Διάλεξη #9 - Security Fundamentals

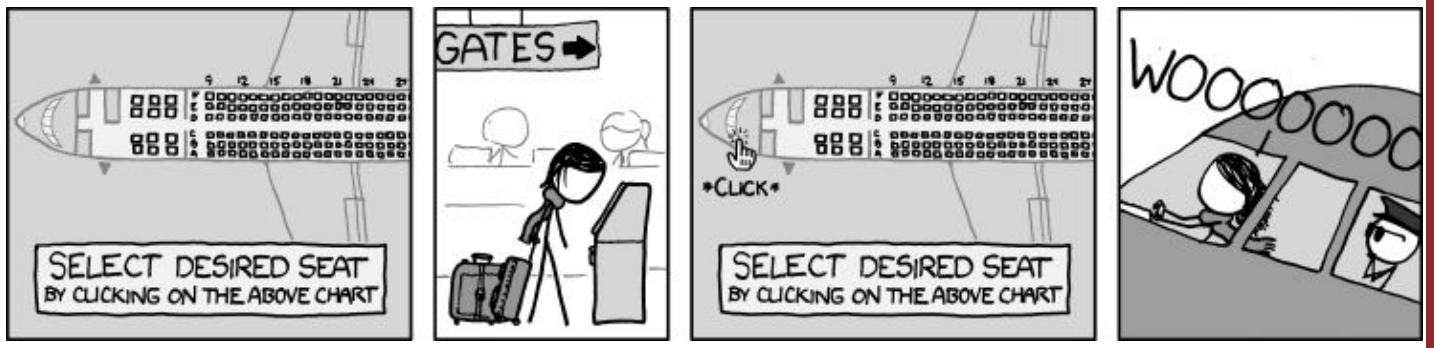
Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

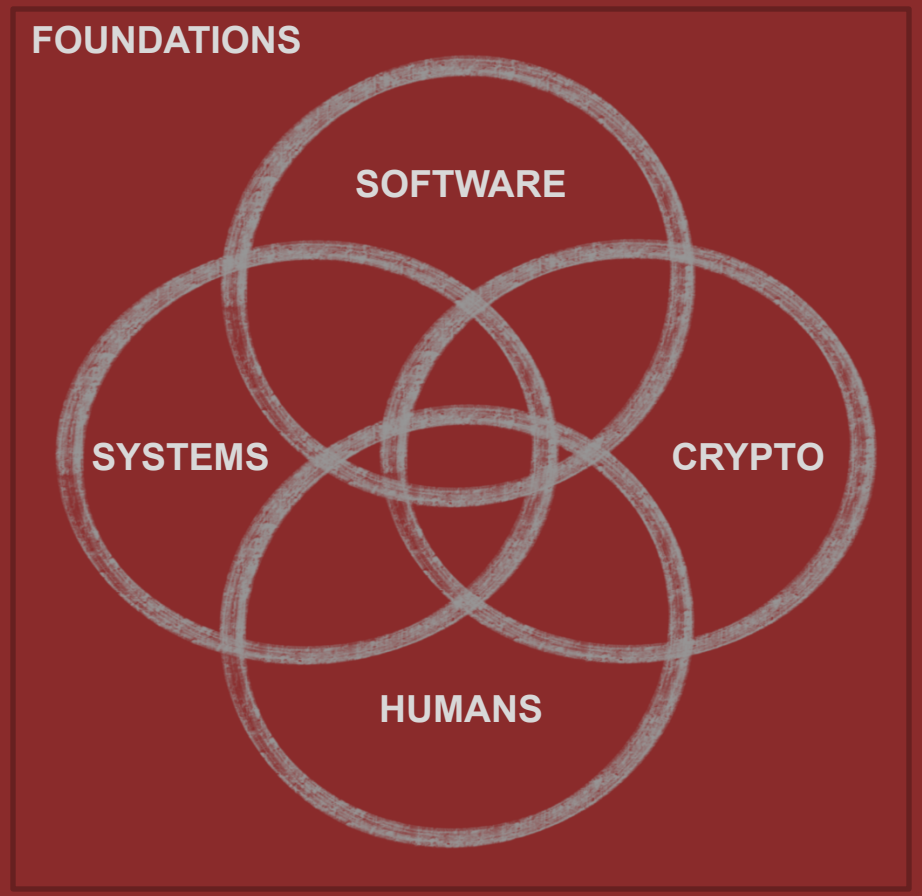
Εισαγωγή στην Ασφάλεια

Θανάσης Αυγερινός

(Security Mindset)

https://xkcd.com/726/





Huge thank you to David Brumley from Carnegie Mellon University for the guidance and content input while developing this class

Ανακοινώσεις / Διευκρινίσεις

- Η εργασία #1 κλείνει στις 24 του μήνα - δεν θα έχουμε παράταση!
  - μην ξεχάσετε το write up!

Ερωτήσεις:

- Δυσκολεύομαι να διαβάσω assembly, τι να κάνω;
  - Χρησιμοποίησε εργαλεία που μπορεί να σε βοηθήσουν. π.χ.
    - https://cloud.binary.ninja/
    - https://dogbolt.org/

Την Προηγούμενη Φορά

1. Bypassing Mitigations
2. Return-Oriented Programming

Σήμερα

- Security Fundamentals
  - Adversaries
  - Threat Models
  - Security Properties
  - Trusted Computing Base (TCB)
  - Security Principles

Adversaries

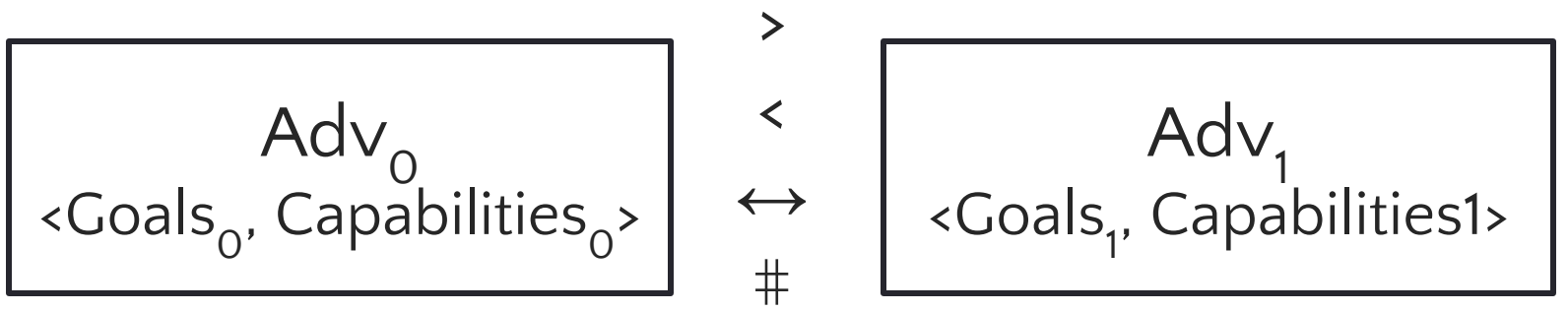
Reminder: Adversary Definition

- Adversary = < Goals, Capabilities >
- Goal: What constitutes success?
  - May involve subgoals
  - Example goal: Gain control of X's data
    - Sub-goal: Reconnaissance: search online for info about X
      - Sub-goal: Access: Guess X's ssh password on Linux lab
        - » Sub-goal: Lateral movement: Use ssh account to move to other services / linked accounts
- Capabilities: What resources can the adversary use?
  - 1 computer or millions?
  - Physical or remote access?
  - Access to source code?

Why don't we include adversary's strategy?

Basic Adversary Metrics

Partial Orders



Implication (A0 → A1):
A0's goals and capabilities are a superset of A1's

Separation (A1 ↛ A0):
A1's goals and/or capabilities are not a superset of A0's

Strict Dominance A0 > A1 = A0 → A1 ∧ A1 ↛ A0
Equivalence: A0 ↔ A1 = A0 → A1 ∧ A1 → A0
Incomparable: A0 # A1 = A0 ↛ A1 ∧ A1 ↛ A0

Example Adversary Comparison

Adversary A0

- Goal: Modify the my-studies website
- Capabilities
  - View the website
  - Send data to the website
  - Modify local browser state
  - Access a normal user account on my-studies
  - Invoke system calls on `my-studies.uoa.gr`

Adversary A1

- Goal: Modify the my-studies website
- Capabilities
  - View the website
  - Send data to the website
  - Modify local browser state
  - ~~Access a normal user account on my-studies~~
  - ~~Invoke system calls on my-studies.uoa.gr~~

$A0 \rightarrow A1 \wedge A1 \not\rightarrow A0$ hence A0 > A1, i.e., A0 strictly dominates A1

Security Mechanism Classification for a property

1. **Prevention**. Prevent issues from happening. Any precautionary measures.
2. **Detection**. Assuming an incident took place, detect them as early and as accurately as possible.
3. **Resilience**. Assuming one or multiple incidents took place, ensure the overall system security degrades gracefully and does not collapse.
4. **Deterrence**. Measures to ensure penalties for actors responsible for security incidents. Policy-based.

Threat Models

Threat Modeling

Threat modeling is a process by which potential threats, such as structural vulnerabilities or the absence of appropriate safeguards, can be identified and enumerated, and countermeasures prioritized.

https://en.wikipedia.org/wiki/Threat_model

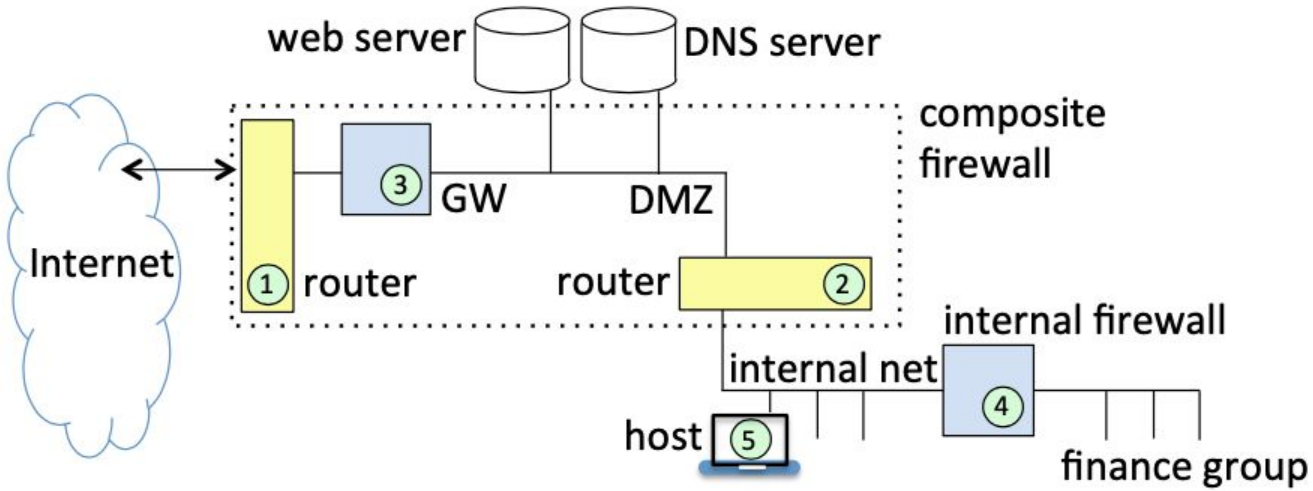
https://www.threatmodelingmanifesto.org/

Threat Model Includes

- Assets
  - What are you protecting?
  - Which matter most?
- System goals (functionality, security)
- Adversary definition – key characteristic
  - Risk assessment – we are in the insurance business!
    - Risk justifies the cost

Systematic Threat Modeling

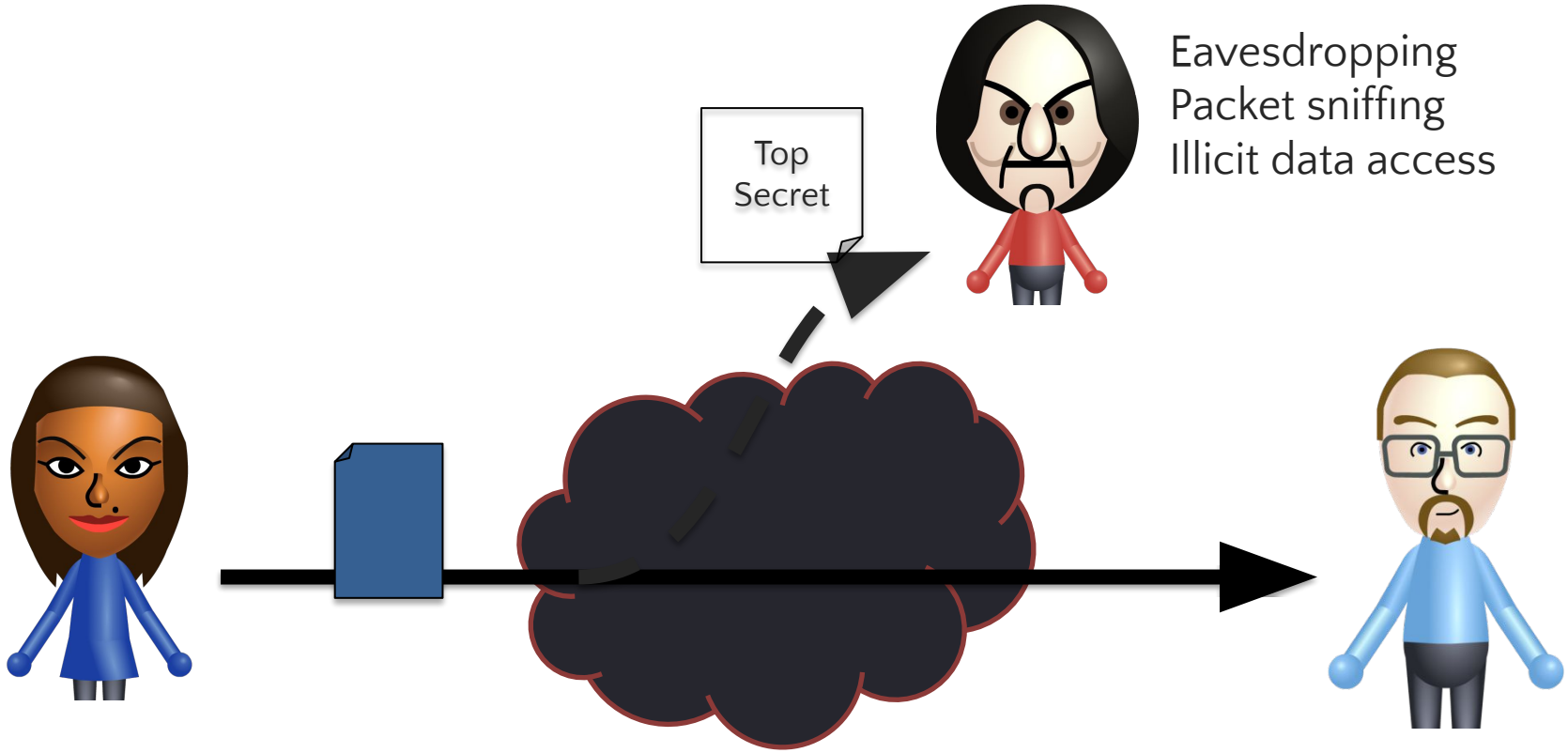
- Diagram-based
- Attack trees
- Checklists
- STRIDE
- MITRE ATT&CK



4 Key Security Properties

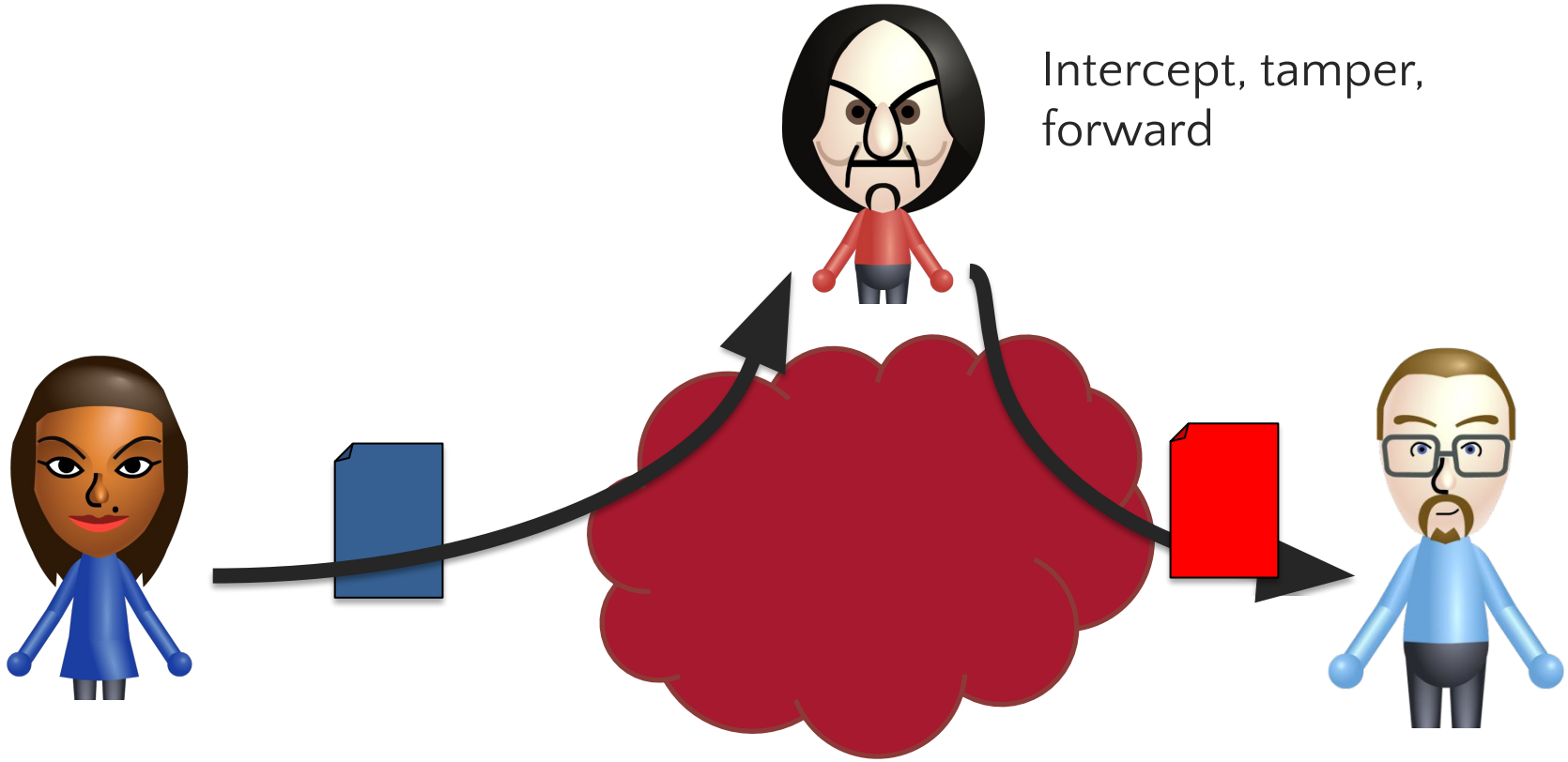
What Properties Do You Care About?

(1) Confidentiality == **Secrecy** == Concealing information



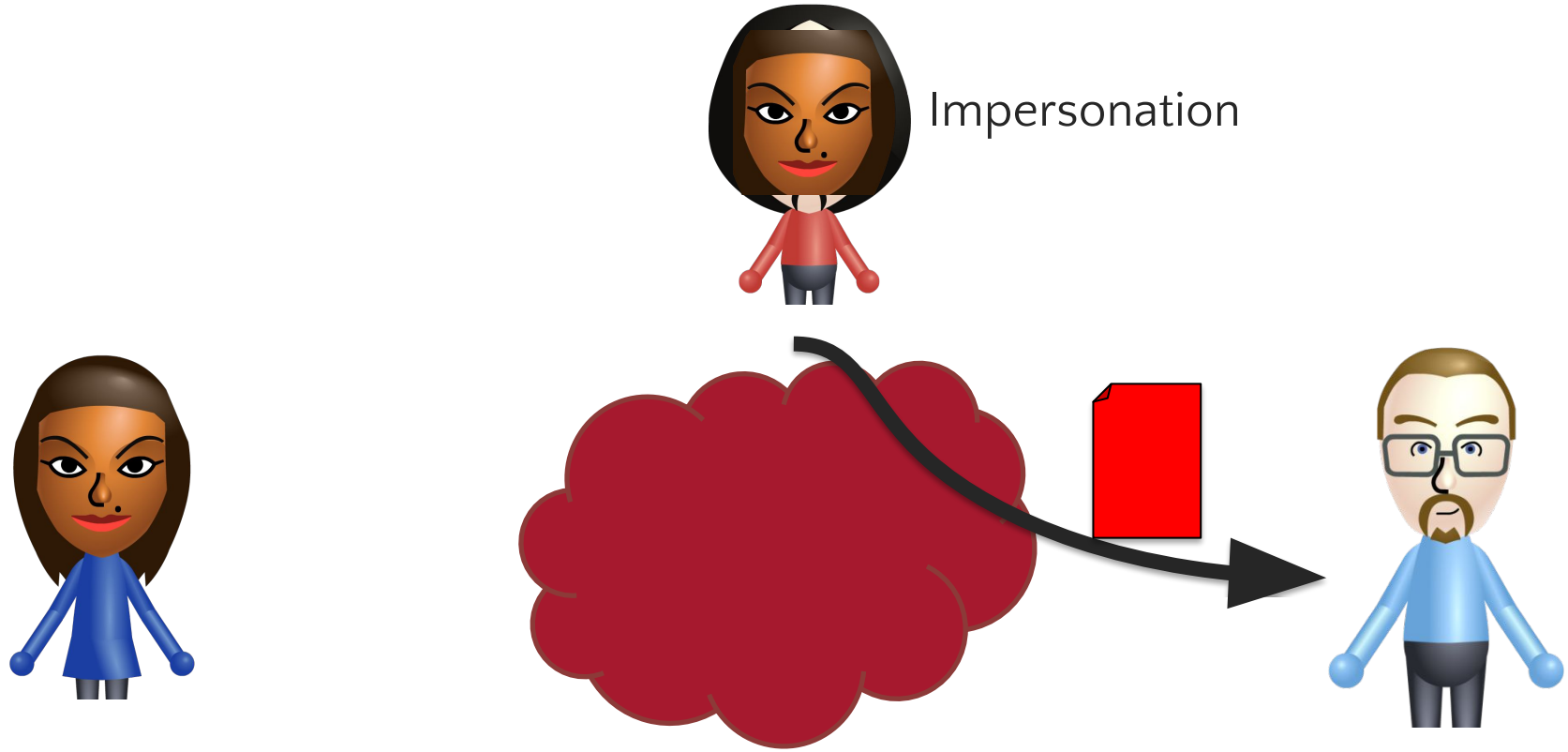
What Properties Do You Care About?

(2) Integrity == Prevention of unauthorized changes



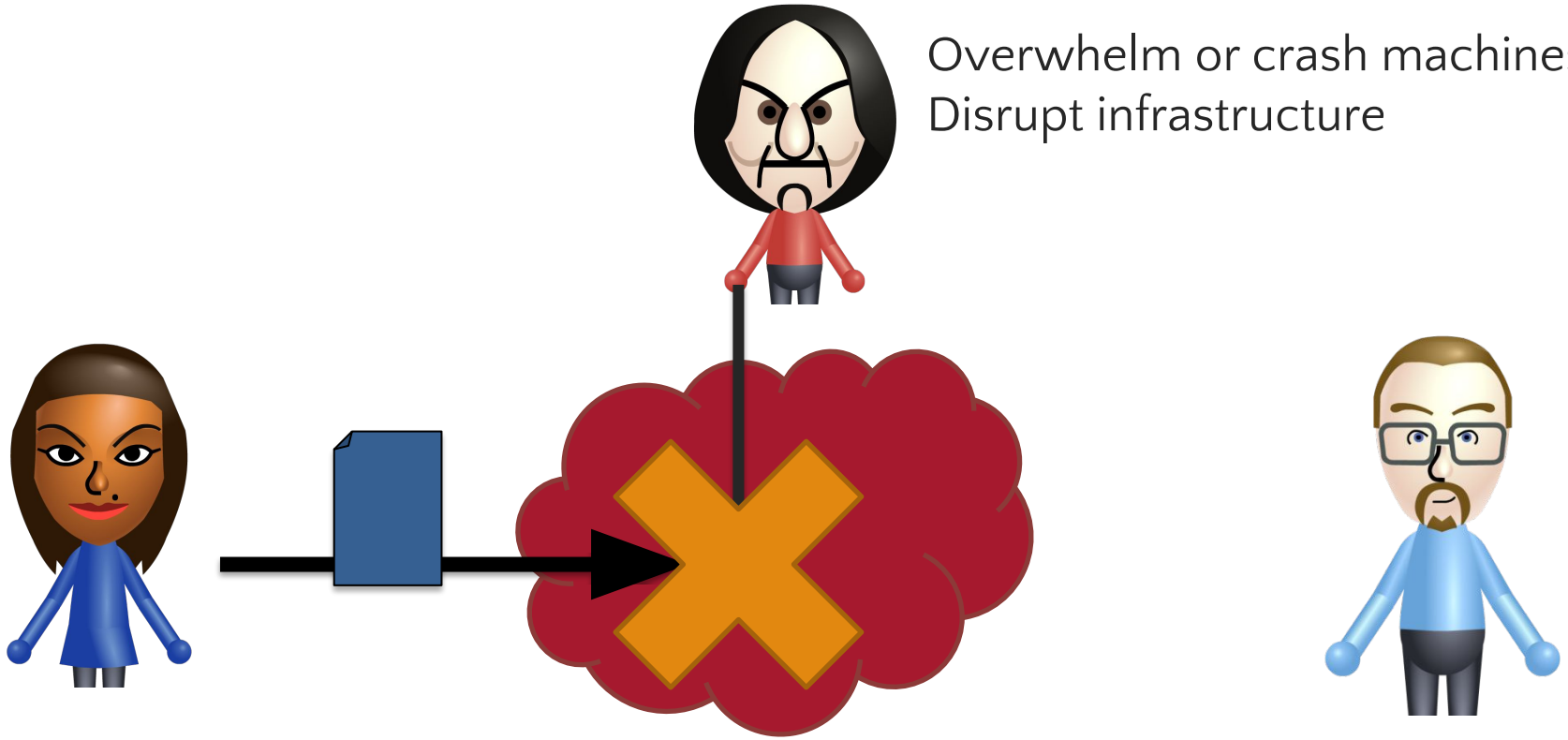
What Properties Do You Care About?

(3) Authenticity == Data and actions attributed to correct person



What Properties Do You Care About?

(4) Availability == Ability to use resources when needed

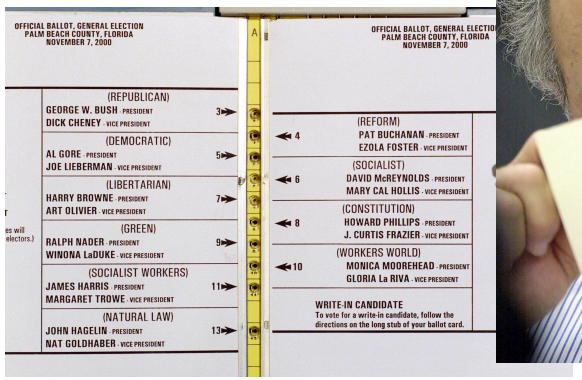


Overwhelm or crash machines,
Disrupt infrastructure

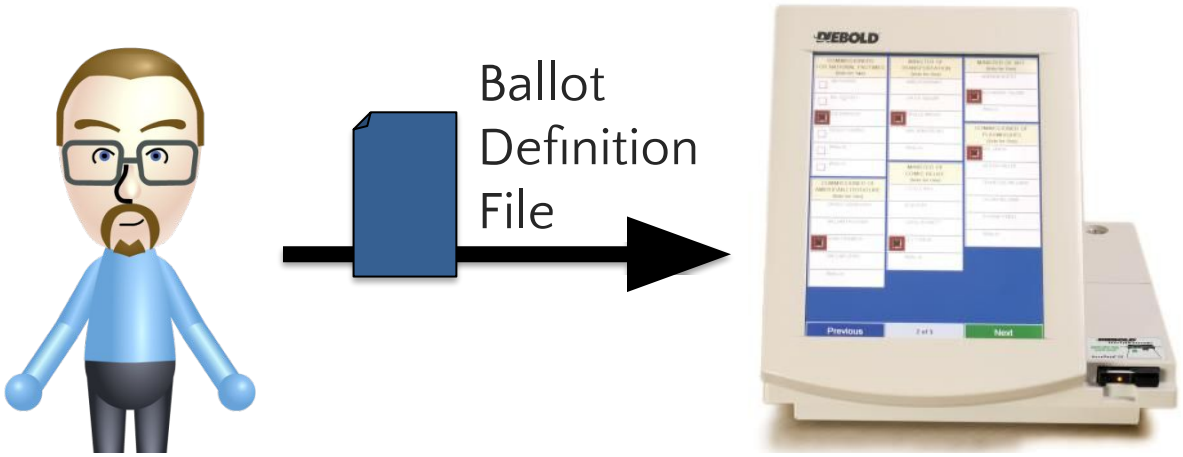
Example: Threat Modeling on E-Voting

Security of E-Voting

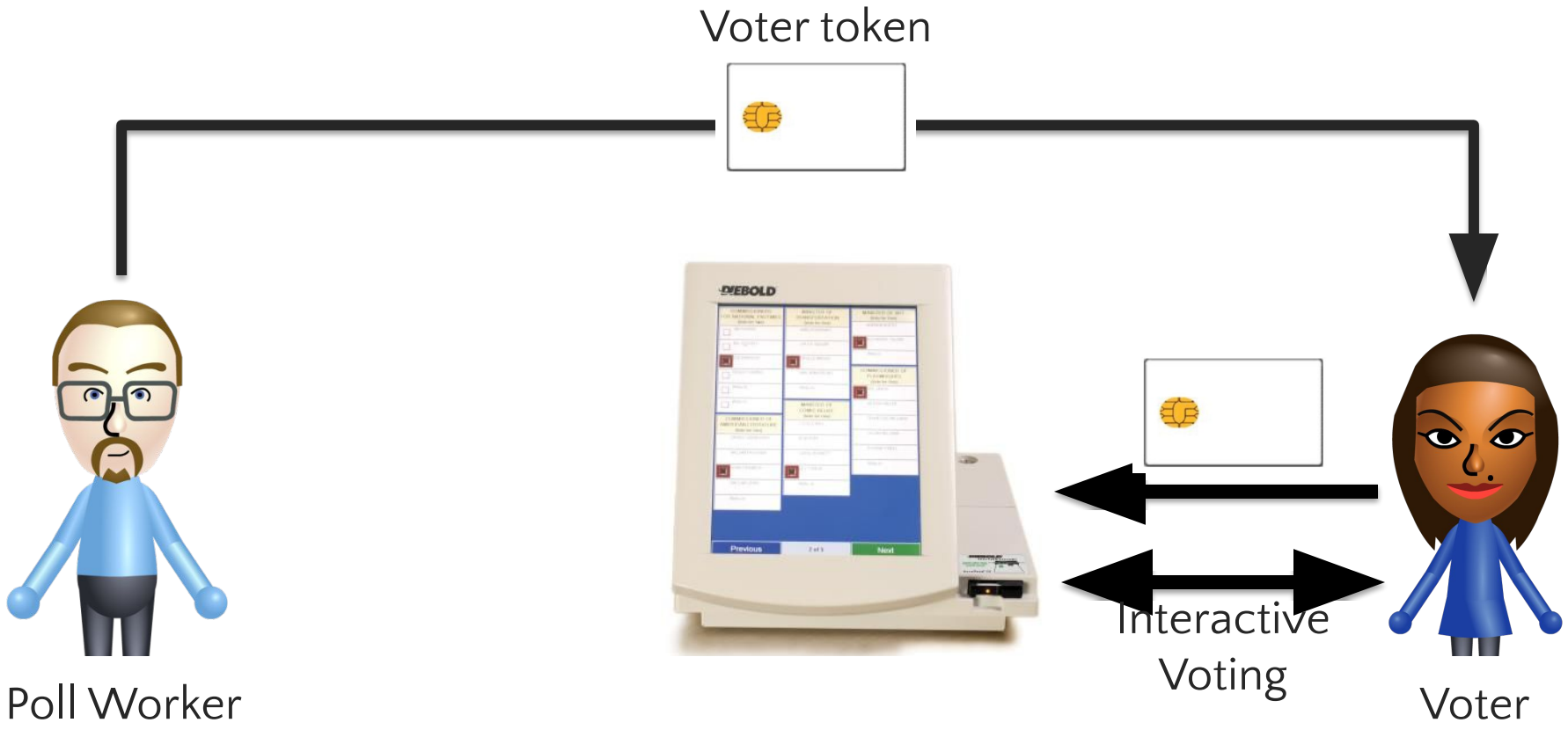
Popular replacement for paper ballots



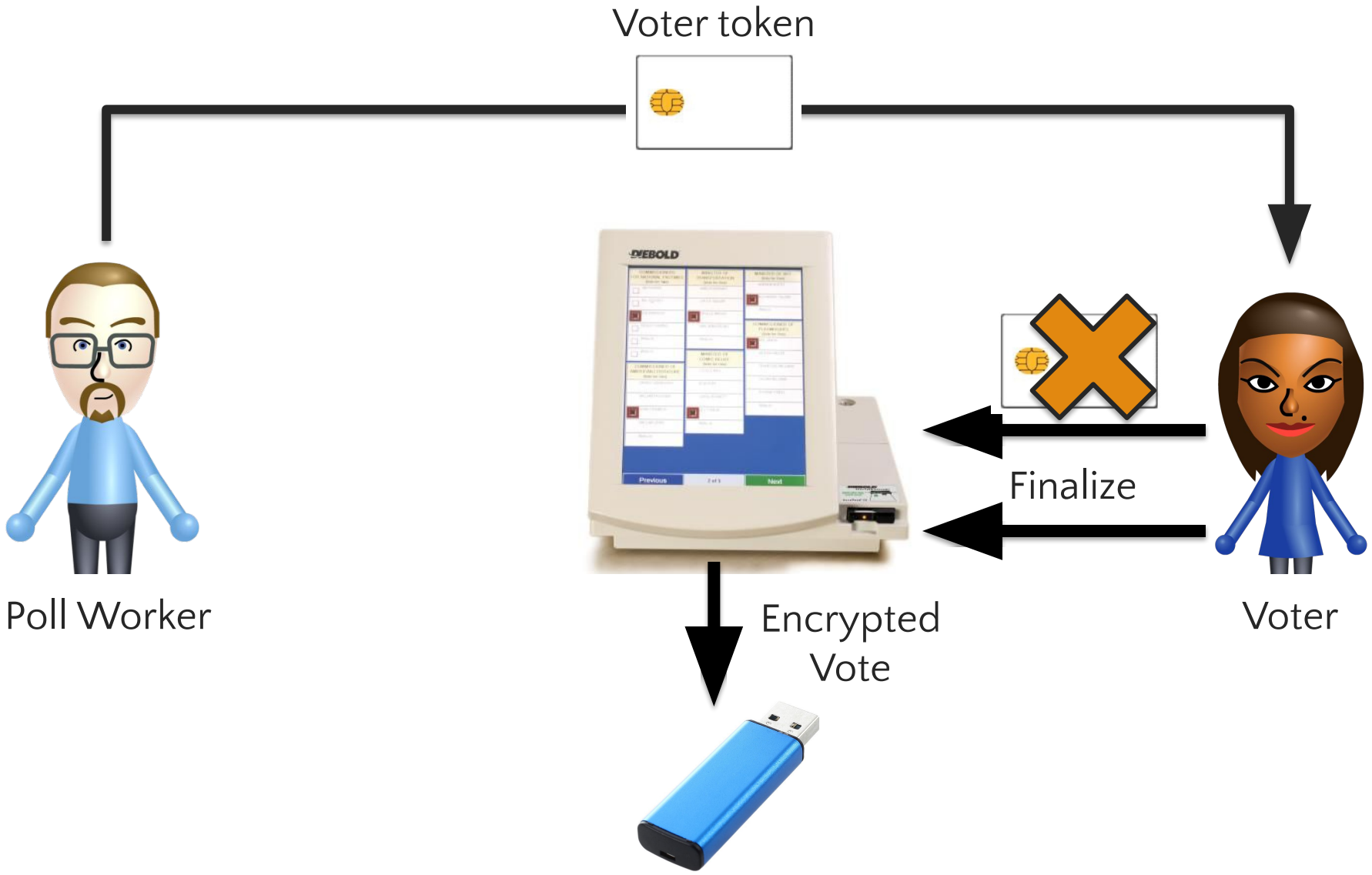
Pre-Election Setup



Active Voting

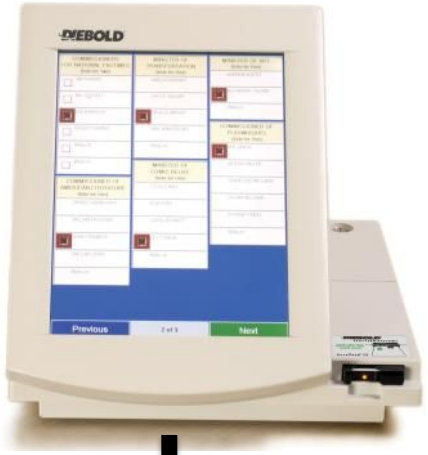


Finalize Ballot

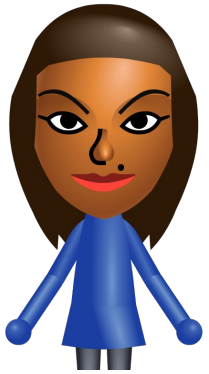


Post-Election Tabulation

Poll Worker

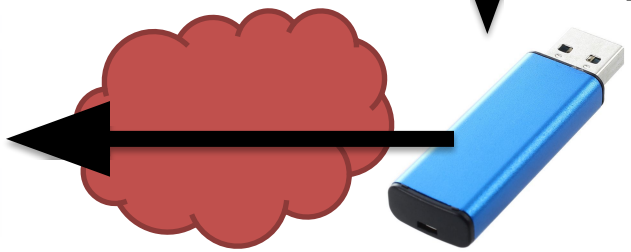


Encrypted Vote



Voter

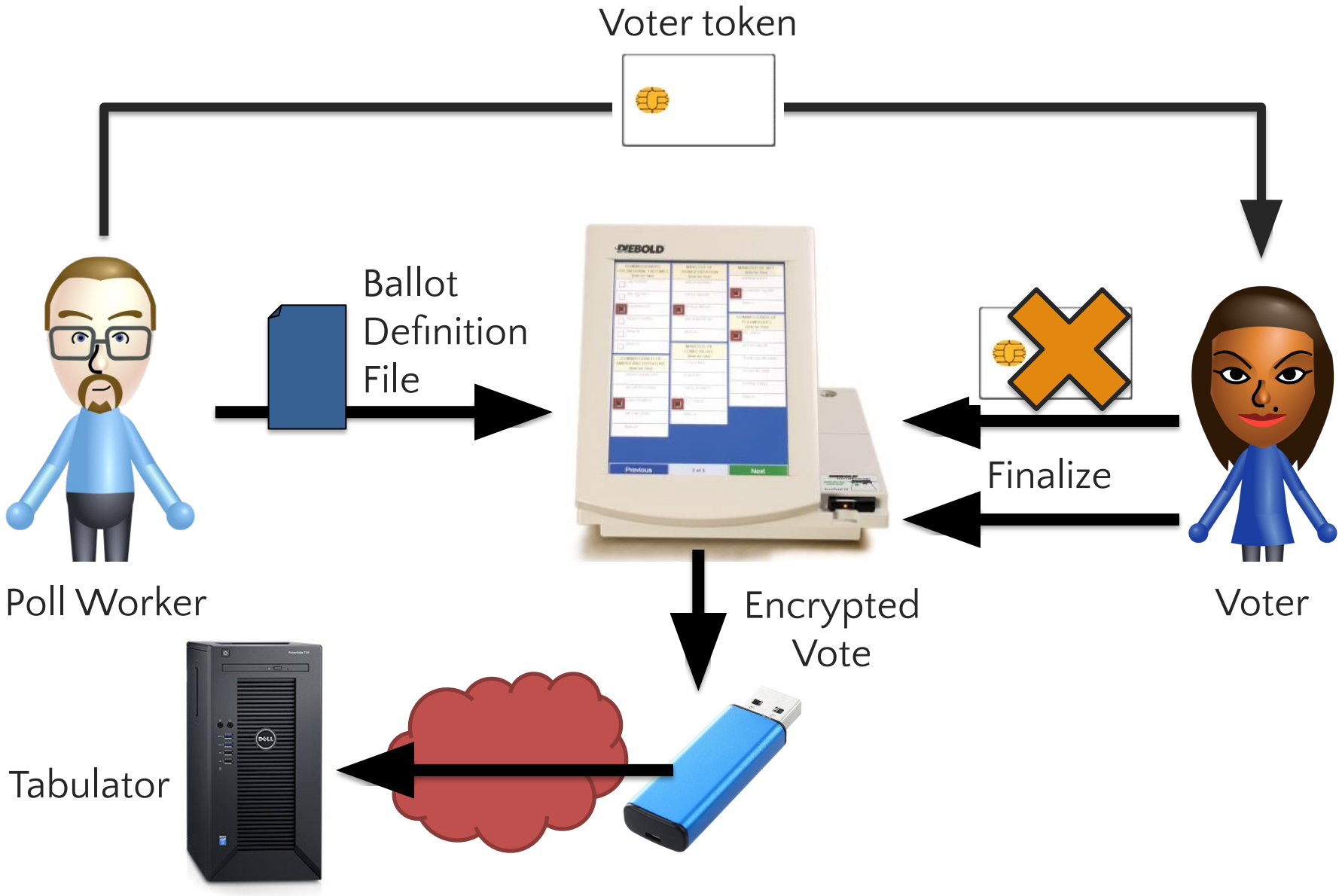
Tabulator



Threat Model for E-Voting

- Assets: Votes
- System goals (properties we want to achieve even when under attack)
  - Functionality
    - Easy to use
    - Produces correct results
    - Produces results promptly
  - Security
    - Votes must remain private (confidentiality)
    - Adversary must not tamper with election outcome
      - By changing votes (integrity)
      - By voting on behalf of others (authenticity)
      - By denying voters the right to vote (availability)

Who Are Potential Adversaries?



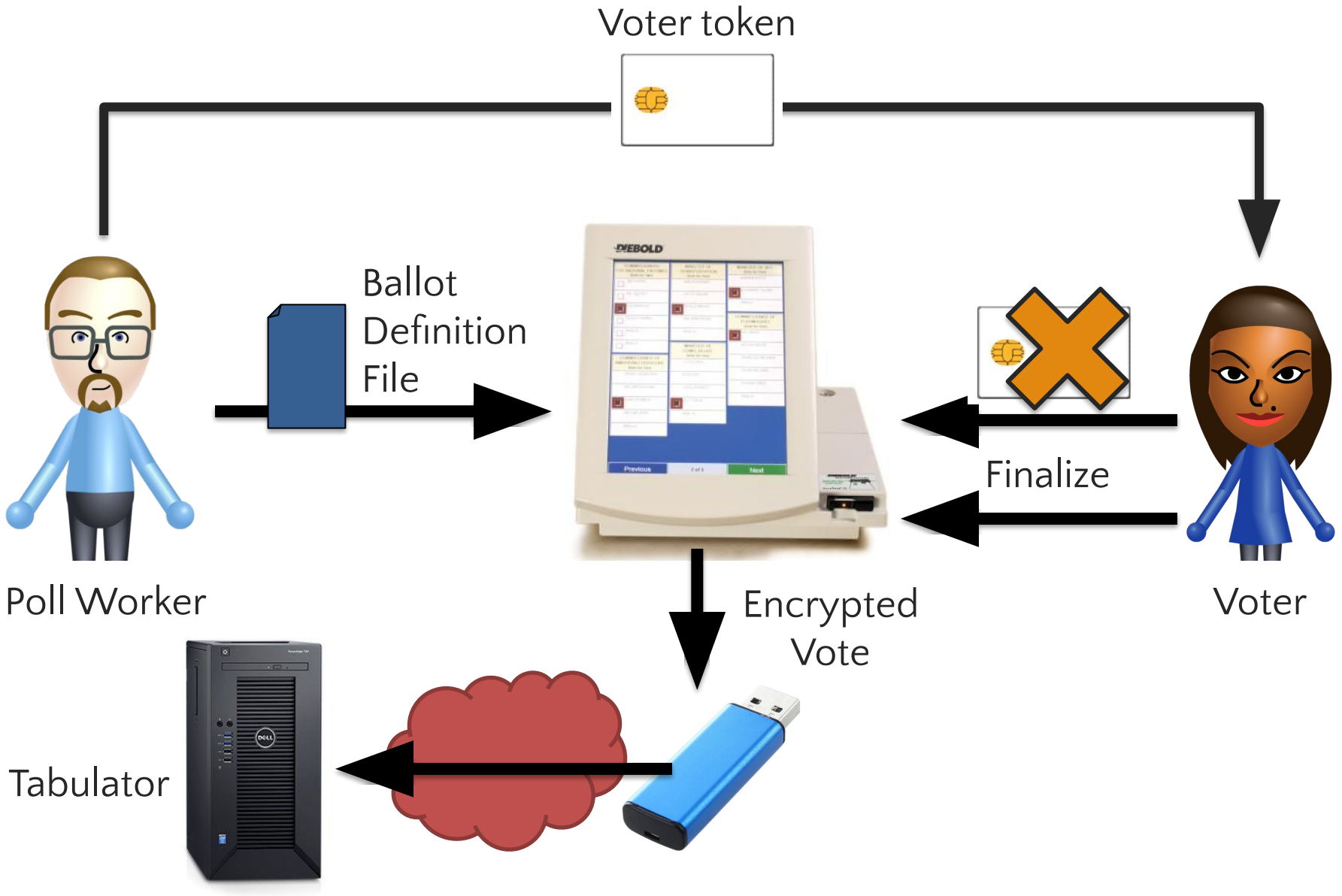
Potential Adversaries

- Voters
- Election officials
- Employees of voting machine manufacturer
  - Software/hardware engineers
  - Maintenance people
- Other engineers
  - Hardware manufacturers
  - Software developers (voting app, OS, compiler)
- Network operators
- Nation states?
- …
- Or any combination of the above

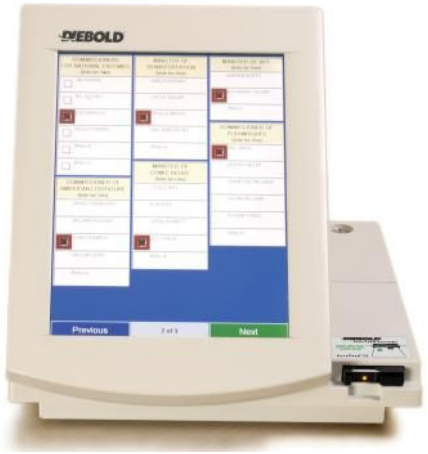
Example Adversary

- Voter
  - Goal: Vote more than once
  - Capabilities:
    - Wear disguises
    - Bring (small) commodity hardware into voting booth
    - (Brief) physical access to the voting machine and token
    - …

What Security Risks Do You See?



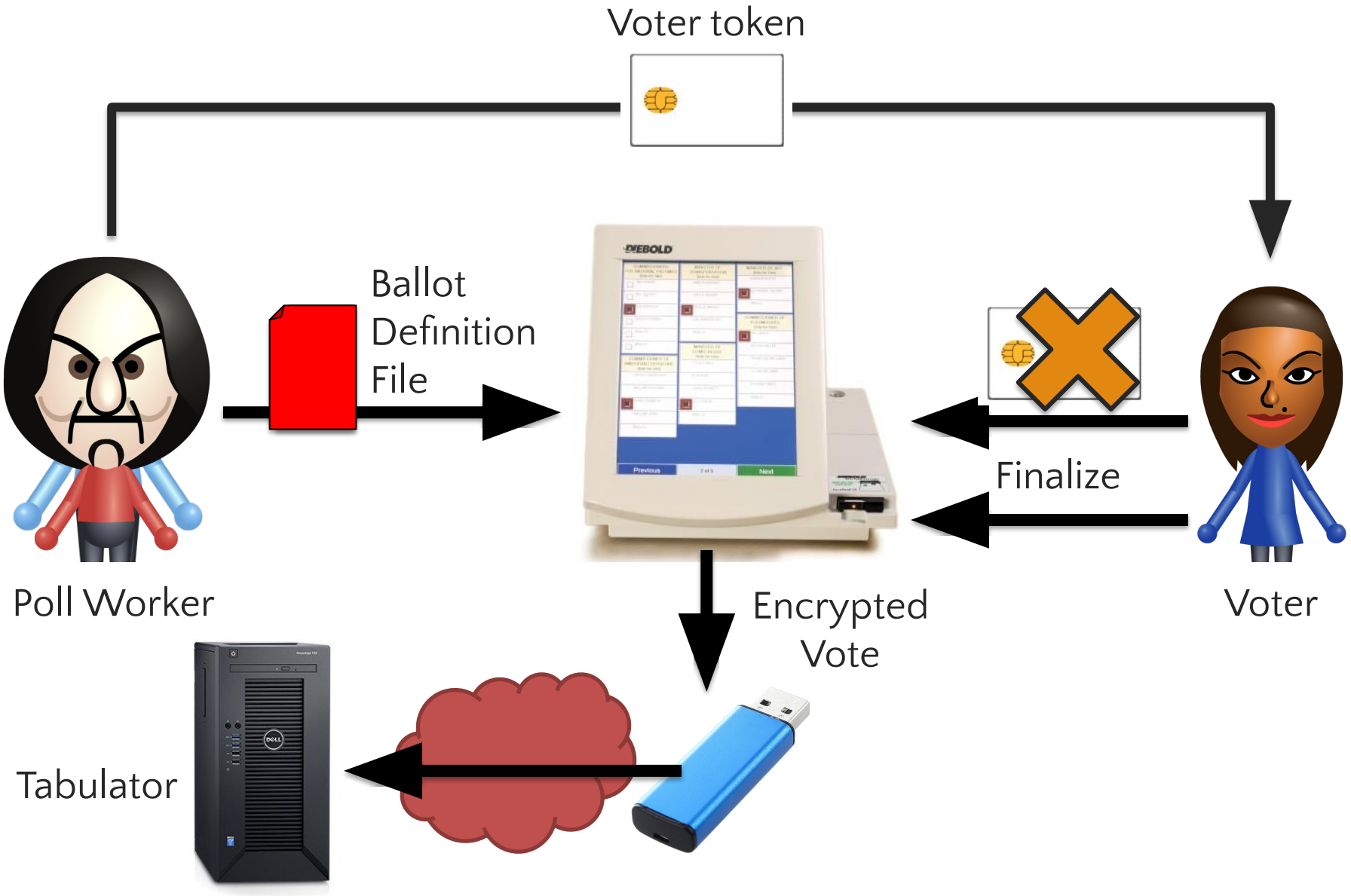
What Does This Machine Do?



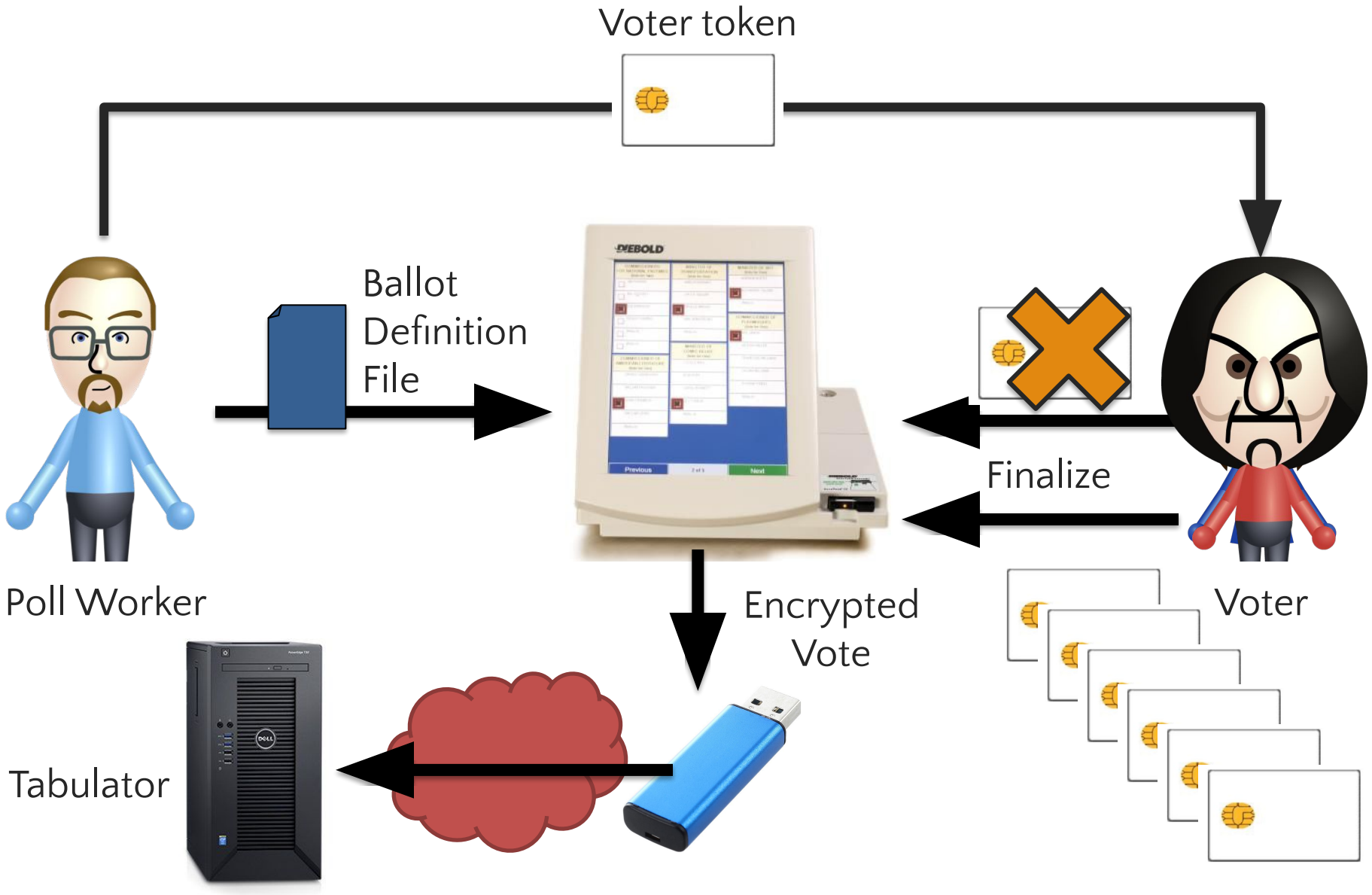
Problem:
An adversary who controls the software or hardware controls the entire voting process

But the machines are locked, right?

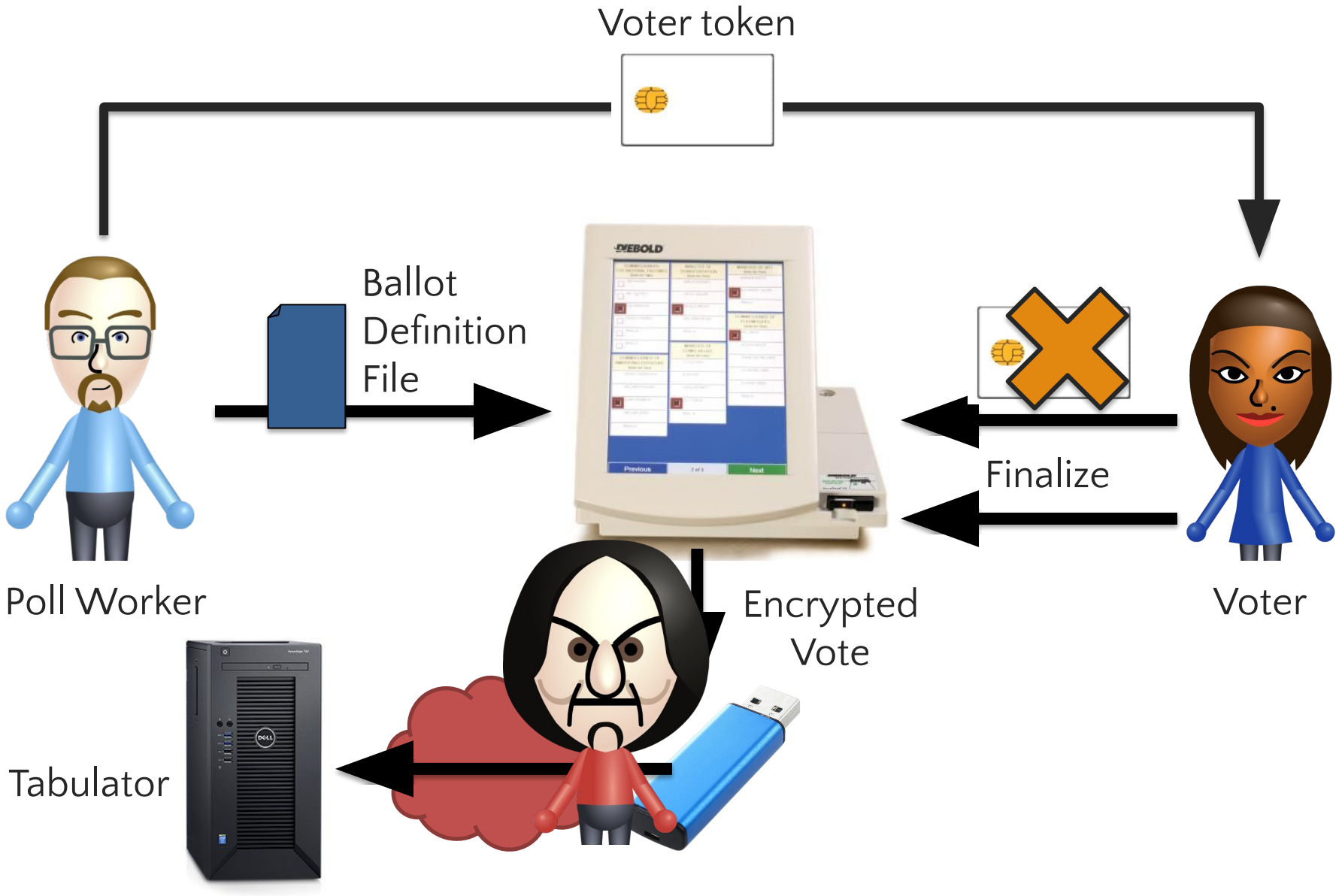
Problem:
Ballot definition files are *not authenticated*



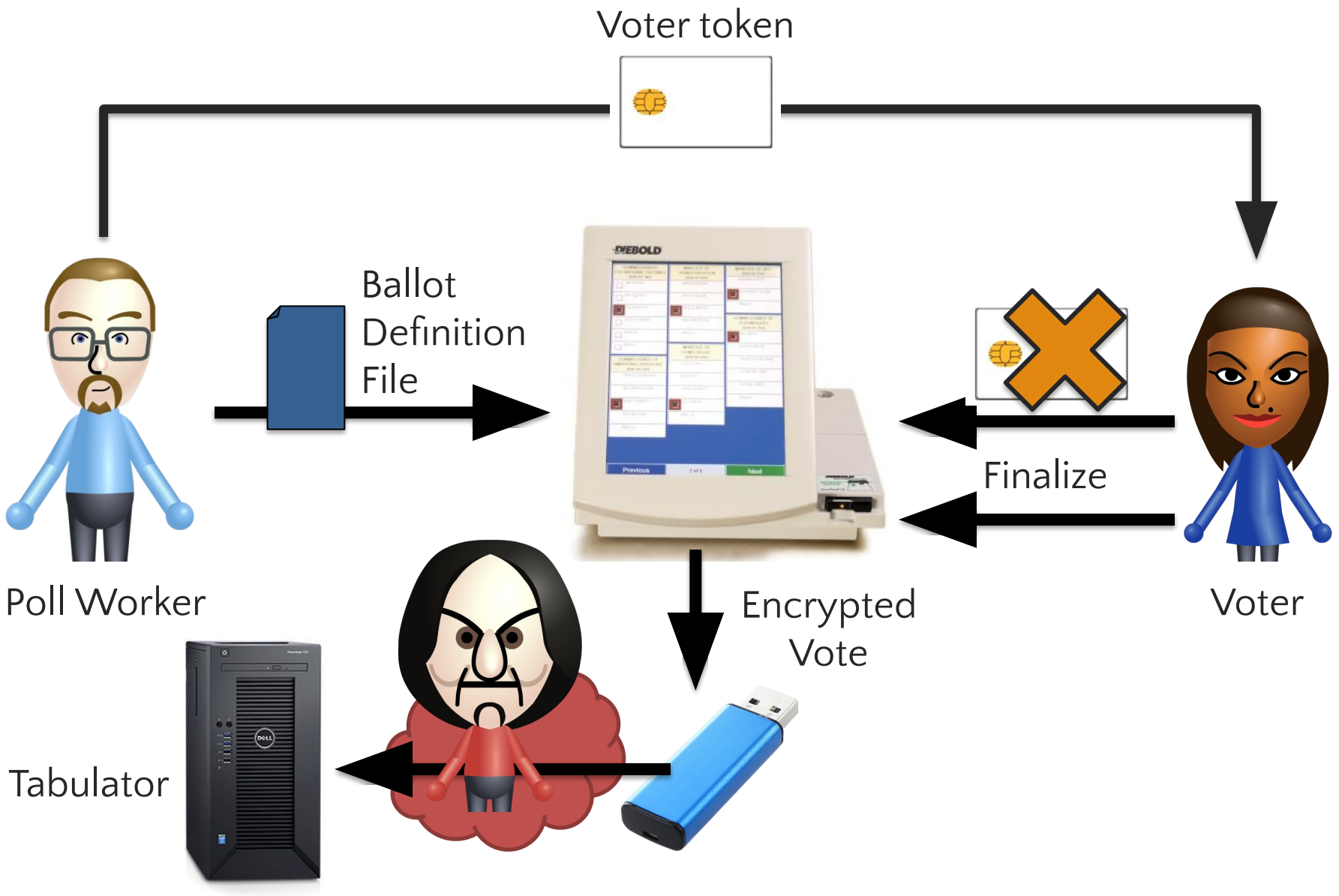
Problem:
Voting machine does *not authenticate* the voter token



Problem:
Encryption key (`F2644hD4`) *hard-coded* into software since (at least) 1998. Votes stored in order cast.



Problem:
Votes *decrypted* before transmission to tabulator



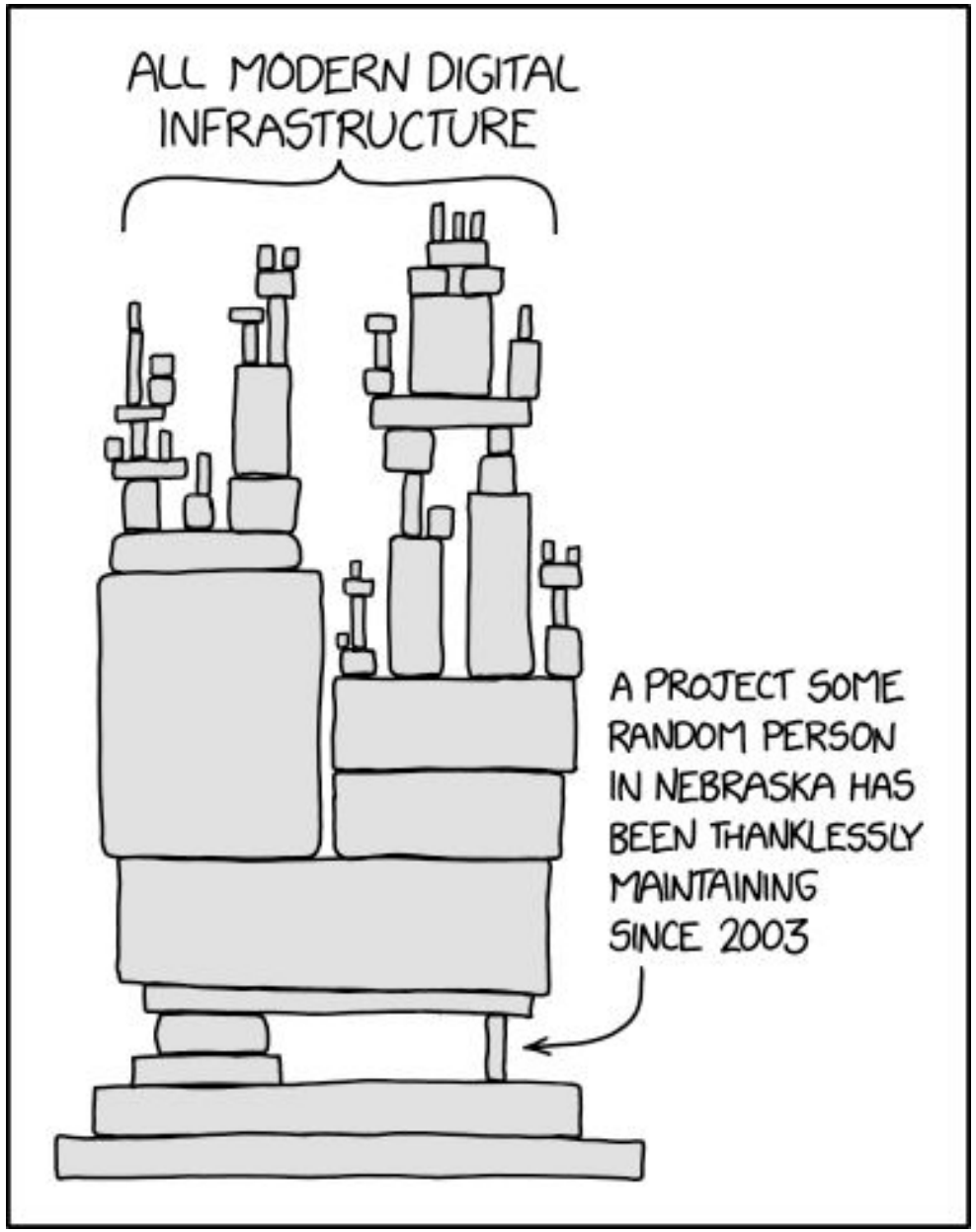
Conclusion: we're gonna be here for a while until we solve security :)

Participation Question

Which of the following is ***TRUE*** about a threat model?

A. It includes system assets, goals, and an adversary definition
B. It focuses on the most important part of the system
C. It must be expressed mathematically
D. It enumerates an attacker's possible strategies

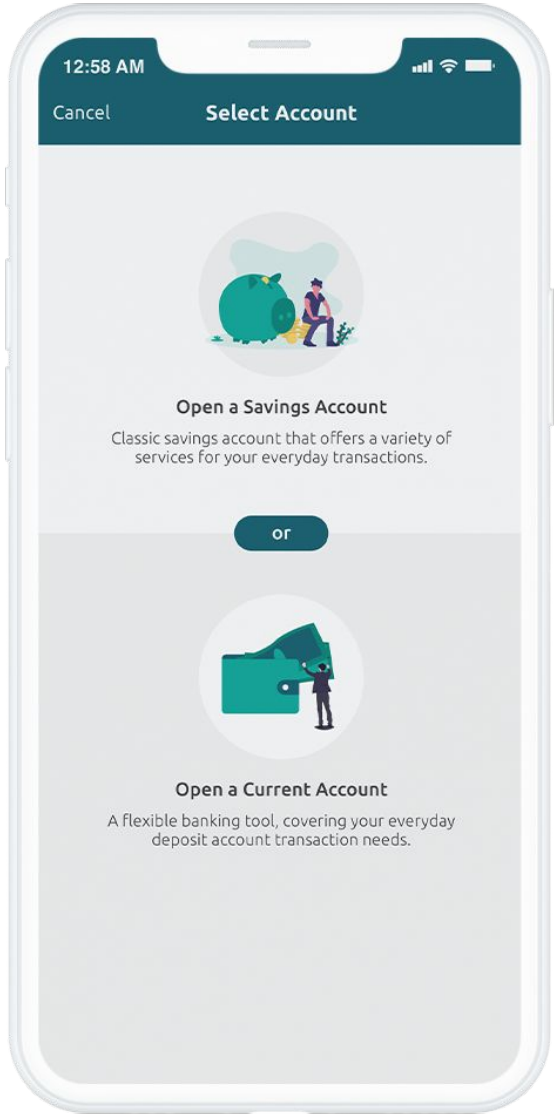
Trusted Computing Base (TCB)



https://xkcd.com/2347/

Why Should I Trust This?

What does this application rely on for its security?

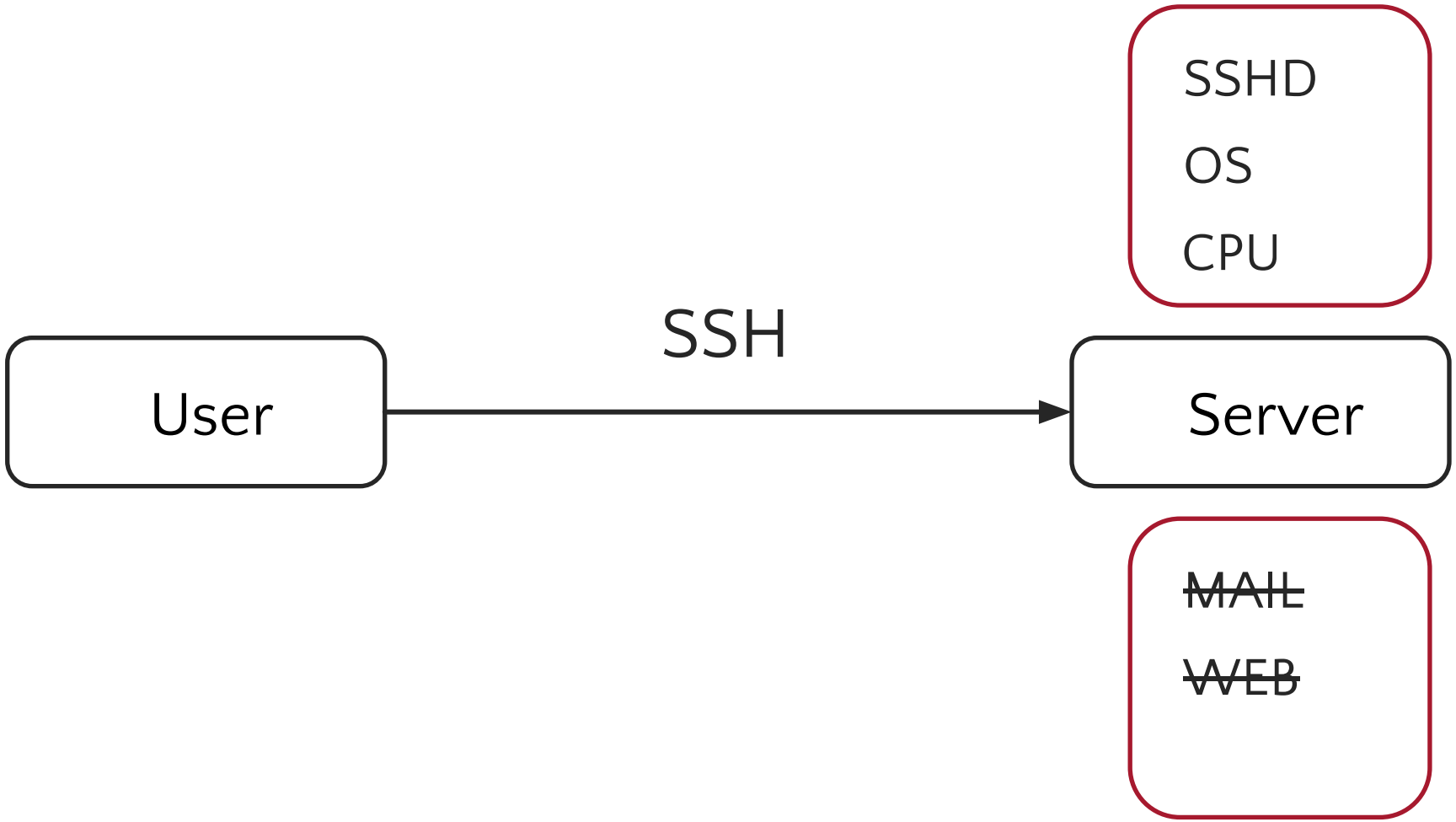


Trusted Computing Base (TCB)

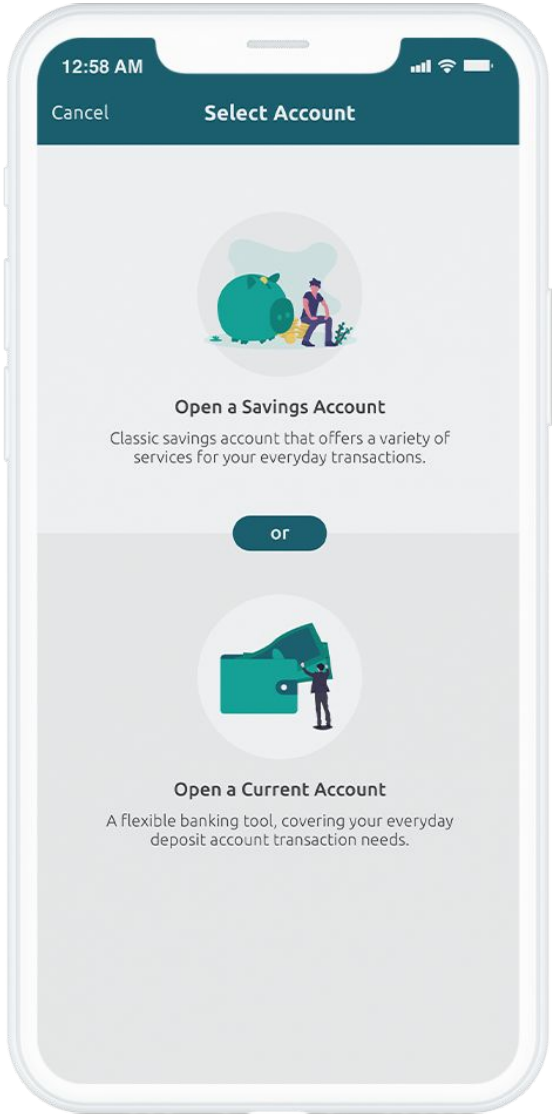
- Component X's TCB is all other components that must operate securely for X to be secure
- Corollary 1: If TCB is secure, X has a chance of being secure
- Corollary 2: If TCB misbehaves, no guarantees about X's security!
- Trusted != Trustworthy

Example of TCB

Example of TCB



What is the TCB here?



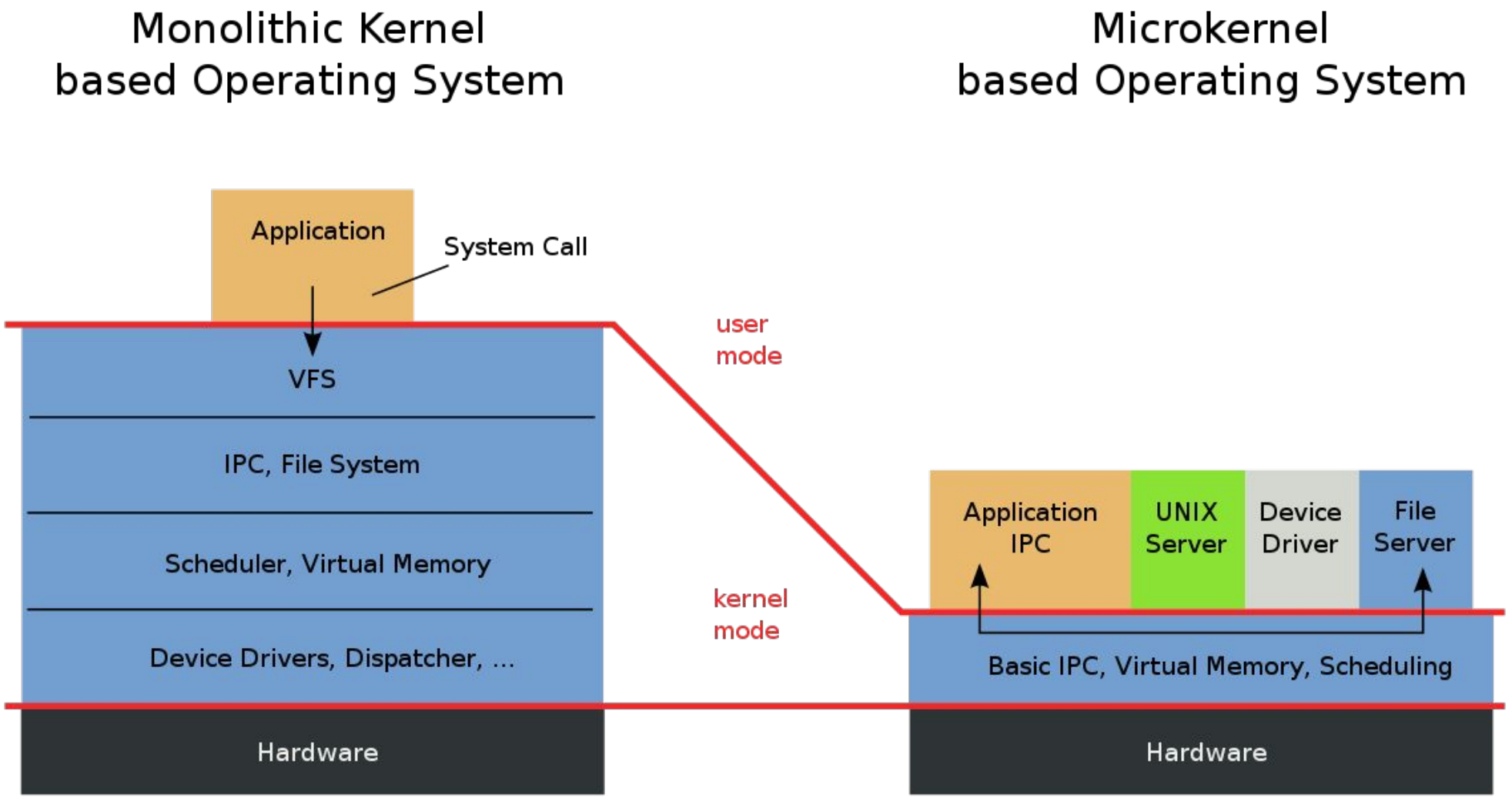
Ideal TCB Design

- Verifiable
  - Implies you want TCB to be as small as possible
    - (even when not verifiable, smaller = less buggy)
- Tamper proof
  - E.g., must prevent messing with the SSHD or OS executables

Why Do We Care About a TCB?

- Securing every piece of a system is hard!
- Identifying the TCB allows us to separate a system into a part that **must** be trusted and a part that **doesn't have to** be
- Can focus security efforts on the trusted piece
  - Reason about security more rigorously
- Caveat: Determining TCB is easier said than done

Example: Operating system kernel



Participation Question

Which of the following is ***NOT*** in the TCB of a *web browser* on your laptop?

A. The laptop's OS
B. JavaScript the browser downloads when you visit a website
C. The laptop's hardware
D. The browser's cryptographic library

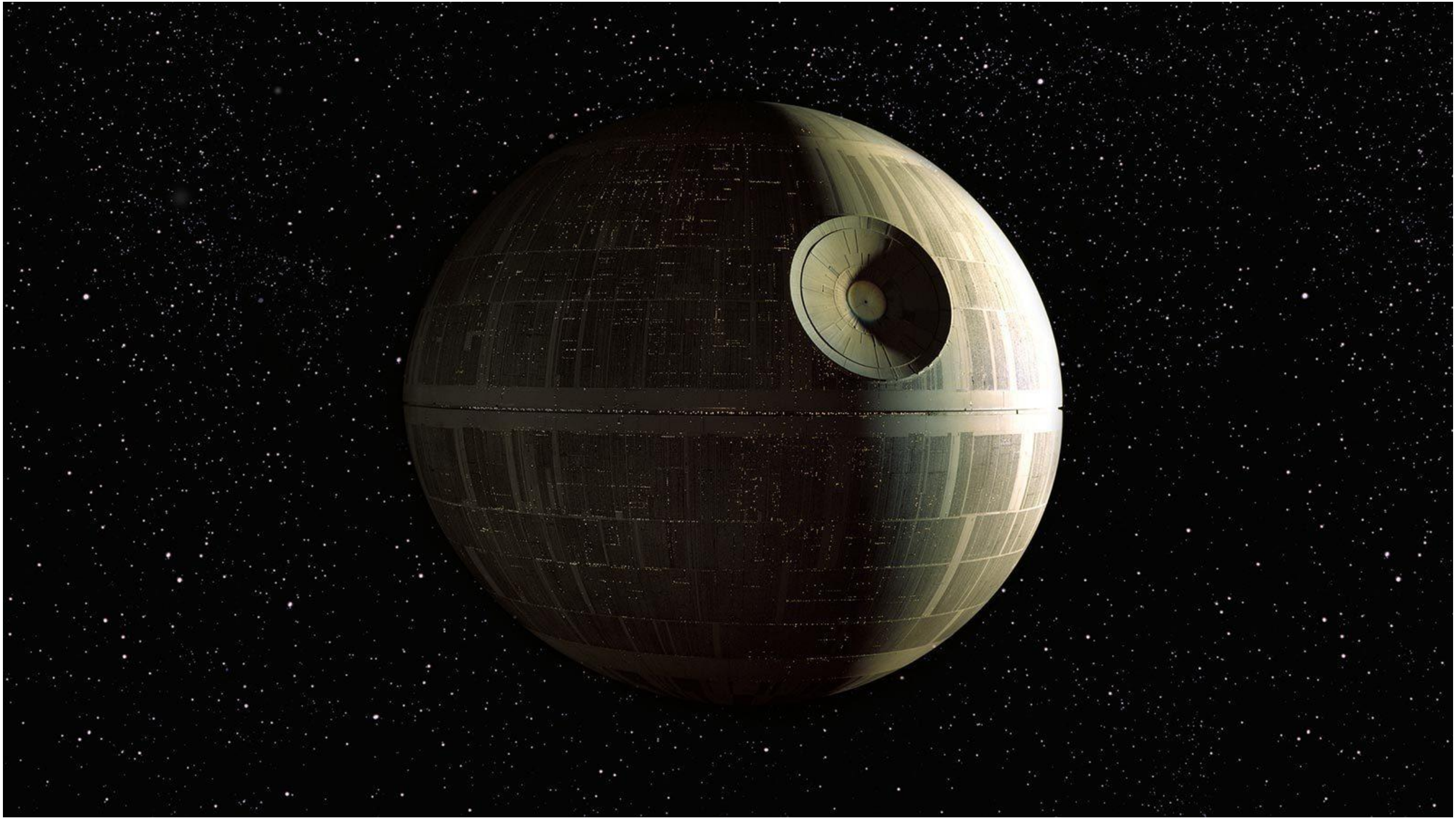
Designing Secure Systems

The Key Principles

- **Economy of mechanism a.k.a. KISS**
- Fail-safe defaults
- Don't rely on security by obscurity
- Complete mediation
- Least privilege
- Separation of duty
- Defense in depth
- Factor in users/acceptance/psychology
- Work factor/economics

} Later

See the reading for more useful principles



Keep It Simple, Stupid (KISS)

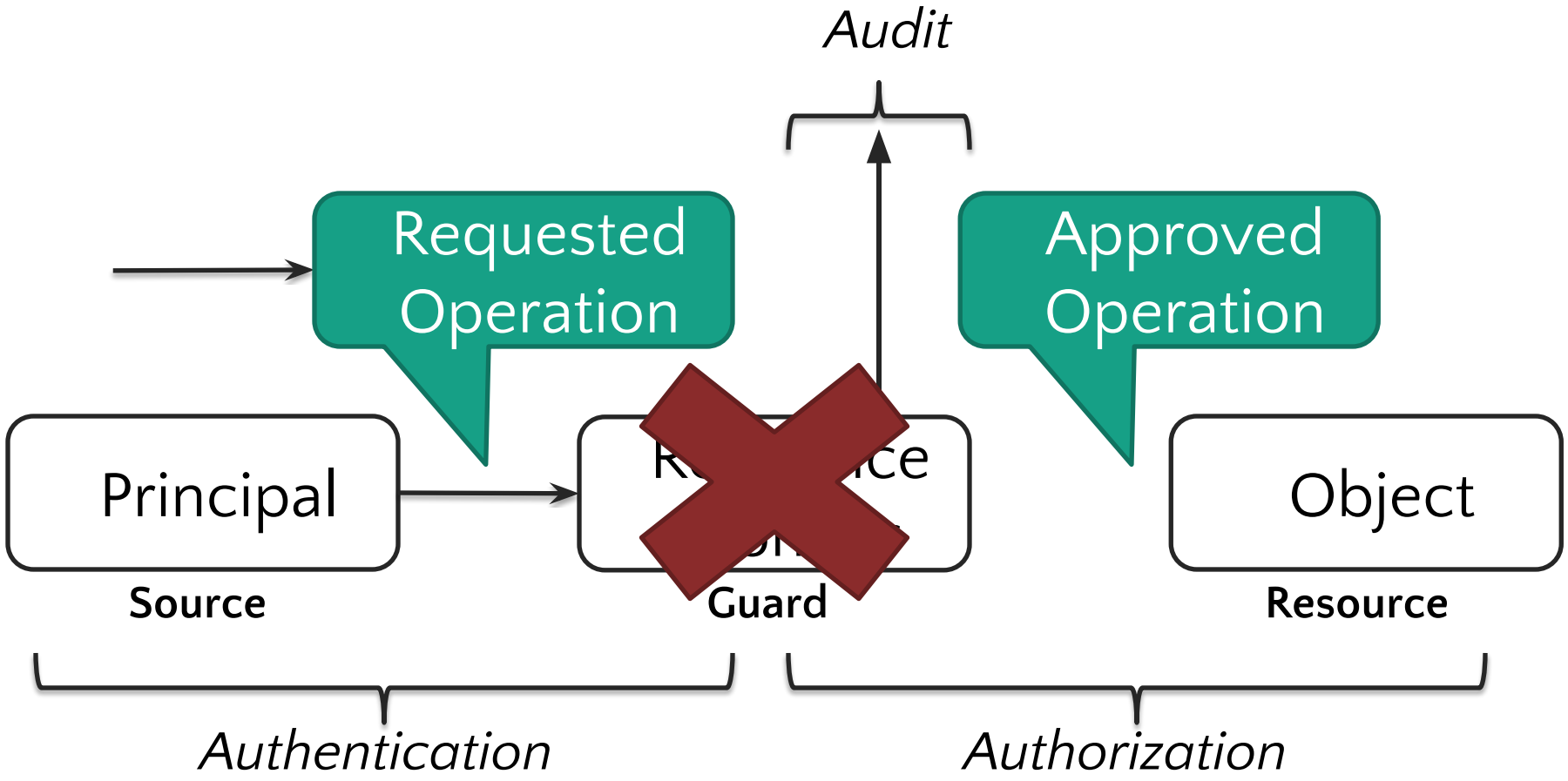
- Rule of thumb:
  1–5 defects per 1K lines of code
- Windows 10 = 50M LOC; Linux 6.7 = 27M LOC
  - In both cases, essentially all in the TCB
- Smaller, simpler TCB is easier to reason about
  - e.g.: seL4 (a formally verified microkernel) = 89k LOC

The Key Principles

- Economy of mechanism a.k.a. KISS
- **Fail-safe defaults**
- Don't rely on security by obscurity
- Complete mediation
- Least privilege
- Separation of duty
- Defense in depth
- Factor in users/acceptance/psychology
- Work factor/economics

} Later

Fail-safe Defaults (Fail Closed)



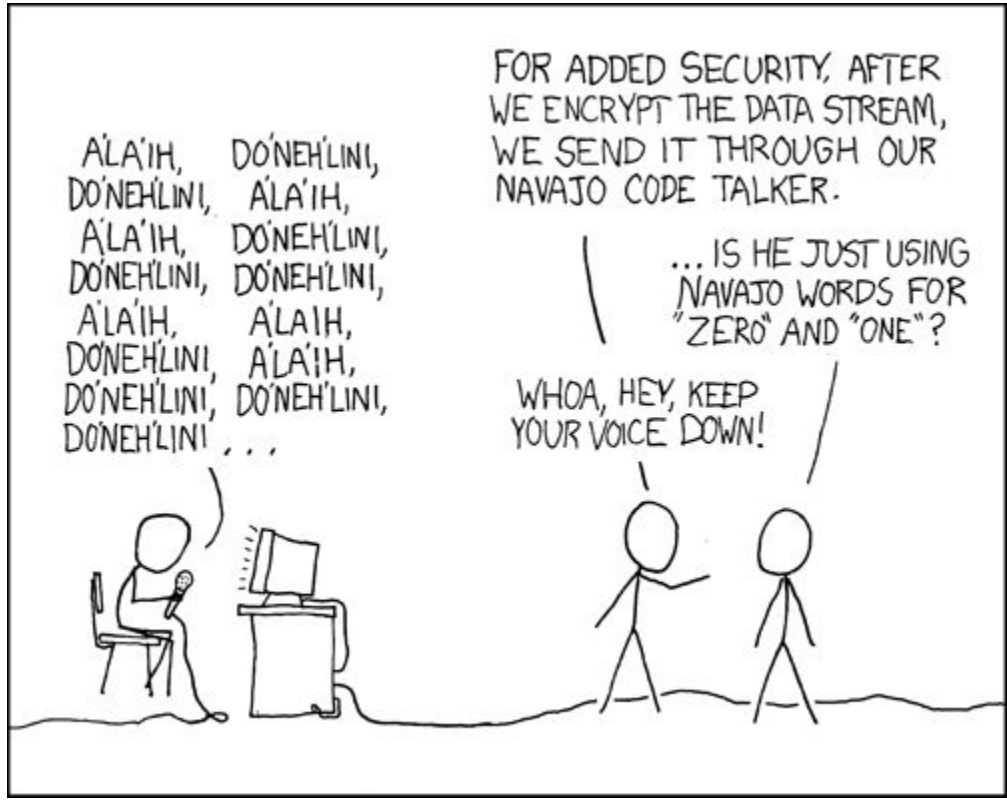
The Key Principles

- Economy of mechanism a.k.a. KISS
- Fail-safe defaults
- **Don't rely on security by obscurity**
- Complete mediation
- Least privilege
- Separation of duty
- Defense in depth
- Factor in users/acceptance/psychology
- Work factor/economics

} Later

No Security by Obscurity

- Common fallacy: System is more secure if the design remains secret
- Keeping designs secret is hard!
  - Someone has to build/implement it
  - Users interact with it
  - The design may be sold to lots of people
- Finding flaws in your own design is hard!



https://xkcd.com/257/

The Key Principles

- Economy of mechanism a.k.a. KISS
- Fail-safe defaults
- Don't rely on security by obscurity
- **Complete mediation**
- Least privilege
- Separation of duty
- Defense in depth
- Factor in users/acceptance/psychology
- Work factor/economics

} Later

Complete Mediation

- Every access to every object is checked by the reference monitor
- Easier said than done!
- TOCTTOU problems

Mediation: TOCTTOU Vulnerabilities

Time-Of-Check-To-Time-Of-Use

```
int openfile(char *path){
  struct stat s;
  if (stat(path,&s) < 0))
    return -1;

  if (!S_ISREG(s.st_mode)){
    error("only regular files allowed");
    return -1;
  }
  return open(path,O_RDONLY)
}
```

Change path

Mediation: TOCTTOU Vulnerabilities

Time-Of-Check-To-Time-Of-Use

```
void withdraw(int w){
  b = getbalance();
  if (b<w)
    error("not enough $$");

  b = b-w;
  send(w)
}
```

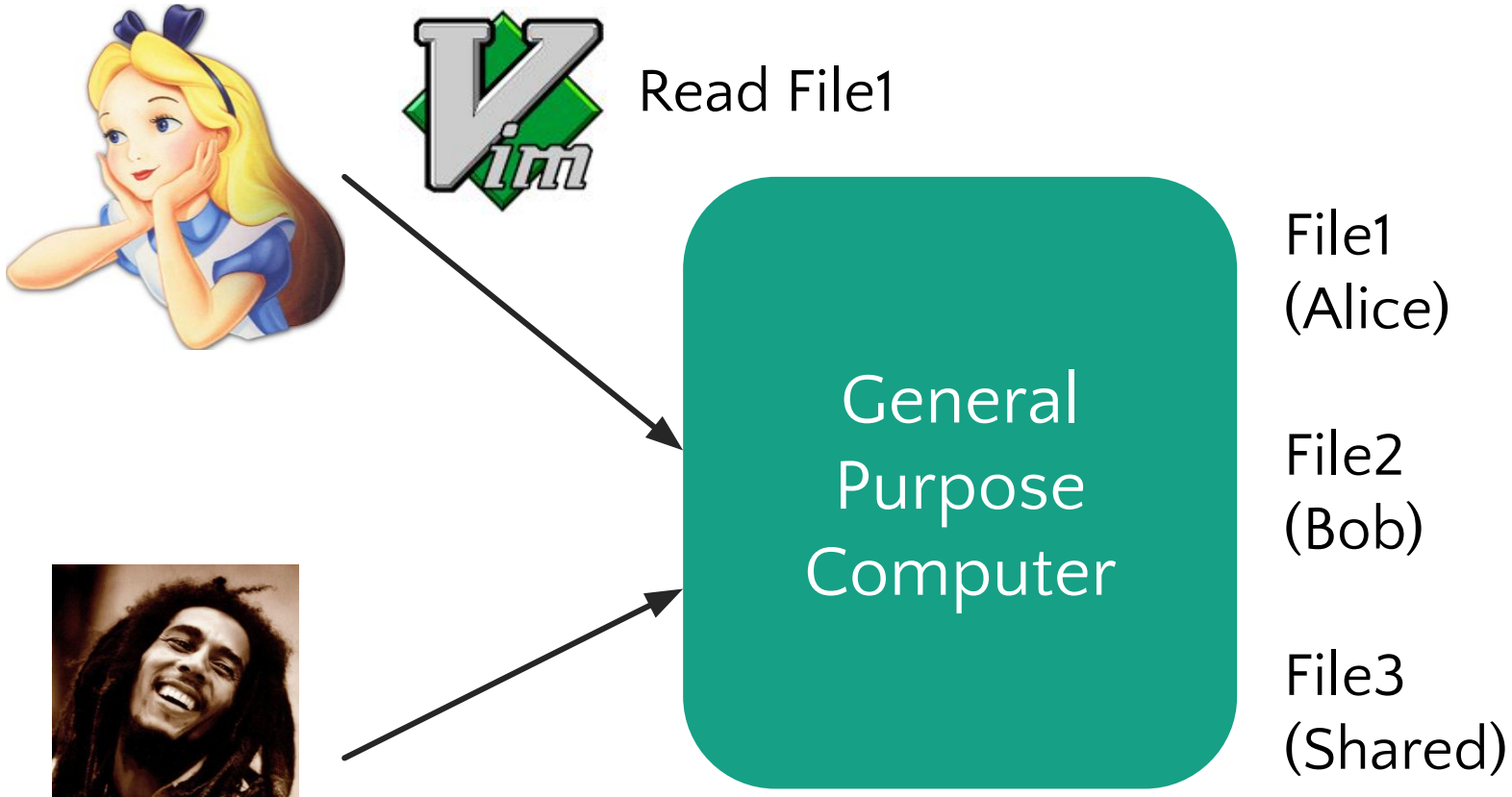
The Key Principles

- Economy of mechanism a.k.a. KISS
- Fail-safe defaults
- Don't rely on security by obscurity
- Complete mediation
- **Least privilege**
- Separation of duty
- Defense in depth
- Factor in users/acceptance/psychology
- Work factor/economics

} Later

Least Privilege

A user or entity should only have access to the specific data, resources and applications needed to complete a required task and nothing more.



The Key Principles

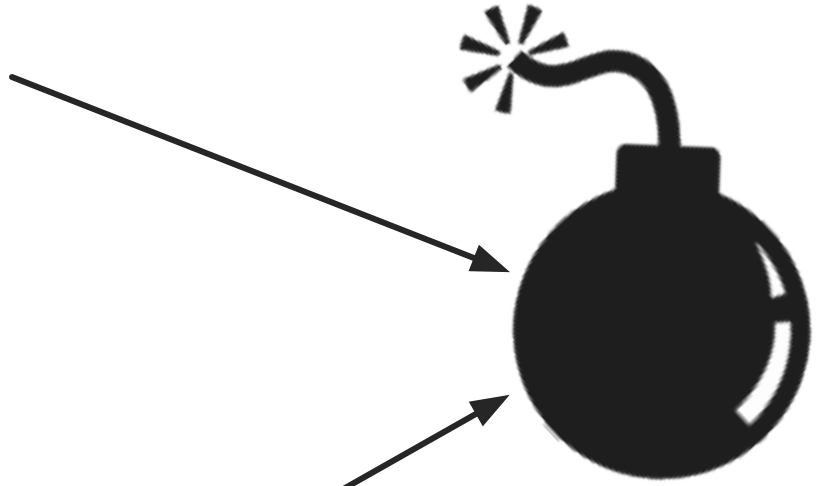
- Economy of mechanism a.k.a. KISS
- Fail-safe defaults
- Don't rely on security by obscurity
- Complete mediation
- Least privilege
- **Separation of duty**
- Defense in depth
- Factor in users/acceptance/psychology
- Work factor/economics

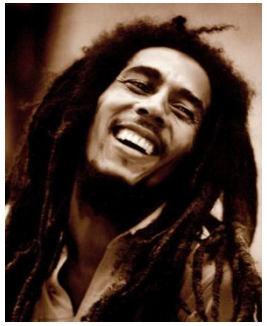
} Later

Separation of Duty (SoD)

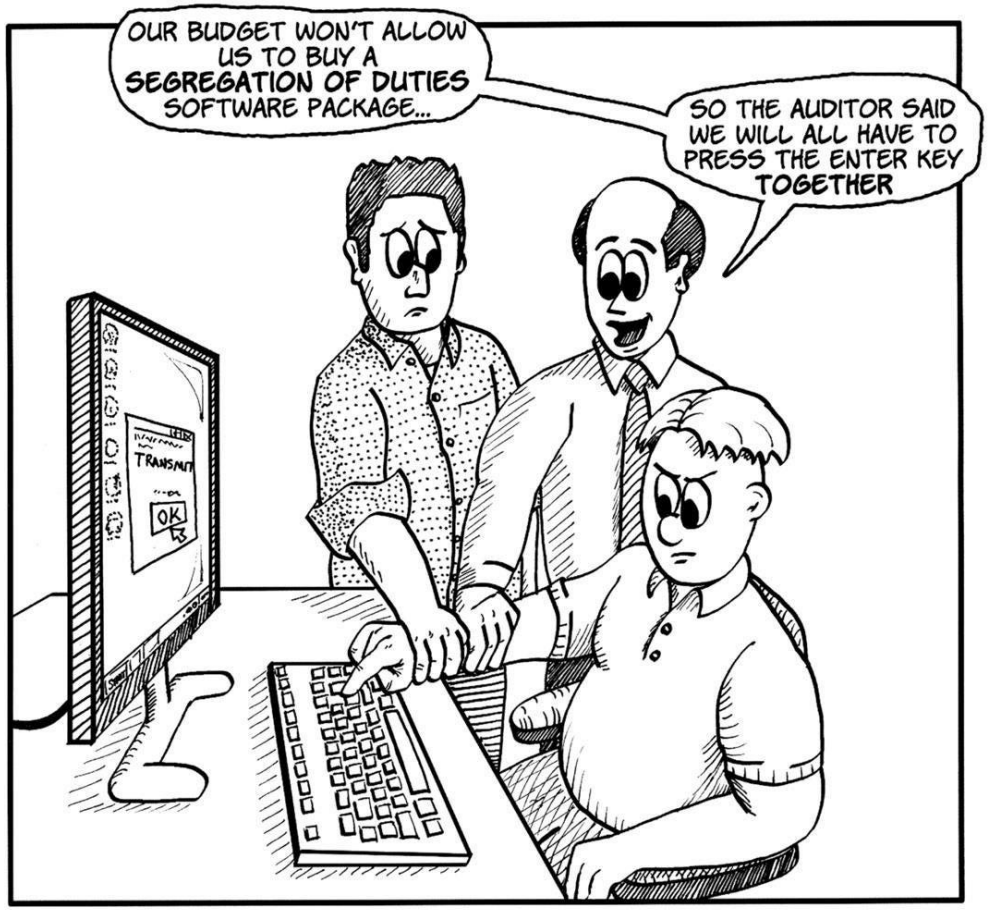
Having more than one person required to complete a task!

Push Pull Request
Launch Missile





Approve Pull Request
Launch Missile



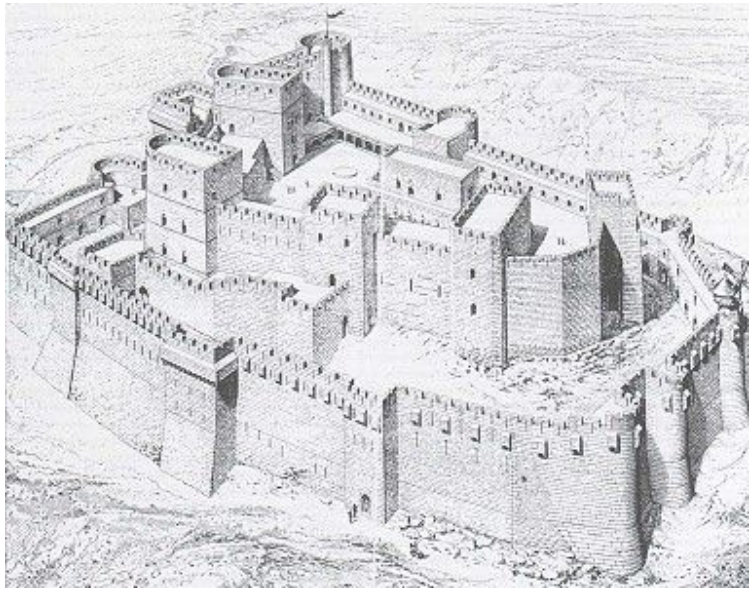
The Key Principles

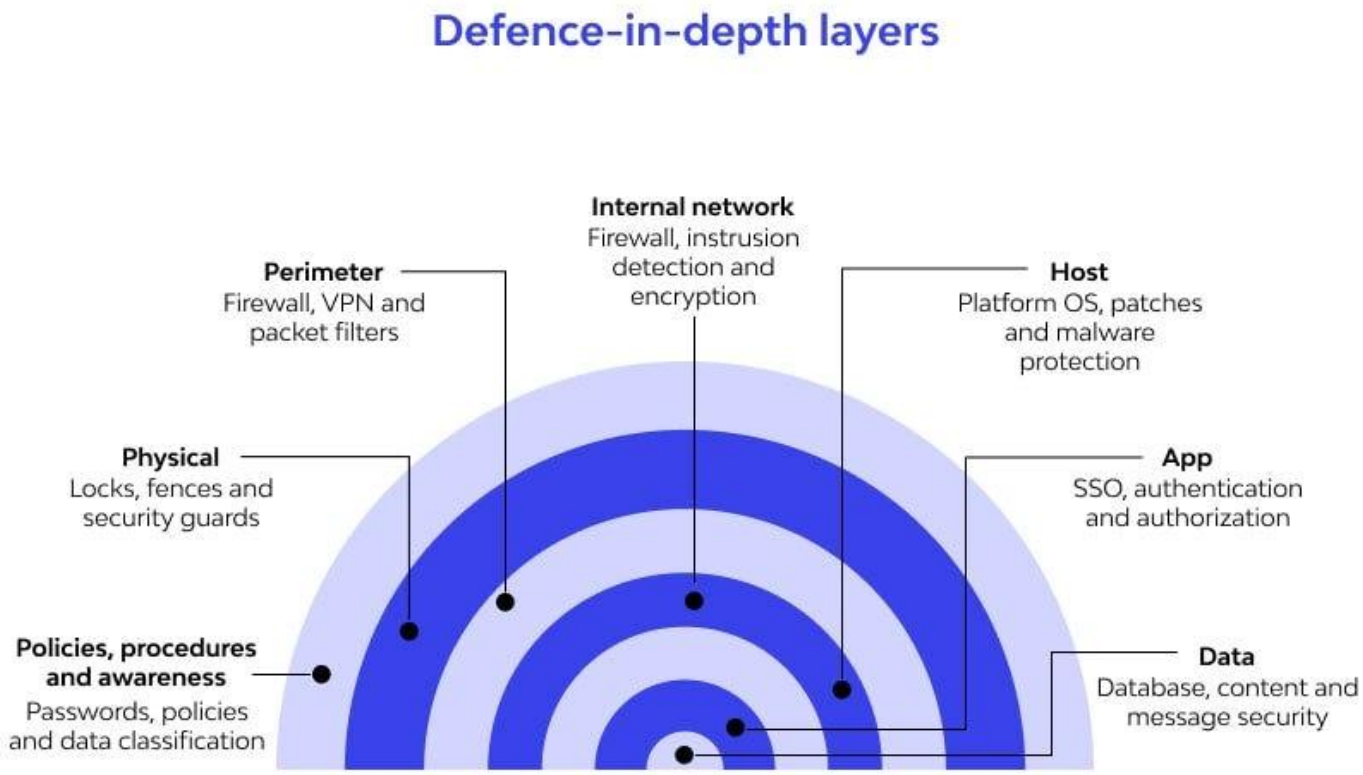
- Economy of mechanism a.k.a. KISS
- Fail-safe defaults
- Don't rely on security by obscurity
- Complete mediation
- Least privilege
- Separation of duty
- **Defense in depth**
- Factor in users/acceptance/psychology
- Work factor/economics

} Later

Defense in Depth

- Few defensive measures are perfect
- Plan for failures
- Beware risk compensation!





Participation Question

Systems based on ACLs (like UNIX) typically deny access entirely if a subject is not listed on the relevant ACL. This is an example of which principle?

A. Fail-safe defaults
B. Complete mediation
C. Separation of duty
D. Defense in depth

Ευχαριστώ και καλή μέρα εύχομαι!

Keep hacking!