Keeping Secrets?

- Suppose we have secret data that only certain users should access
- Is DAC enough to prevent leaks?

```
charlie@DESKTOP:~$ groups
charlie topsecret
```

Keeping Secrets?

- Suppose we have secret data that only certain users should access
- Is DAC enough to prevent leaks?

```
charlie@DESKTOP:~$ groups
charlie topsecret
charlie@DESKTOP:~$ ls –la /top-secret-intel/
drwxr-xr-x 0 root root      512 Jan  8 14:55 .
drwxr-xr-x 0 root root      512 Oct 11 19:58 ..
-rw-r----- 1 root topsecret 896 Jan 29 22:47 northkorea.pdf
```

Keeping Secrets?

- Suppose we have secret data that only certain users should access
- Is DAC enough to prevent leaks?

```
charlie@DESKTOP:~$ groups
charlie topsecret
charlie@DESKTOP:~$ ls –la /top-secret-intel/
drwxr-xr-x 0 root root        512 Jan  8 14:55 .
drwxr-xr-x 0 root root        512 Oct 11 19:58 ..
-rw-r----- 1 root topsecret  896 Jan 29 22:47 northkorea.pdf
```

Keeping Secrets?

- Suppose we have secret data that only certain users should access
- Is DAC enough to prevent leaks?

```
charlie@DESKTOP:~$ groups
charlie topsecret
charlie@DESKTOP:~$ ls —la /top-secret-intel/
drwxr-xr-x 0 root root      512 Jan  8 14:55 .
drwxr-xr-x 0 root root      512 Oct 11 19:58 ..
-rw-r----- 1 root topsecret 896 Jan 29 22:47 northkorea.pdf
charlie@DESKTOP:~$ groups mallory
mallory secret
```

Keeping Secrets?

- Suppose we have secret data that only certain users should access
- Is DAC enough to prevent leaks?

```
charlie@DESKTOP:~$ groups
charlie topsecret
charlie@DESKTOP:~$ ls –la /top-secret-intel/
drwxr-xr-x 0 root root      512 Jan  8 14:55 .
drwxr-xr-x 0 root root      512 Oct 11 19:58 ..
-rw-r----- 1 root topsecret 896 Jan 29 22:47 northkorea.pdf
charlie@DESKTOP:~$ groups mallory
mallory secret
```

Keeping Secrets?

- Suppose we have secret data that only certain users should access
- Is DAC enough to prevent leaks?

```
charlie@DESKTOP:~$ groups
charlie topsecret
charlie@DESKTOP:~$ ls –la /top-secret-intel/
drwxr-xr-x 0 root root      512 Jan  8 14:55 .
drwxr-xr-x 0 root root      512 Oct 11 19:58 ..
-rw-r----- 1 root topsecret 896 Jan 29 22:47 northkorea.pdf
charlie@DESKTOP:~$ groups mallory
mallory secret
charlie@DESKTOP:~$ ls –la /home/mallory
drwxrwxrwx 0 mallory mallory   512 Jan  8 14:55 .
drwxr-xr-x 0 root    root      512 Oct 11 19:58 ..
```

Keeping Secrets?

- Suppose we have secret data that only certain users should access
- Is DAC enough to prevent leaks?

```
charlie@DESKTOP:~$ groups
charlie topsecret
charlie@DESKTOP:~$ ls –la /top-secret-intel/
drwxr-xr-x 0 root root      512 Jan  8 14:55 .
drwxr-xr-x 0 root root      512 Oct 11 19:58 ..
-rw-r----- 1 root topsecret 896 Jan 29 22:47 northkorea.pdf
charlie@DESKTOP:~$ groups mallory
mallory secret
charlie@DESKTOP:~$ ls –la /home/mallory
drwxrwxrwx 0 mallory mallory   512 Jan  8 14:55 .
drwxr-xr-x 0 root    root      512 Oct 11 19:58 ..
```

Keeping Secrets?

- Suppose we have secret data that only certain users should access
- Is DAC enough to prevent leaks?

```
charlie@DESKTOP:~$ groups
charlie topsecret
charlie@DESKTOP:~$ ls –la /top-secret-intel/
drwxr-xr-x 0 root root      512 Jan  8 14:55 .
drwxr-xr-x 0 root root      512 Oct 11 19:58 ..
-rw-r----- 1 root topsecret 896 Jan 29 22:47 northkorea.pdf
charlie@DESKTOP:~$ groups mallory
mallory secret
charlie@DESKTOP:~$ ls –la /home/mallory
drwxrwxrwx 0 mallory mallory   512 Jan  8 14:55 .
drwxr-xr-x 0 root    root      512 Oct 11 19:58 ..
charlie@DESKTOP:~$ cp /top-secret-intel/northkorea.pdf /home/mallory
charlie@DESKTOP:~$ ls –l /home/mallory
-rw-r----- 1 charlie charlie 896 Jan 29 22:47 northkorea.pdf
charlie@DESKTOP:~$ chmod ugo+rw /home/mallory/northkorea.pdf
```

Keeping Secrets?

- Suppose we have secret data that only certain users should access
- Is DAC enough to prevent leaks?

```
charlie@DESKTOP:~$ groups
charlie topsecret
charlie@DESKTOP:~$ ls –la /top-secret-intel/
drwxr-xr-x 0 root root      512 Jan  8 14:55 .
drwxr-xr-x 0 root root      512 Oct 11 19:58 ..
-rw-r----- 1 root topsecret 896 Jan 29 22:47 northkorea.pdf
charlie@DESKTOP:~$ groups mallory
mallory secret
charlie@DESKTOP:~$ ls –la /home/mallory
drwxrwxrwx 0 mallory mallory   512 Jan  8 14:55 .
drwxr-xr-x 0 root    root      512 Oct 11 19:58 ..
charlie@DESKTOP:~$ cp /top-secret-intel/northkorea.pdf /home/mallory
charlie@DESKTOP:~$ ls –l /home/mallory
-rw-r----- 1 charlie charlie 896 Jan 29 22:47 northkorea.pdf
charlie@DESKTOP:~$ chmod ugo+rw /home/mallory/northkorea.pdf
```

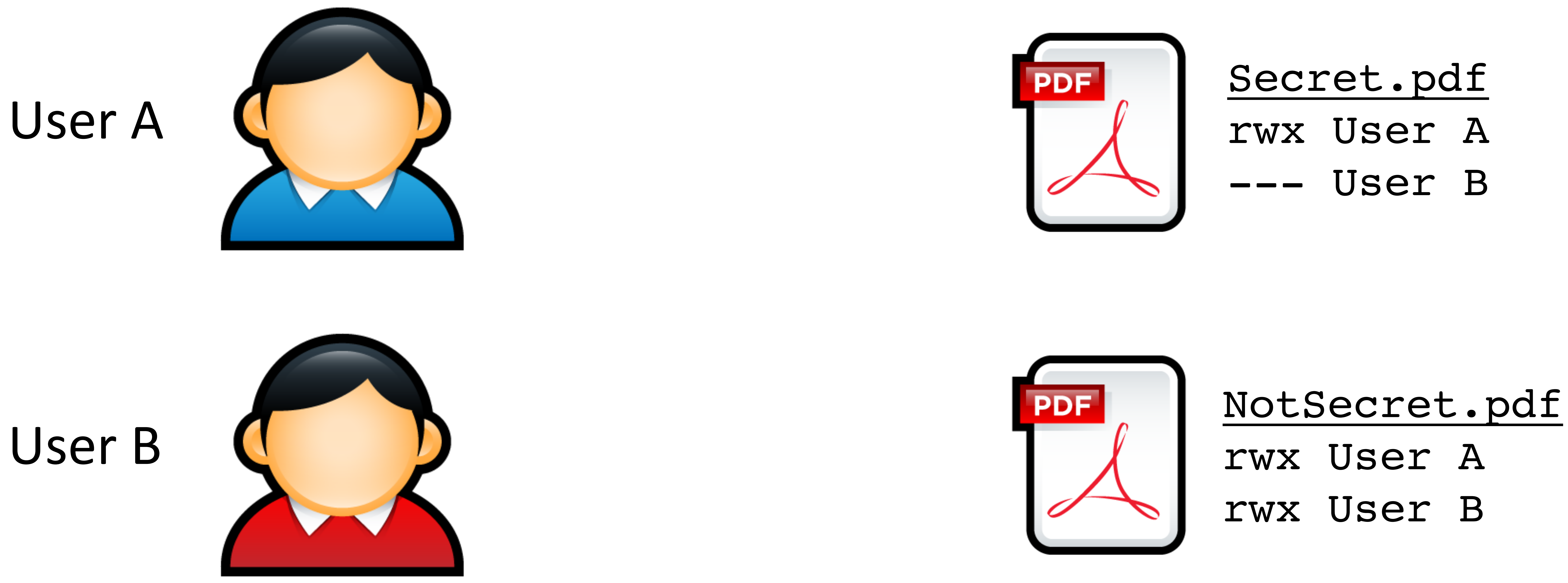
Keeping Secrets?

- Suppose we have secret data that only certain users should access
- Is DAC enough to prevent leaks?

```
charlie@DESKTOP:~$ groups
charlie topsecret
charlie@DESKTOP:~$ ls –la /top-secret-intel/
drwxr-xr-x 0 root root      512 Jan  8 14:55 .
drwxr-xr-x 0 root root      512 Oct 11 19:58 ..
-rw-r----- 1 root topsecret 896 Jan 29 22:47 northkorea.pdf
charlie@DESKTOP:~$ groups mallory
mallory secret
charlie@DESKTOP:~$ ls –la /home/mallory
drwxrwxrwx 0 mallory mallory   512 Jan  8 14:55 .
drwxr-xr-x 0 root    root      512 Oct 11 19:58 ..
charlie@DESKTOP:~$ cp /top-secret-intel/northkorea.pdf /home/mallory
charlie@DESKTOP:~$ ls –l /home/mallory
-rw-r----- 1 charlie charlie 896 Jan 29 22:47 northkorea.pdf
charlie@DESKTOP:~$ chmod ugo+rw /home/mallory/northkorea.pdf
```

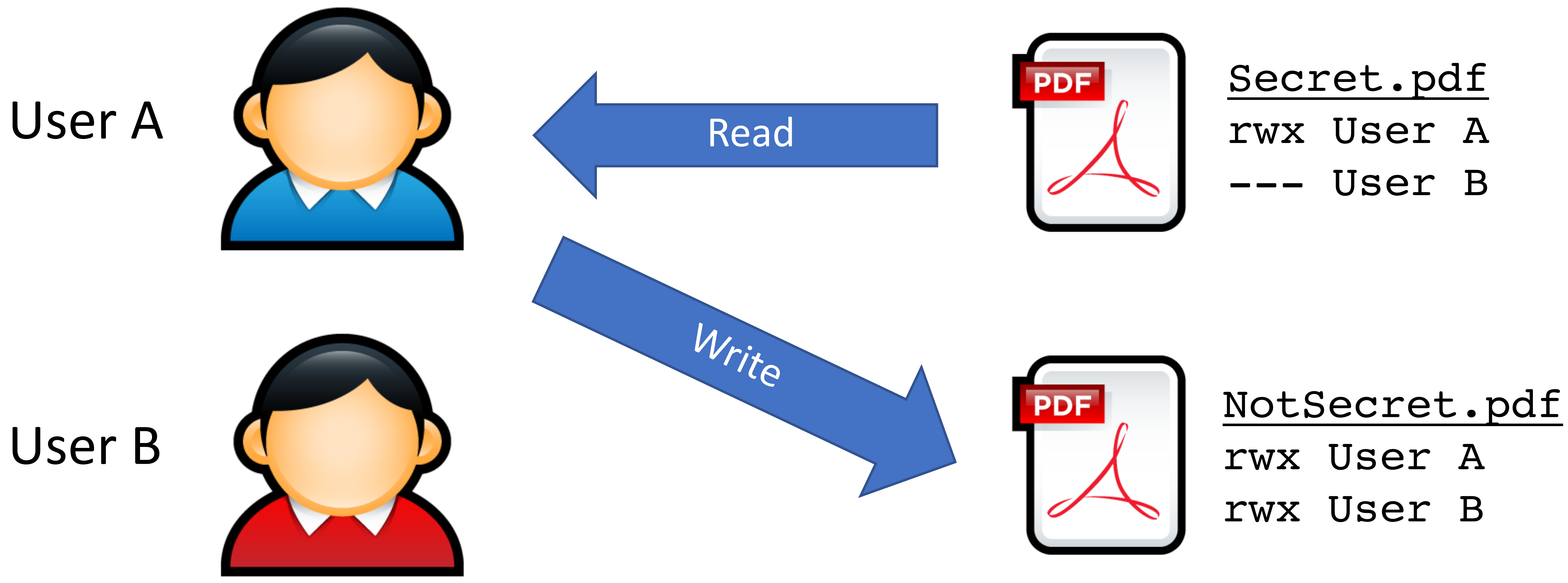
Failure of DAC

- DAC cannot prevent the leaking of secrets



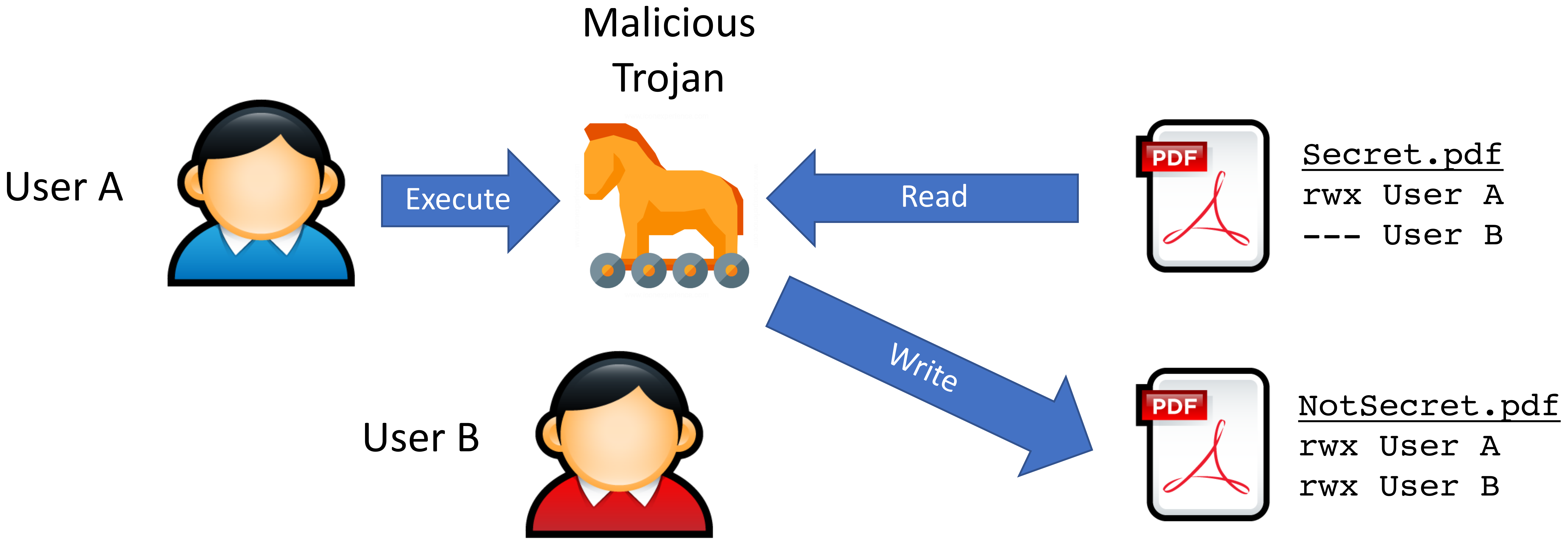
Failure of DAC

- DAC cannot prevent the leaking of secrets



Failure of DAC

- DAC cannot prevent the leaking of secrets



Mandatory Access Control

Mandatory Access Control Goals

- Restrict the access of subjects to objects based on a system-wide policy

Bell-Lapadula (1973)

"No read , no write "

System Model:

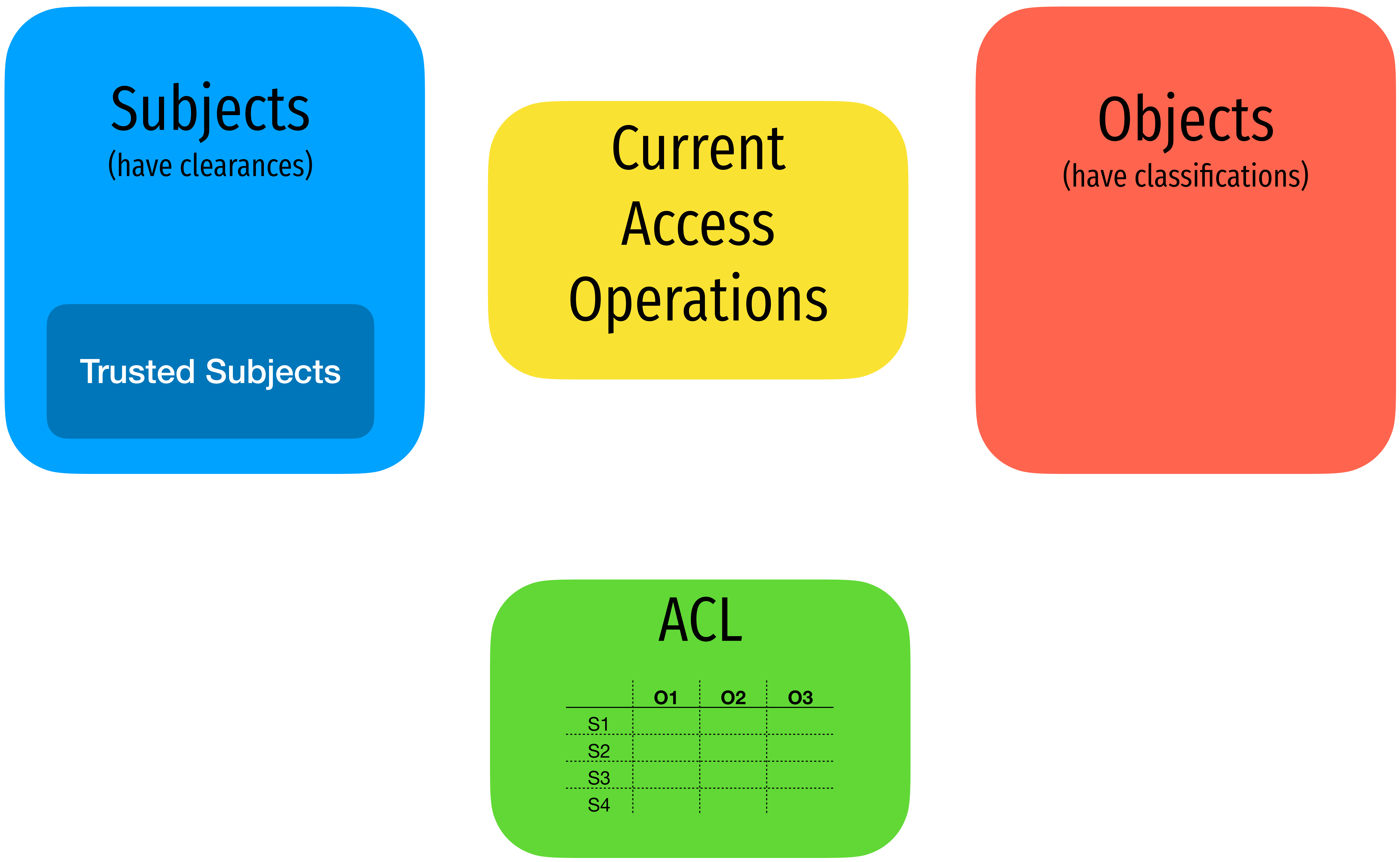
Security Policy:

BLP System Model

Clearances:

Classifications:

BLP System State



Elements of the Bell-LaPadula Model

Subjects

$L_m(s)$: maximum level

$L_c(s)$: current level

Top Secret

Secret

Confidential

Discretionary Access Control Matrix

Defined by the administrator

	o_1	o_2	o_3
s_1	RW	RX	
s_2	R	RWX	RW
s_3		RWX	

Objects

$L(o)$: level

Top Secret

Secret

Confidential

Unclassified

Simplified Bell-LaPadula Example

- Assume $L_m(s) = L_c(s)$ is always true

Simplified Bell-LaPadula Example

- Assume $L_m(s) = L_c(s)$ is always true
- ★-property
  - *s* can read *o* iff $L(s) >= L(o)$ (**no read up**)
  - *s* can write *o* iff $L(s) <= L(o)$ (**no write down**)

Confidential

Top Secret

Secret

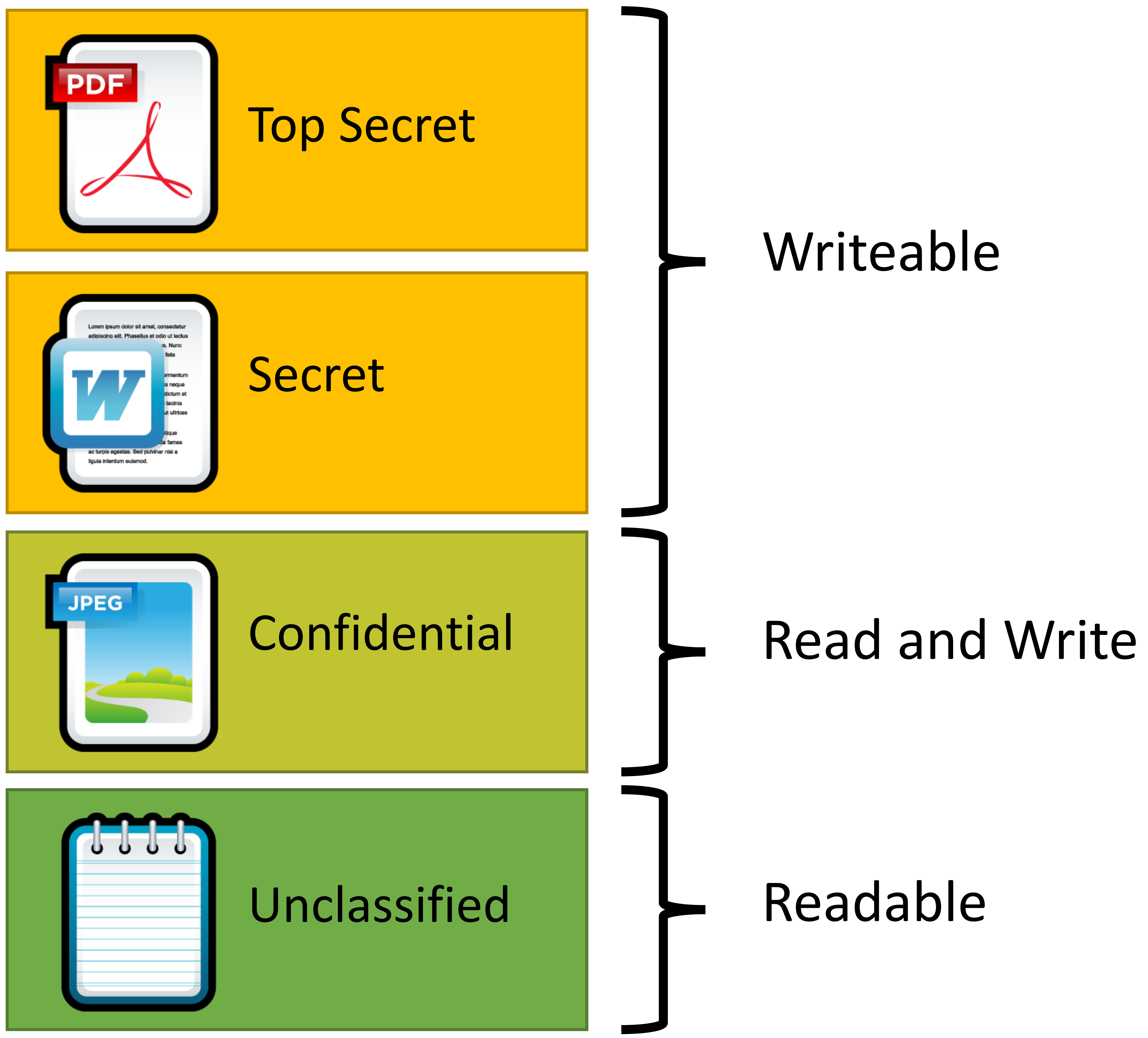
Confidential

Unclassified

Simplified Bell-LaPadula Example

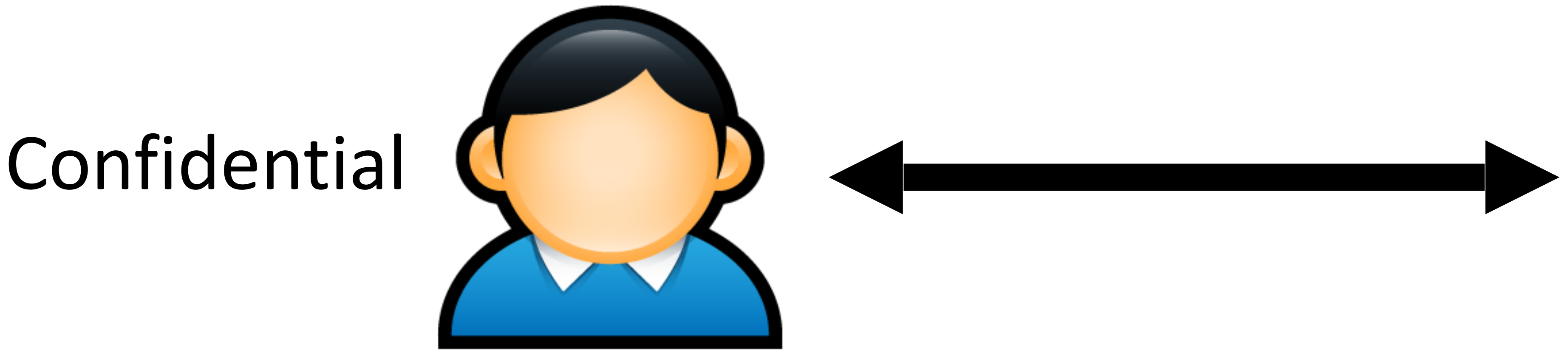
- Assume $L_m(s) = L_c(s)$ is always true
- ★-property
  - *s* can read *o* iff $L(s) >= L(o)$ (**no read up**)
  - *s* can write *o* iff $L(s) <= L(o)$ (**no write down**)

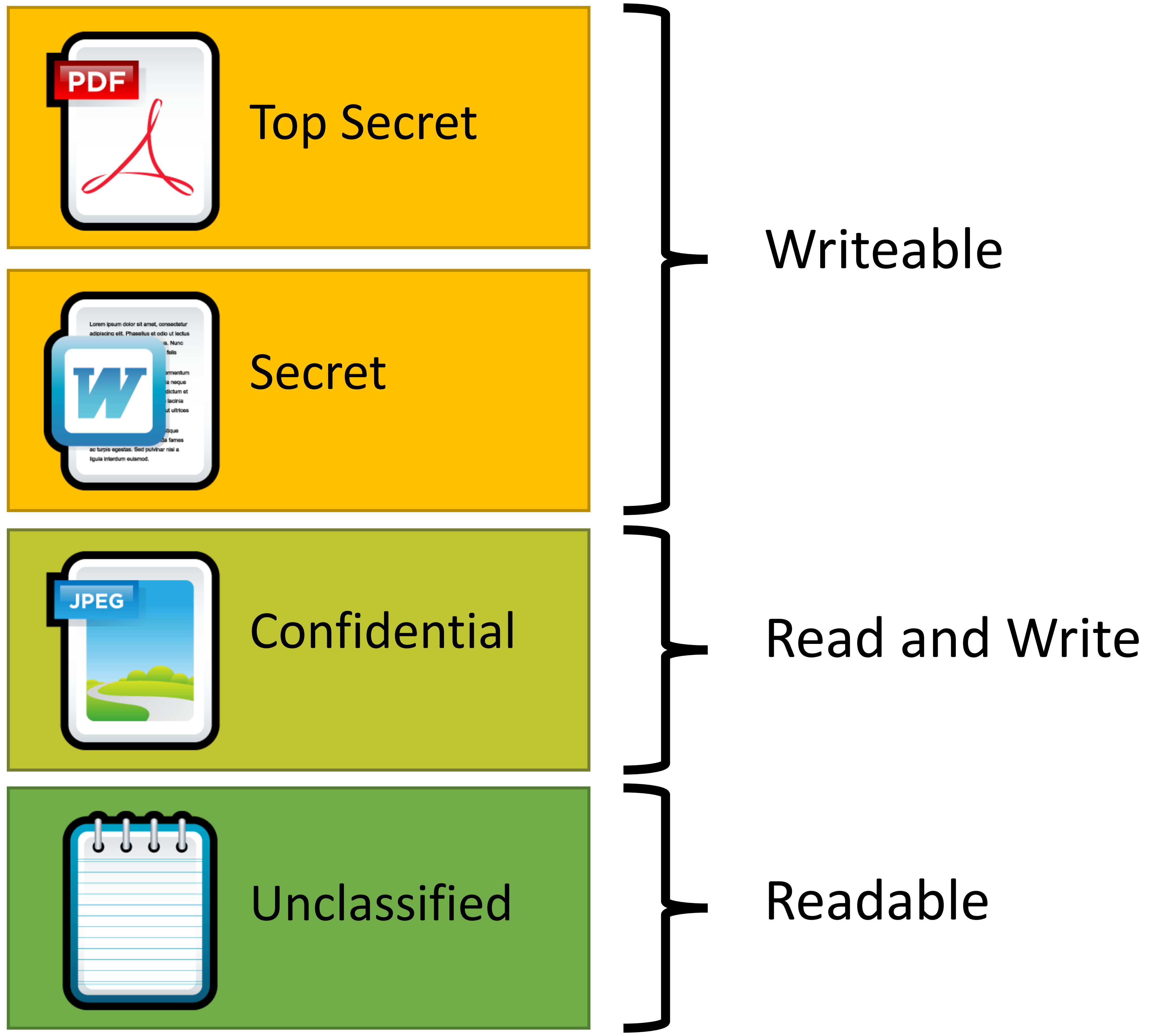
Confidential



Simplified Bell-LaPadula Example

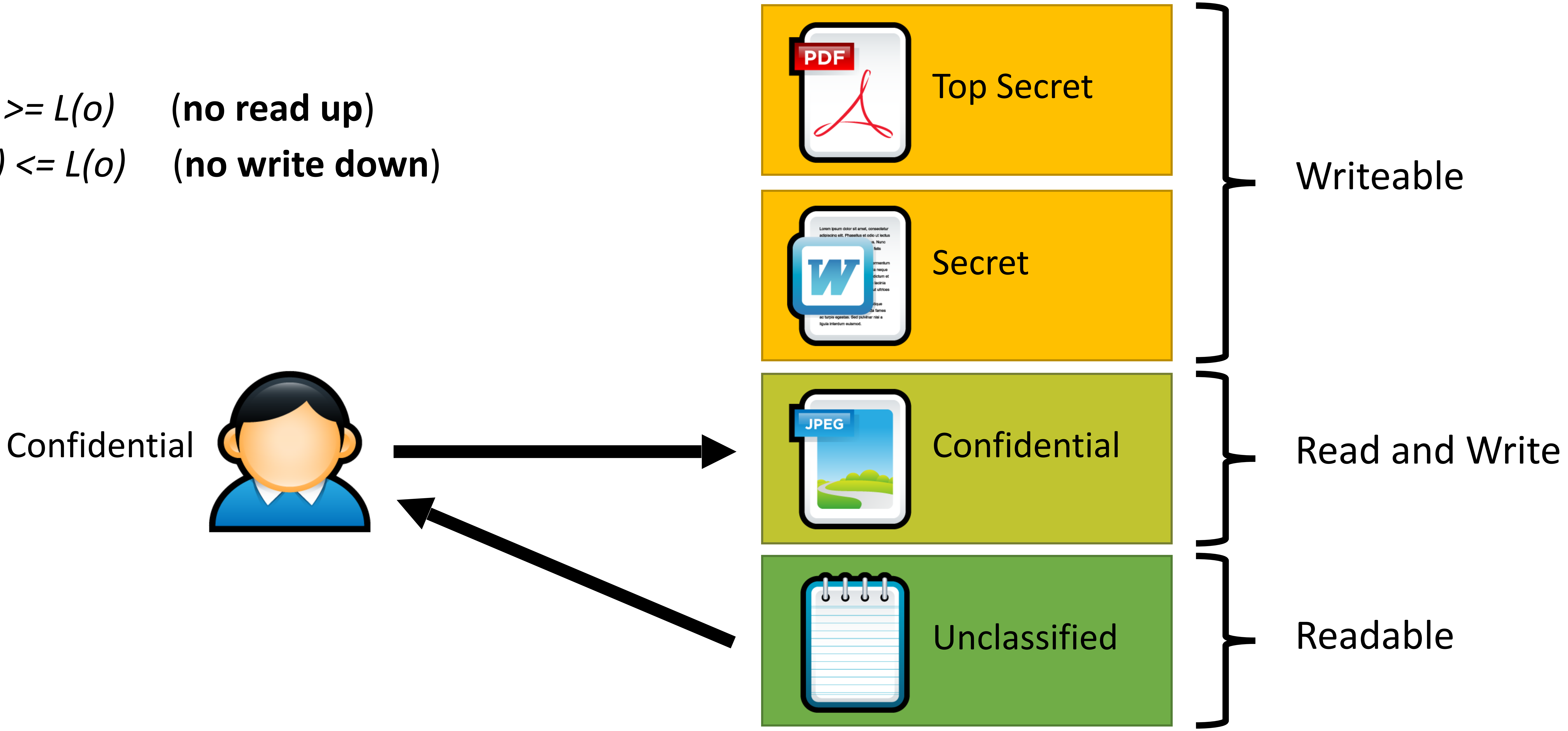
- Assume $L_m(s) = L_c(s)$ is always true
- ★-property
  - *s* can read *o* iff $L(s) >= L(o)$ (**no read up**)
  - *s* can write *o* iff $L(s) <= L(o)$ (**no write down**)





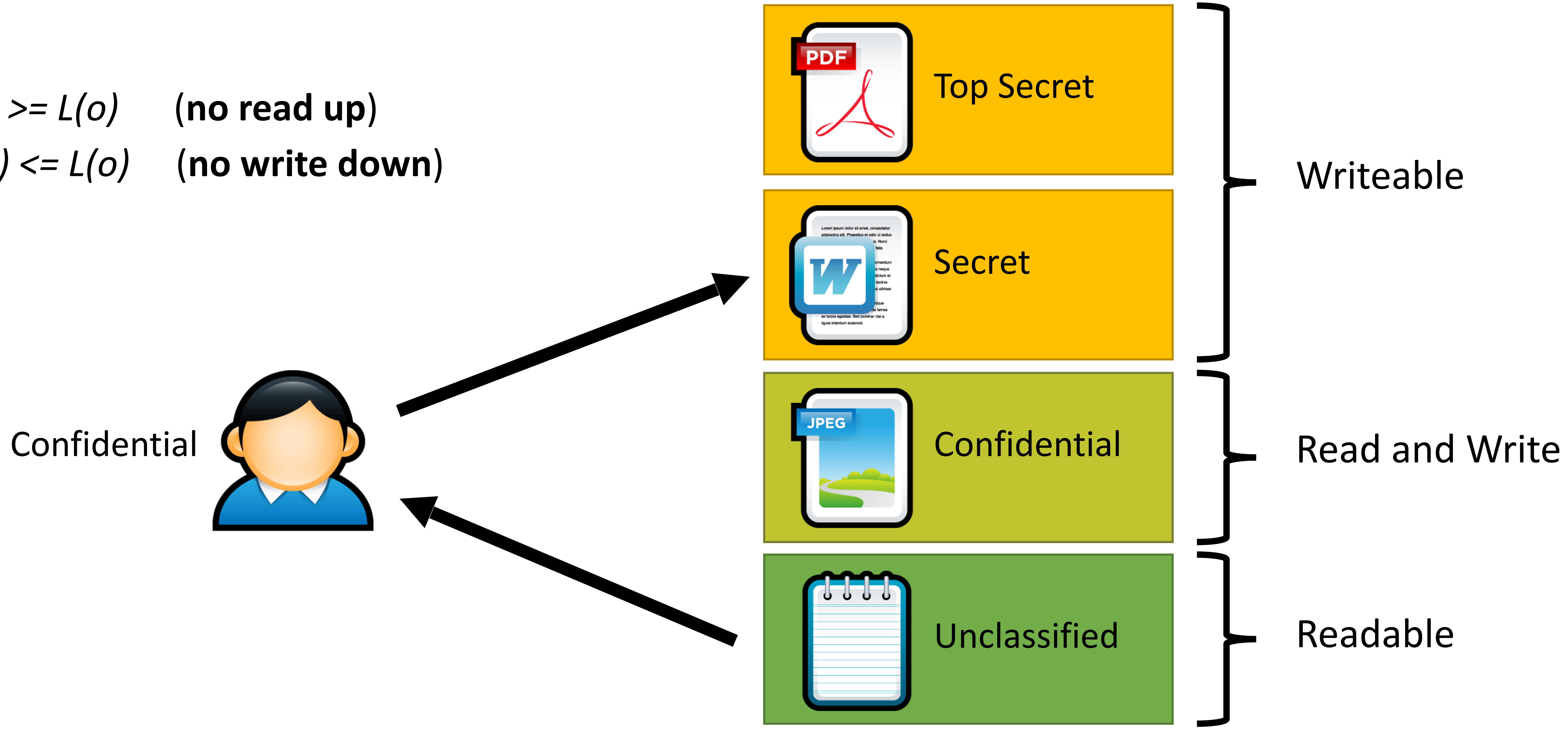
Simplified Bell-LaPadula Example

- Assume $L_m(s) = L_c(s)$ is always true
- ★-property
  - *s* can read *o* iff $L(s) >= L(o)$ (**no read up**)
  - *s* can write *o* iff $L(s) <= L(o)$ (**no write down**)



Simplified Bell-LaPadula Example

- Assume $L_m(s) = L_c(s)$ is always true
- ★-property
  - *s* can read *o* iff $L(s) >= L(o)$ (**no read up**)
  - *s* can write *o* iff $L(s) <= L(o)$ (**no write down**)



BLP Idea

A computer system is in a state, and undergoes state transitions whenever an operation occurs..

System is secure if all transitions satisfy 3 properties:

Simple:

Star:

Discretionary:

BLP Idea

A computer system is in a state, and undergoes state transitions whenever an operation occurs..

System is secure if all transitions satisfy 3 properties:

Simple: S can read O if S has higher clearance

Star: S can write O if S has lower clearance.

Discretionary: Every access allowed by ACL.

Users are trusted

Subjects are not trusted. (Malware)

App armor

Whenever a protected program runs regardless of UID, AppArmor controls:

- The POSIX capabilities it can have (even if it is running as root)
- The directories/files it can read/write/execute

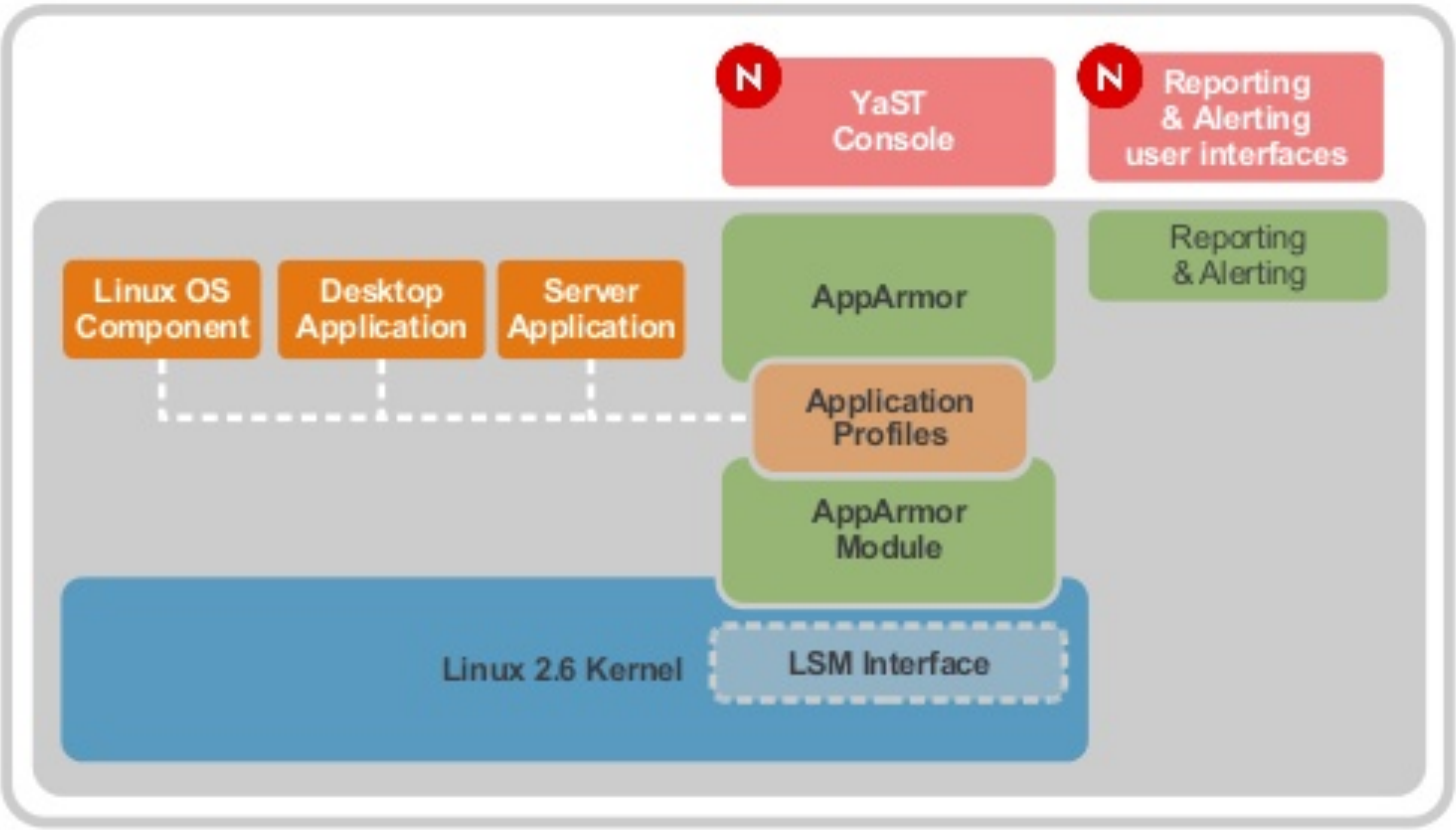
Example security profile for `ntpd`

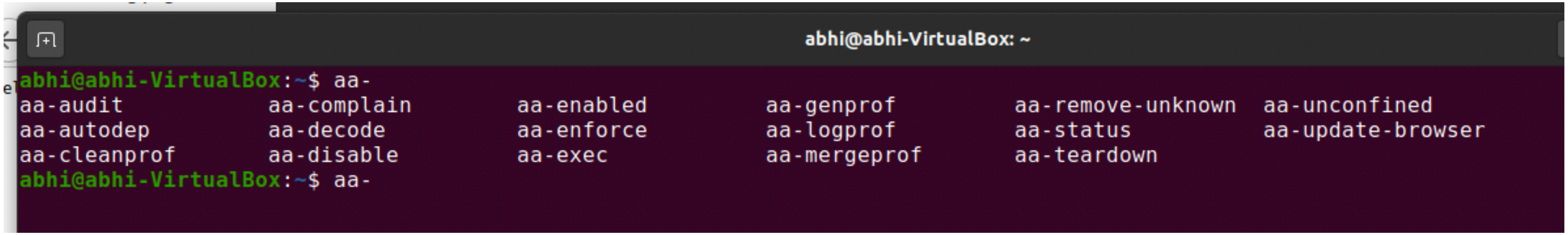
```
/usr/sbin/ntpd {
  #include <abstractions/base>
  #include <abstractions/nameservice>

  capability ipc_lock,
  capability net_bind_service,
  capability sys_time,
  capability sys_chroot,
  capability setuid,

  /etc/ntp.conf                          r,
  /etc/ntp/drift*                        rwl,
  /etc/ntp/keys                          r,
  /etc/ntp/step-tickers                  r,
  /tmp/ntp*                              rwl,
  /usr/sbin/ntpd                         rix,
  /var/log/ntp                           w,
  /var/log/ntp.log                       w,
  /var/run/ntpd.pid                      w,
  /var/lib/ntp/drift                     rwl,
  /var/lib/ntp/drift.TEMP                rwl,
  /var/lib/ntp/var/run/ntp/ntpd.pid      w,
  /var/lib/ntp/drift/ntp.drift           r,
  /drift/ntp.drift.TEMP                  rwl,
  /drift/ntp.drift                       rwl,
}
```


Slide from Novell/defcon 2015

Apparmor

AppArmor Architecture




abhi@abhi-VirtualBox: ~
abhi@abhi-VirtualBox:~$ aa-
aa-audit aa-complain aa-enabled aa-genprof aa-remove-unknown aa-unconfined
aa-autodep aa-decode aa-enforce aa-logprof aa-status aa-update-browser
aa-cleanprof aa-disable aa-exec aa-mergeprof aa-teardown
abhi@abhi-VirtualBox:~$ aa-

Apparmor

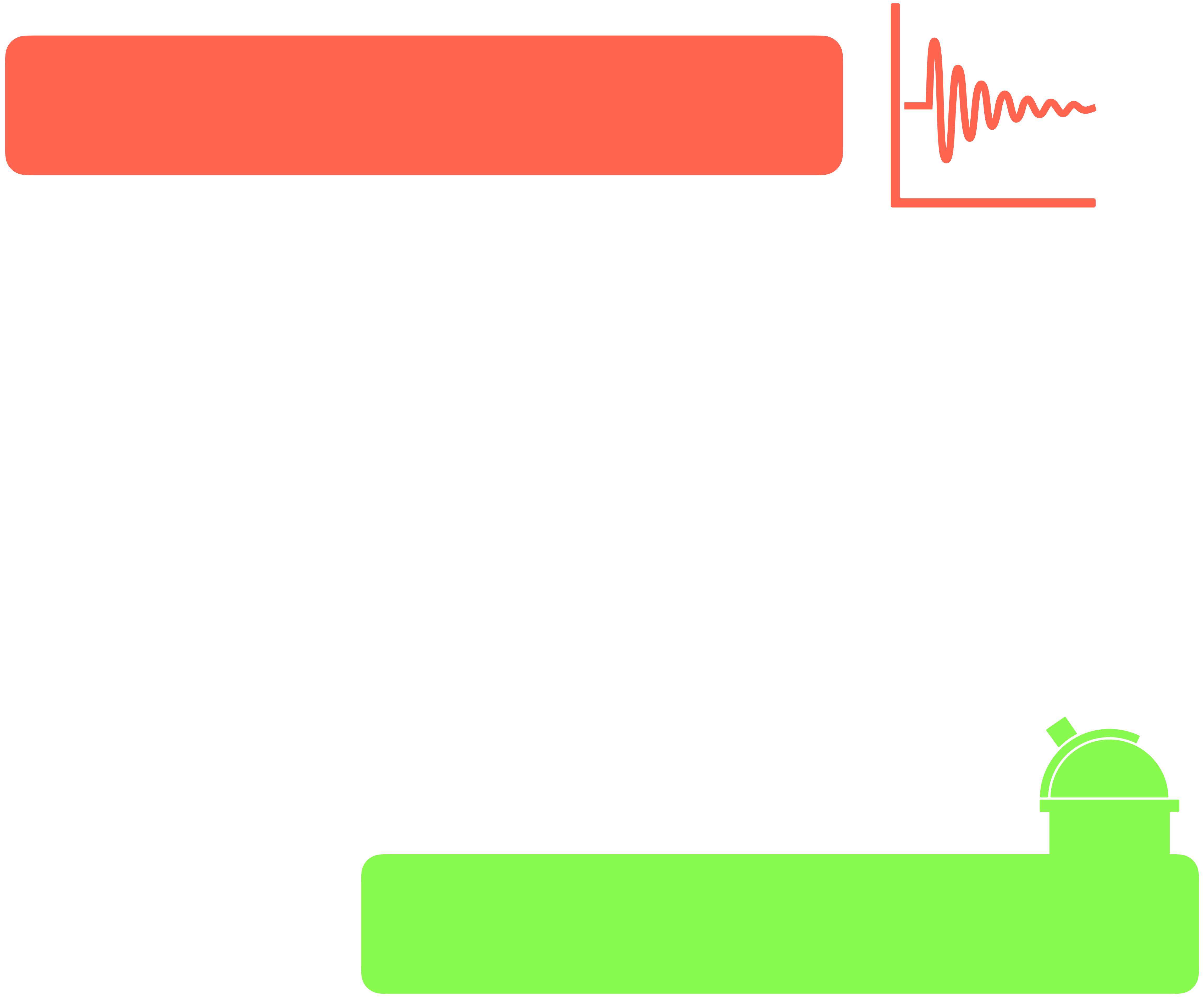

```
# vim:syntax=apparmor
#include <tunables/global>

/usr/sbin/tcpdump {
  #include <abstractions/base>
  #include <abstractions/nameservice>
  #include <abstractions/user-tmp>

  capability net_raw,
  capability setuid,
  capability setgid,
  capability dac_override,
  capability chown,
  network raw,
  network packet,

  # for -D
  @{PROC}/bus/usb/ r,
  @{PROC}/bus/usb/** r,

  # for finding an interface
  /dev/ r,
  @{PROC}/[0-9]*/net/dev r,
  /sys/bus/usb/devices/ r,
  /sys/class/net/ r,
  /sys/devices/**/net/** r,

  # for -j
  capability net_admin,

  # for tracing USB bus, which libpcap supports
  /dev/usbmon* r,
  /dev/bus/usb/ r,
  /dev/bus/usb/** r,

  # for init_etherarray(), with -e
  /etc/ethers r,

  # for USB probing (see libpcap-1.1.x/pcap-usb-linux.c:probe_devices())
  /dev/bus/usb/**/[0-9]* w,

  # for -z
  /{usr/,}bin/gzip ixr,
  /{usr/,}bin/bzip2 ixr,

  # for -F and -w
  audit deny @{HOME}/.* mrwkl,
  audit deny @{HOME}/.*/ rw,
/etc/apparmor.d/usr.sbin.tcpdump
```

Not Enough

TopSecret.pdf
rwx User A
--- User B

NotSecret.pdf
rwx User A
rwx User B

Not Enough: Covert channels

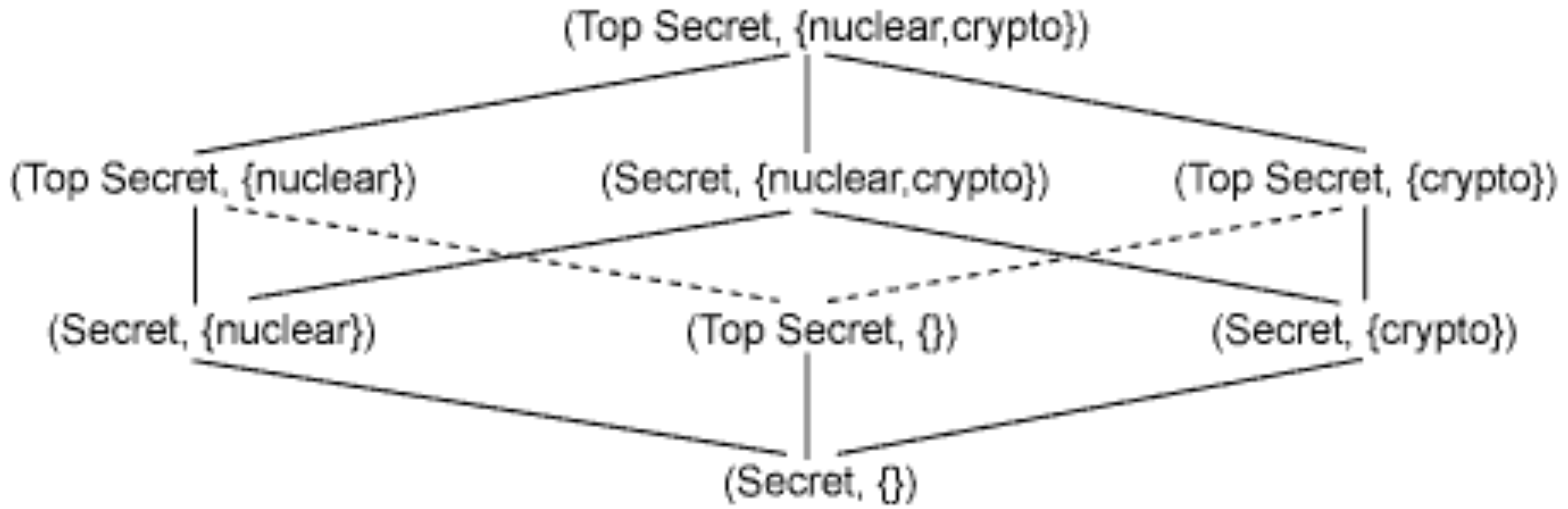


Security Lattice

Compartments:

Ordering between (Level, Compartment)

Lattice


(Top Secret, {nuclear,crypto})
(Top Secret, {nuclear})
(Secret, {nuclear,crypto})
(Top Secret, {crypto})
(Secret, {nuclear})
(Top Secret, {})
(Secret, {crypto})
(Secret, {})

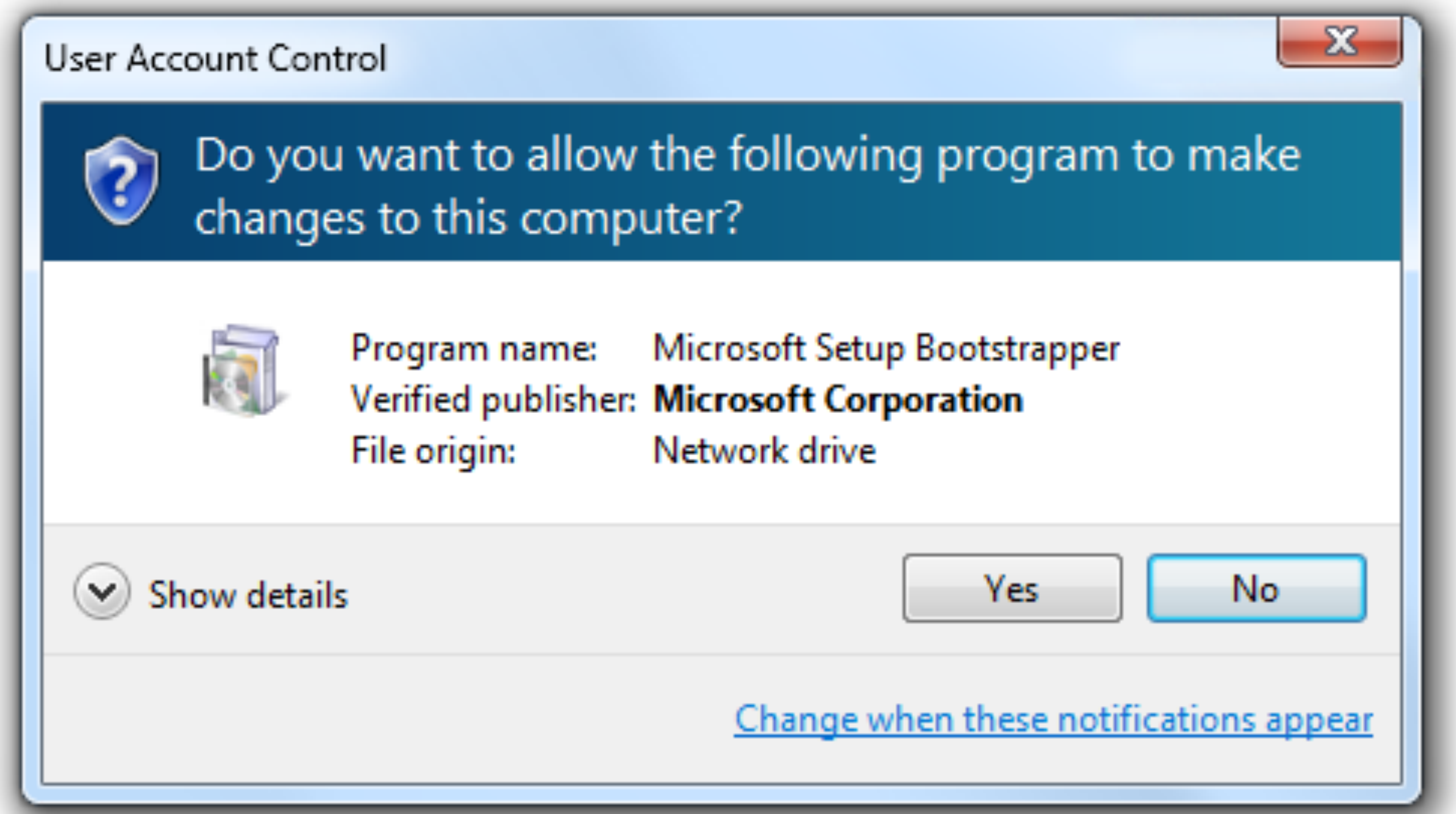
Need-to-Know policy

Integrity Protection in Practice

- Mandatory Integrity Control in Windows
  - Since Vista
  - Four integrity levels: Low, Medium, High, System
  - Each process assigned a level
    - Processes started by normal users are Medium
    - Elevated processes have High
  - Some processes intentionally run as Low
    - Internet Explorer in protected mode
  - Ring policy
    - Reading and writing do not change integrity level

Integrity Protection in Practice

- Mandatory Integrity Control in Windows
  - Since Vista
  - Four integrity levels: Low, Medium, High, System
  - Each process assigned a level
    - Processes started by normal users are Medium
    - Elevated processes have High
  - Some processes intentionally run as Low
    - Internet Explorer in protected mode
  - Ring policy
    - Reading and writing do not change integrity level



Confidentiality? What else?

Biba Integrity Policy

Biba Integrity Model

- Proposed in 1975
- Like Bell-LaPadula, security model with provable properties based on a state transition model
  - Each subject has an integrity level
  - Each object has an integrity level
  - Integrity levels are totally ordered (high → low)
- Integrity levels in Biba are not the same as security levels in Bell-LaPadula
  - Some high integrity data does not need confidentiality
  - Examples: stock prices, official statements from the president

Possible Mandatory Policies in Biba

1. Strict integrity
   - *s* can read *o* iif $i(s) <= i(o)$ (**no read down**)
   - *s* can write *o* iff $i(s) >= i(o)$ (**no write up**)

Possible Mandatory Policies in Biba

1. Strict integrity
   - *s* can read *o* iif *i(s) <= i(o)* (**no read down**)
   - *s* can write *o* iff *i(s) >= i(o)* (**no write up**)
2. Subject low-water mark
   - *s* can always read *o*; afterward *i(s) = min(i(s), i(o))* (**subject tainting**)
   - *s* can write *o* iff *i(s) >= i(o)* (**no write up**)

Possible Mandatory Policies in Biba

1. Strict integrity
   - *s* can read *o* iif *i(s) <= i(o)* (**no read down**)
   - *s* can write *o* iff *i(s) >= i(o)* (**no write up**)
2. Subject low-water mark
   - *s* can always read *o*; afterward *i(s) = min(i(s), i(o))* (**subject tainting**)
   - *s* can write *o* iff *i(s) >= i(o)* (**no write up**)
3. Object low-water mark
   - *s* can read *o* iif *i(s) <= i(o)* (**no read down**)
   - *s* can always write *o*; afterward *o(s) = min(i(s), i(o))* (**object tainting**)

Possible Mandatory Policies in Biba

1. Strict integrity
   - *s* can read *o* iif *i(s) <= i(o)* (**no read down**)
   - *s* can write *o* iff *i(s) >= i(o)* (**no write up**)
2. Subject low-water mark
   - *s* can always read *o*; afterward *i(s) = min(i(s), i(o))* (**subject tainting**)
   - *s* can write *o* iff *i(s) >= i(o)* (**no write up**)
3. Object low-water mark
   - *s* can read *o* iif *i(s) <= i(o)* (**no read down**)
   - *s* can always write *o*; afterward *o(s) = min(i(s), i(o))* (**object tainting**)
4. Low-water mark integrity audit
   - *s* can always read *o*; afterward *i(s) = min(i(s), i(o))* (**subject tainting**)
   - *s* can always write *o*; afterward *o(s) = min(i(s), i(o))* (**object tainting**)

Possible Mandatory Policies in Biba

1. Strict integrity
   - s can read o iif $i(s) <= i(o)$ (**no read down**)
   - s can write o iff $i(s) >= i(o)$ (**no write up**)
2. Subject low-water mark
   - s can always read o; afterward $i(s) = min(i(s), i(o))$ (**subject tainting**)
   - s can write o iff $i(s) >= i(o)$ (**no write up**)
3. Object low-water mark
   - s can read o iif $i(s) <= i(o)$ (**no read down**)
   - s can always write o; afterward $o(s) = min(i(s), i(o))$ (**object tainting**)
4. Low-water mark integrity audit
   - s can always read o; afterward $i(s) = min(i(s), i(o))$ (**subject tainting**)
   - s can always write o; afterward $o(s) = min(i(s), i(o))$ (**object tainting**)
5. Ring
   - s can read any object o
   - s can write o iff $i(s) >= i(o)$ (**no write up**)

Biba Strict Integrity Example

- Strict integrity
  - *s* can read *o* iif $i(s) <= i(o)$ (**no read down**)
  - *s* can write *o* iff $i(s) >= i(o)$ (**no write up**)

Medium Integrity 

High Integrity


Medium Integrity

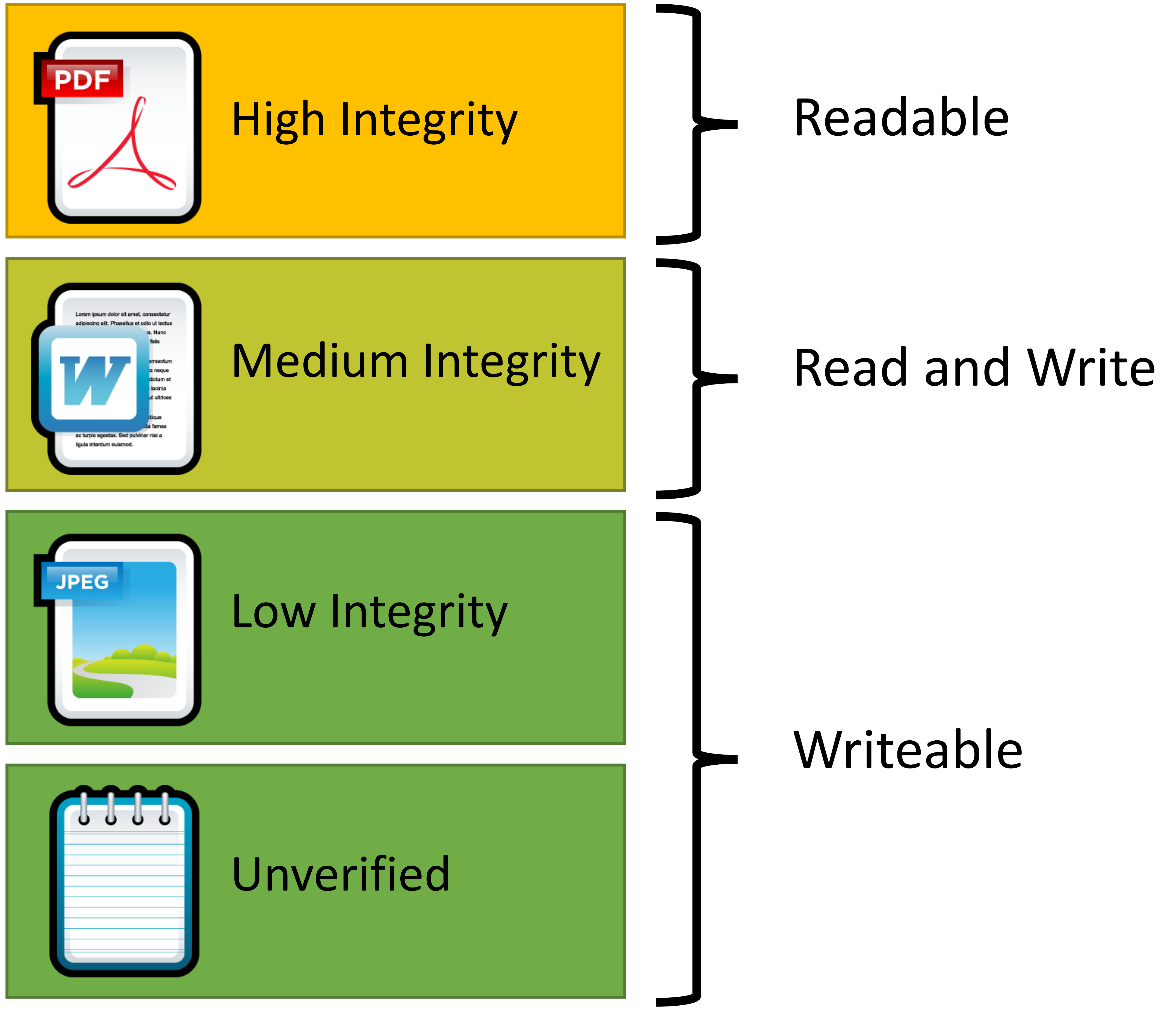
Low Integrity


Unverified

Biba Strict Integrity Example

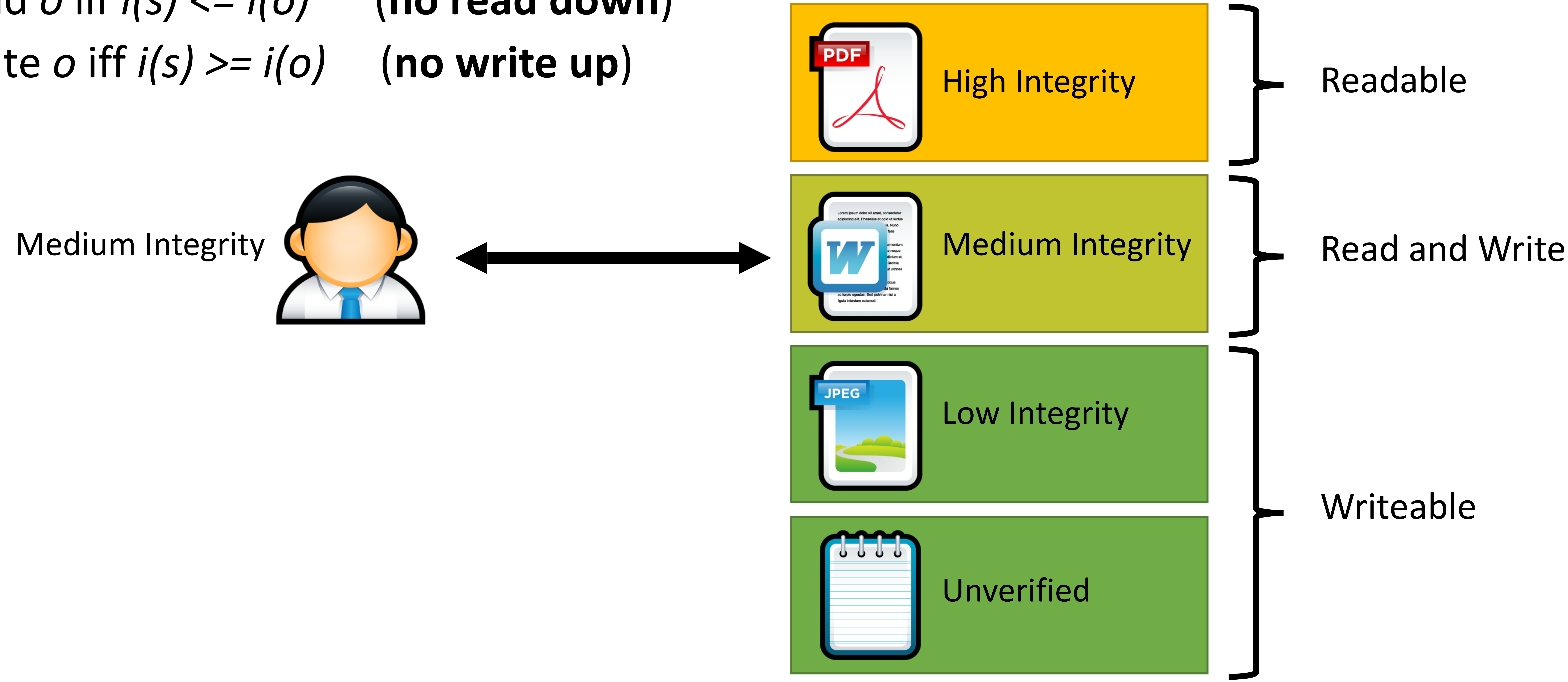
- Strict integrity
  - *s* can read *o* iif *i(s)* <= *i(o)* **(no read down)**
  - *s* can write *o* iff *i(s)* >= *i(o)* **(no write up)**

Medium Integrity



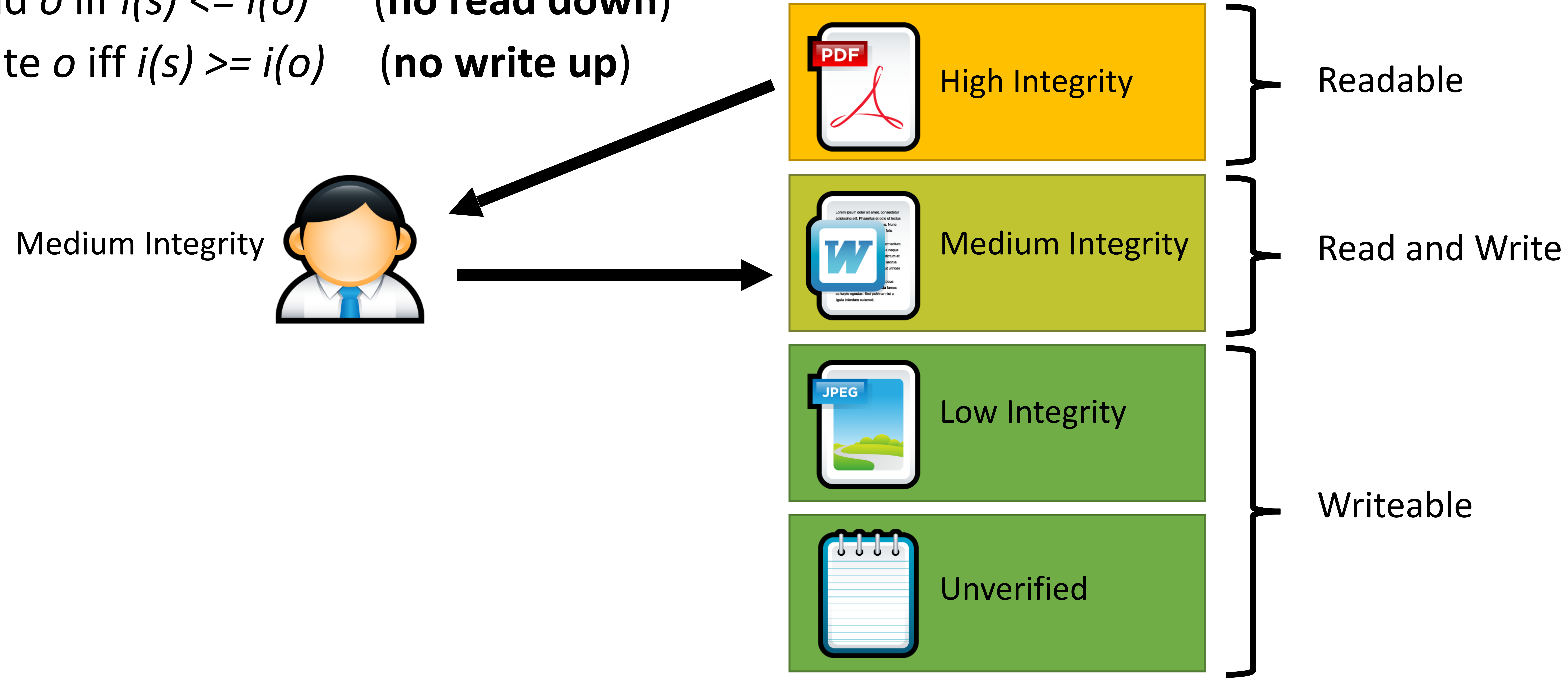
Biba Strict Integrity Example

- Strict integrity
  - *s* can read *o* iif *i(s) <= i(o)* **(no read down)**
  - *s* can write *o* iff *i(s) >= i(o)* **(no write up)**



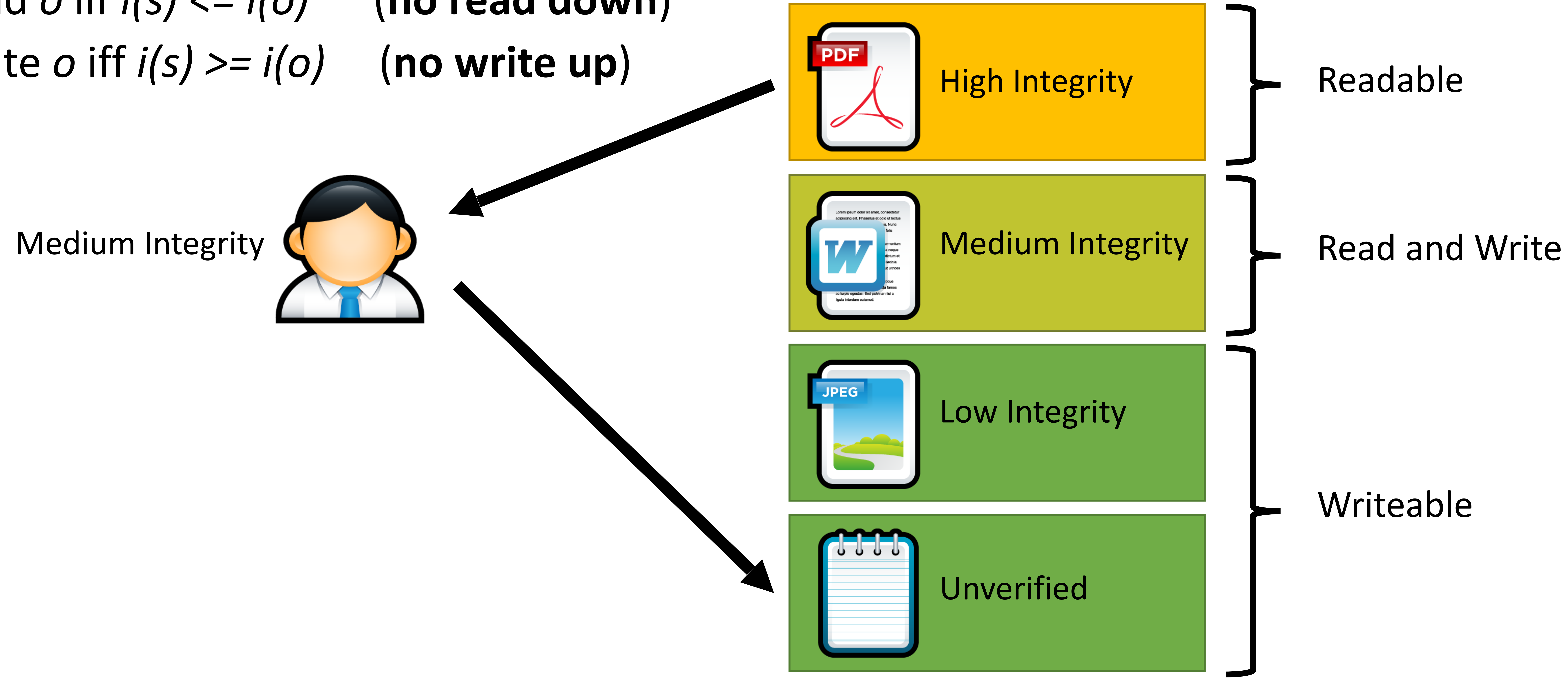
Biba Strict Integrity Example

- Strict integrity
  - *s* can read *o* iif *i(s)* <= *i(o)* **(no read down)**
  - *s* can write *o* iff *i(s)* >= *i(o)* **(no write up)**



Biba Strict Integrity Example

- Strict integrity
  - *s* can read *o* iif *i(s) <= i(o)* **(no read down)**
  - *s* can write *o* iff *i(s) >= i(o)* **(no write up)**



Practical Example of Biba Integrity

- Military chain of command
  - Generals may issue orders to majors and privates
  - Majors may issue orders to privates, but not generals
  - Privates may only take orders

Comparison

BPL

- Offers confidentiality
- “Read down, write up”
- Focuses on controlling reads
- Theoretically, no requirement that subjects be trusted
  - Even malicious programs can’t leak secrets they don’t know

Biba

Comparison

BPL

- Offers confidentiality
- “Read down, write up”
- Focuses on controlling reads
- Theoretically, no requirement that subjects be trusted
  - Even malicious programs can’t leak secrets they don’t know

Biba

- Offers integrity

Comparison

BPL

- Offers confidentiality
- “Read down, write up”
- Focuses on controlling reads
- Theoretically, no requirement that subjects be trusted
  - Even malicious programs can’t leak secrets they don’t know

Biba

- Offers integrity
- “Read up, write down”

Comparison

BPL

- Offers confidentiality
- “Read down, write up”
- Focuses on controlling reads
- Theoretically, no requirement that subjects be trusted
  - Even malicious programs can’t leak secrets they don’t know

Biba

- Offers integrity
- “Read up, write down”
- Focuses on controlling writes

Comparison

BPL

- Offers confidentiality
- “Read down, write up”
- Focuses on controlling reads
- Theoretically, no requirement that subjects be trusted
  - Even malicious programs can’t leak secrets they don’t know

Biba

- Offers integrity
- “Read up, write down”
- Focuses on controlling writes
- Subjects must be trusted
  - A malicious program can write bad information

Covert and Side Channels

Caveats of Bell-LaPadula

Caveats of Bell-LaPadula

- ★-property prevents **overt** leakage of information
  - Does not address covert channels

Caveats of Bell-LaPadula

- ★-property prevents **overt** leakage of information
  - Does not address covert channels
- What does this mean?

Covert Channels

- Access control is defined over “legitimate” channels
  - Read/write an object
  - Send/receive a packet from the network
  - Read/write shared memory
- However, isolation in real systems is imperfect
  - Actions have observable side-effects

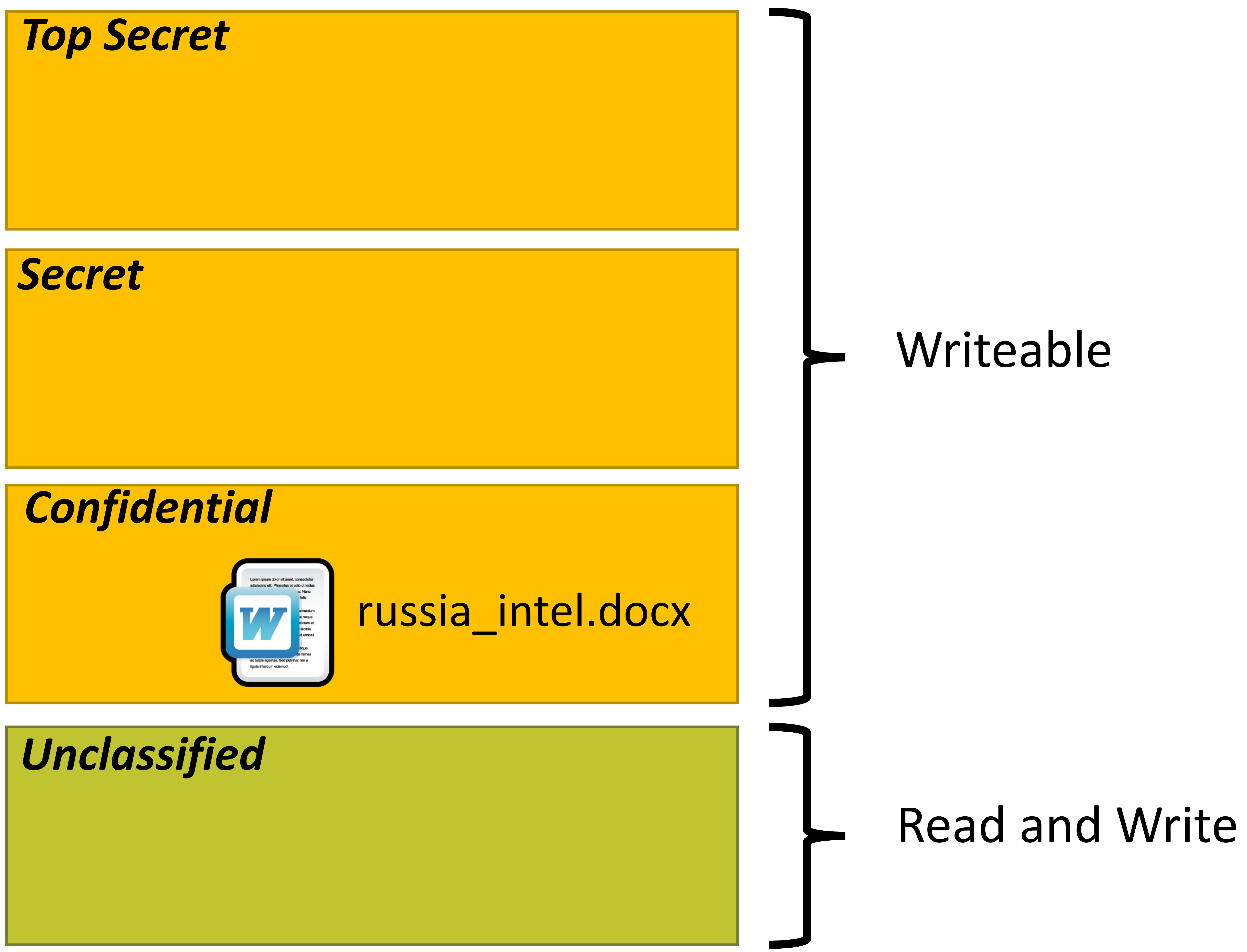
Covert Channels

- Access control is defined over “legitimate” channels
  - Read/write an object
  - Send/receive a packet from the network
  - Read/write shared memory
- However, isolation in real systems is imperfect
  - Actions have observable side-effects
- External observations can create covert channels
  - Communication via unintentional channels
  - Examples:
    - Existence of file(s) or locks on file(s)
    - Measure the timing of events
    - CPU cache (e.g. Meltdown and Spectre)

Simple Example

Unclassified

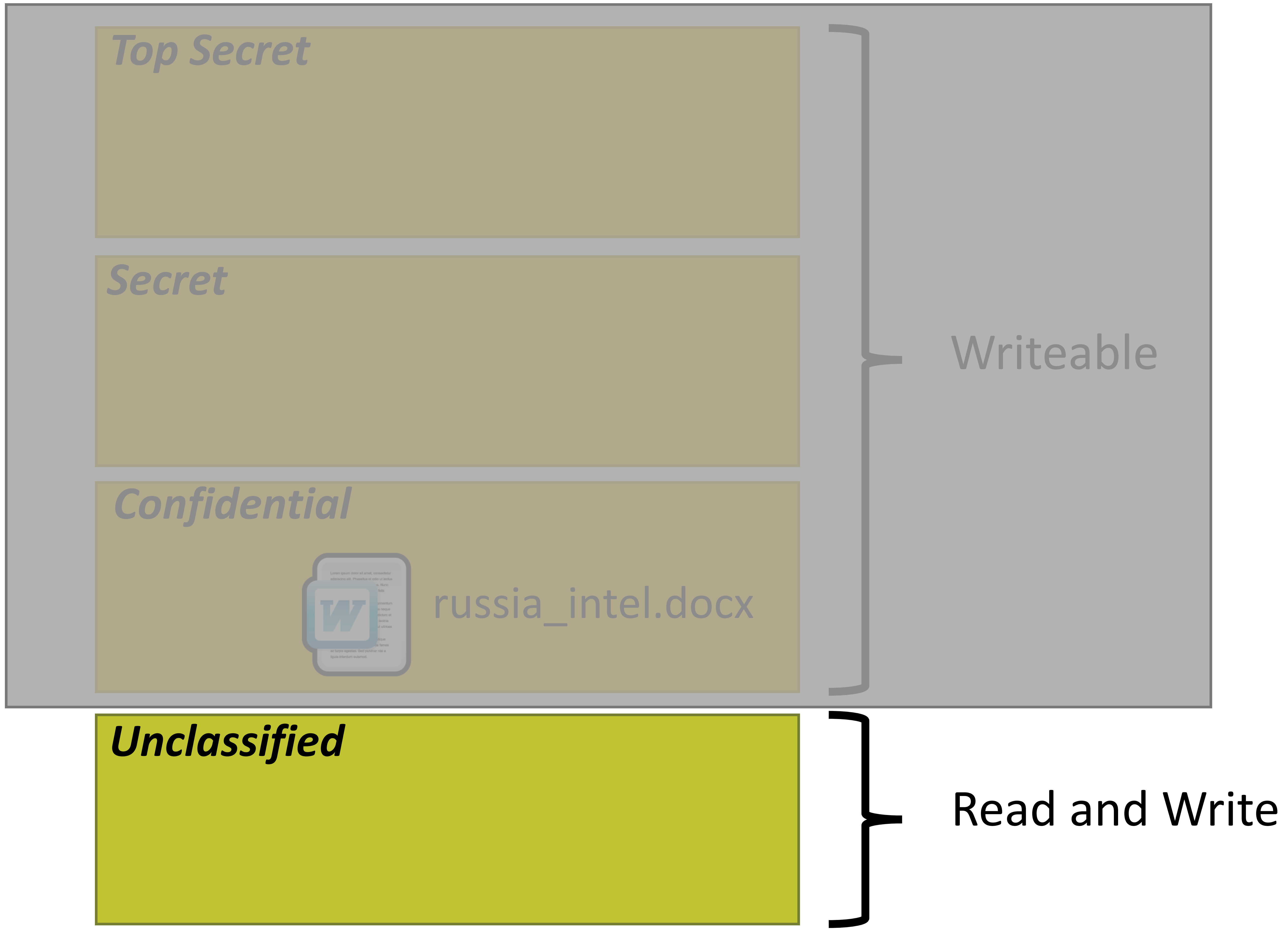
Bell-LaPadula MAC



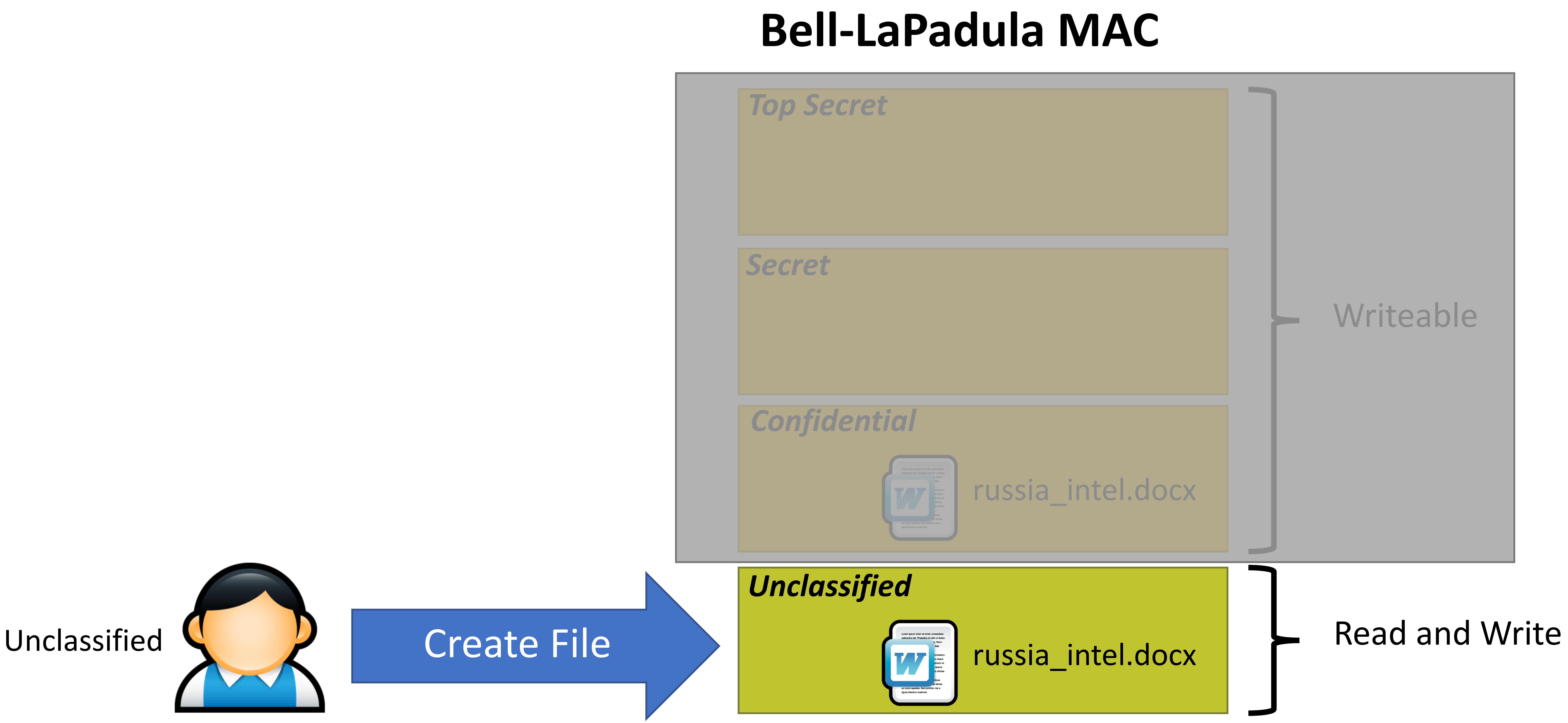
Simple Example

Unclassified

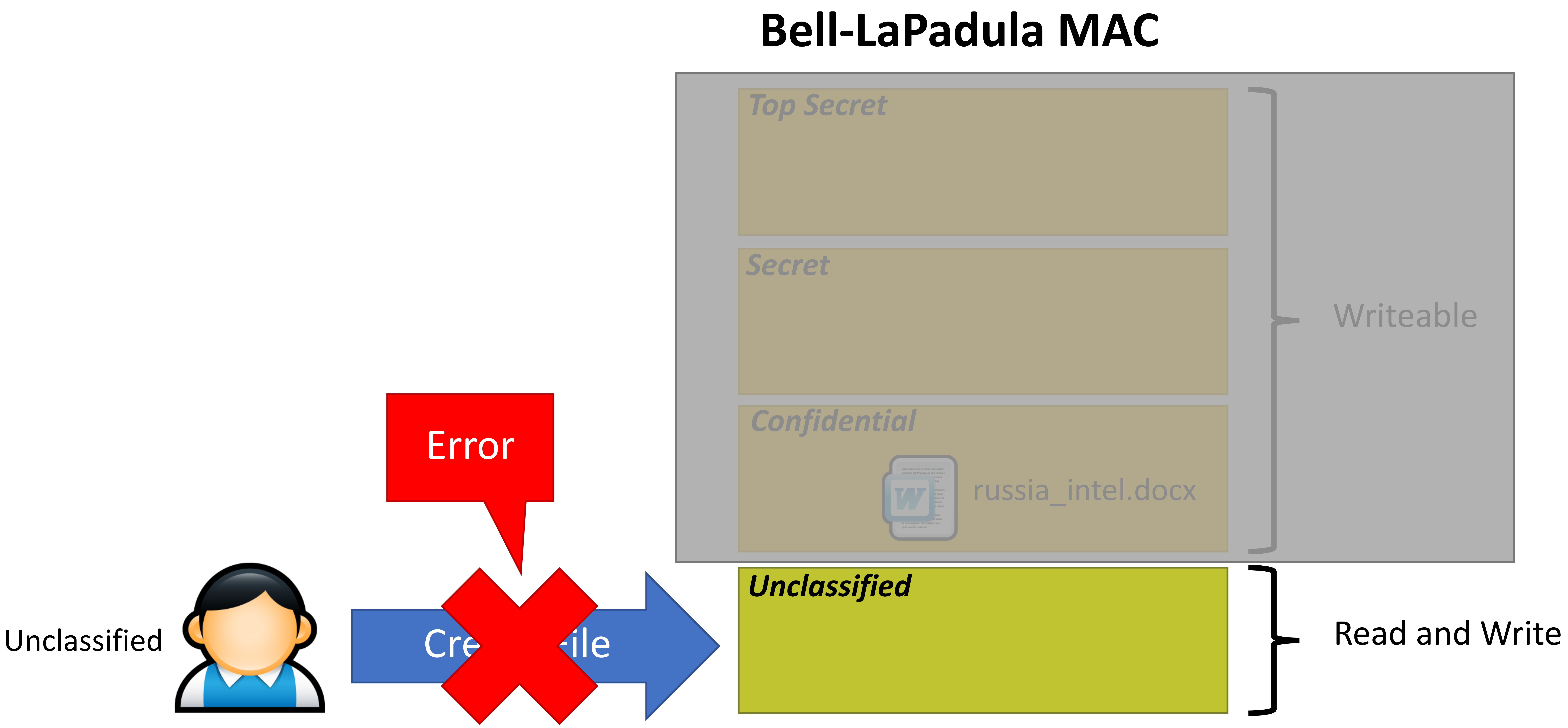
Bell-LaPadula MAC



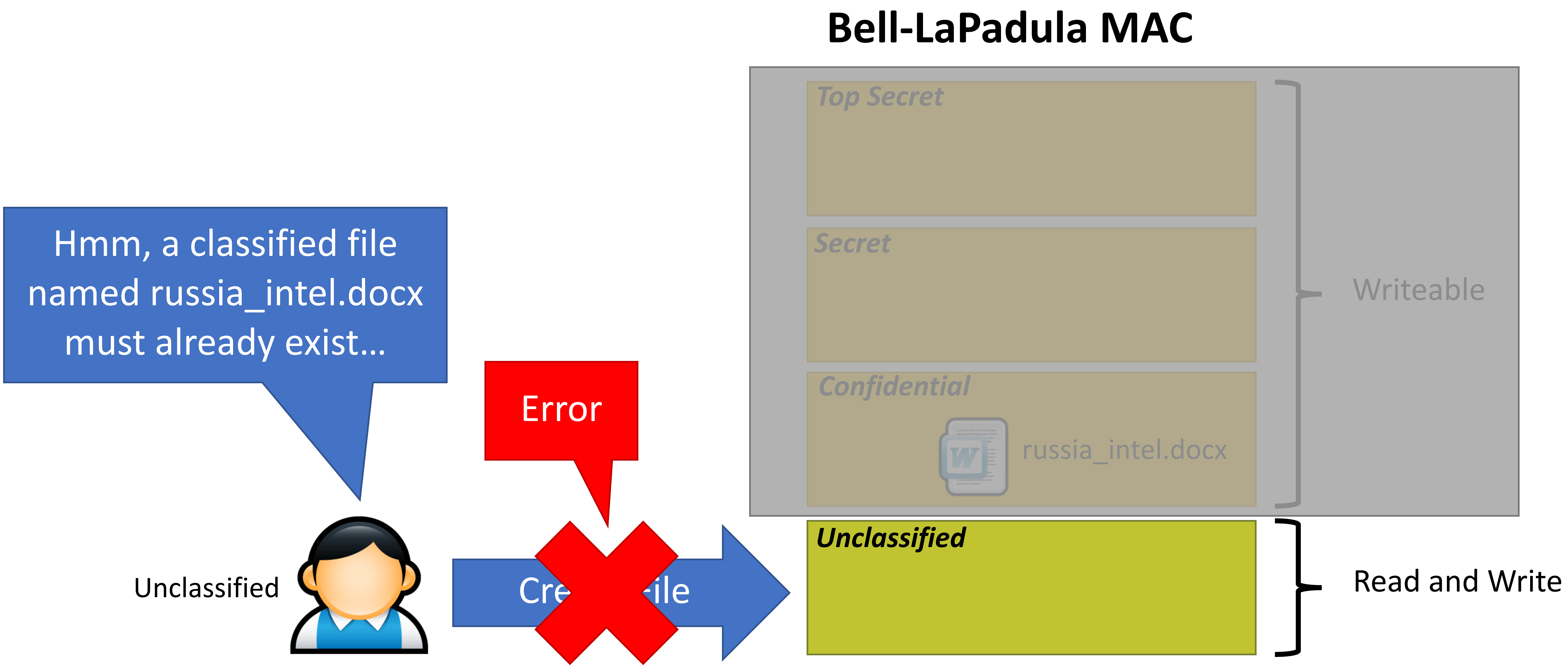
Simple Example



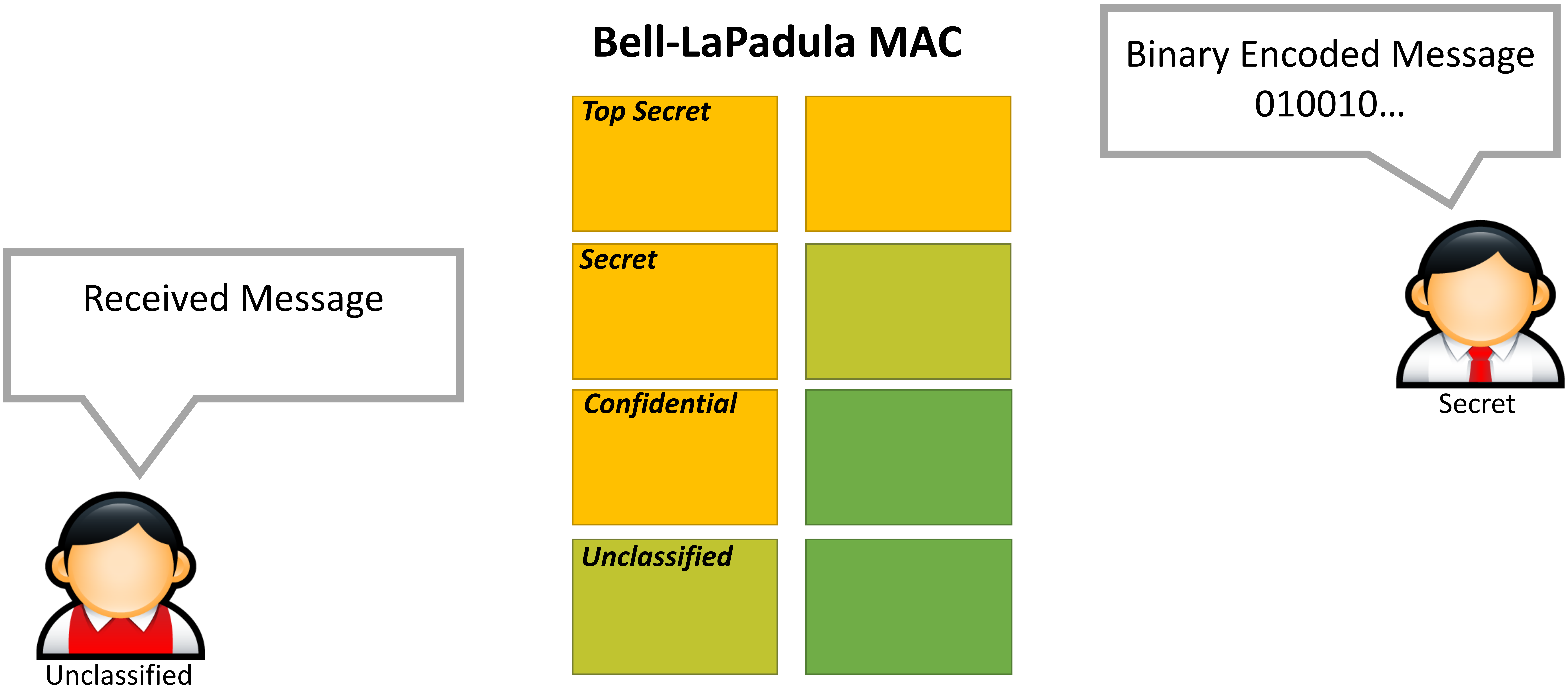
Simple Example



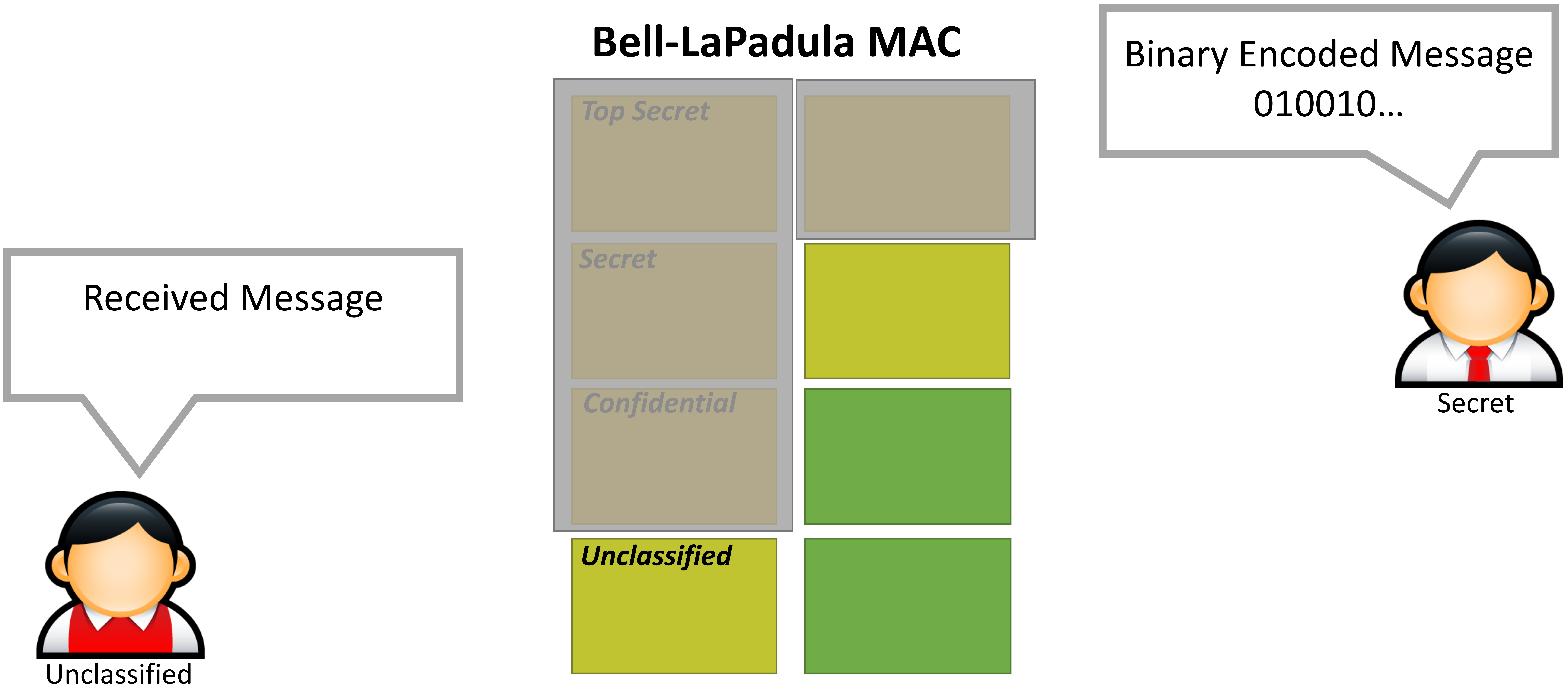
Simple Example



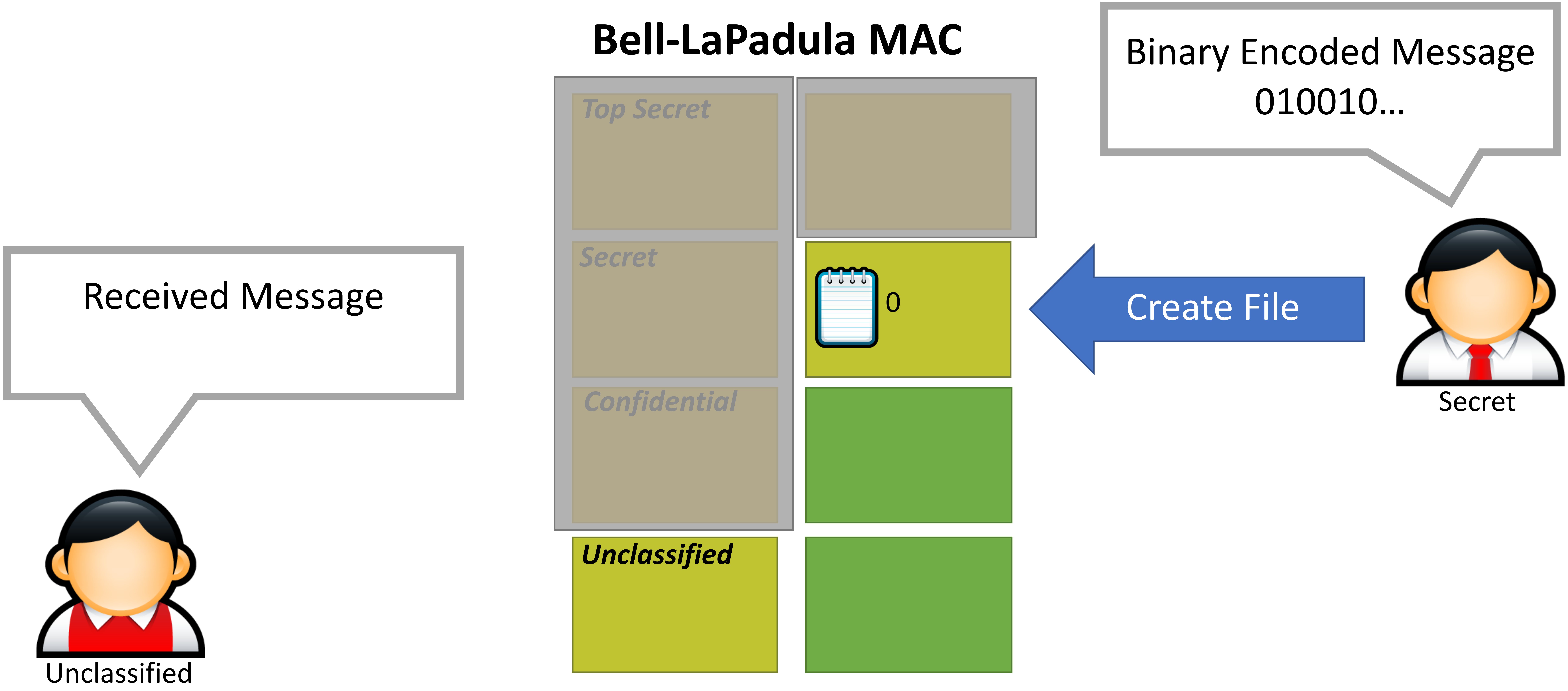
Exploiting a Covert Channel



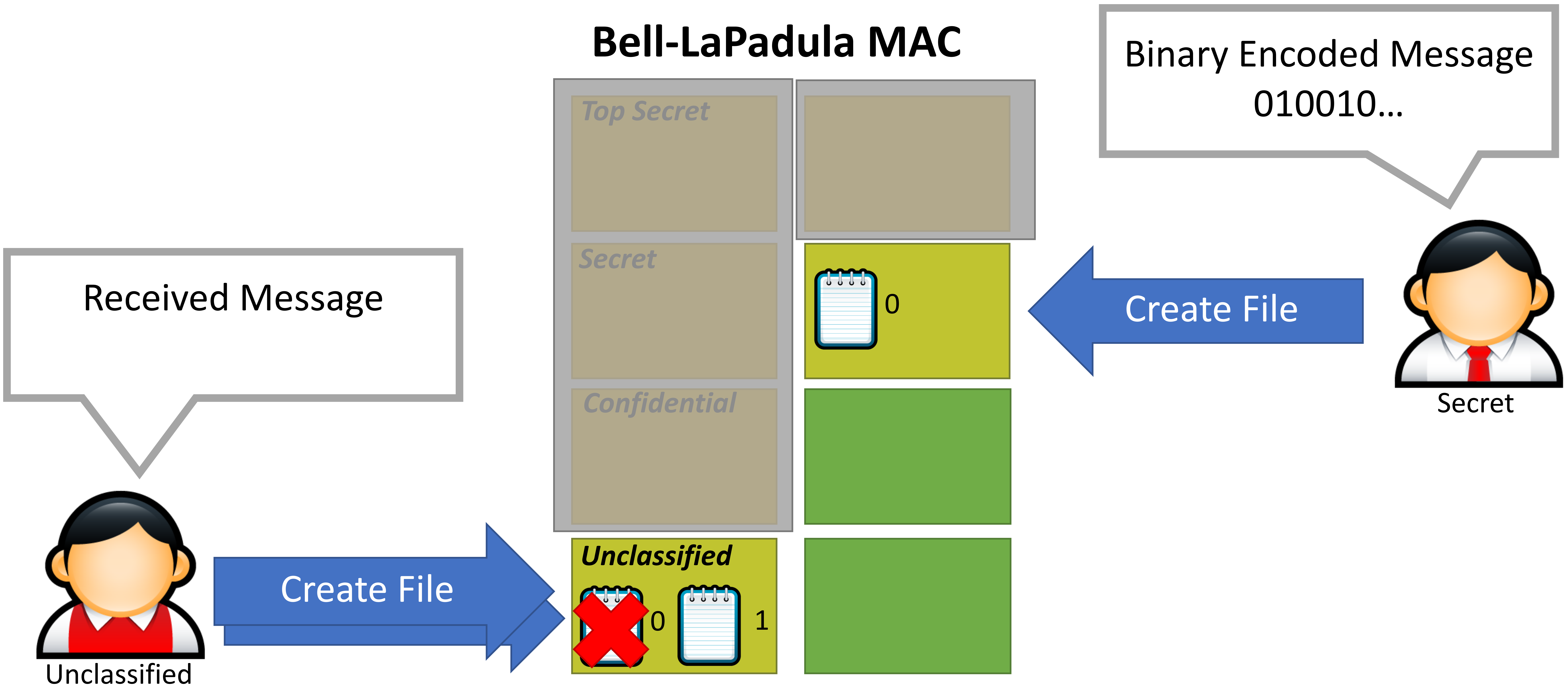
Exploiting a Covert Channel



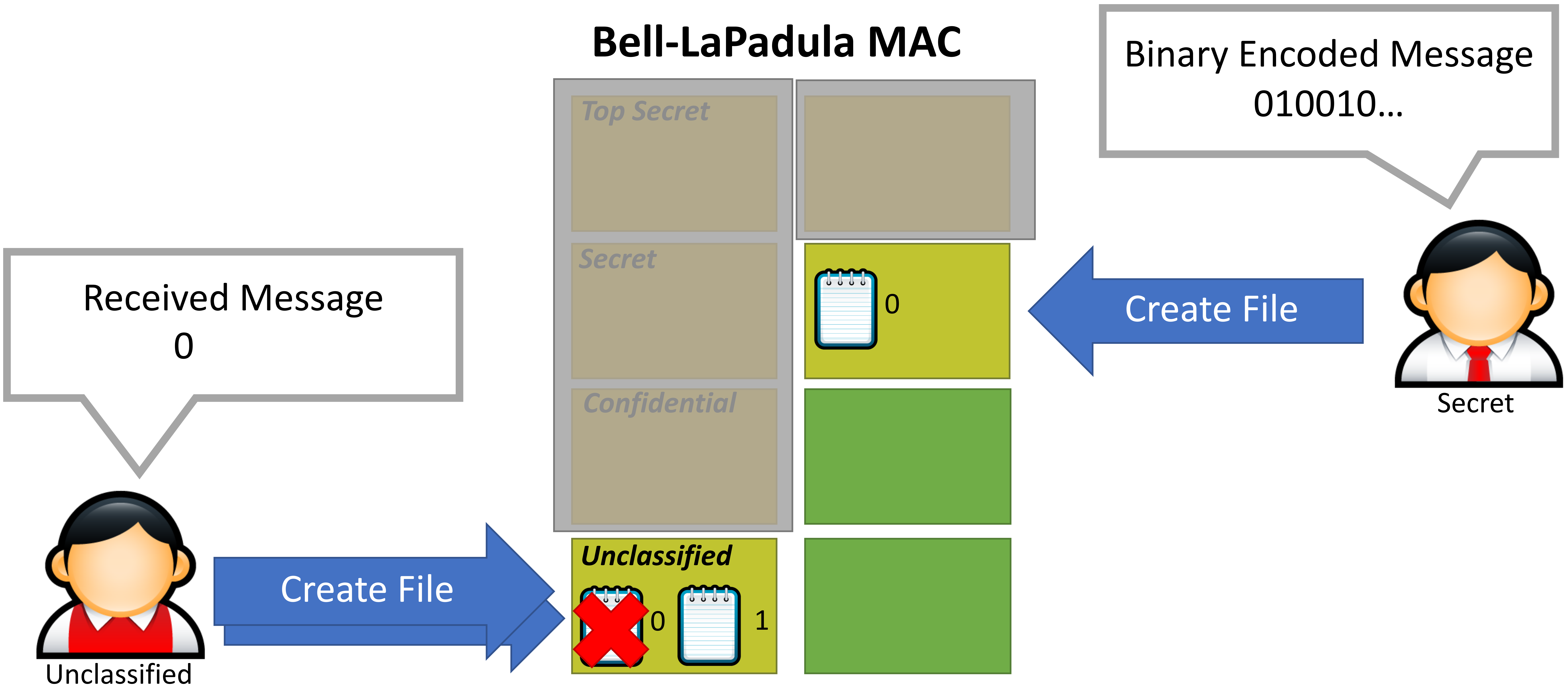
Exploiting a Covert Channel



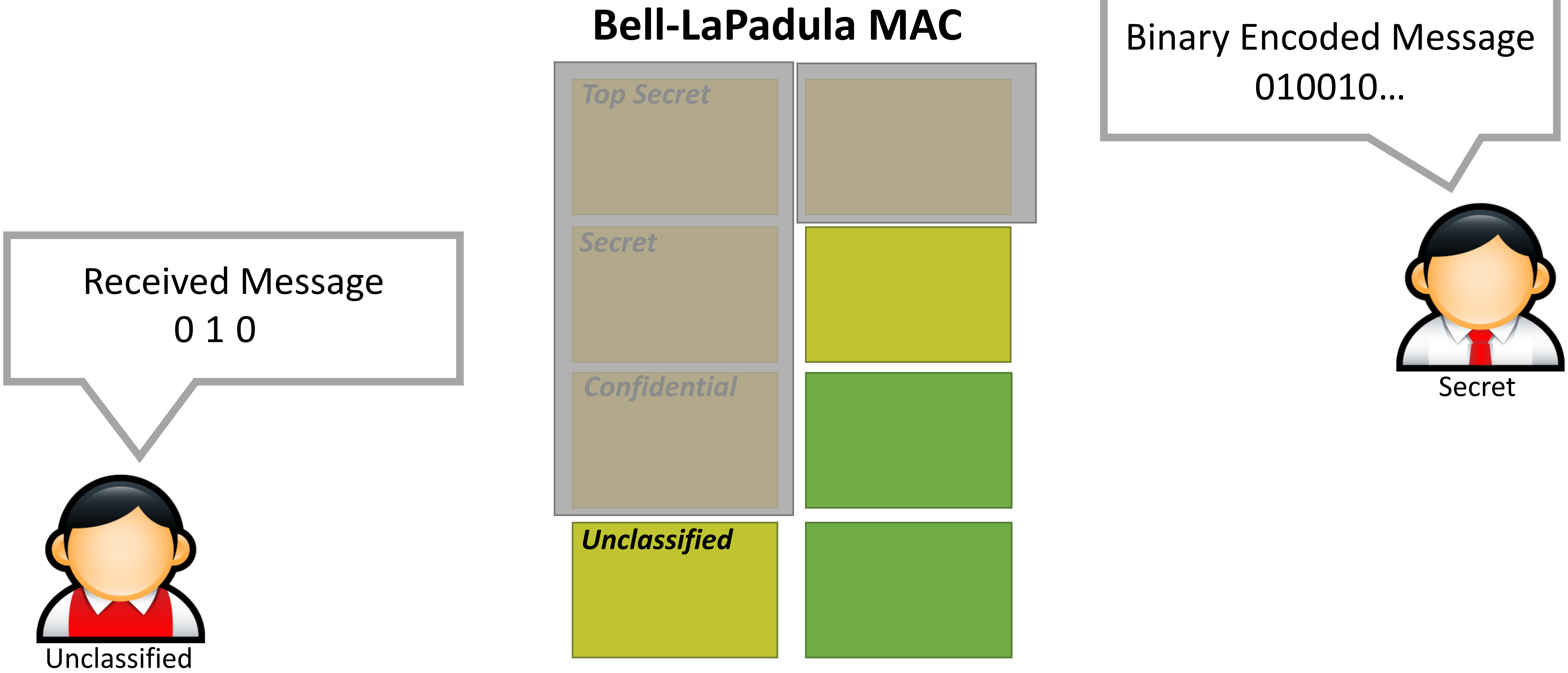
Exploiting a Covert Channel



Exploiting a Covert Channel



Exploiting a Covert Channel



Leveraging Covert Channels

- Covert channels are typically noisy
  - Based on precise timing of events
  - May result in encoding errors, i.e. errors in data transmission
  - Communication is probabilistic
- Information theory and coding theory can be applied to make covert channels more robust
  - Naïve approach: duplicate the data *n* times
  - Better approach: uses Forward Error Correction (FEC) coding
  - Zany approach: use Erasure Coding

Bell-LaPadula and Covert Channels

- Covert channels are not blocked by the ★-property
- It is very hard, perhaps impossible, to block all covert channels
  - May appear in program code
  - Or operating system code
  - Or in the hardware itself (e.g. CPU covert channels)

Bell-LaPadula and Covert Channels

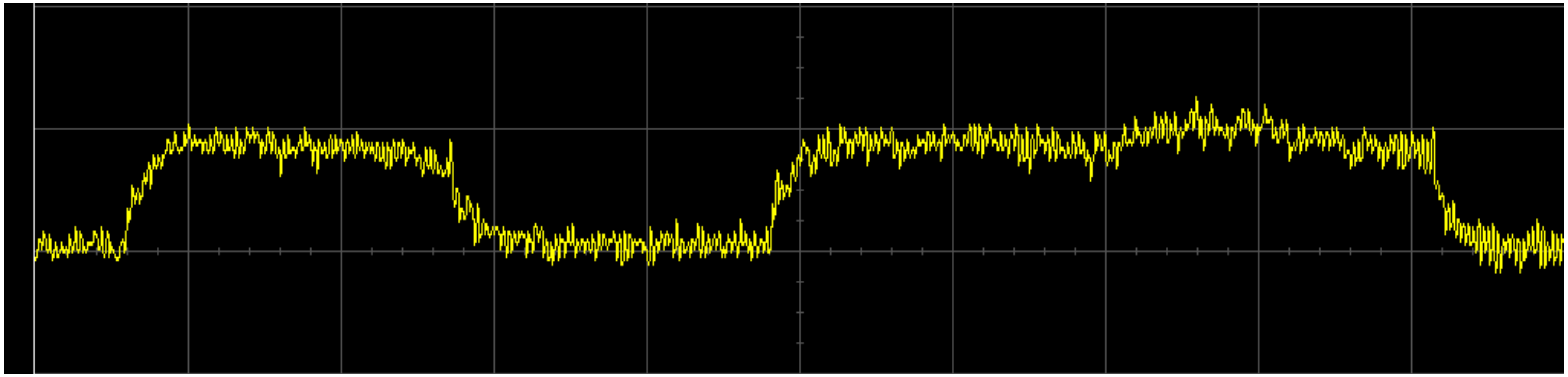
- Covert channels are not blocked by the ★-property
- It is very hard, perhaps impossible, to block all covert channels
  - May appear in program code
  - Or operating system code
  - Or in the hardware itself (e.g. CPU covert channels)
- Potential mitigations:
  - Limit the bandwidth of covert channels by enforcing rate limits
    - Warning: may negatively impact system performance
  - Intentionally make channels noisier by using randomness to introduce "chaff"
    - Warning: slows down attacks, but may not stop them
  - Use anomaly detection to identify subjects using a covert channel
    - Warning: may result in false positives
    - Warning: no guarantee this will detect all covert channels

Side Channel Attacks

- Side channels result from inadvertent information leakage
  - Timing – e.g., password recovery by timing keystrokes
  - Power – e.g., crypto key recovery by power fluctuations
  - RF emissions – e.g., video signal recovery from video cable EM leakage
- Virtually any shared resource can be used

Side Channel Attack Example

- Victim is decrypting RSA data
  - Key is not known to the attacker
  - Encryption process is not directly accessible to the attacker
- Attacker is logged on to the same machine as the victim
  - Secret key can be deciphered by observing the CPU voltage
  - Short peaks = no multiplication (0 bit), long peaks = multiplication (1 bit)



Real Side Channel Attacks

- CPU voltage attacks against RSA
- Keystroke timing attacks against SSH
- Timing and CPU cache attacks against AES
- RF radiation attacks against computer monitors!
  - Attacker can observe what is on your screen
- CPU cache attacks against process isolation
  - Meltdown and Spectre
  - Also leverage a covert channel ;)