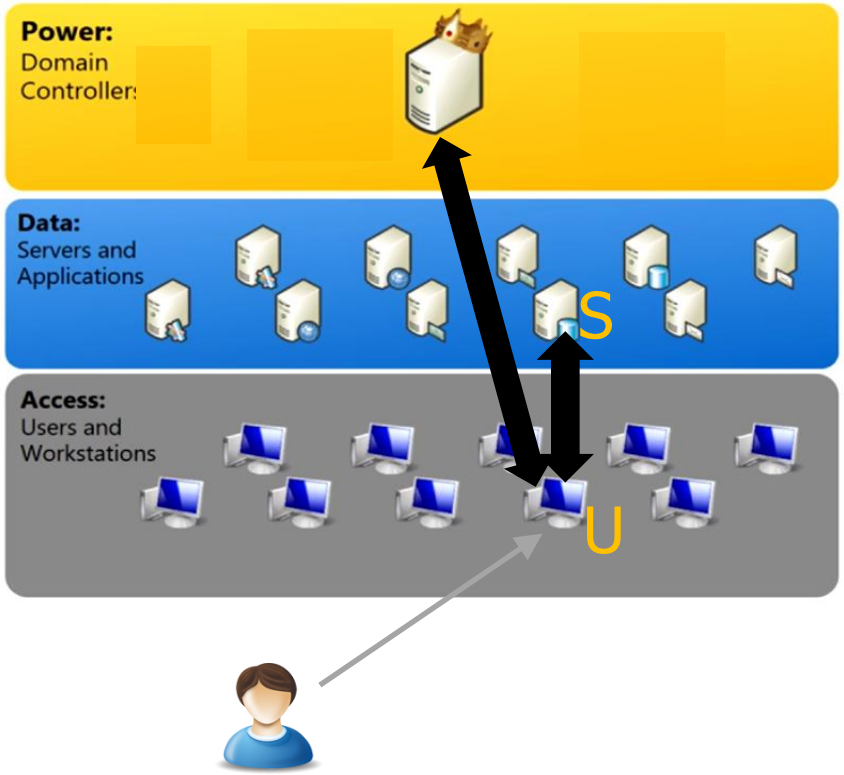
U, PWD-U on Keyboard+Screen

Workstation needs to:

1. **Authenticate** U
2. Make sure U has **access rights** for interactive logon on that workstation

Usually: **Kerberos**

Network Logon:
Implementation DOMAIN account



Domain account U is in interactive logon

Network logon on service S

S needs to:

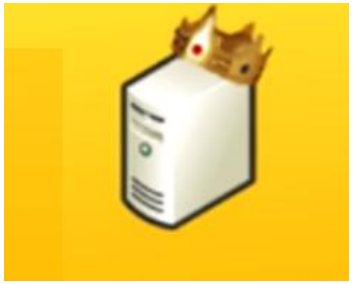
1. **Authenticate** U
2. Make sure U has **access rights** for network logon on S

Usually: **Kerberos**

LDAP: Double Meaning

- **L**ightweight **D**irectory **Access Protocol**

DIRECTORY SERVICE



- A **standard** for **naming** and **describing** IT objects:
  - **Every object** has a **name** and a set of **attributes**
  - ACLs describe who can read/write what

- A **protocol** for interacting with a Directory Service (server that stores those descriptions)

Example (just to have an idea) (I-a)

User account

```
DistinguishedName: CN=John.Doe,OU=Marketing,OU=Departments,DC=corp,DC=example,DC=com
sAMAccountName: John.Doe
UserPrincipalName: John.Doe@corp.example.com
DisplayName: John Doe
GivenName: John
Surname: Doe
Description: Marketing Specialist
Department: Marketing
Title: Marketing Specialist
Mail: John.Doe@corp.example.com
TelephoneNumber: +1 555-1234
MemberOf: CN=Marketing_Team,OU=Marketing,OU=Departments,DC=corp,DC=example,DC=com
AccountExpires: Never
PasswordLastSet: 2025-03-01
LogonCount: 112
LastLogonTimestamp: 2025-03-10
```

Domain `corp.example.com`

Example
(just to have an idea) (I-b)

My description at UniTS
>60 attributes

...		
accountExpires	Integer8	1 0x0
cn	DirectoryString	1 BARTOLI ALBERTO [5943]
lastLogonTimestamp	Integer8	1 2/10/2023 13:22
mail	DirectoryString	1 bartoli.alberto@units.it
mAPIRecipient	Boolean	1 FALSE
name	DirectoryString	1 BARTOLI ALBERTO [5943]
...		

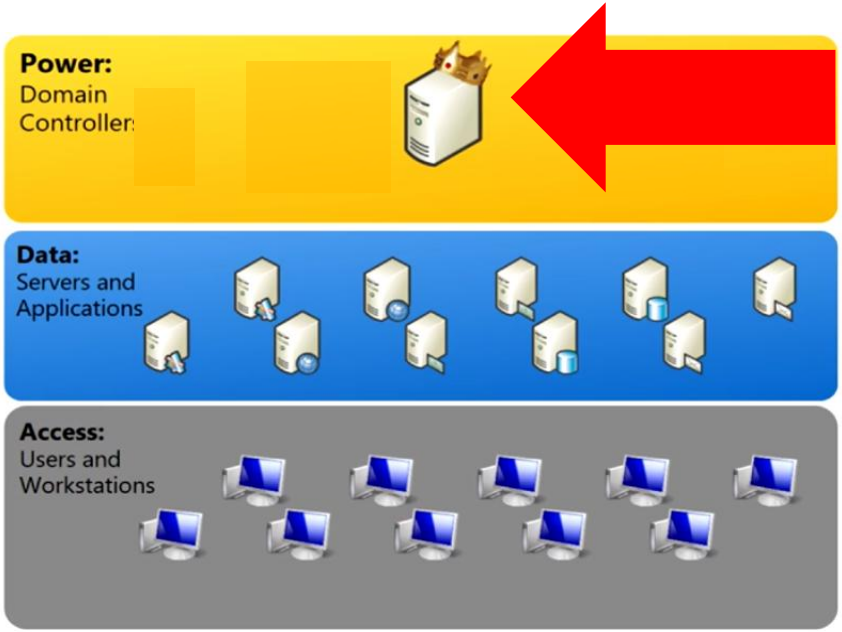
Example (just to have an idea) (II)

Service account

```
DistinguishedName: CN=SQLService,OU=Services,DC=corp,DC=example,DC=com
sAMAccountName: SQLService
UserPrincipalName: SQLService@corp.example.com
ServicePrincipalName: MSSQLSvc/SQL-SERVER-02.corp.example.com:1433
Description: Service account for SQL Server authentication
MemberOf: CN=SQL_Admins,OU=Admins,DC=corp,DC=example,DC=com
AccountExpires: Never
PasswordLastSet: 2025-03-01
LogonCount: 3402
LastLogonTimestamp: 2025-03-10
AccountType: User (Service)
```

Domain `corp.example.com`

Keep in mind



- Describes **everything**:
  - Accounts
  - **Credentials**
  - **Access rights**
  - …

- Reading Credentials…
- Modifying ACLs…