CSE P564:
Computer Security and Privacy

Autumn 2024

David Kohlbrenner
dkohlbre@cs

UW Instruction Team: David Kohlbrenner, Yoshi Kohno, Franziska Roesner. Thanks to Dan Boneh, Dieter Gollmann, Dan Halperin, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials

Hello!

- Instructor: David Kohlbrenner (he/him)

TA Staff

- Kirandeep (Kiran) Kaur

- Sela Navot

Course Plan

- Lectures: in-person
  - Lectures are recorded to Panopto (please attend!)
    - * Recordings include student speech/video/chat and will not be shared outside the class
- Office hours:
  - Wednesdays 5-6pm, in-person
  - Vote on the others! (Ed board)
- Evaluation
  - Labs
  - Paper summaries
  - Final project; **no exams**
  - Participation/in-class exercises

Required notification

If there is fire or other building emergency requiring evacuation, you will hear the fire alarm system activate. Upon hearing the alarm, please exit the room through the front or back. Assume all alarm activations are real and respond promptly. Once you have left the room, look to your left and right to find the nearest illuminated EXIT sign; they are green. You will follow them to the nearest exit.

DO NOT USE ELEVATORS.

Once you have exited the building, the evacuation assembly point for this building is [see building exit route signs posted in building hallway for this information]. Please wait at the evacuation assembly point until the ‘all clear’ has been given, after which you may re-enter.

Course Resource Cheat Sheet

- **Course website:** Schedule, assignment details, readings, policies
- **Classrooms:** Lectures
- **Zoom/In-person:** Office hours
- **Panopto:** Lecture recordings
- **Ed:** Discussion board, Announcements
- **Email:** Reach course staff privately

Gradescope In-class activities

- We’ll do a lot of breakouts in class

- Breakouts are four parts:
  - Think (solo)
  - Pair (discuss with neighbors)
  - Writeup (Summarize your discussion on Gradescope)
  - Share (Full-class discussion)

What Does “Security” Mean to You?

1) Spend a minutes defining security in the context of computing/technology to yourself.
2) Discuss with your neighbors
3) Summarize on Gradescope
4) Share!

Try putting some answers in Gradescope

Why Systems Fail

Systems may fail for many reasons, including:

- Reliability deals with accidental failures
- Usability deals with problems arising from operating mistakes made by users
- Design and goal oversights deals with oversights, errors, and omissions during the design process
- Security deals with intentional failures created by intelligent parties
  - Security is about computing in the presence of an adversary
  - But security, reliability, usability, and design/goals oversights are all related

Challenges: What is “Security”?

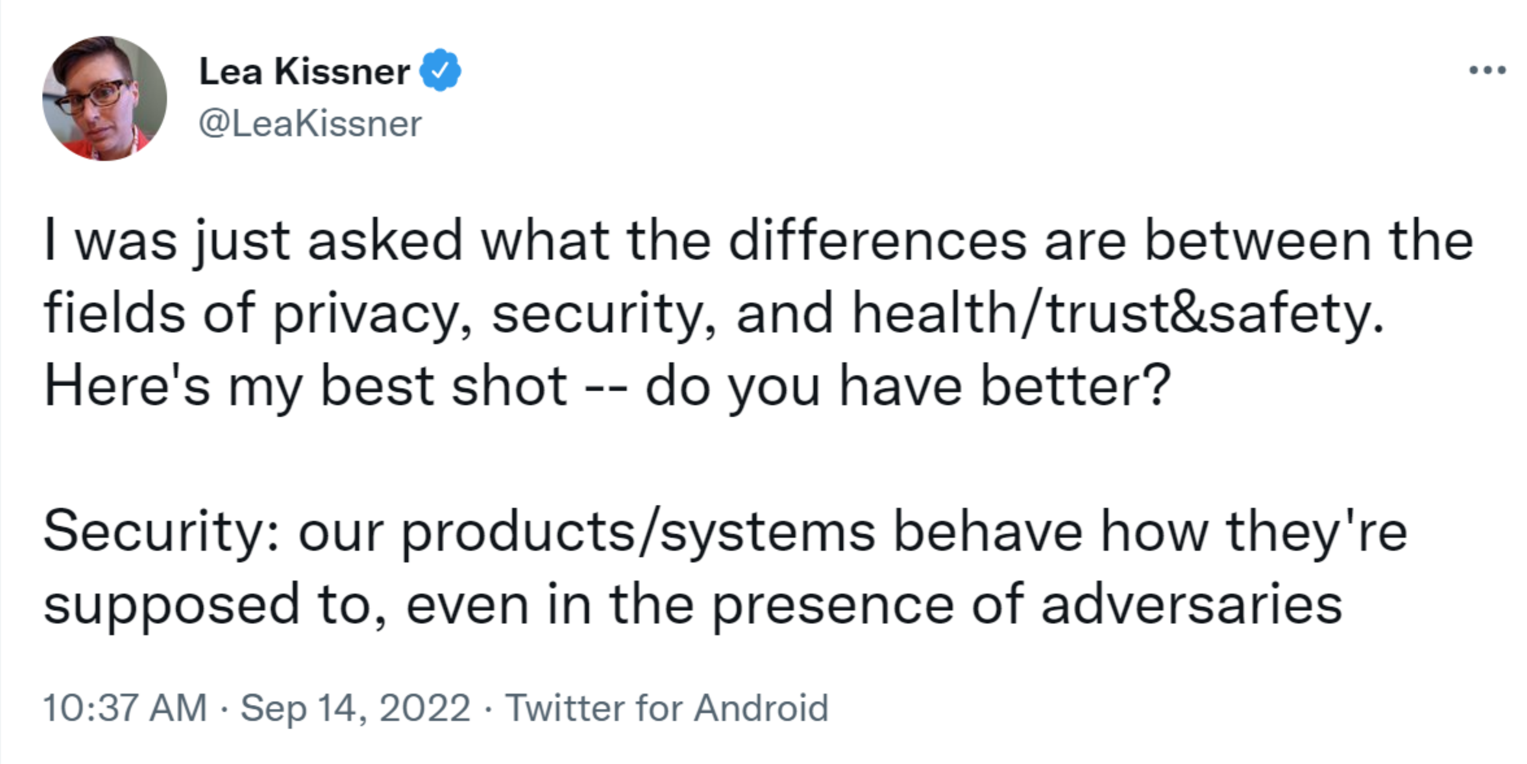
- What does security mean?
  - Often the hardest part of building a secure system is figuring out what security means (“threat modeling”)
  - Who are the **stakeholders** for which we are considering “security”?
  - What are the **assets** to protect?
  - What are the **threats** to those assets?
  - Who are the **adversaries**, and what are their **resources**?
  - What is the **security policy or goals**?
- **Perfect security does not exist!**
  - Security is not a binary property
  - Security is about risk management

Privacy?

- Privacy often strongly overlaps security

- Privacy may also consider when systems *work as intended!*

- Not a hard-and-fast distinction
  - Privacy and security are generally intertwined

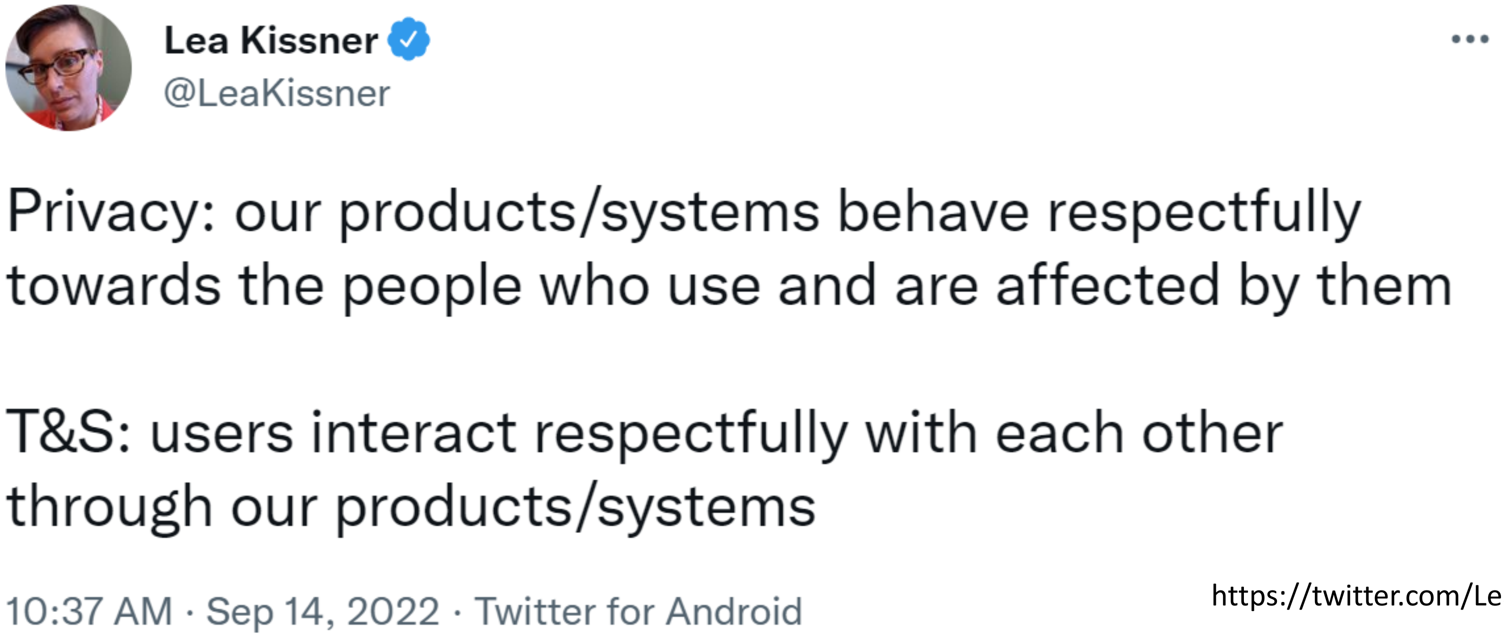


Lea Kissner
@LeaKissner

I was just asked what the differences are between the fields of privacy, security, and health/trust&safety. Here's my best shot -- do you have better?

Security: our products/systems behave how they're supposed to, even in the presence of adversaries

10:37 AM · Sep 14, 2022 · Twitter for Android



Lea Kissner
@LeaKissner

Privacy: our products/systems behave respectfully towards the people who use and are affected by them

T&S: users interact respectfully with each other through our products/systems

10:37 AM · Sep 14, 2022 · Twitter for Android

https://twitter.com/LeaKissner/status/1570104506477867008

Two Key Themes of this Course

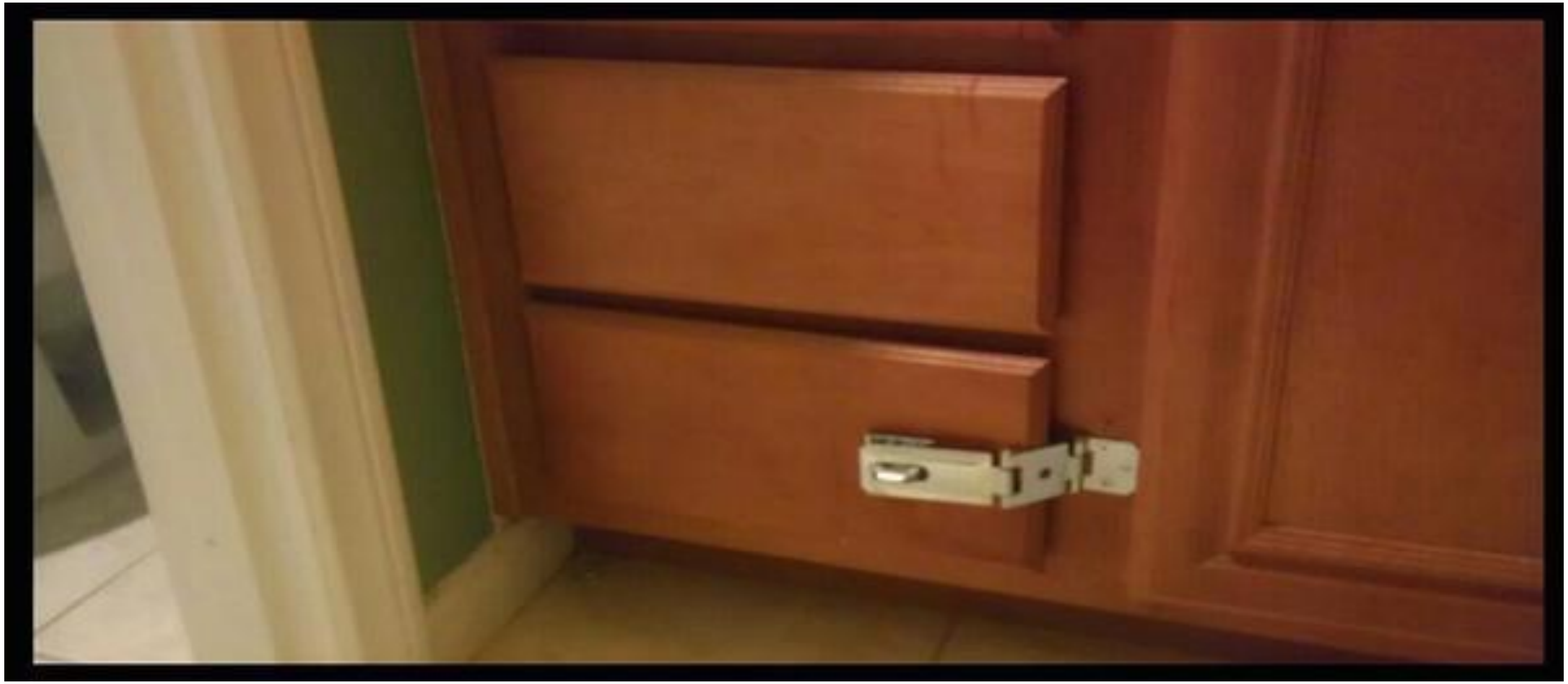
1. How to **think** about security and privacy
   - The "Security Mindset"
   - (This mindset will be valuable even outside of the security context, e.g., to consider diverse stakeholders of a system)

2. **Technical aspects of security and privacy**
   - Vulnerabilities and attack techniques
   - Defensive technologies
   - Topics range widely, tell us if something you'd like isn't covered

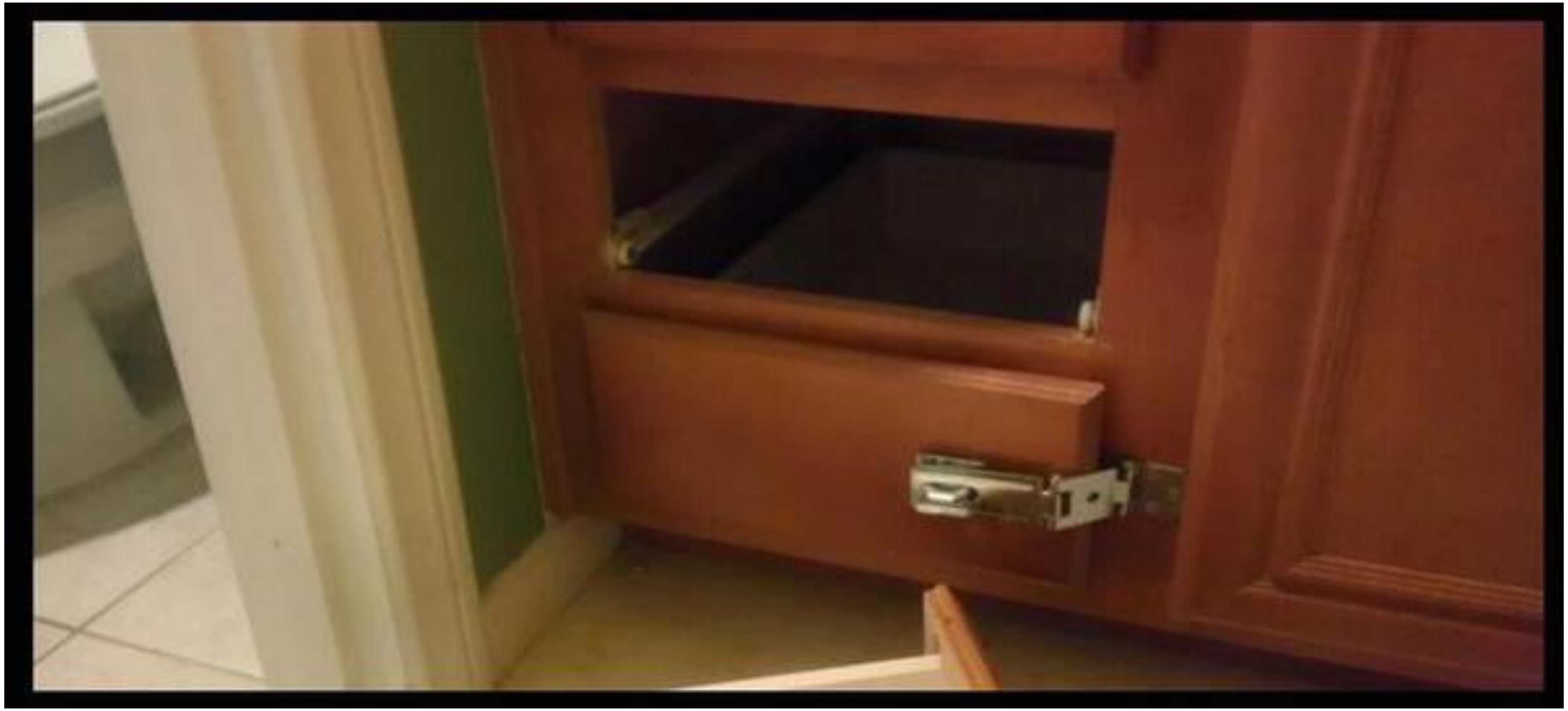
Theme 1: Security Mindset

- Thinking critically about designs, challenging assumptions
- Being curious, thinking like an attacker
- “That new product X sounds awesome, I can’t wait to use it!” versus
    - “That new product X sounds cool, but I wonder what would happen if someone did Y with it; I wonder if the designers thought of Z…”
- Why it’s important
    - Technology changes, so learning to think like a security person is more important than learning specifics of today’s systems
    - Will help you design better systems/solutions
    - Interactions with broader context: law, policy, ethics, etc.

Security Mindset Example



Security Mindset Example



Learning the Security Mindset

- Several approaches for developing “The Security Mindset” and for exploring the broader contextual issues surrounding computer security
  - Reading papers!
  - In class discussions and activities
  - Labs
  - Participation in Ed discussion board (e.g., asking about news stories, technologies)

What This Course is Not About

- Not a comprehensive course on computer security
  - Impossible to cover everything in one quarter

– Not about all of the latest and greatest attacks
  - Read news, ask questions, discuss on Ed

- Not a course on ethical, legal, or economic issues
  - We will touch on these issues, but the topic is huge

- Not a course on how to “break into” systems
  - Yes, we will learn about attacks ... but the ultimate goal is to develop an understanding of attacks so that you can build more secure systems

Security: Not Just for PCs

smartphones

wearables

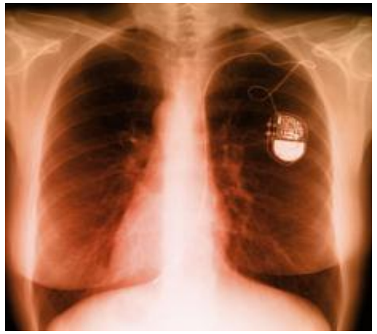
voting machines

RFID

game platforms

EEG headsets

mobile sensing platforms



medical devices

cars

airplanes

Communication

- dkohlbre@cs.washington.edu
  - Use this if something is sensitive, personal, confidential, etc.
- csep564-staff@cs.washington.edu
  - Use this to reach all course staff (including instructor)
- Ed Discussion Board
  - Default platform for questions, supports private and public posts.
  - Also announcements
- We will do our best to be responsive, but please be professional, and plan ahead!

Course Materials

- Readings
  - There are required (weekly) research readings
  - There are many suggested readings, these have no summaries

- Lectures
  - Lectures are where the bulk of information is
  - Lectures are interactive, and have discussion periods
  - Please attend!

Guest Lectures

- We may have some guest lectures during the quarter
  - They will be announced ahead of time if we’re having one

Course webpage

Grading, labs, etc.

Ethical security work

- We’re going to discuss how to break things
    - Please be responsible with this knowledge
    - Understanding flaws is critical to fixing them

In-Class Participation

- Trying to bring the best of online, in-person
  - In-class discussions, polls, and other online tools

- **Main component: Lightly graded in-class activities**
  - Gradescope submission, effort not correctness

Late Submission Policy

- 5 free late days, no questions asked
  - Cumulative, throughout the quarter
  - Use up to 3 for one submission
  - Don’t ask us about more late days, use these _first_

- Otherwise, assignments will be dropped 20% per calendar day.
  - Late days will be rounded up
  - 26 hours late is 2 days (40% deduction)

Discussion Board

- We've set up a Ed Discussion Board for this course

- Please use it to discuss labs and other general class materials

- You can also use it to exercise the "security mindset"
  - Discussions of how movies get security right or wrong
  - Discussions of news articles about security (or not about security, but that miss important security-related things)
  - Discussions about security flaws you observe in the real world

Generative AI Tools (aka ChatGPT)

- Tl;dr: We heavily discourage using these tools
  - You may *not* use them to solve assignments/questions
  - You may (with disclosure) ask basic factual questions
  - See course webpage for full policy

- Our experience has been that GenAI solutions to security problems are *particularly bad*
  - This is backed up by studies (so far) on the security quality of GenAI code

Announcements

- We will use Ed for **announcements**
  - It will send an email to you for announcements

Final Project

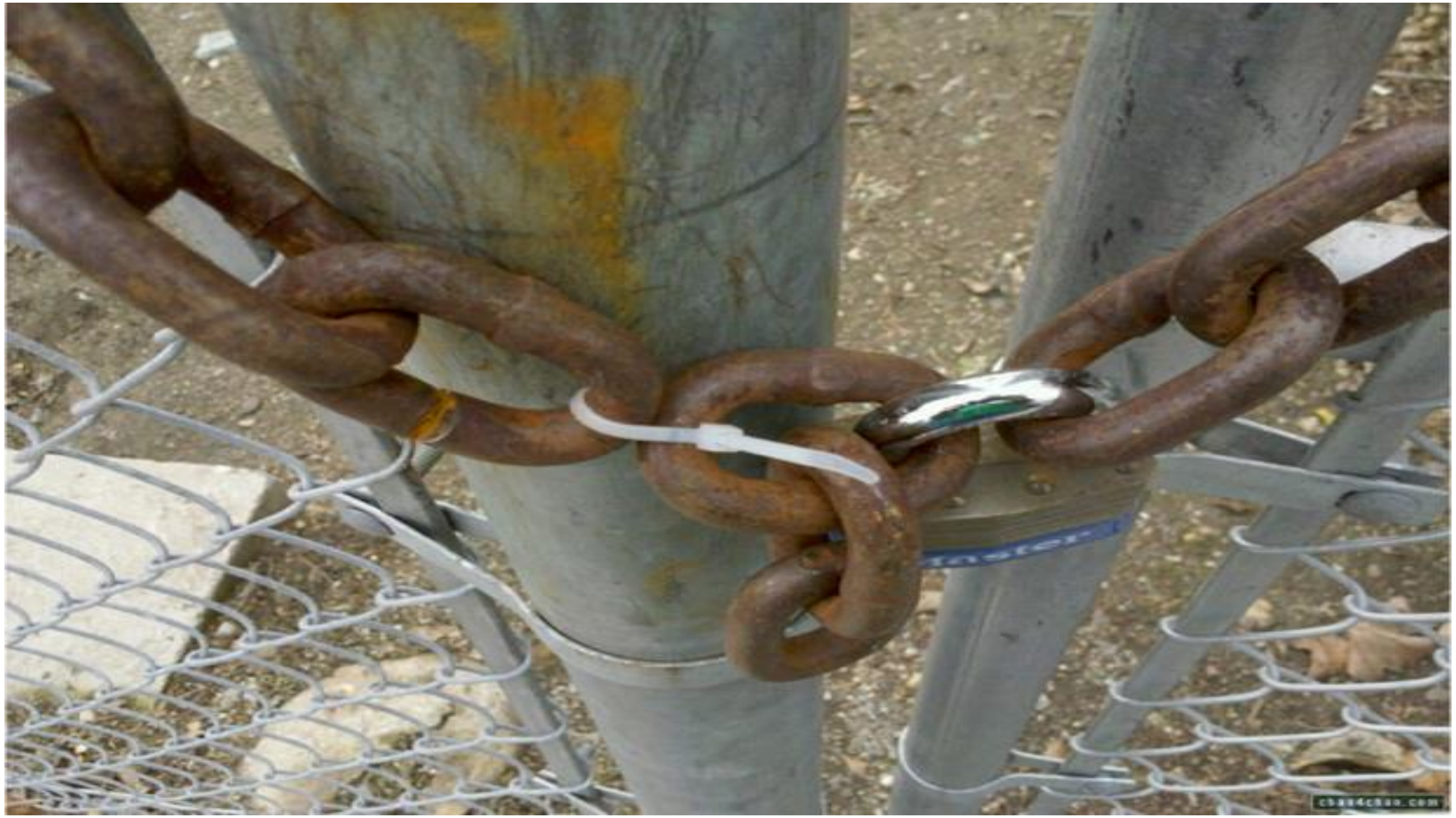
- **No midterm or final exam!**
- Final project will require you to find and fix vulnerabilities in a medium (~1000 lines of C code) piece of software.
- You will also need to explain your decisions and evaluation of the vulnerabilities

Prerequisites (CSE 484)

- Assume: Working knowledge of C and assembly
  - One of the labs will involve writing buffer overflow attacks in C
  - You will need a detailed understanding of x86 architecture, stack layout, calling conventions, etc.
- Assume: Working knowledge of software engineering tools for Unix environments (gdb, etc)
- Assume: Working knowledge of JavaScript
- **Assume: Ability to learn new programming languages / skills easily**

Discussion

- **Everyone** in this class **deserves** to be in this class!!
- We are **all** coming to this course with **different backgrounds** and experiences
- There are **no bad questions**; never belittle a questioner or their question; always be supportive
- Instructors / staff aren't always aware of everything, so **please call our attention to things as needed**
  - E.g., someone might harm someone else with what they say without ever realizing that what they said is harmful; that harm still exists, regardless of whether there was an intent to harm



Threat Modeling

Threat Modeling

- Assets: What are we trying to protect?
- Adversaries: Who might try to attack, and why?
- Vulnerabilities: How might the system be weak?
- Threats: What actions might an adversary take to exploit vulnerabilities?
- Risk: How important are assets? How likely is exploit?
- Possible Defenses: What mitigations do we have available?
- Not “traditional” threat modeling, but important (both in general, and to help better understand the system prior to threat modeling):
  - Benefits: Who might the system benefit, and how?
  - Harms: Who might the system harm, and how?

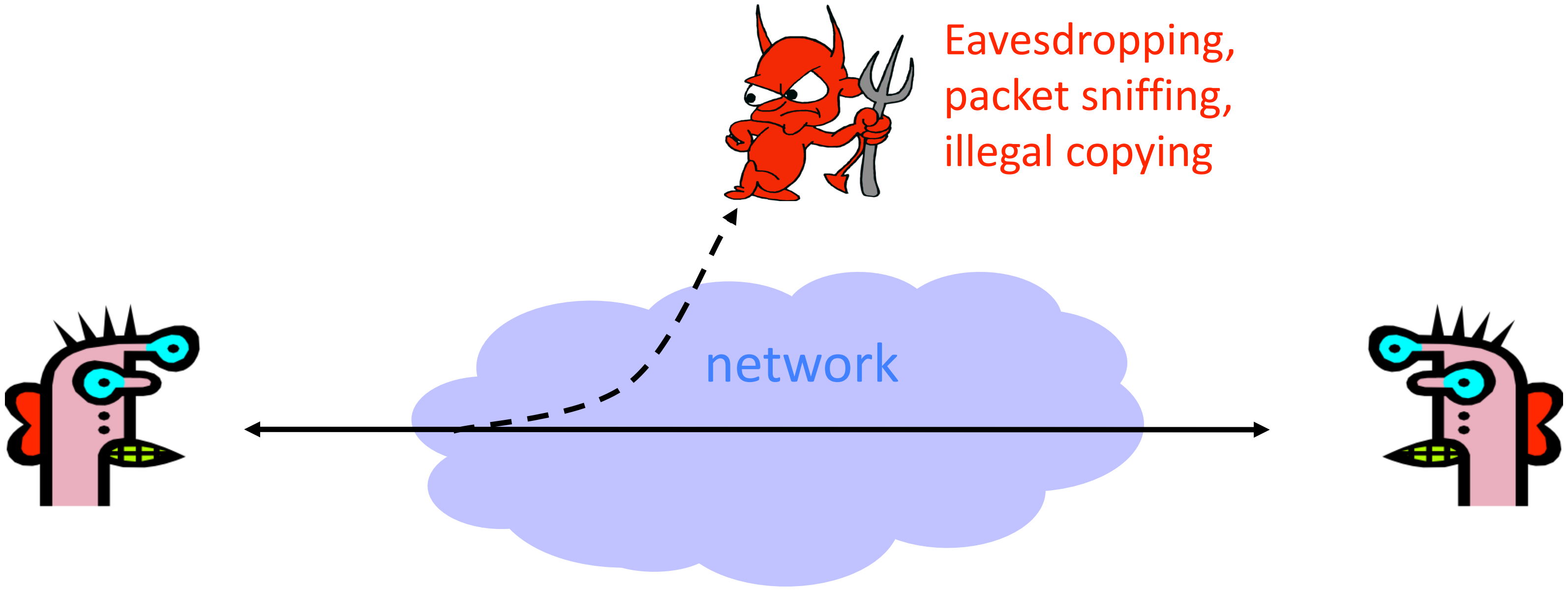
What’s *Security*, Anyway?

- Common general security goals: “CIA”
  - Confidentiality
  - Integrity
  - Availability

- Or the extension: CPIAAU (Parkerian Hexad)
  - Control
  - Authenticity
  - Utility

Confidentiality (Privacy)

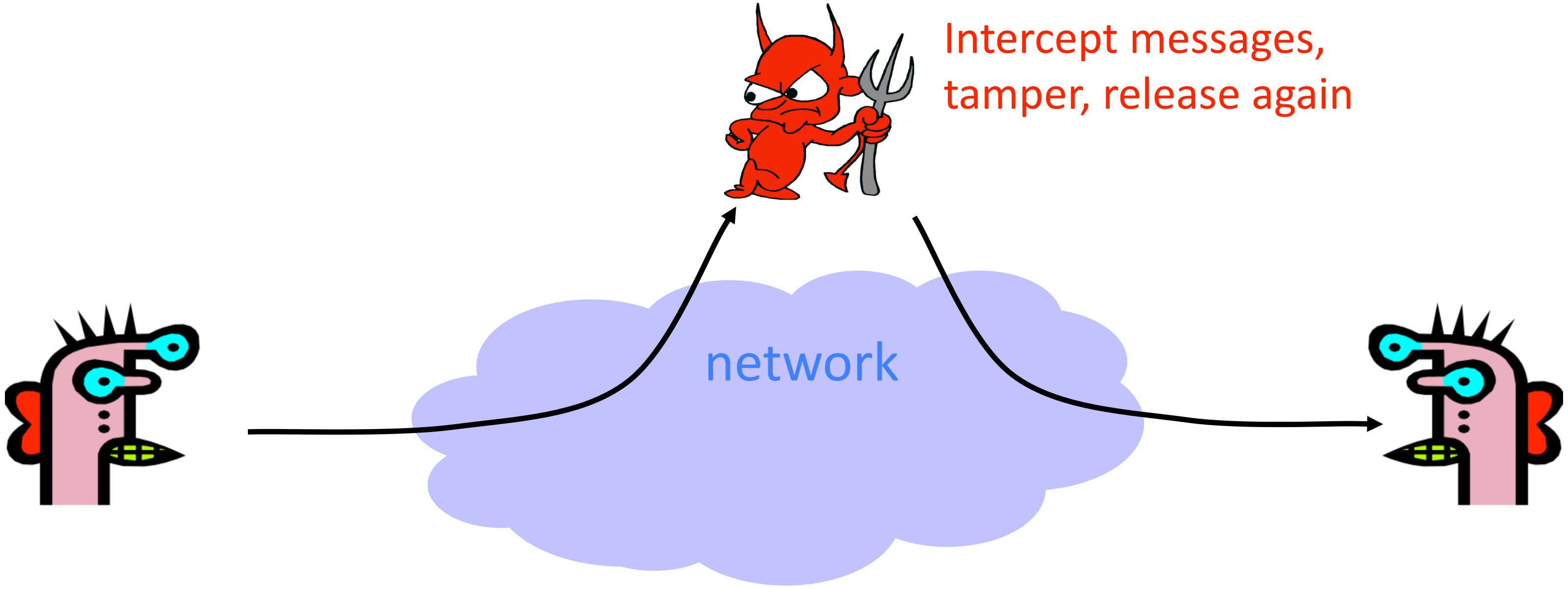
- Confidentiality is concealment of information.



Eavesdropping, packet sniffing, illegal copying

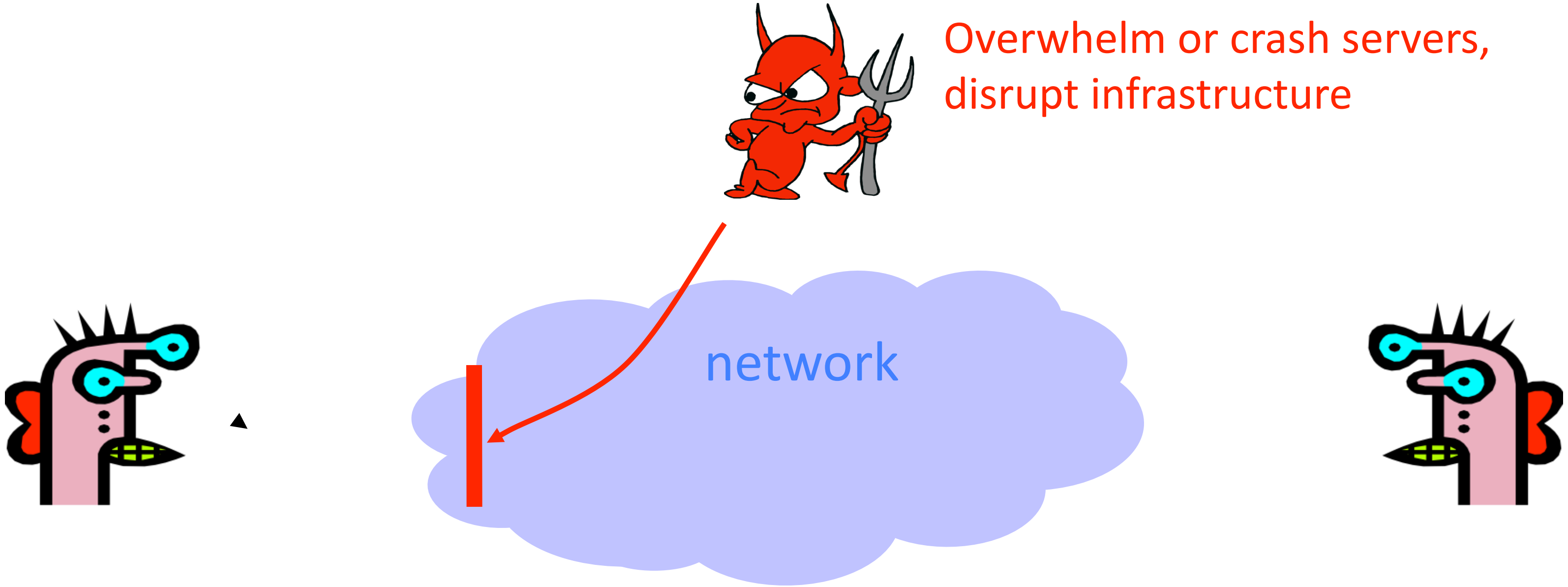
Integrity

- Integrity is prevention of unauthorized changes.



Availability

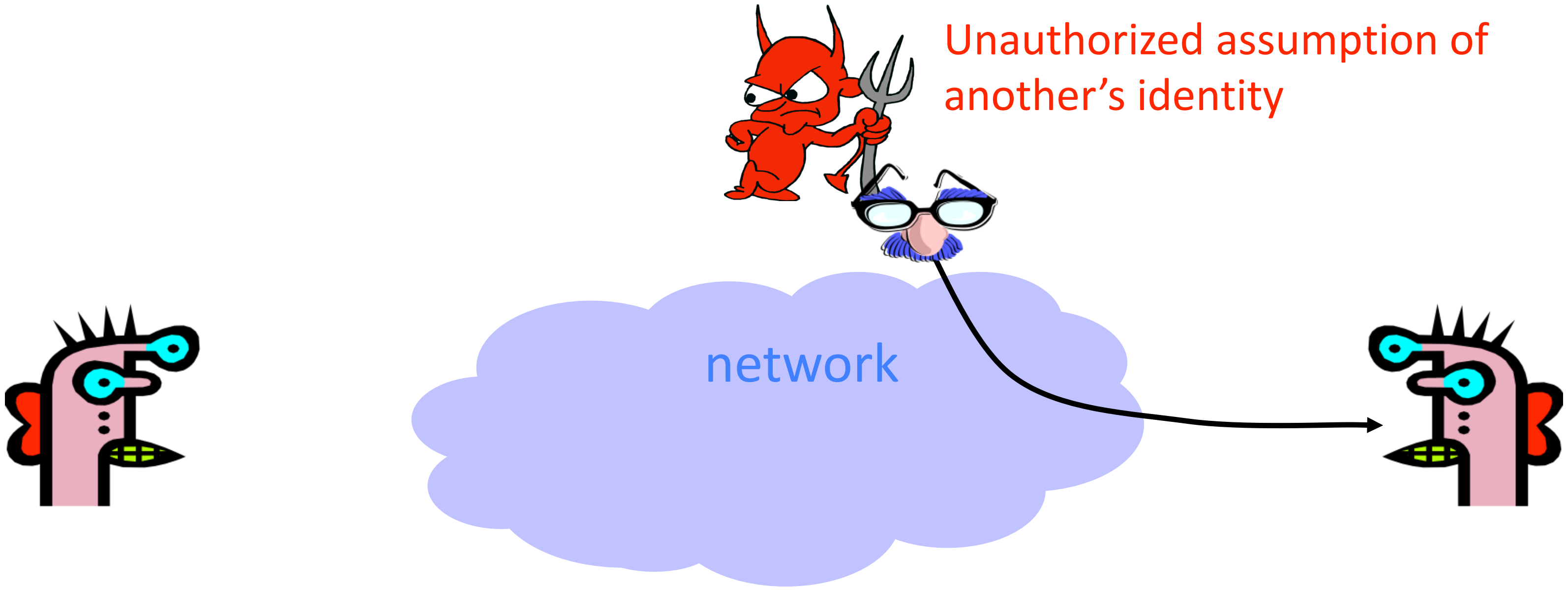
- Availability is ability to use information or resources.



Overwhelm or crash servers,
disrupt infrastructure

Authenticity

- Authenticity is knowing who you're talking to.



Unauthorized assumption of another's identity

Threat Modeling

- There's no such thing as perfect security
  - But, attackers have limited resources
  - **Make them pay unacceptable costs / take on unacceptable risks to succeed!**

- Defining security per context: identify assets, adversaries, motivations, threats, vulnerabilities, risk, possible defenses

Threat Modeling Example: Electronic Voting

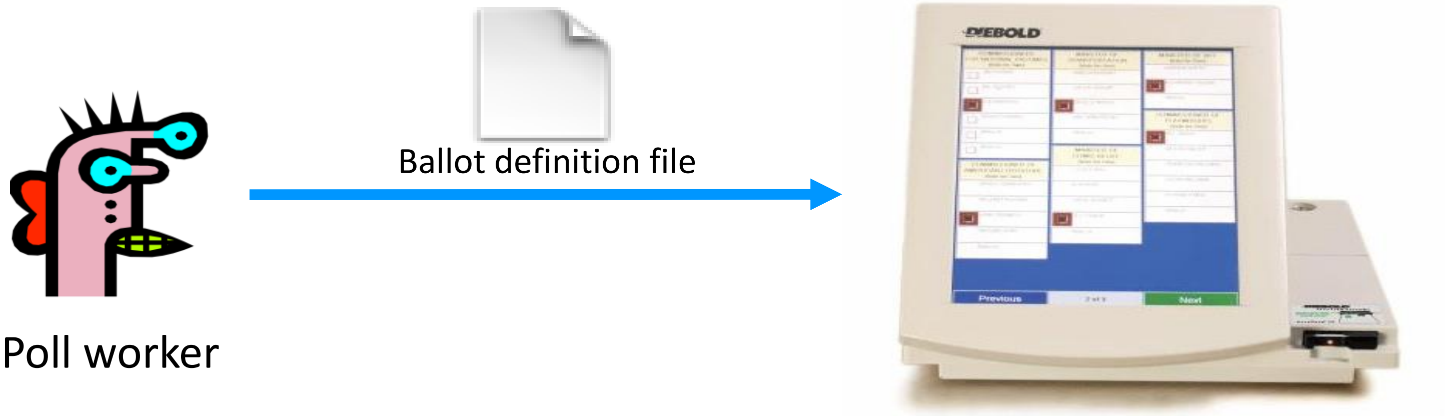
- Popular replacement to traditional paper ballots

AP

Before we get into the system itself

- Think about how in-person electronic voting must work
  - Machines
  - Votes
  - Voters
  - Etc.
- Lightning round threat modeling!
  - Assets?
  - Adversaries?
  - Risks (What might happen if assets are compromised?)
  - Defenses?

Pre-Election

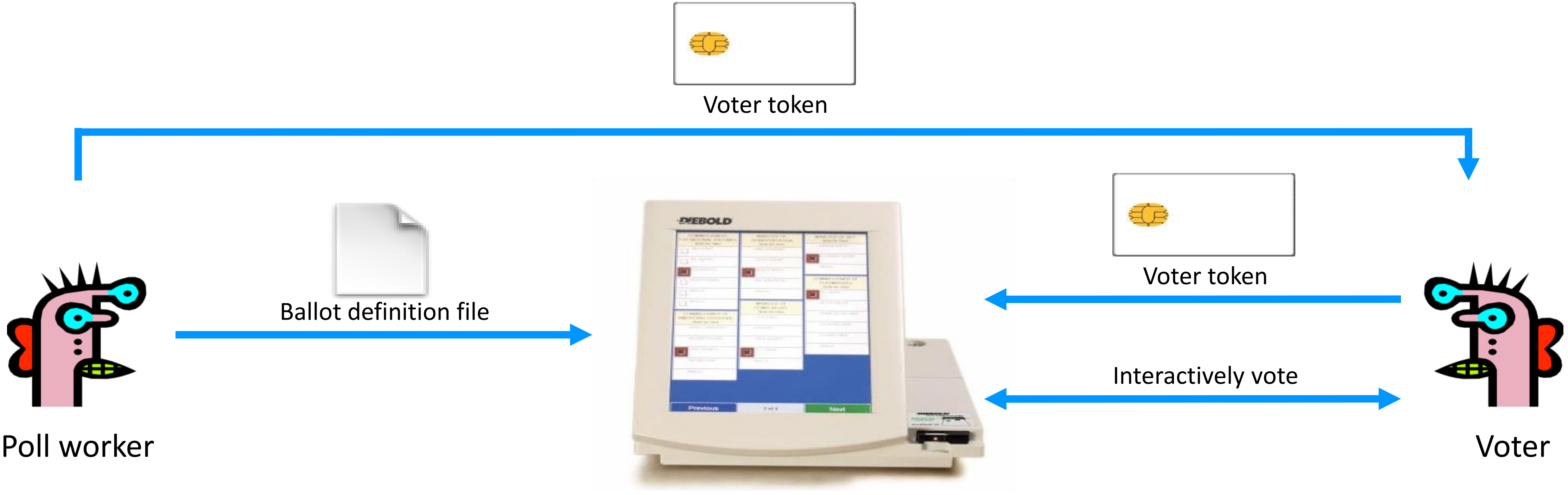


Ballot definition file

Poll worker

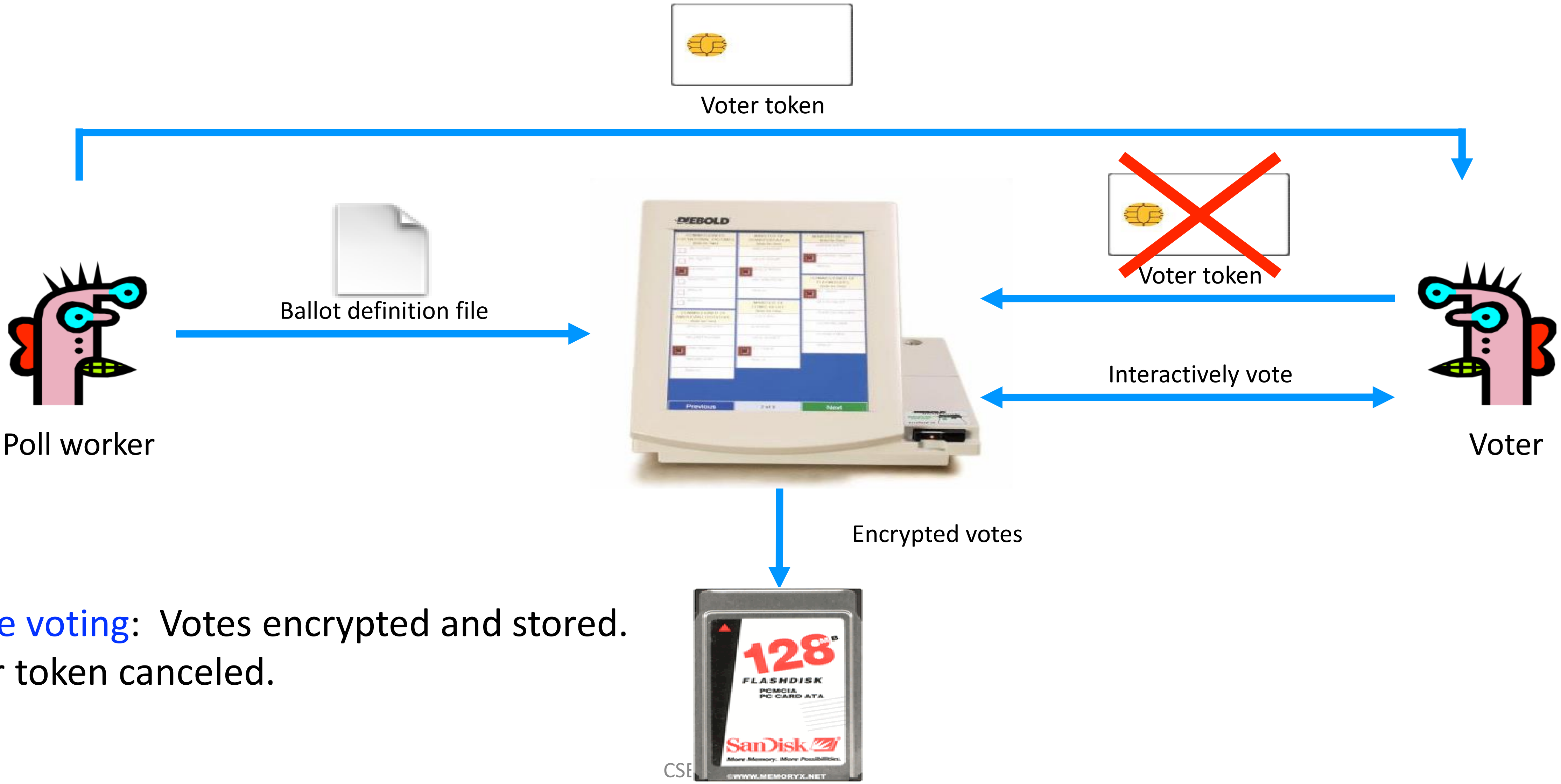
Pre-election: Poll workers load “ballot definition files” on voting machine.

Active Voting



Active voting: Voters obtain single-use tokens from poll workers. Voters use tokens to activate machines and vote.

Active Voting



Voter token

Poll worker

Ballot definition file

Voter token

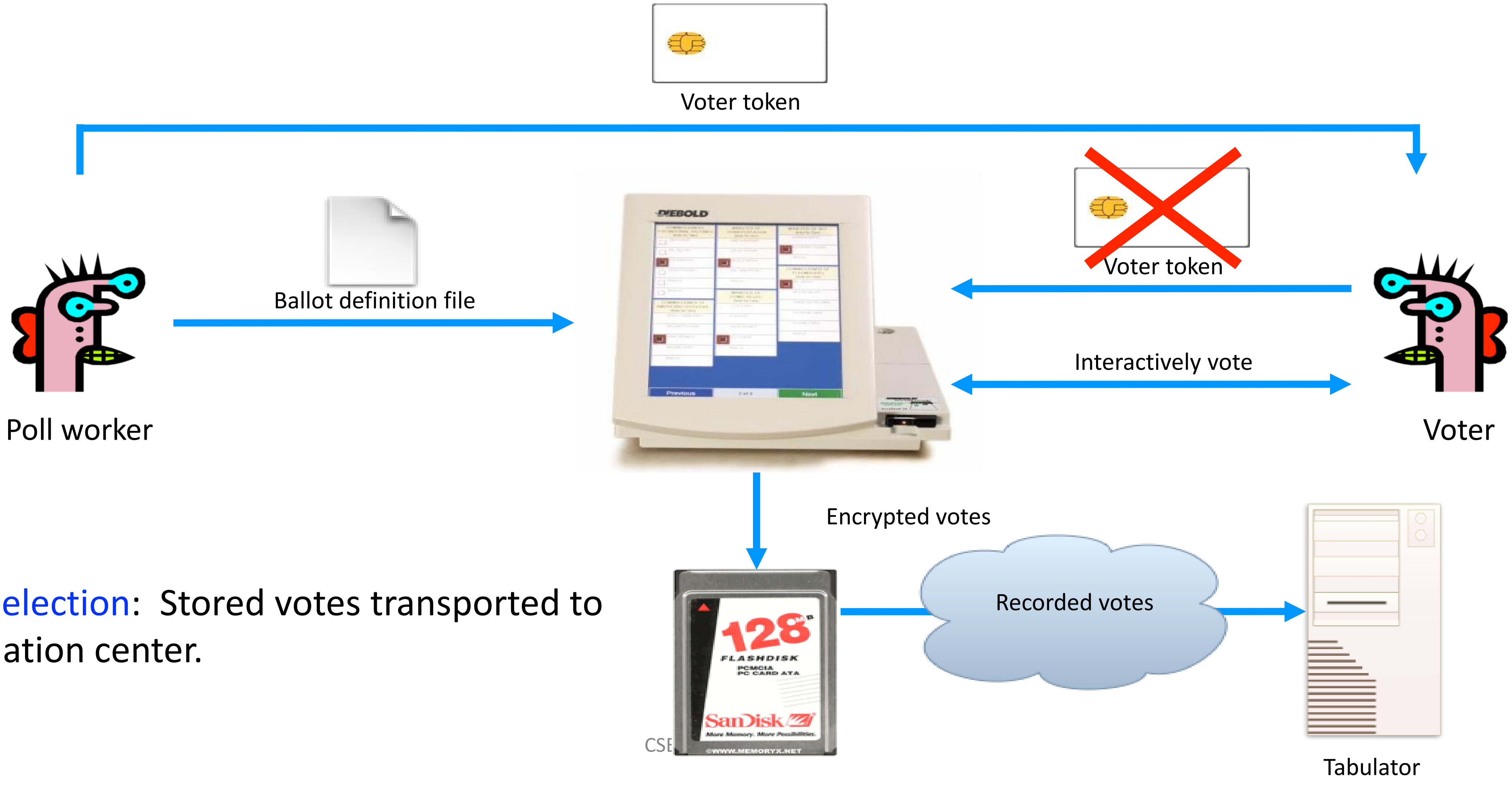
Interactively vote

Voter

Encrypted votes

Active voting: Votes encrypted and stored. Voter token canceled.

Post-Election



Post-election: Stored votes transported to tabulation center.

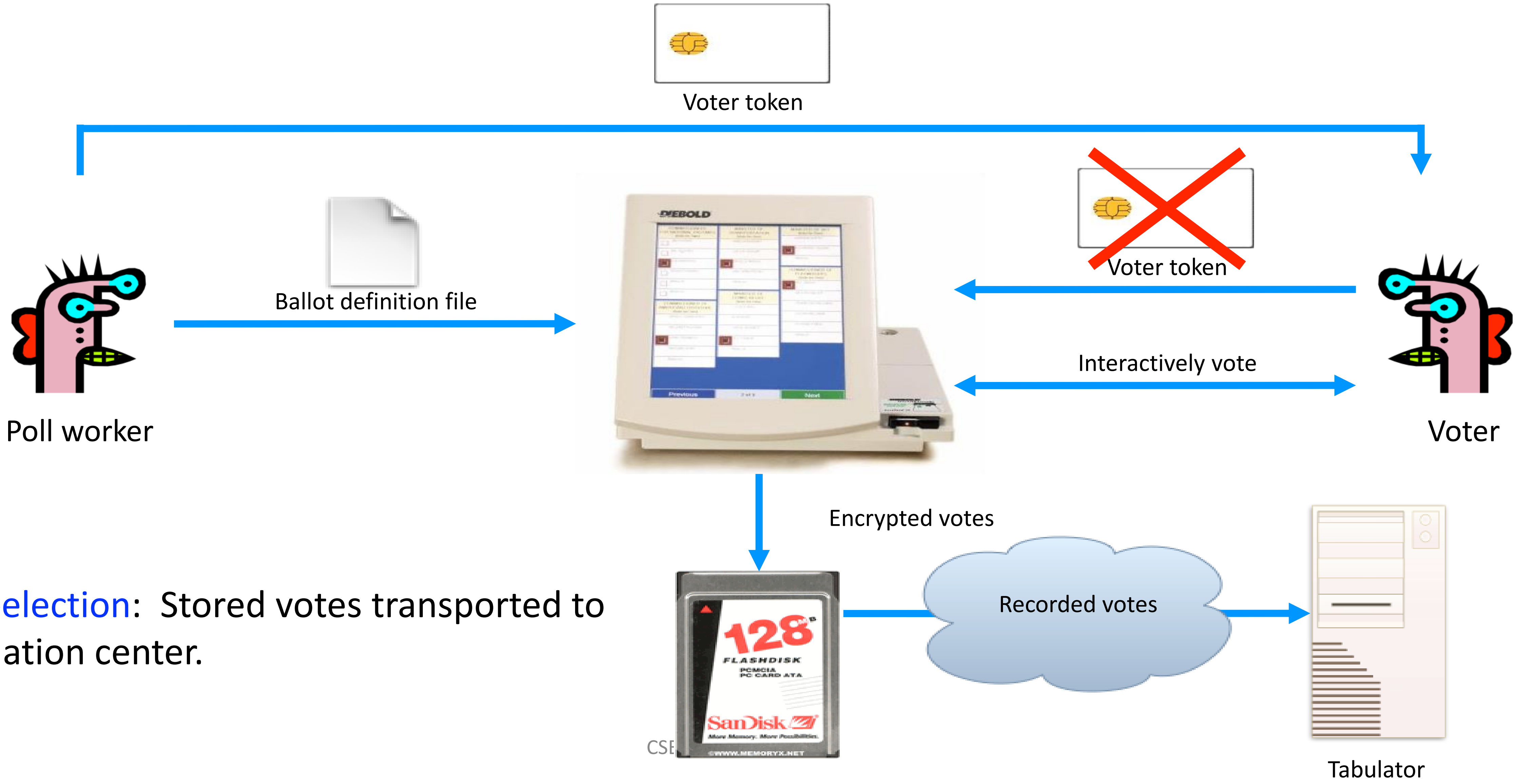
Security and E-Voting (Simplified)

- Functionality goals:
  - Easy to use, reduce mistakes/confusion, make voting more accessible
- Security goals:
  - Pollev.com/dkohlbre

Threat modeling – In detail

- Gradescope
- Fill out the questions while discussing with your neighbors
  - Everyone should submit their own
  - **Polish not required, get down some good ideas!**

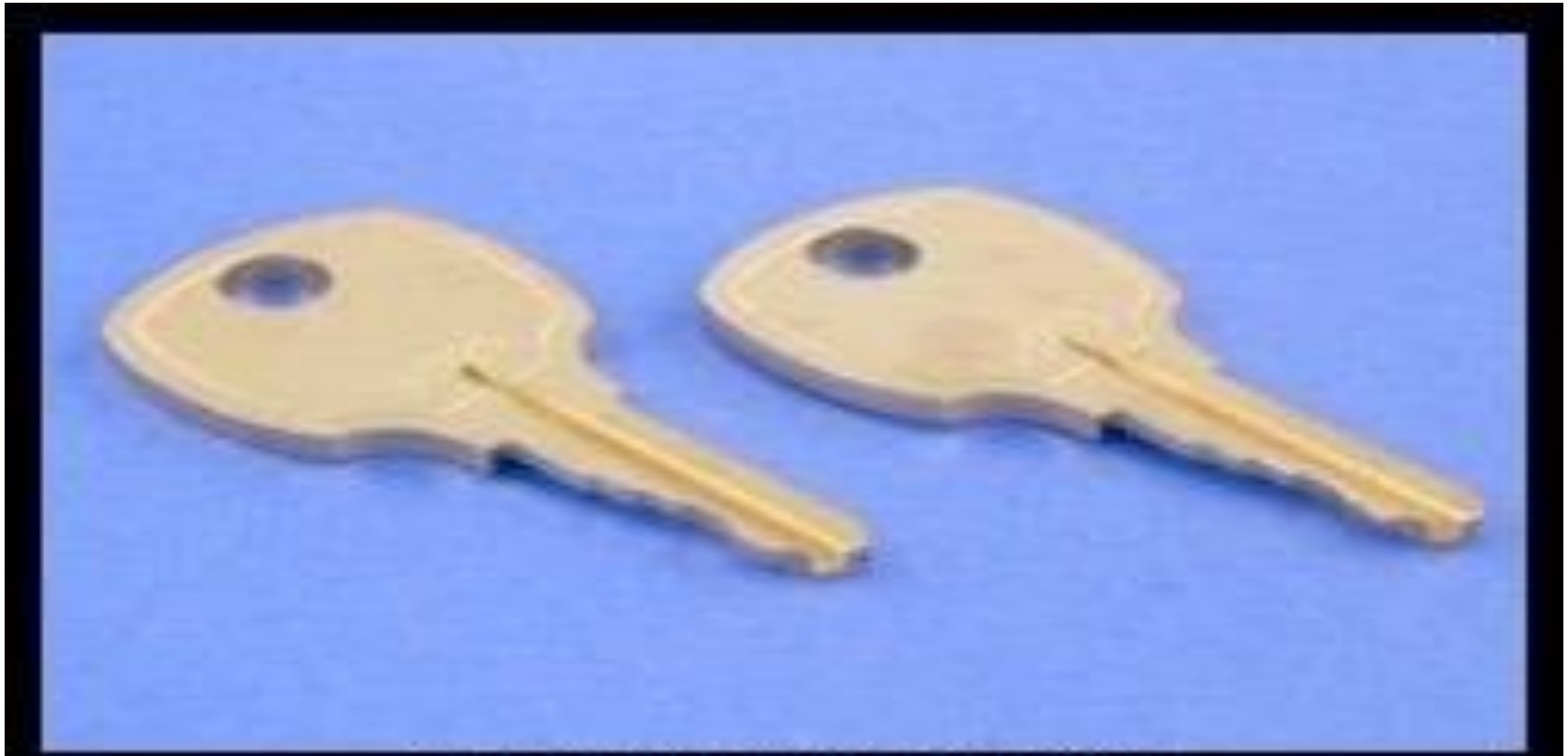
Post-Election



Post-election: Stored votes transported to tabulation center.

What Software is Running?

Problem: An adversary (e.g., a poll worker, software developer, or company representative) able to control the software or the underlying hardware could do whatever they wanted.

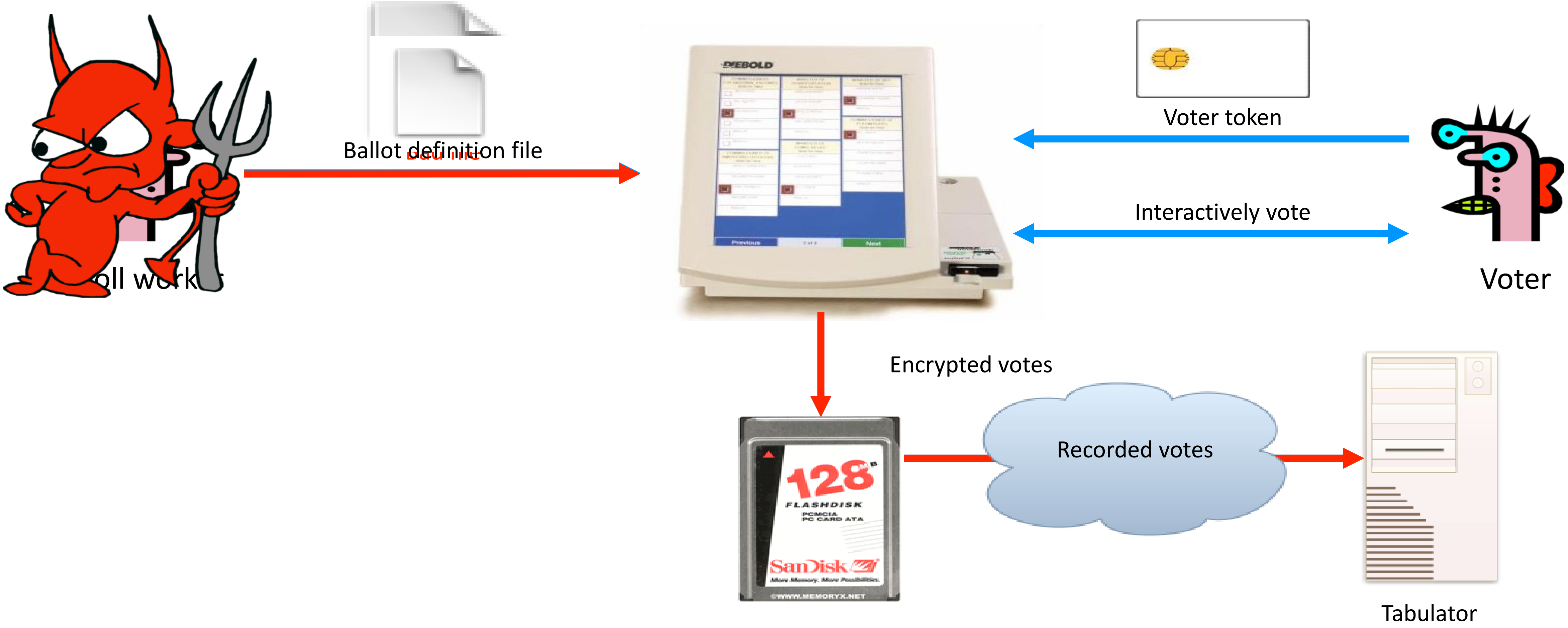


KEYS TO THE KINGDOM

Photo taken from Diebold's online store. The keys that open every Diebold touch-screen voting machine. Working copies have been made from the photo.

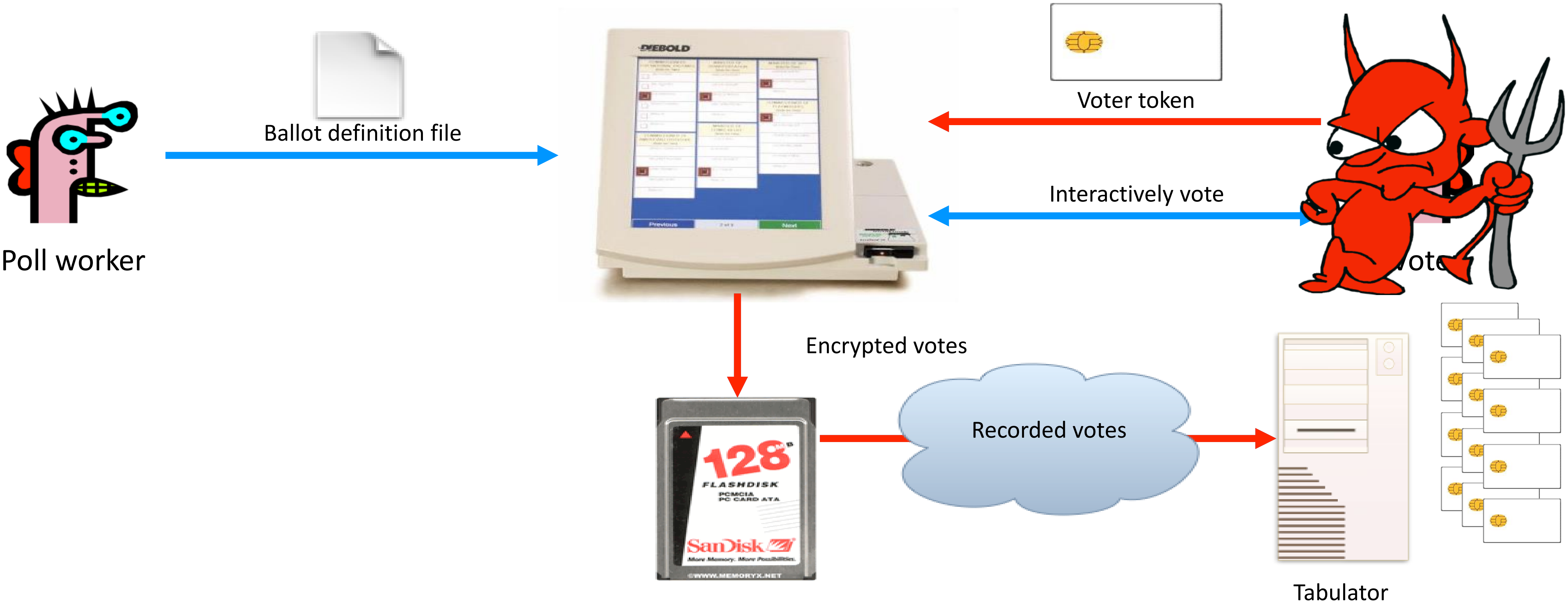
Ballot definition files are not authenticated.

Example attack: A malicious poll worker could modify ballot definition files so that votes cast for “Mickey Mouse” are recorded for “Donald Duck.”



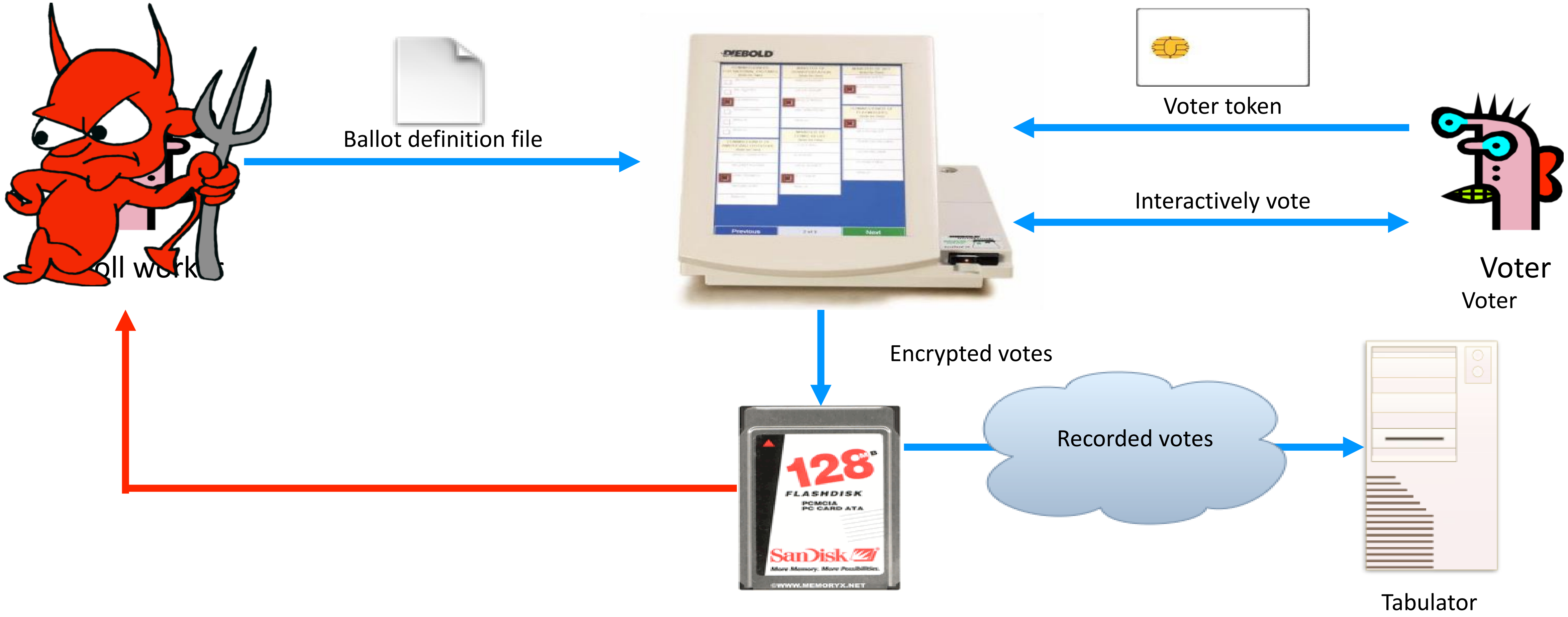
No authentication from token to terminal

Example attack: A regular voter could make their own voter token and vote multiple times.



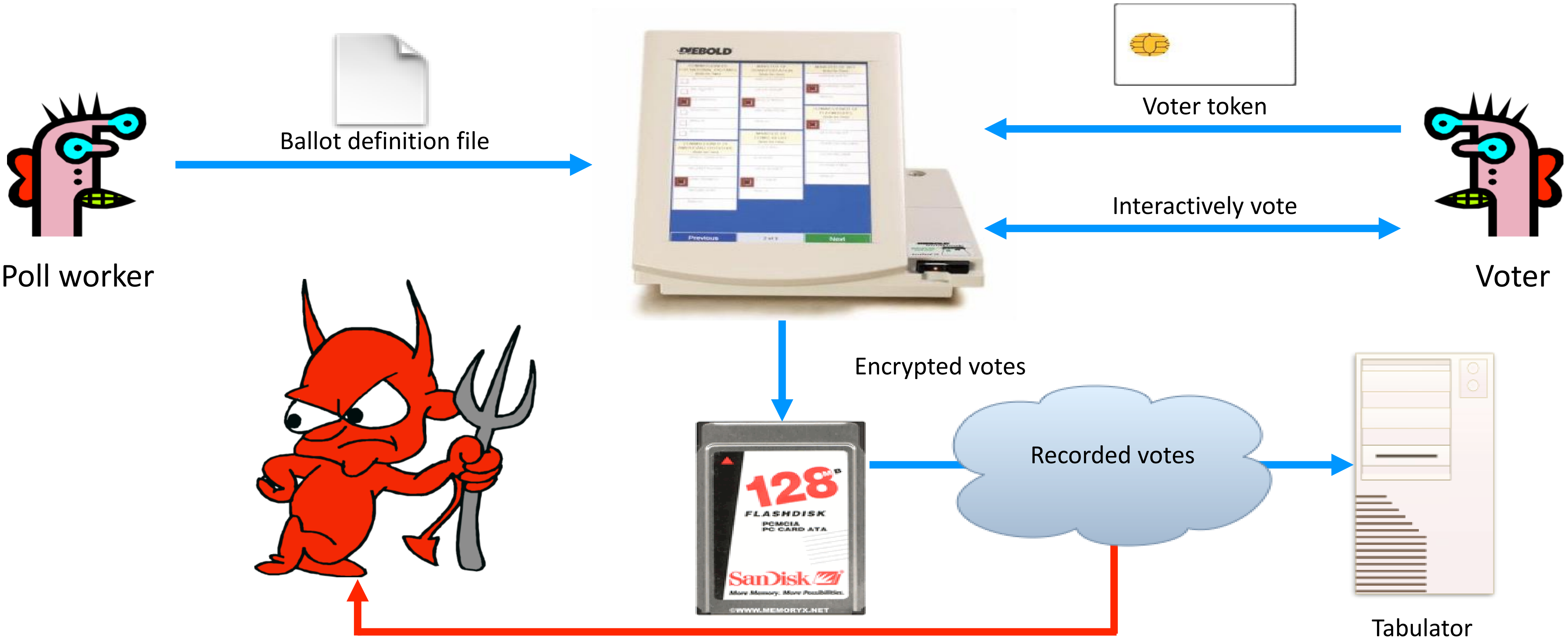
Encryption key (“F2654hD4”) hard-coded 1998. Votes stored in the order cast.

Example attack: A poll worker could determine how voters vote.



Problem: When votes transmitted to tabulator over the Internet or a dialup connection, they are decrypted first; the cleartext results are sent to the tabulator.

Example attack: A sophisticated outsider could determine how voters vote.



Thinking about Defense

Approaches to Defense

- Prevention
    - Stop an attack
- Detection
    - Detect an ongoing or past attack
- Response and Resilience
    - Respond to / recover from attacks

- The threat of a response may be enough to deter some attackers

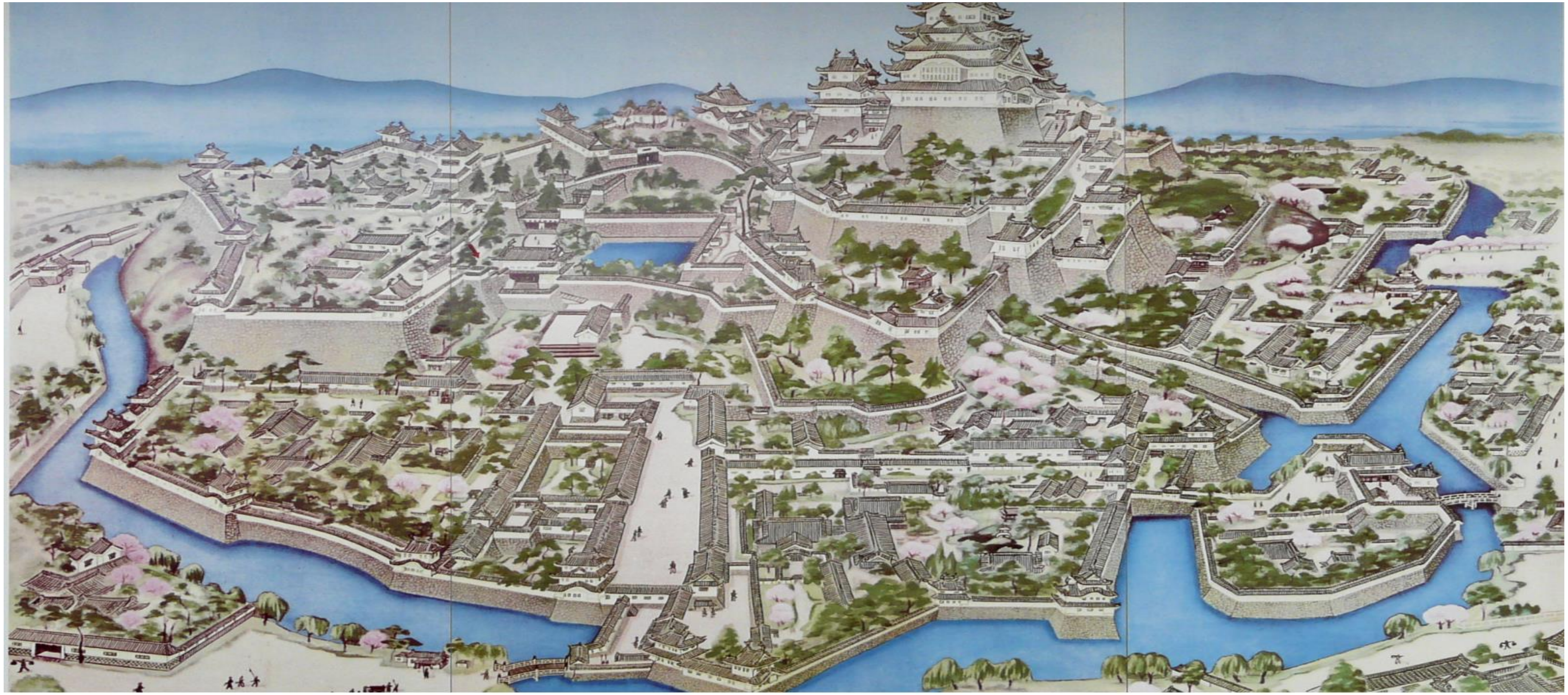
Whole System is Critical

- Securing a system involves a whole-system view
  - Cryptography
  - Implementation
  - People
  - Physical security
  - Everything in between

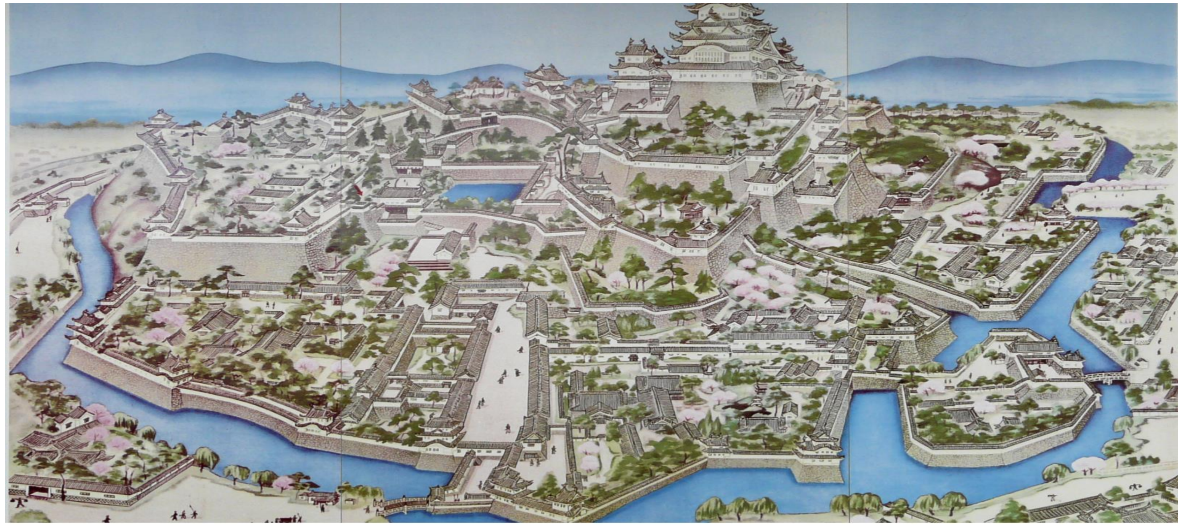
- This is because “security is only as strong as the weakest link,” and security can fail in many places
  - No reason to attack the strongest part of a system if you can walk right around it.

Asymmetric advantages in security

Asymmetric advantages in security

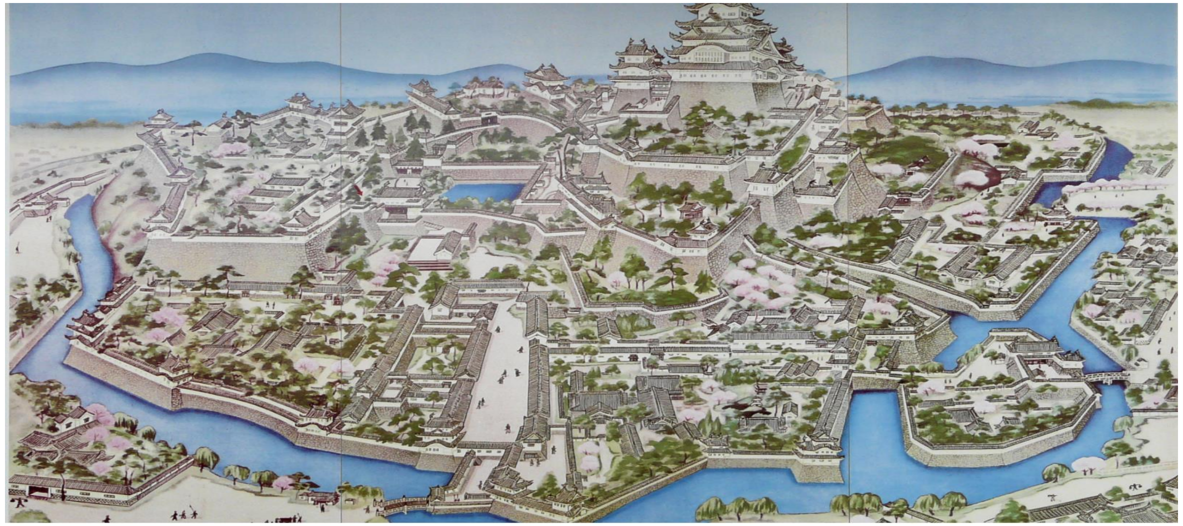


Attacker’s Asymmetric Advantage



- Attacker only needs to win one time, not all the time
- Attackers are professional attackers (maybe)

Defender’s Asymmetric Advantage



- The attacker only succeeds while undetected
- Defender is on ‘home turf’
- Defender has (hopefully) more resources than the attacker
- If the defender can spot them one time, they win

Podcast: https://risky.biz/category/risky-business-news/

From Policy to Implementation

- After you’ve figured out what security means to your application, there are still challenges:
  - Requirements bugs and oversights
    - Incorrect or problematic goals
  - Design bugs and oversights
    - Poor use of cryptography
    - Poor sources of randomness
    - ...
  - Implementation bugs and oversights
    - Buffer overflow attacks
    - ...
  - Is the system **usable**?

Many Participants

- Many parties involved
  - System developers
  - Companies deploying the system
  - The end users
  - The adversaries (possibly one of the above)
- Different parties have different goals
  - System developers and companies may wish to optimize cost
  - End users may desire security, privacy, and usability
    - Side question: Do system developers / companies really understand the needs and values of all their users? Or all stakeholders who might be impacted by the system?
  - But the relationship between these goals is quite complex (e.g., will customers choose features or security?)

Better News

- There are a lot of defense mechanisms
    - We’ll study some, but by no means all, in this course
- It’s important to understand their limitations
    - “If you think cryptography will solve your problem, then you don’t understand cryptography… and you don’t understand your problem” -- Bruce Schneier (… maybe)

The x86(_64)

First technical component of the course

- Understanding classic binary vulnerabilities and exploitation techniques
- We’ll be doing everything on x86 (32 or 64bit)
- Code will be in C
    - Lab 1: very little C
    - FP: More C
- Isn’t this… archaic?

Reminders

- Manual memory management
- Strings are ‘just’ arrays of bytes, *no length field*
  - Strings only end when there is a NUL(NULL) byte.
- Pointers/integers/etc are all just bytes

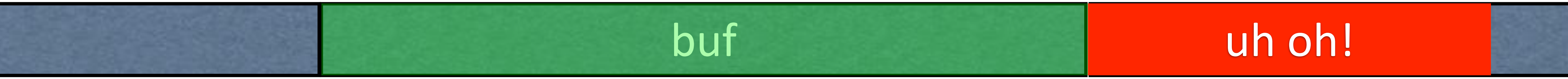
C strings

```
void main(int argc, char* argv[]){
    char string1[32];

    memset(string1, 'a', 32);
    string1[31] = 0x00; // or, '\0'

    printf("String1: %s\n", string1);

    memset(string1, 'a', 32);

    printf("String1, again: %s\n", string1);
}
```

Attacks on Memory Buffers

- Buffer is a pre-defined data storage area inside computer memory (stack or heap)
- Typical situation:
  - A function takes some input that it writes into a pre-allocated buffer.
  - The developer forgets to check that the size of the input isn't larger than the size of the buffer.
  - Uh oh.
    - "Normal" bad input: crash
    - "Adversarial" bad input : take control of execution

Stack Buffers

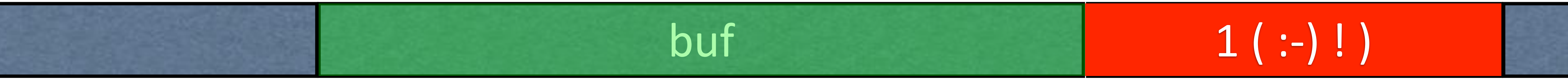


- Suppose Web server contains this function

```
void func(char *str) {
    char buf[126];
    ...
    strcpy(buf,str);
    ...
}
```

- No bounds checking on strcpy()
- If str is longer than 126 bytes
  - Program may crash
  - Attacker may change program behavior

Example: Changing Flags



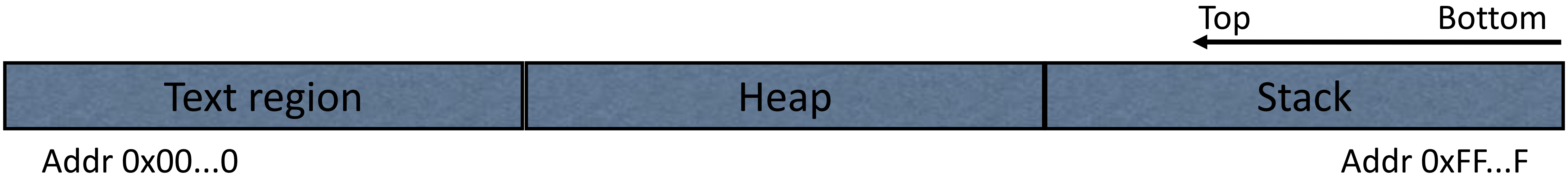
- Suppose Web server contains this function

```
void func(char *str) {
    byte auth = 0;
    char buf[126];
    ...
    strcpy(buf,str);
    ...
}
```

- Authenticated variable non-zero when user has extra privileges
- Morris worm also overflowed a buffer to overwrite an authenticated flag in fingerd

Memory Layout

- Text region: Executable code of the program
- Heap: Dynamically allocated data
- Stack: Local variables, function return addresses; grows and shrinks as functions are called and return



Stack Buffers

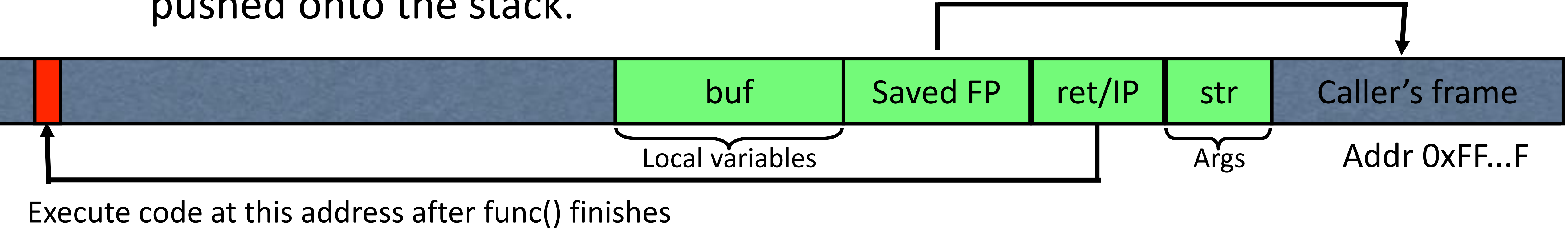
- Suppose Web server contains this function:

```
void func(char *str) {
        char buf[126];
        strcpy(buf,str);
}
```

Allocate local buffer
(126 bytes reserved on stack)

Copy argument into local buffer

- When this function is invoked, a new frame (activation record) is pushed onto the stack.



Execute code at this address after func() finishes

What if Buffer is Overstuffed?

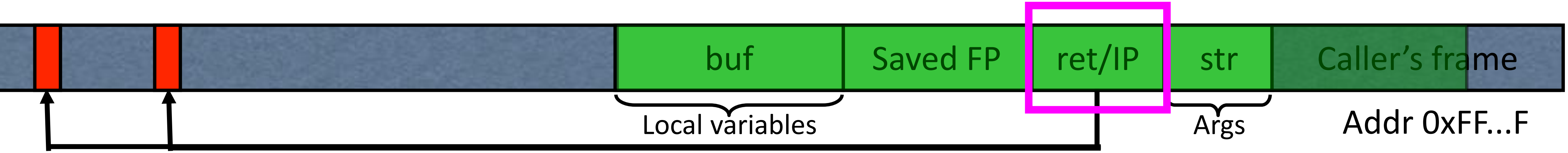
- Memory pointed to by str is copied onto stack…

```
void func(char *str) {
        char buf[126];
        strcpy(buf,str);
}
```

strcpy does NOT check whether the string at *str contains fewer than 126 characters

- If a string longer than 126 bytes is copied into buffer, it will overwrite adjacent stack locations.

This will be interpreted as return address!



buf

Saved FP

ret/IP

str

Caller’s frame

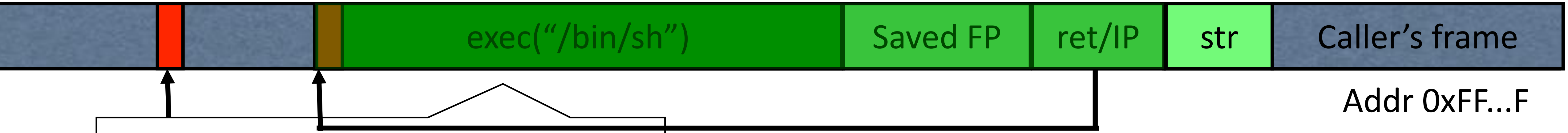
Local variables

Args

Addr 0xFF...F

Executing Attack Code

- Suppose buffer contains attacker-created string
  - For example, str points to a string received from the network as the URL



Attacker puts actual assembly instructions into his input string, e.g., binary code of execve("/bin/sh")

In the overflow, a pointer back into the buffer appears in the location where the system expects to find return address

- When function exits, code in the buffer will be executed, giving attacker a shell **("shellcode")**
  - Root shell if the victim program is setuid root