Introduction

CMSC 414: Computer and Network Security

Spring 2015

What is computer & network security?

- Normally, we are concerned with <u>correctness</u>
  - Does the software achieve the desired behavior?
- Security is a form of correctness
  - Does the software prevent “undesired” behavior?

What is computer & network security?

- Normally, we are concerned with <u>correctness</u>
  - Does the software achieve the desired behavior?
- Security is a form of correctness
  - Does the software prevent “undesired” behavior?

The key difference:

Security involves an adversary
who is active and malicious.

Attackers seek to circumvent protective measures.

What are “undesired” behaviors?

What are “undesired” behaviors?

- Reveals info users wish to hide (confidentiality)
    - Corporate secrets
    - Private data; personally identifying information (PII)

What are “undesired” behaviors?

- Reveals info users wish to hide (confidentiality)
  - Corporate secrets
  - Private data; personally identifying information (PII)

- Modifies information or functionality (integrity)
  - Destroys records
  - Changes data in-flight (think “the telephone game”)
  - Installs unwanted software (spambot, spyware, etc.)

What are “undesired” behaviors?

- Reveals info users wish to hide (confidentiality)
  - Corporate secrets
  - Private data; personally identifying information (PII)

- Modifies information or functionality (integrity)
  - Destroys records
  - Changes data in-flight (think “the telephone game”)
  - Installs unwanted software (spambot, spyware, etc.)

- Denies access to a service (availability)
  - Crashing a website for political reasons
  - Denial of service attack
  - Variant: *fairness*

What are "undesired" behaviors?

- Reveals info users wish to hide (confidentiality)
  - Corporate secrets
  - Private data; personally identifying information (PII)
- Modifies information or functionality (integrity)
  - Destroys records
  - Changes data in-flight (think "the telephone game")
  - Installs unwanted software (spambot, spyware, etc.)
- Denies access to a service (availability)
  - Crashing a website for political reasons
  - Denial of service attack
  - Variant: *fairness*

This is a subset

Attacks are common

From just the past 9 months or so:

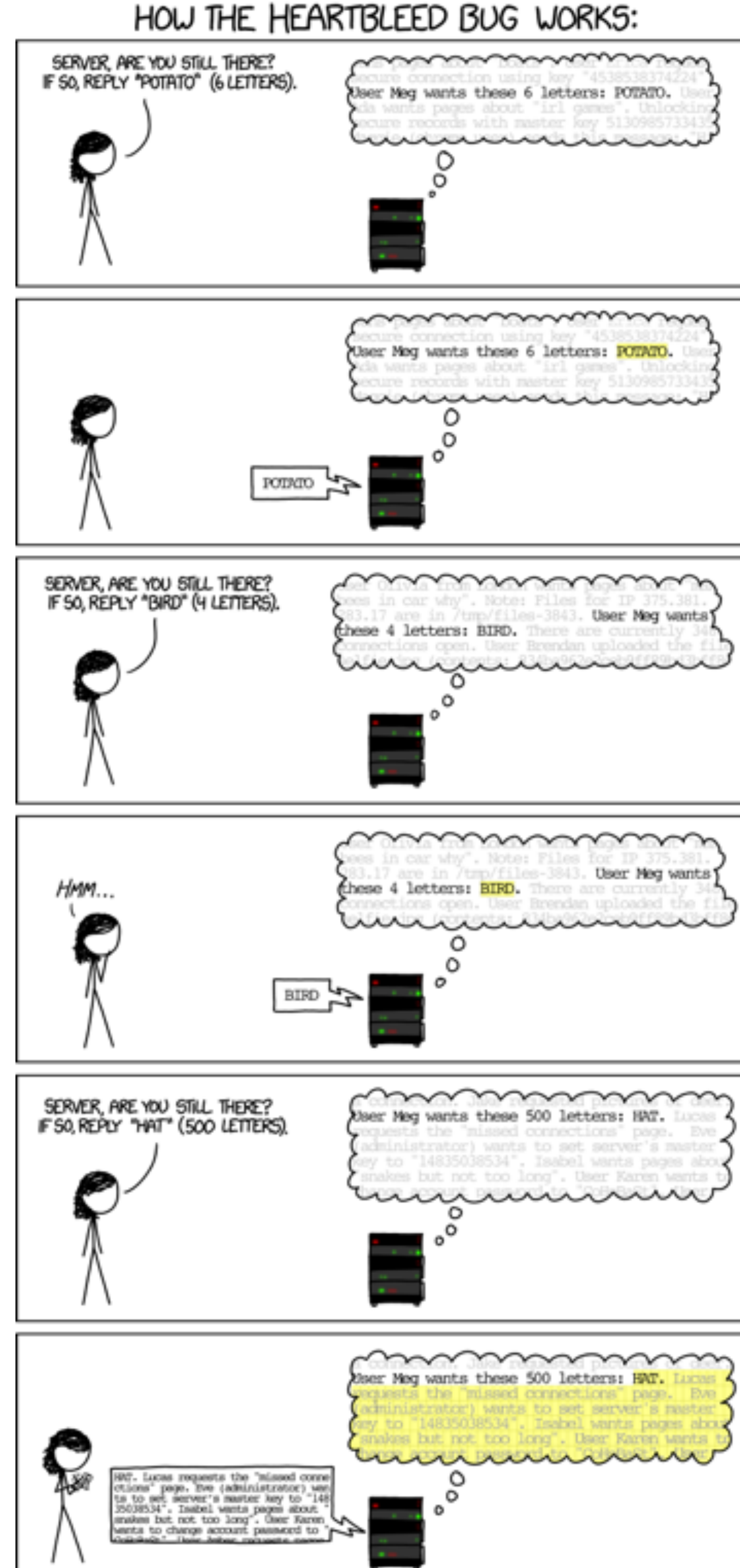
Why are attacks common?

Why are attacks common?

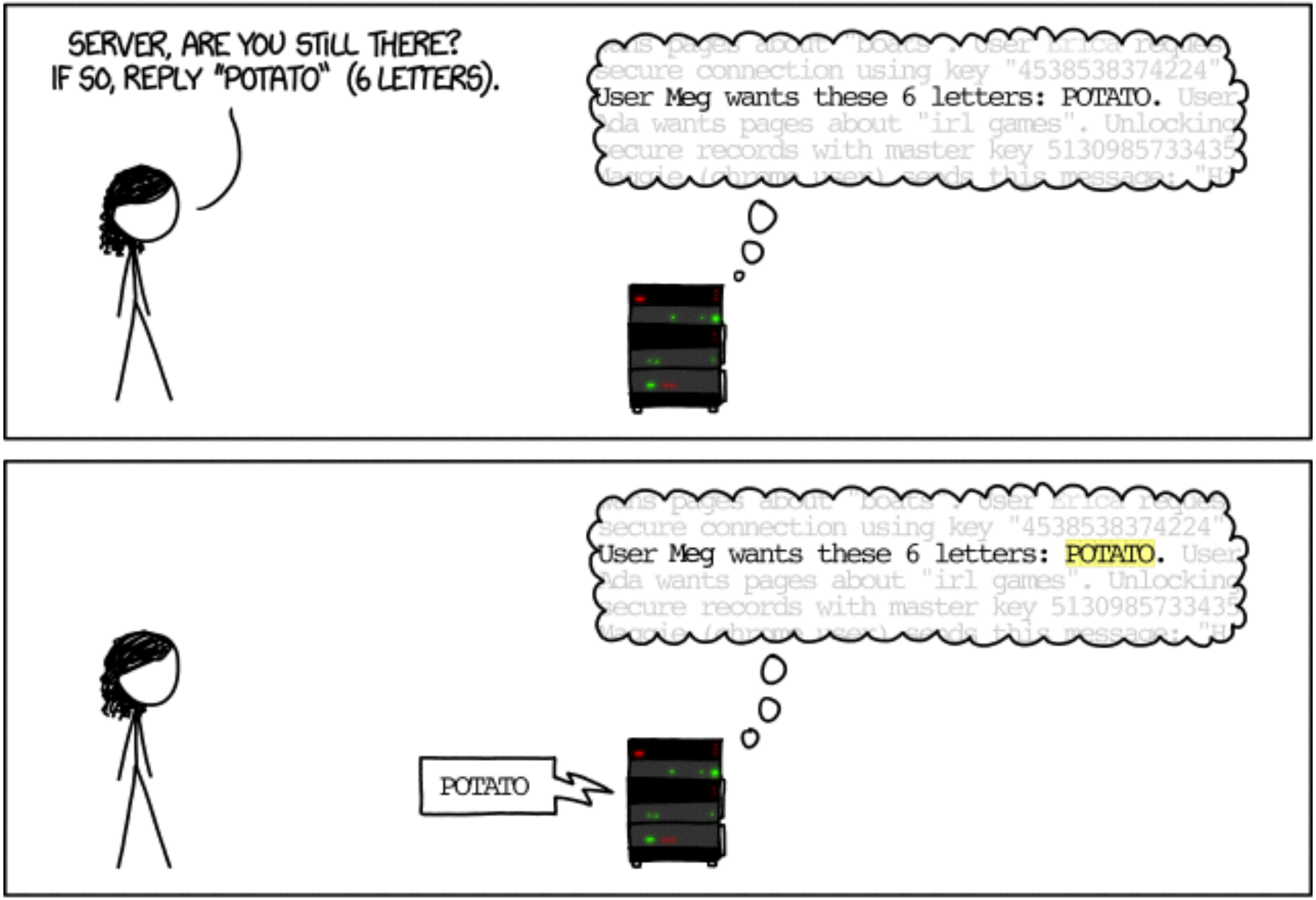
- Security is a property of the systems *we build*
- Many attacks begin by exploiting a vulnerability
  - Vulnerability = software defect that can be exploited to yield an undesired behavior
  - Software defect = the code doesn't "behave correctly"
- Software defects arise due to
  - flaws in the design and/or
  - bugs in the implementation

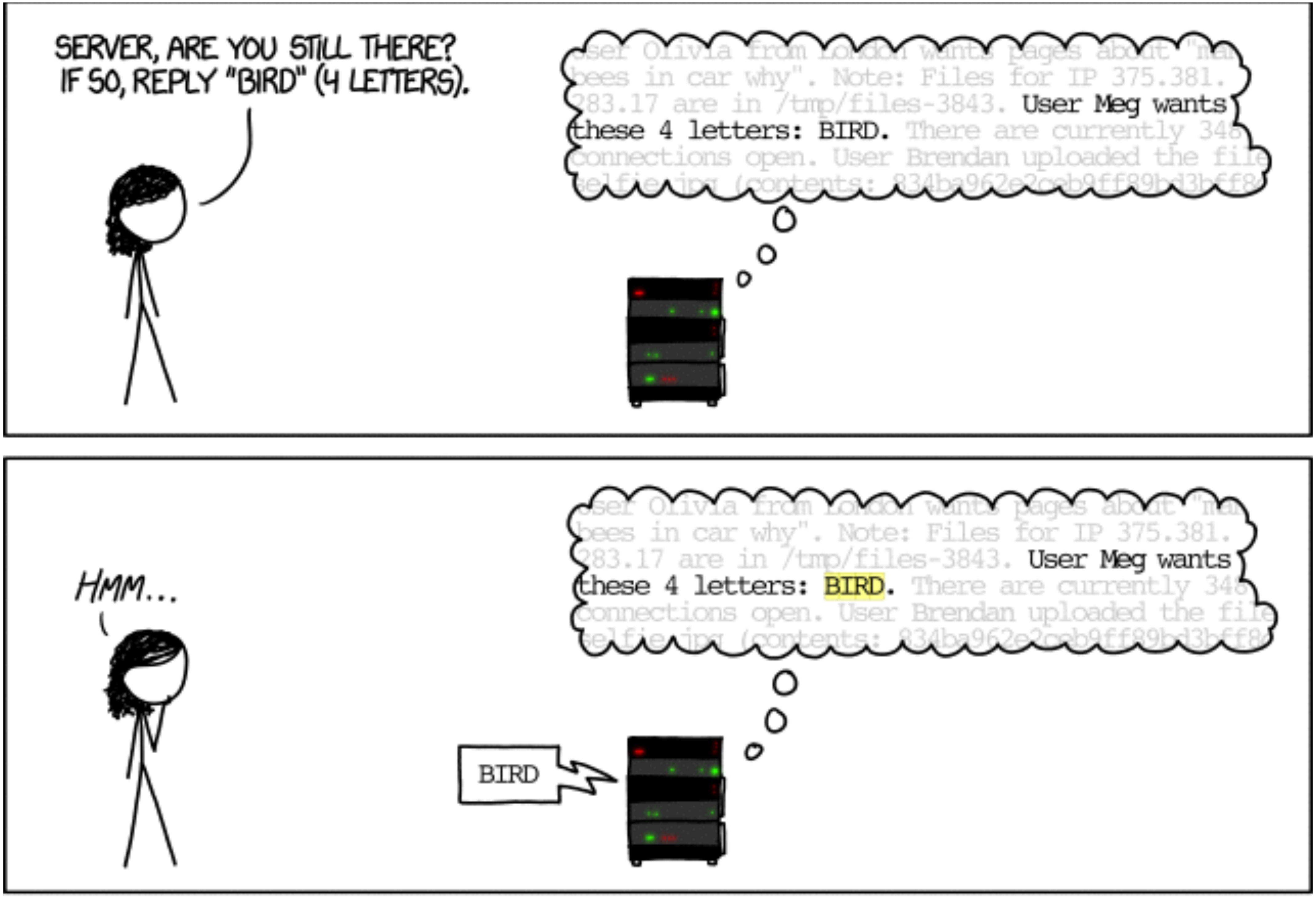
Heartbleed

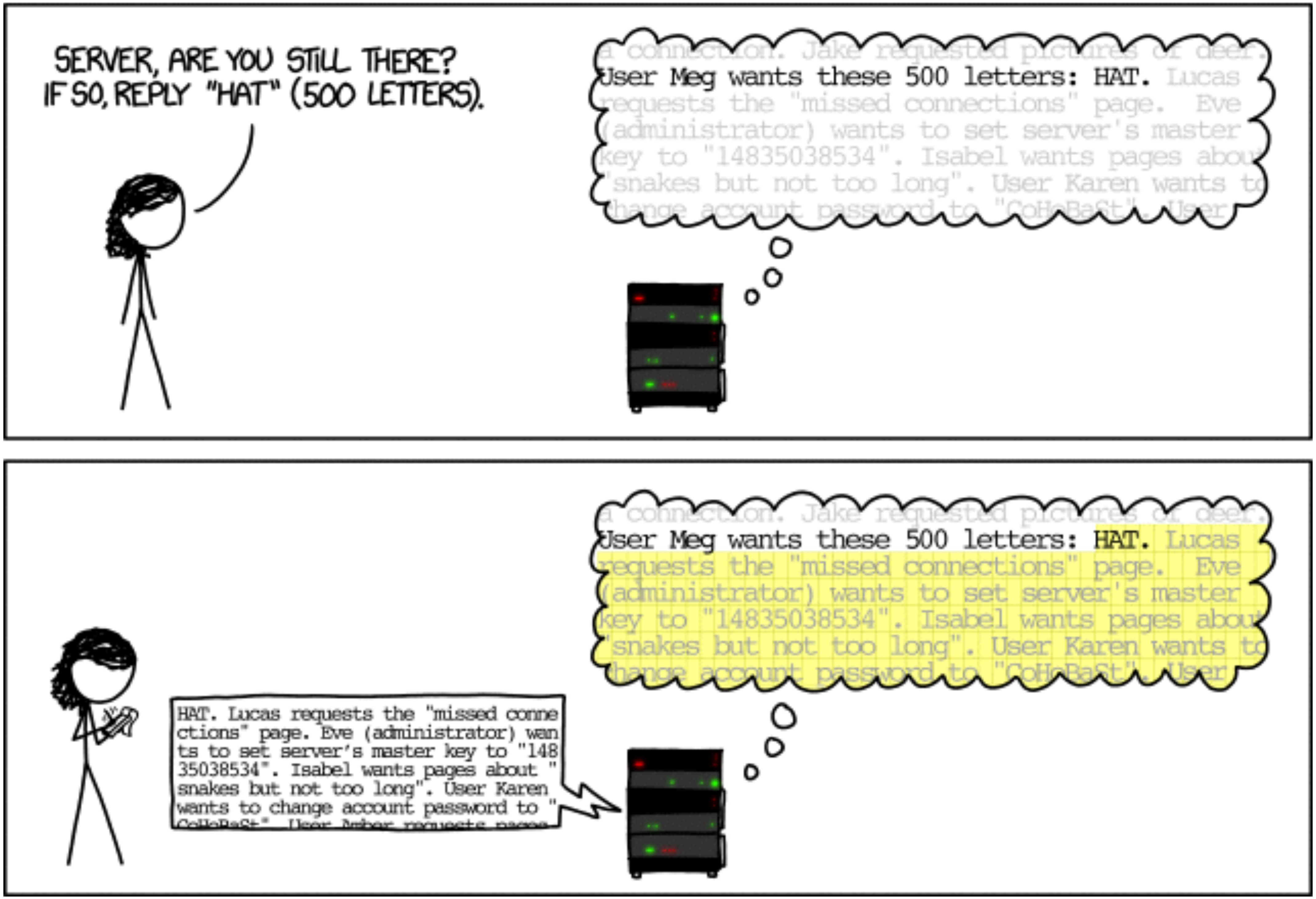
- SSL is the de facto protocol for secure online communication
- Heartbleed was a vulnerability in the most popular SSL server
    - A malformed packet allows you to see server memory
- Fix: don’t let the user just tell you how much data to give back
- This was a design flaw



HOW THE HEARTBLEED BUG WORKS:




SERVER, ARE YOU STILL THERE? IF SO, REPLY "BIRD" (4 LETTERS).
ser Olivia from London wants pages about "ma
bees in car why". Note: Files for IP 375.381.
283.17 are in /tmp/files-3843. User Meg wants
these 4 letters: BIRD. There are currently 346
connections open. User Brendan uploaded the file
selfie.jpg (contents: 834ba962e2ceb9ff89bd3bff8
HMM...
ser Olivia from London wants pages about "ma
bees in car why". Note: Files for IP 375.381.
283.17 are in /tmp/files-3843. User Meg wants
these 4 letters: BIRD. There are currently 346
connections open. User Brendan uploaded the file
selfie.jpg (contents: 834ba962e2ceb9ff89bd3bff8
BIRD

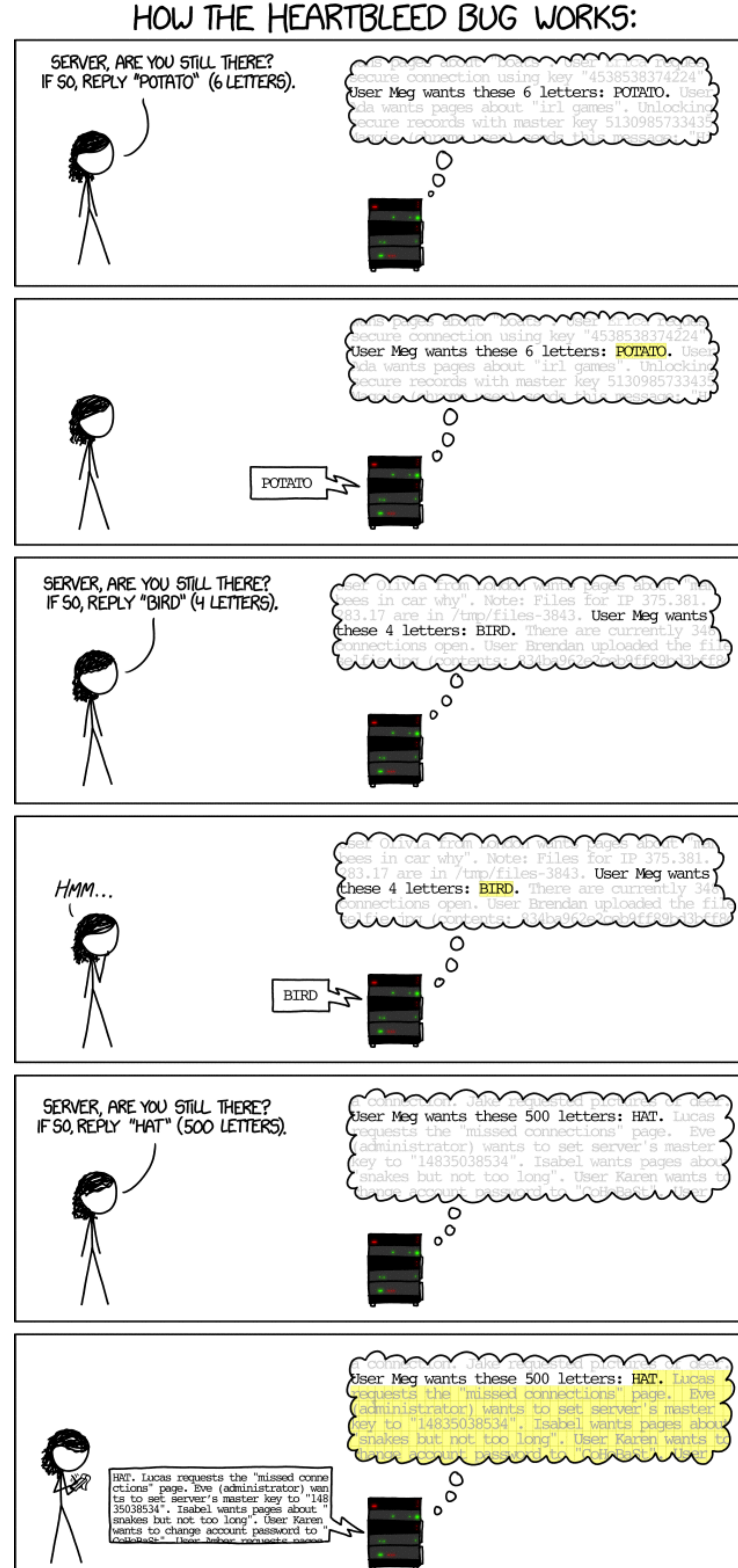


User passwords, private keys, personal information…

~40% of “secure” web servers vulnerable

Heartbleed

- SSL is what gets us encrypted communication with websites
- Heartbleed was a vulnerability in the most popular SSL server
  - A malformed packet allows you to see server memory
- Fix: don't let the user just tell you how much data to give back
- This was a design flaw



RSA 2011 breach

1. **Carefully crafted Flash program**. When run by the vulnerable Flash player, allows the attacker to execute arbitrary code on the running machine.

2. This program could be **embedded in an Excel spreadsheet**, and run automatically when the spreadsheet was opened.

3. Spreadsheet **attached to an email**, masquerading as a trusted party ("spear phishing")
   - You can forge any "From" address

Why are attacks common?

- Because attacks derive from design flaws or implementation bugs
- But all software has bugs: so what?
- A *normal user* never sees most bugs
  - Post-deployment bugs are usually rare corner cases
- Too expensive to fix every bug
  - Only fix what's likely to affect normal users

Why are attacks common?

Attackers are not normal users

- Normal users avoid bugs/flaws
- Adversaries seek them out and try to *exploit* them

Why are attacks common?

Attackers are not normal users

- Normal users avoid bugs/flaws
- Adversaries seek them out and try to *exploit* them

This extends beyond software:

Attacks are possible even with perfect software

Why are attacks common?

Because it's **profitable**

And because a system is
*only as secure as its **weakest link***

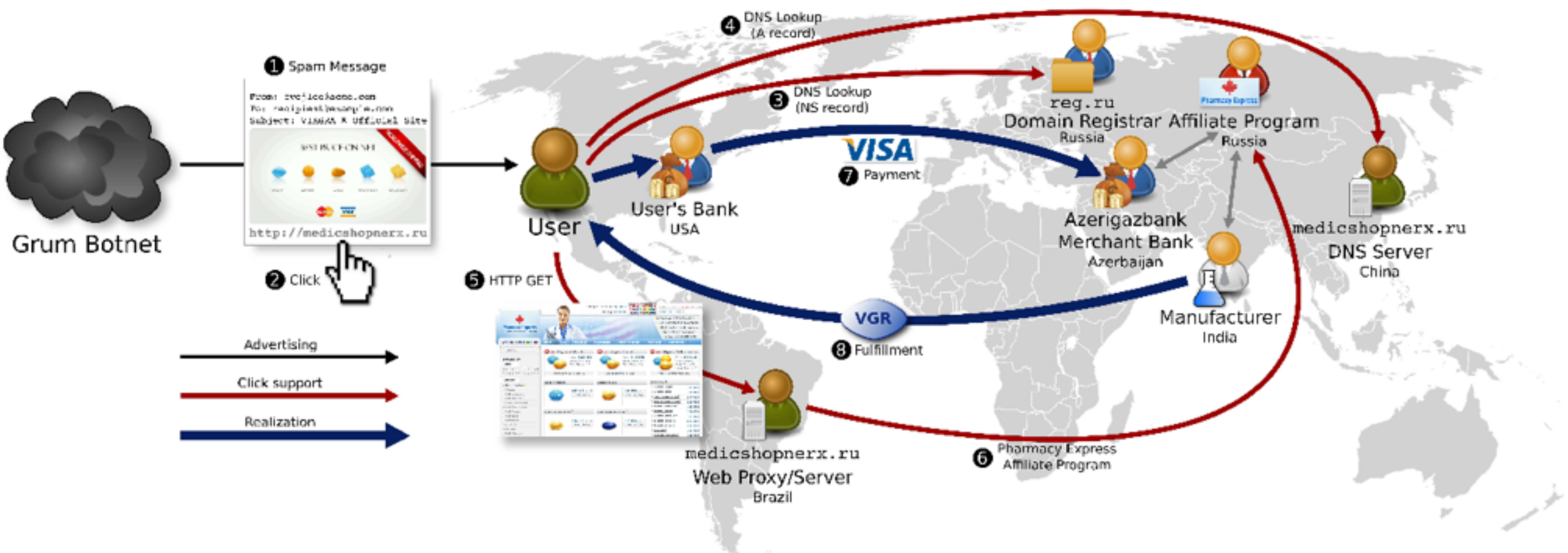


Figure 1: Infrastructure involved in a single URL's value chain, including advertisement, click support and realization steps.

In order to achieve security, we must:

Be able to eliminate bugs and design flaws and/or make them harder to exploit.

In order to achieve security, we must:

Be able to eliminate bugs and design flaws and/or make them harder to exploit.

Be able to think like attackers.

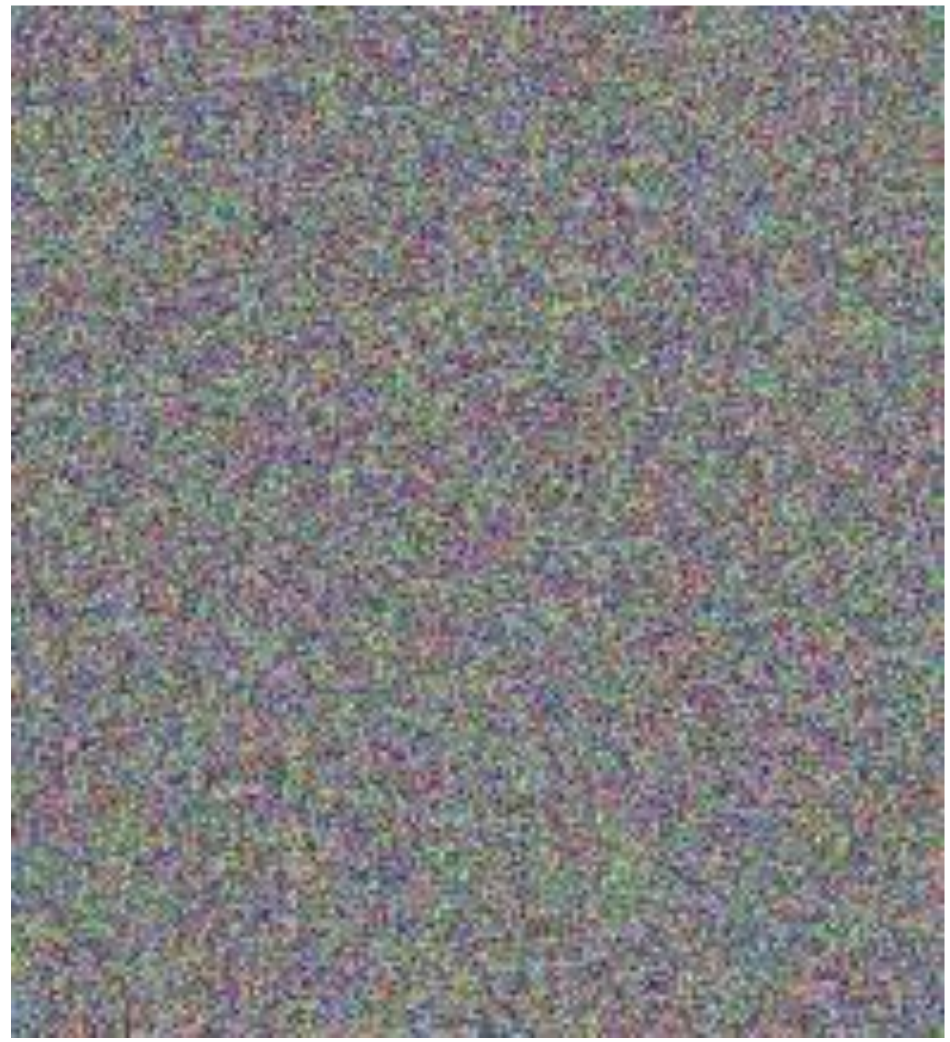
In order to achieve security, we must:

Be able to eliminate bugs and design flaws and/or make them harder to exploit.

Be able to think like attackers.

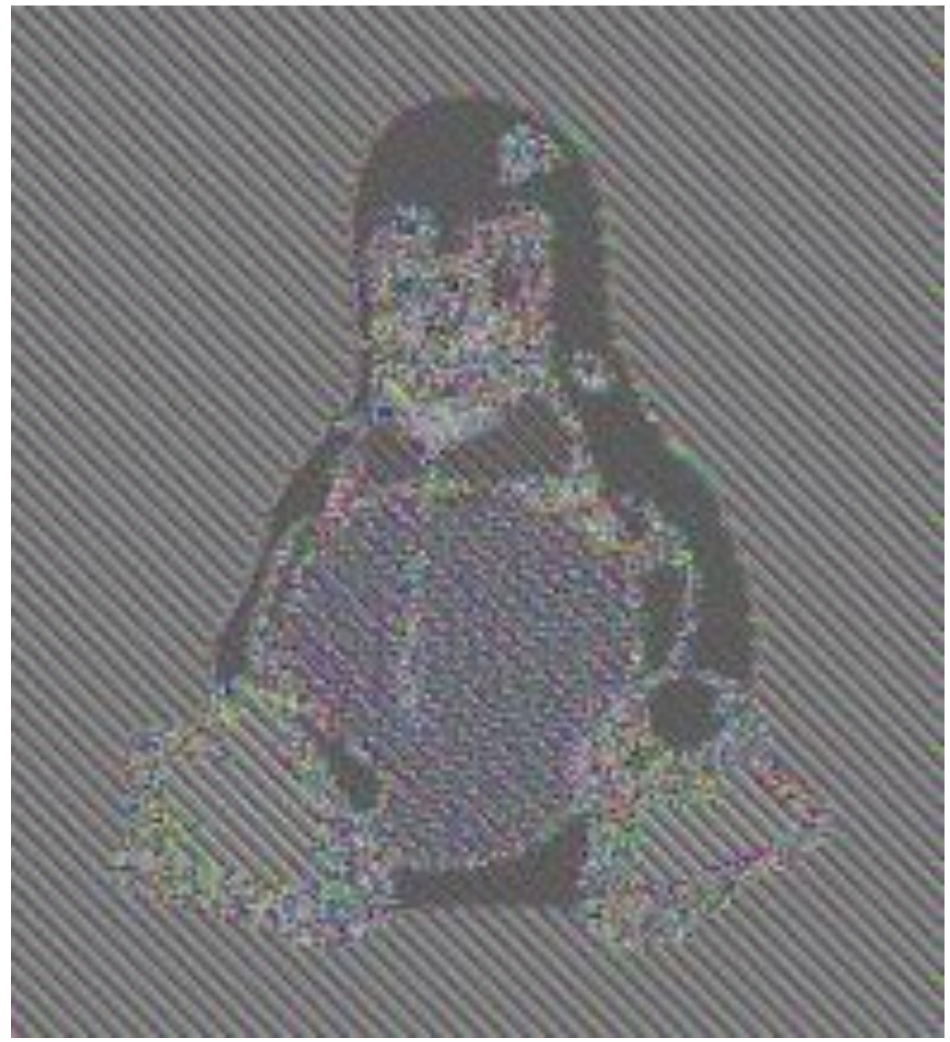
Develop a foundation for deeply understanding the systems we use and build.

Widespread misuse of crypto



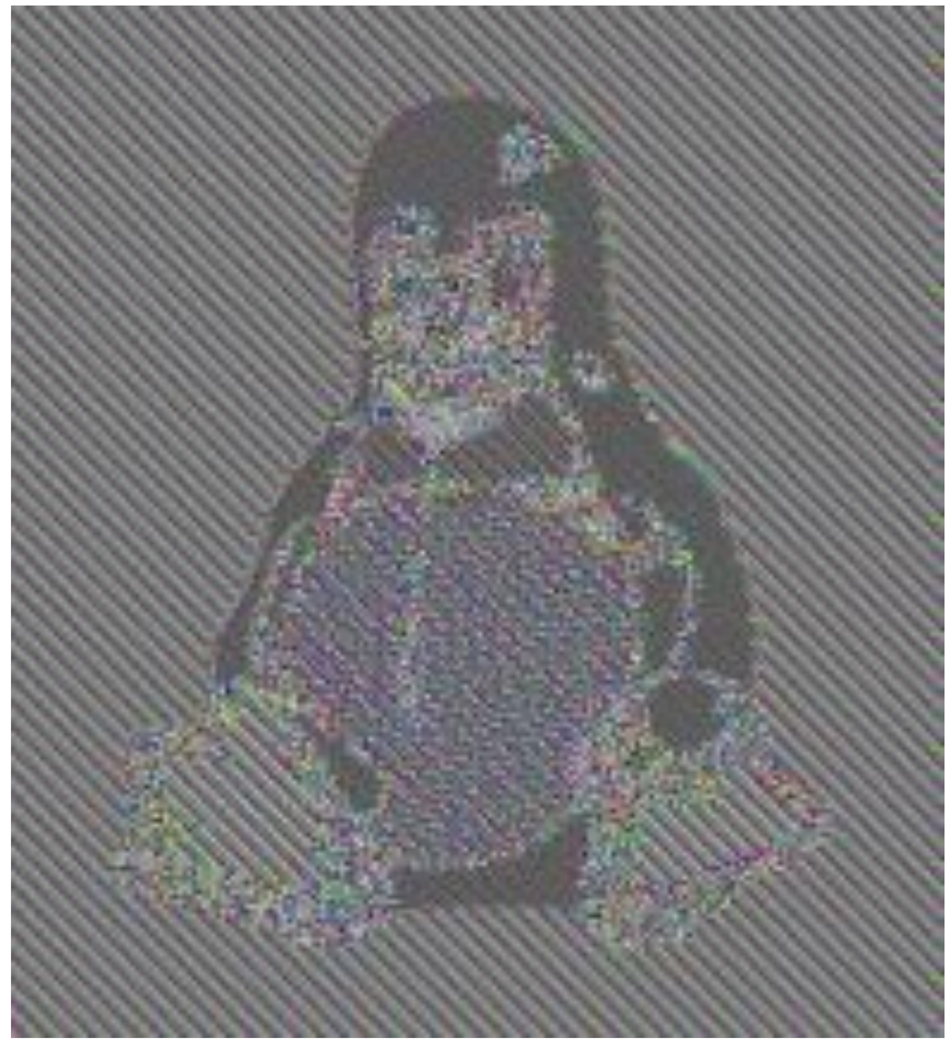
This is an encrypted image

Widespread misuse of crypto



This is an encrypted image

Widespread misuse of crypto



This is an encrypted image

50% of Android apps that use crypto encrypt in this manner

In order to achieve security, we must:

Be able to eliminate bugs and design flaws and/or make them harder to exploit.

Be able to think like attackers.

Develop a foundation for deeply understanding the systems we use and build.

In order to achieve security, we must:

Be able to eliminate bugs and design flaws and/or make them harder to exploit.

Be able to think like attackers.

Develop a foundation for deeply understanding the systems we use and build.

Software	**Hardware**	**Protocols**
Users	**Law**	**Economics**

The Goals of CMSC 414

Be able to eliminate bugs and design flaws
and/or make them harder to exploit.

Be able to think like attackers.

Develop a foundation for deeply understanding
the systems we use and build.

Software **Hardware** **Protocols**

Users **Law** **Economics**

This time

- What is security?
- Administrative
- Analyzing a system's security
  1. Summarize the system
  2. Identify the assets
  3. Identify the adversaries & threats
  4. Identify the vulnerabilities
- Trusting trust

Administrative

Communicating

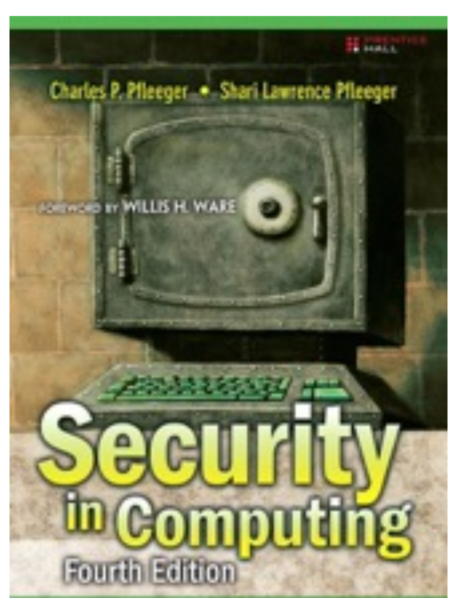
- Resources and all this info will be on the class website
  - http://www.cs.umd.edu/class/2015/cmsc414
- Who
  - Me: Dave Levin (dml@cs.umd.edu)
  - TAs: Frank Cangialosi, Yi Qian, and Chengxi Ye
  - Office hours are on the website
  - If my office hours don't work for you, email and set up a time
- We will be using Piazza
  - You should have been added; let me know if you haven't

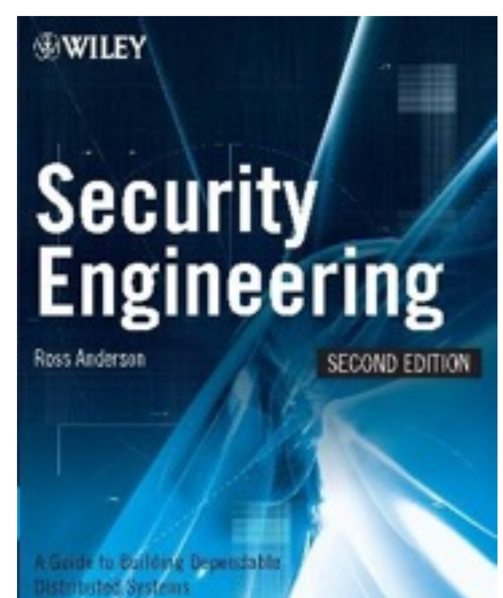
Administrative

Textbooks

- None required
  - Mostly in-class and papers posted on website

- Recommended texts, if you are so inclined
  - “Security in Computing”, Pfleeger & Pfleeger
  - “Security Engineering”, Ross Anderson
    - Free online: http://www.cl.cam.ac.uk/~rja14/book.html





Administrative

Outside reading

- The best way to learn is to reinforce
- *Lots* of security resources (something is always breaking).
  - Krebs on security
  - Bruce Schneier's blog
  - Any other favorites? Let us know on Piazza

What’s in this course?

What's in this course?

Software Security

How do we build software that is secure?

Memory safety

Malware

Web security

Static analysis

Design principles

What's in this course?

Software Security

Crypto

What it is, and how to use it responsibly

A black-box approach to crypto

Designing protocols that *use* crypto

Authentication: proving who you are

Anonymity: hiding who you are

What’s in this course?

Software Security

Crypto

Network Security

Attacks on TCP & DNS

Botnets

Underground spam economies

How to build secure networked systems.

What's in this course?

Software Security	***How do we build software that is secure?***
Crypto	***What it is, and how to use it responsibly.***
Network Security	***How to build secure networked systems.***

Attacks and defenses across all of these

Brief Listing of the Top 25

This is a brief listing of the Top 25 items, using the general ranking.

NOTE: 16 other weaknesses were considered for inclusion in the Top 25, but their general scores were not high enough. They are listed in a separate "On the Cusp" page.

Rank	Score	ID	Name
[1]	93.8	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
[2]	83.3	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
[3]	79.0	CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
[4]	77.7	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
[5]	76.9	CWE-306	Missing Authentication for Critical Function
[6]	76.8	CWE-862	Missing Authorization
[7]	75.0	CWE-798	Use of Hard-coded Credentials
[8]	75.0	CWE-311	Missing Encryption of Sensitive Data
[9]	74.0	CWE-434	Unrestricted Upload of File with Dangerous Type
[10]	73.8	CWE-807	Reliance on Untrusted Inputs in a Security Decision
[11]	73.1	CWE-250	Execution with Unnecessary Privileges
[12]	70.1	CWE-352	Cross-Site Request Forgery (CSRF)

Brief Listing of the Top 25

This is a brief listing of the Top 25 items, using the general ranking.

NOTE: 16 other weaknesses were considered for inclusion in the Top 25, but their general scores were not high enough. They are listed in a separate "On the Cusp" page.

Rank	Score	ID	Name
[1]	93.8	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
[2]	83.3	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
[3]	79.0	CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
[4]	77.7	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
[5]	76.9	CWE-306	Missing Authentication for Critical Function
[6]	76.8	CWE-862	Missing Authorization
[7]	75.0	CWE-798	Use of Hard-coded Credentials
[8]	75.0	CWE-311	Missing Encryption of Sensitive Data
[9]	74.0	CWE-434	Unrestricted Upload of File with Dangerous Type
[10]	73.8	CWE-807	Reliance on Untrusted Inputs in a Security Decision
[11]	73.1	CWE-250	Execution with Unnecessary Privileges
[12]	70.1	CWE-352	Cross-Site Request Forgery (CSRF)

Ethics and legality

- You will be learning about (and implementing and *launching*) attacks, many of which are in active use today.
- ***This is not an invitation to use them without the explicit written consent of all parties involved***
- If you want to try something out, then *let me know* and I will try to help create a safe environment
- This is not just a question of ethics; to do otherwise would risk violating UMD policies and MD/USA laws

Prerequisite knowledge

- You should be reasonably proficient in C and Unix
- You should also be creative and resourceful (those who try to attack your systems will be!)
- Otherwise, this course **won't** require any prior knowledge in networking or crypto

What’re grades based on?

- Grade breakdown
  - Projects (5 x ~10% each)
  - Midterms (2 x 12% each)
  - Final (25%)
  - Meet your instructor (1%)

Meet your instructor (that's me!)

- You come by my office at some point during the semester and we chat

- Gives me a chance to get to know each of you, learn about your interests, chat plans/research…

- Again: if you are booked during my office hours, just email me to set up a time.

Midterms & Exams

Expected dates

Midterm #1:	Midterm #2:	Final exam:
Mar. 5	Apr. 14	May 18
12%	**12%**	**25%**

Please see the syllabus for information about excused absences

A brief whirlwind tour of some things to come

How do you know you’re really talking to bankofamerica.com?

Brief, high-level walkthrough of certificates

Is anything really “secure”?

Is anything really "secure"?

- Security requires context
  - What is the ***threat model***? What can the attacker do?
  - What are the ***assets*** you seek to protect?
  - Whom and what do you ***trust***?

Is anything really “secure”?

- Security requires context
  - What is the ***threat model***? What can the attacker do?
  - What are the ***assets*** you seek to protect?
  - Whom and what do you ***trust***?
- “Trust no one!”
  - That’s the spirit!
  - But how did you compile your code again?
  - Who built your OS? Your hardware?…

Is anything really “secure”?

- Security requires context
  - What is the ***threat model***? What can the attacker do?
  - What are the ***assets*** you seek to protect?
  - Whom and what do you ***trust***?

- “Trust no one!”
  - That’s the spirit!
  - But how did you compile your code again?
  - Who built your OS? Your hardware?…

Required reading
“Reflections on Trusting Trust”
Ken Thompson

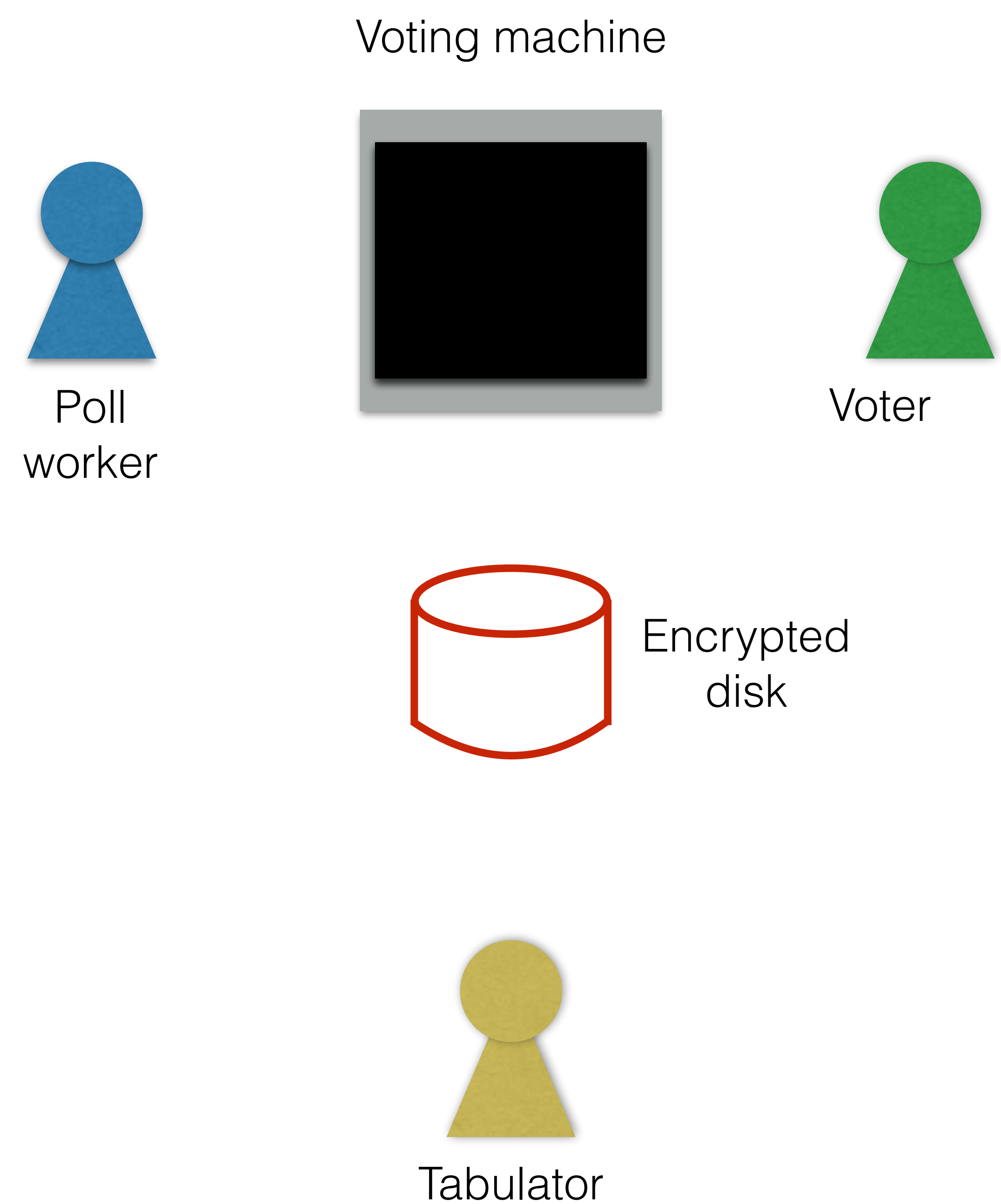
Analyzing security

“Security mindset”

- One of the goals for this course is to develop a “security mindset”
- That is, the ability to view a potentially large, complex system and be able to reason about:
  - What are some of the potential security threats?
  - What are the hidden assumptions? (and are the explicit assumptions likely to be true?)
  - How can we mitigate the risks of the system?
- Be creative! (attackers will be)

E-voting analysis

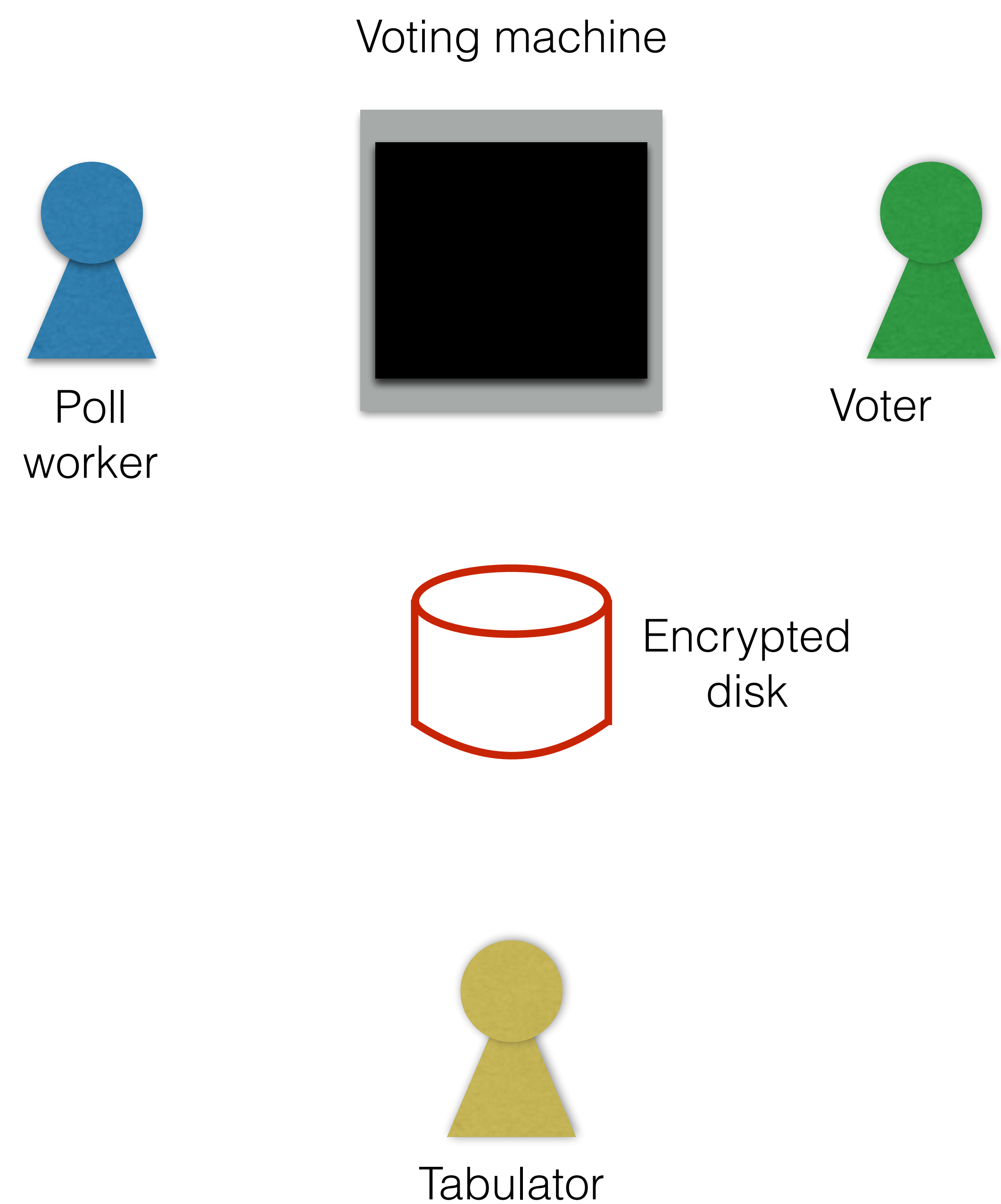
1. Summarize the system as clearly and concisely as possible



E-voting analysis

1. Summarize the system as clearly and concisely as possible

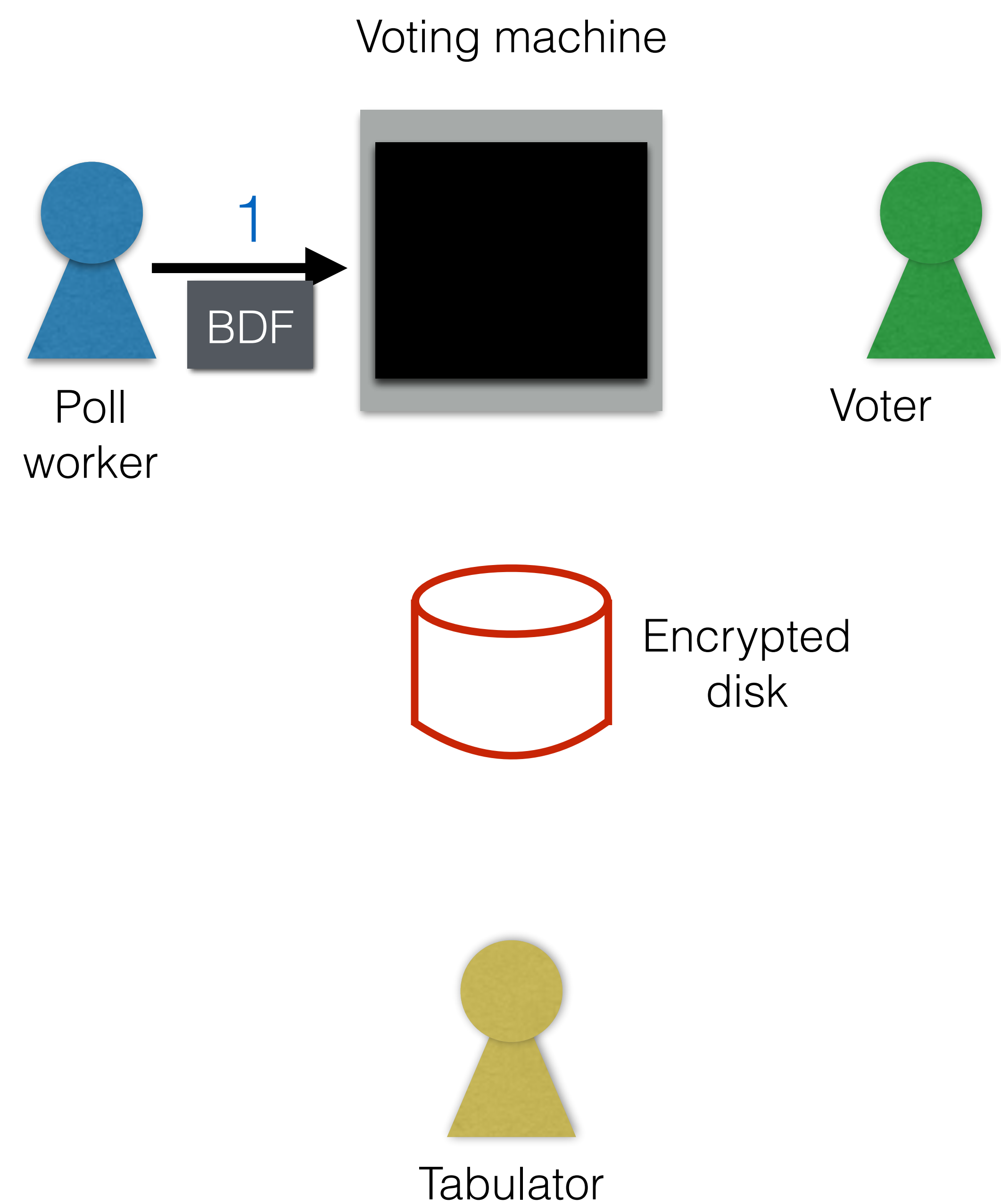
1. Pre-election phase
   - Poll worker loads a “ballot definition file” (defines who’s running, colors on the screen, and many more things) on the voting machines with, e.g., USB



E-voting analysis

1. Summarize the system as clearly and concisely as possible

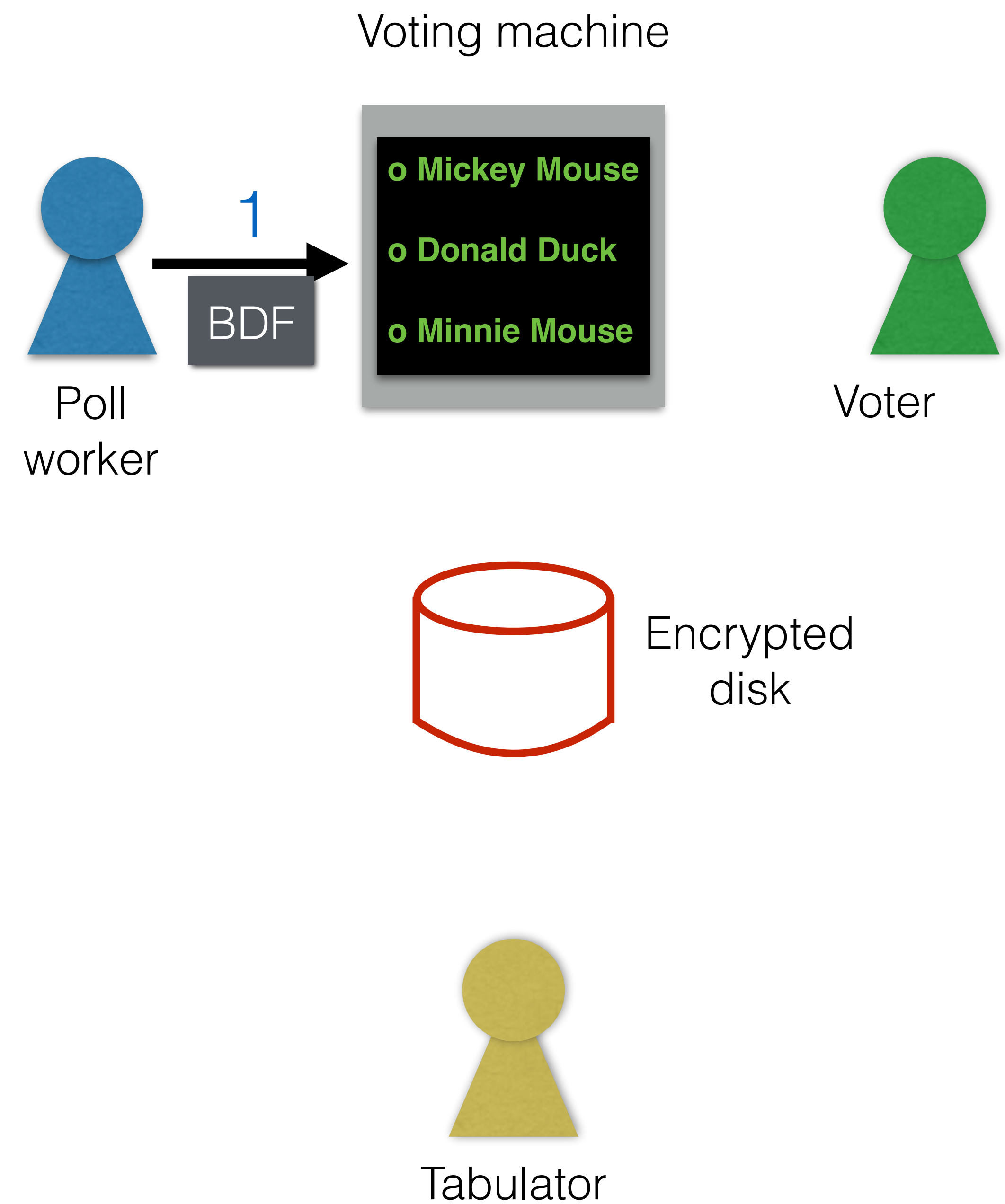
1. Pre-election phase
   - Poll worker loads a “ballot definition file” (defines who’s running, colors on the screen, and many more things) on the voting machines with, e.g., USB



E-voting analysis

1. Summarize the system as clearly and concisely as possible

1. Pre-election phase
   - Poll worker loads a "ballot definition file" (defines who's running, colors on the screen, and many more things) on the voting machines with, e.g., USB

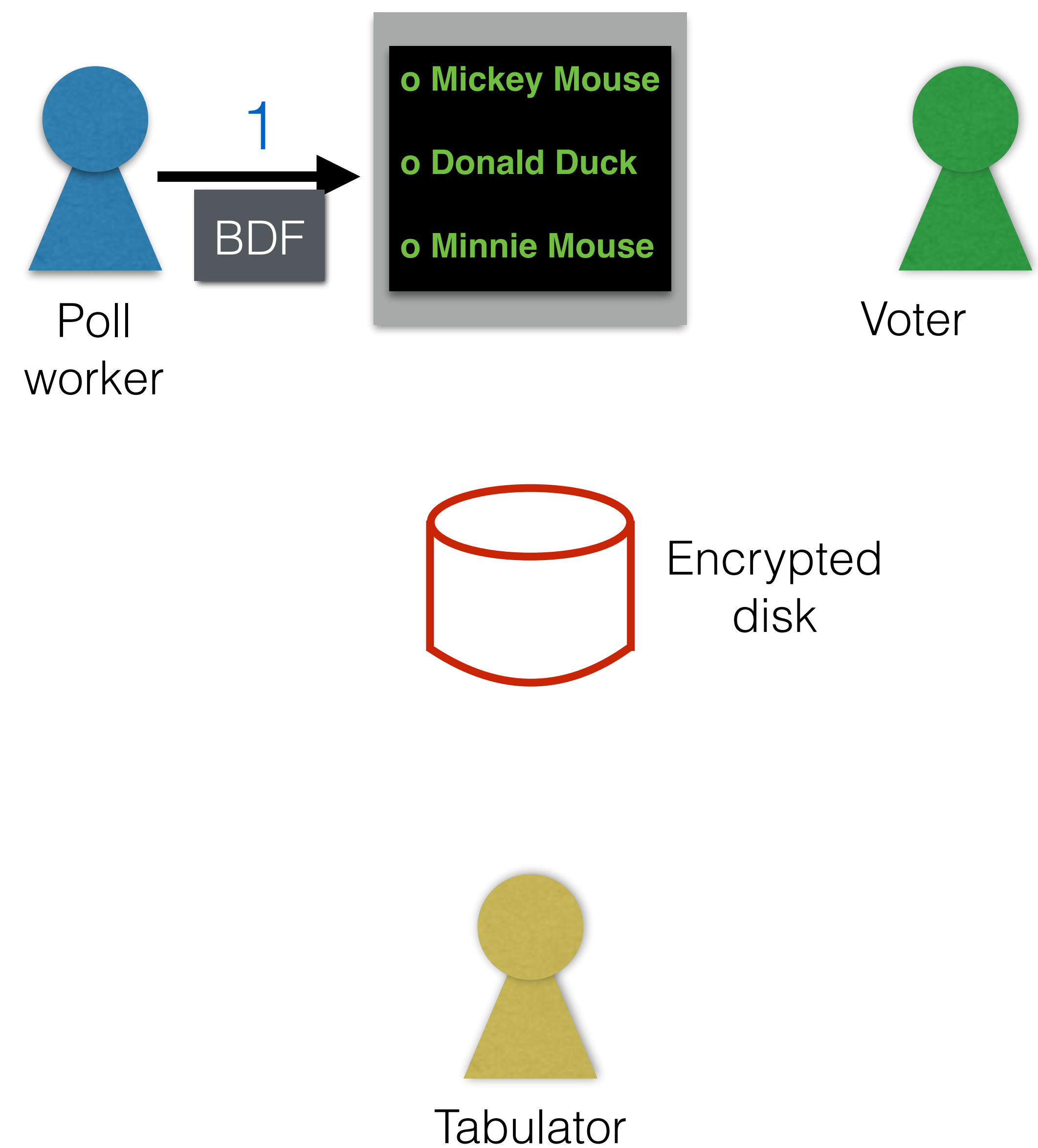


E-voting analysis

1. Summarize the system as clearly and concisely as possible

1. Pre-election phase
   - Poll worker loads a “ballot definition file” (defines who’s running, colors on the screen, and many more things) on the voting machines with, e.g., USB

2. Voting phase
   (a) Voter obtains a single-use token from poll workers (on smartcard)
   (b) Voter uses the token to interactively vote
   (c) Vote stored encrypted on disk
   (d) Voter token canceled

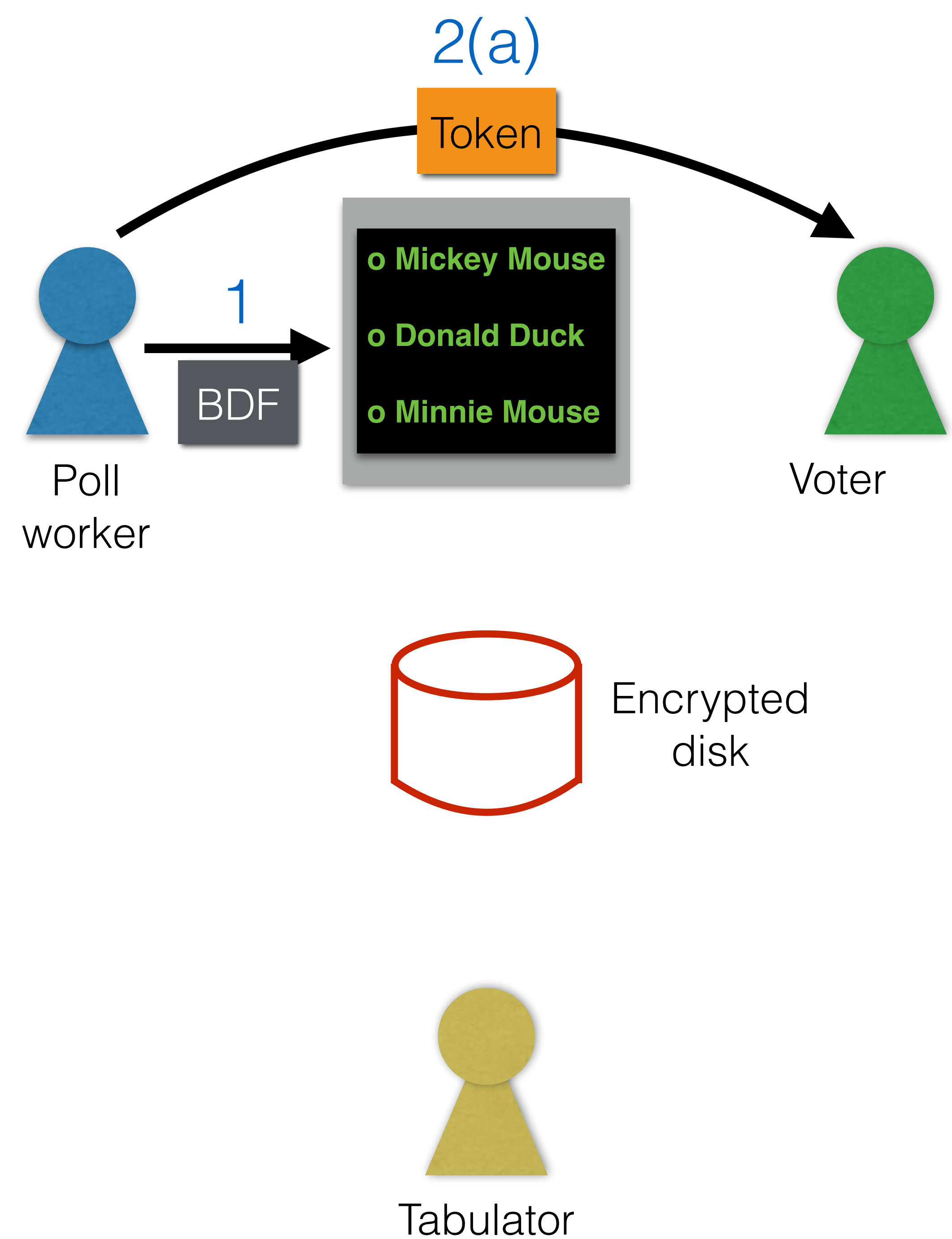


E-voting analysis

1. Summarize the system as clearly and concisely as possible

1. Pre-election phase
   - Poll worker loads a “ballot definition file” (defines who’s running, colors on the screen, and many more things) on the voting machines with, e.g., USB

2. Voting phase
   (a) Voter obtains a single-use token from poll workers (on smartcard)
   (b) Voter uses the token to interactively vote
   (c) Vote stored encrypted on disk
   (d) Voter token canceled

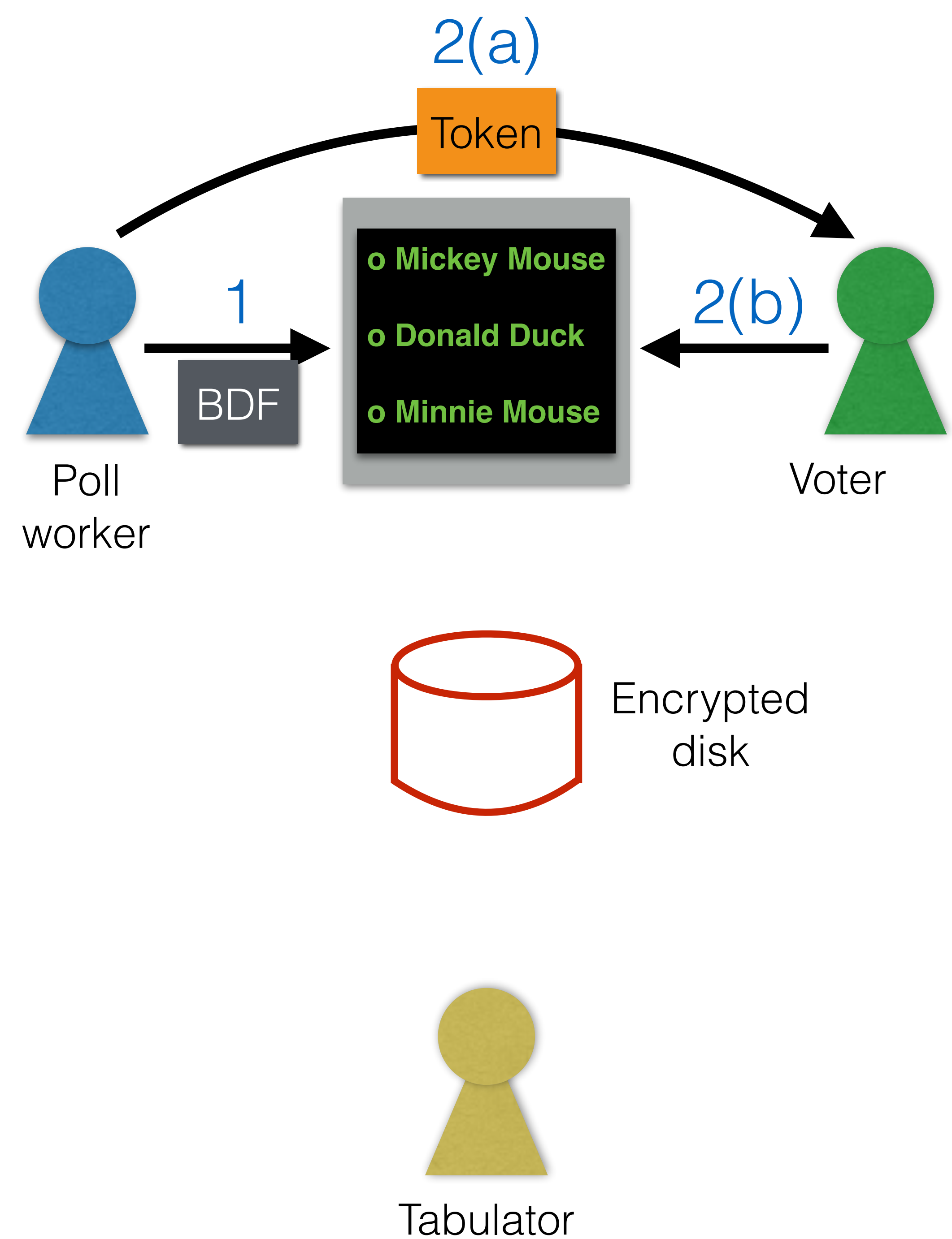


E-voting analysis

1. Summarize the system as clearly and concisely as possible

1. Pre-election phase
   - Poll worker loads a "ballot definition file" (defines who's running, colors on the screen, and many more things) on the voting machines with, e.g., USB

2. Voting phase
   (a) Voter obtains a single-use token from poll workers (on smartcard)
   (b) Voter uses the token to interactively vote
   (c) Vote stored encrypted on disk
   (d) Voter token canceled

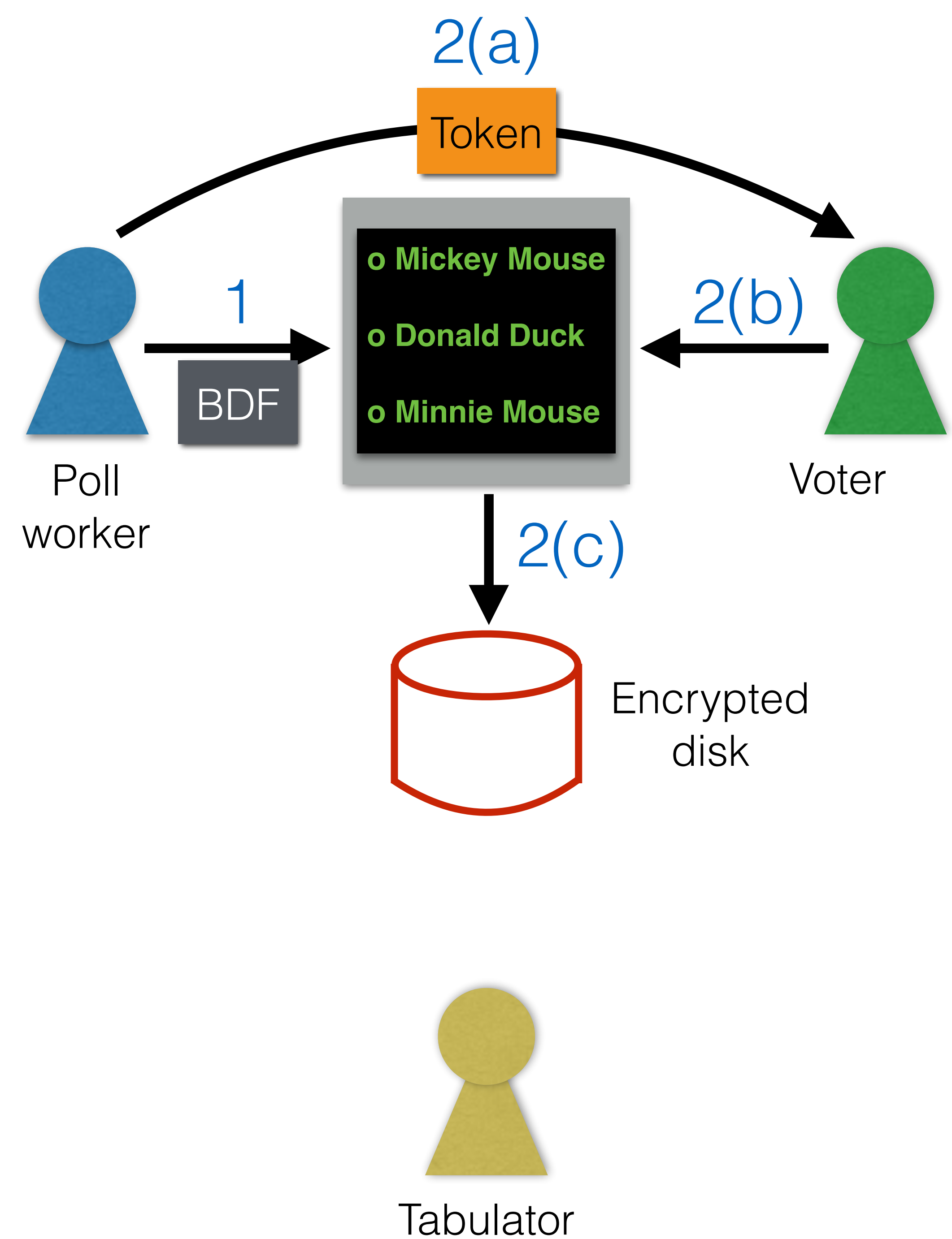


E-voting analysis

1. Summarize the system as clearly and concisely as possible

1. Pre-election phase
   - Poll worker loads a "ballot definition file" (defines who's running, colors on the screen, and many more things) on the voting machines with, e.g., USB

2. Voting phase
   (a) Voter obtains a single-use token from poll workers (on smartcard)
   (b) Voter uses the token to interactively vote
   (c) Vote stored encrypted on disk
   (d) Voter token canceled



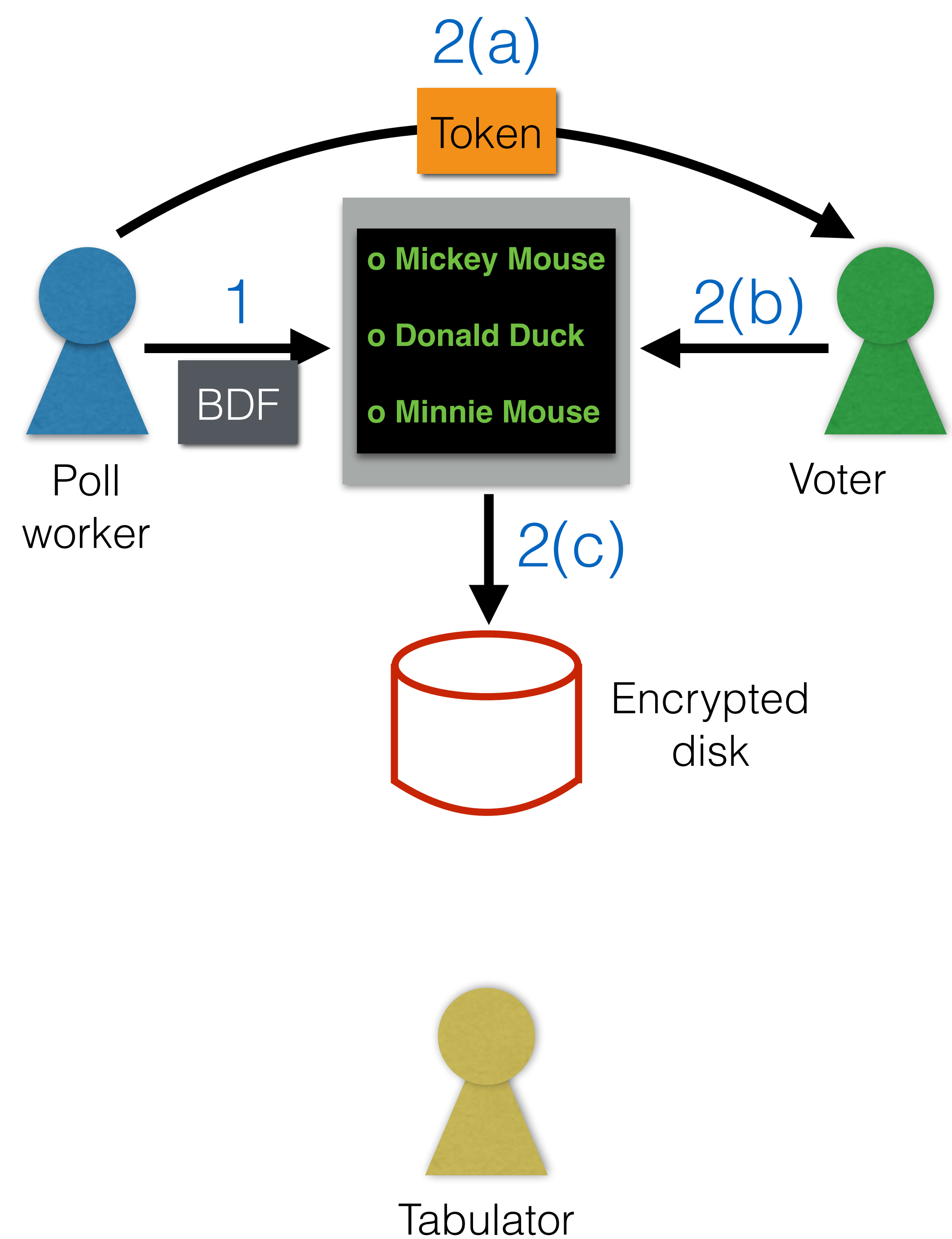
E-voting analysis

1. Summarize the system as clearly and concisely as possible

1. Pre-election phase
   - Poll worker loads a “ballot definition file” (defines who’s running, colors on the screen, and many more things) on the voting machines with, e.g., USB

2. Voting phase
   (a) Voter obtains a single-use token from poll workers (on smartcard)
   (b) Voter uses the token to interactively vote
   (c) Vote stored encrypted on disk
   (d) Voter token canceled

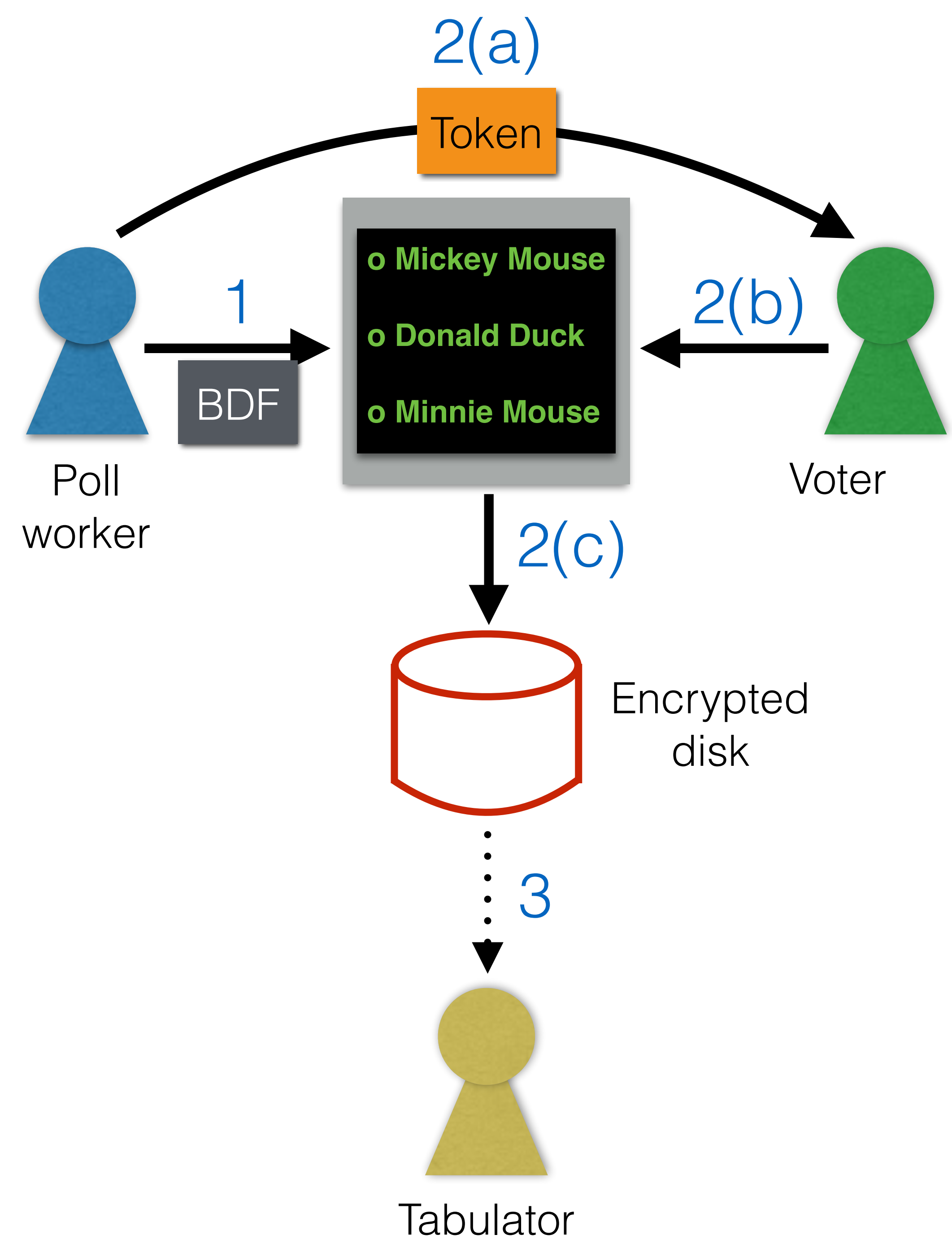
3. Post-election phase
   - Stored votes decrypted and transported to tabulator
   - Tabulator counts and announces vote



E-voting analysis

1. Summarize the system as clearly and concisely as possible

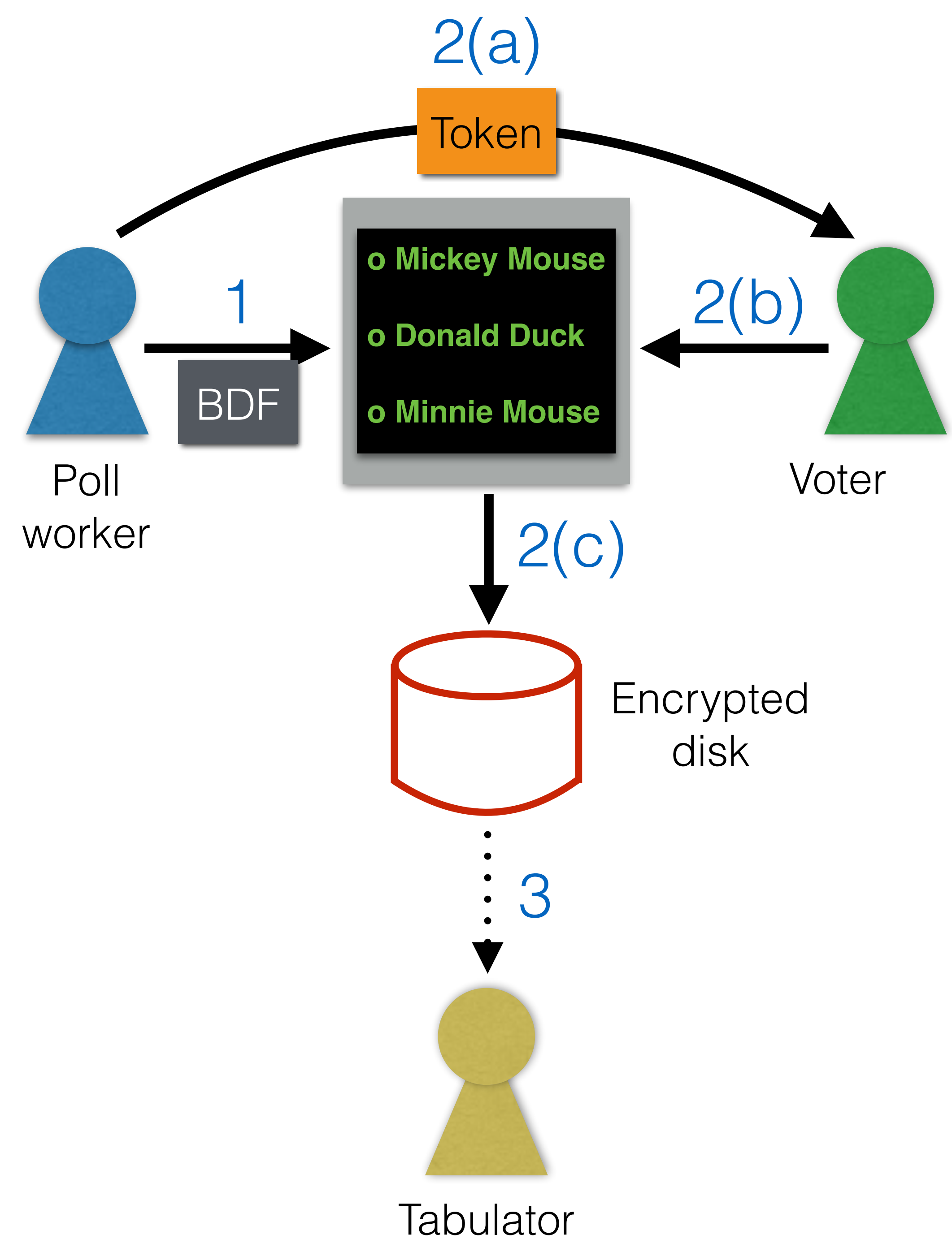
1. Pre-election phase
   - Poll worker loads a “ballot definition file” (defines who’s running, colors on the screen, and many more things) on the voting machines with, e.g., USB
2. Voting phase
   (a) Voter obtains a single-use token from poll workers (on smartcard)
   (b) Voter uses the token to interactively vote
   (c) Vote stored encrypted on disk
   (d) Voter token canceled
3. Post-election phase
   - Stored votes decrypted and transported to tabulator
   - Tabulator counts and announces vote



E-voting analysis

2. Identify the assets / goals of the system

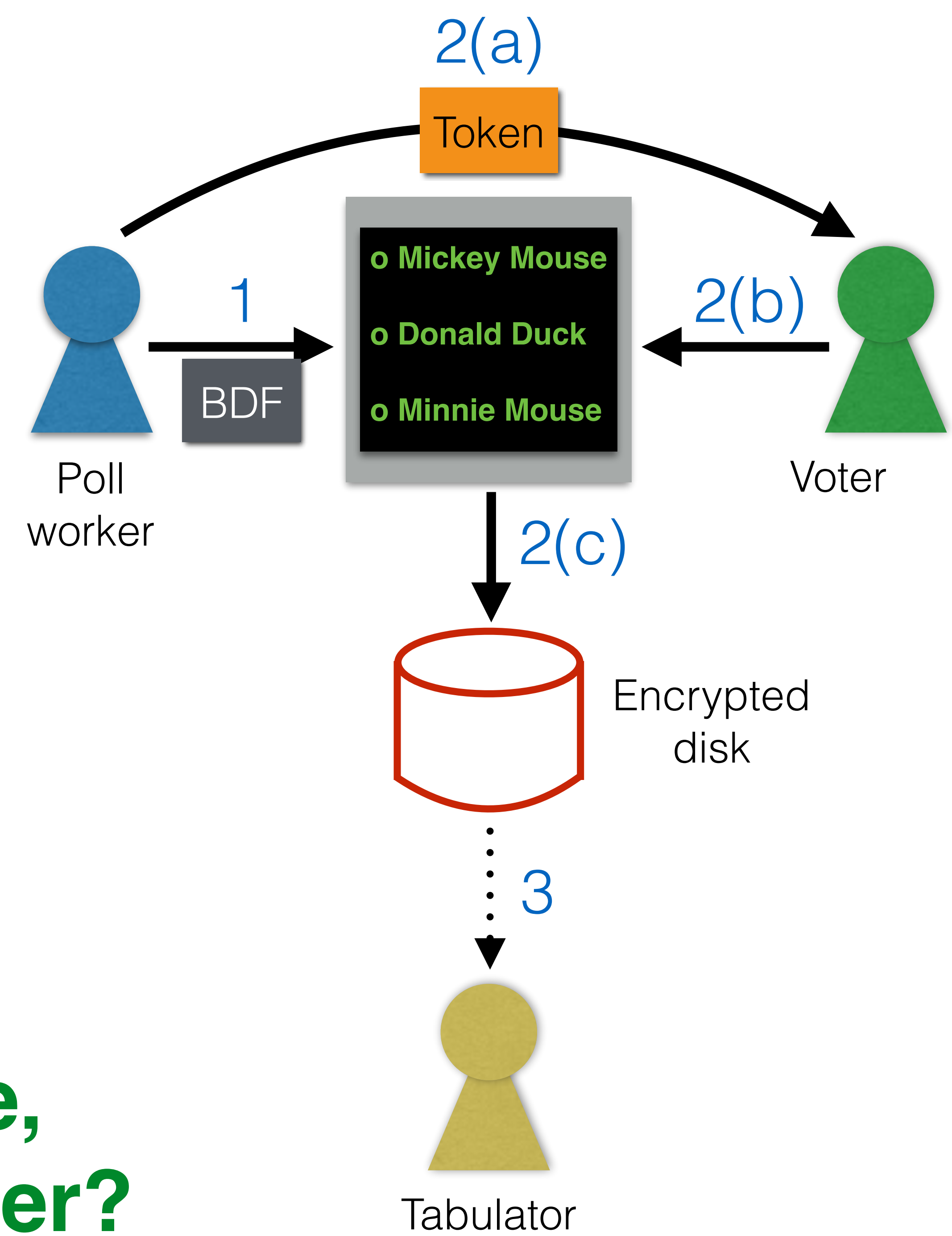
- Confidentiality (can't steal data)
- Integrity (can't modify data)
- Availability (system stays up)
- Usability (no undue burden on users)



E-voting analysis

2. Identify the assets / goals of the system

- Confidentiality (can't steal data)
- Integrity (can't modify data)
- Availability (system stays up)
- Usability (no undue burden on users)

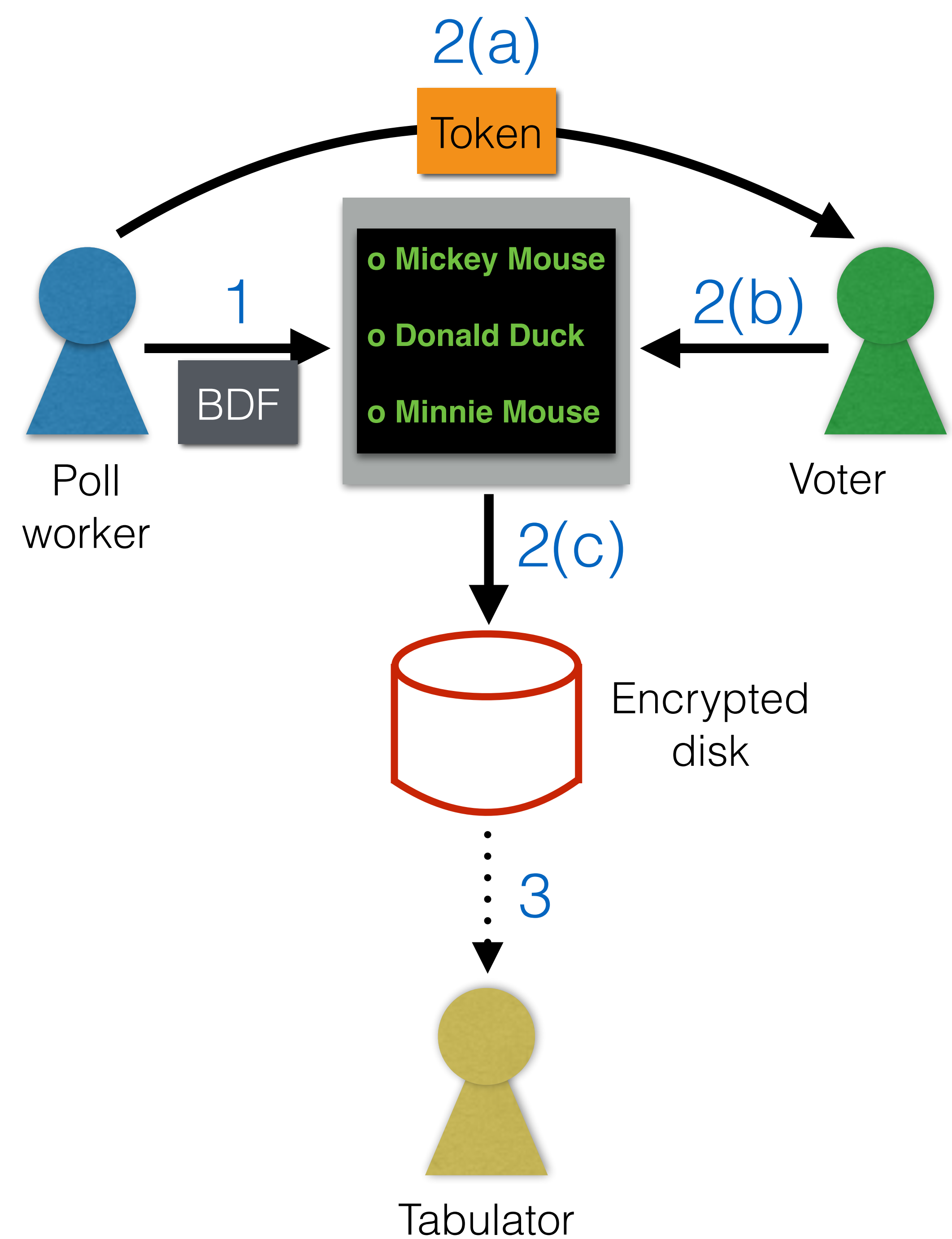


Is it ok if the attacker *can* do these, so long as you can *catch* him or her?

E-voting analysis

2. Identify the assets / goals of the system

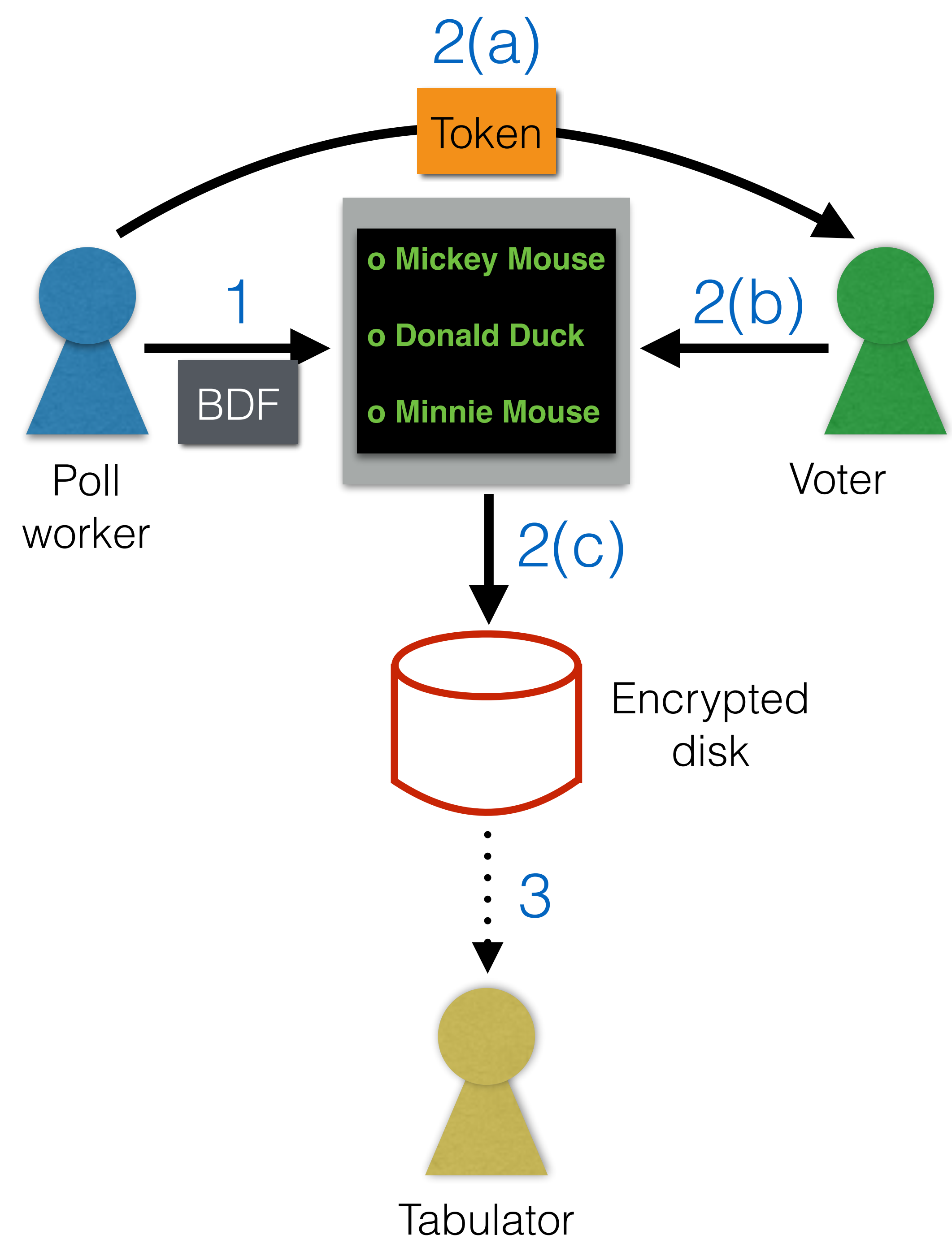
- Confidentiality (can't steal data)
- Integrity (can't modify data)
- Availability (system stays up)
- Usability (no undue burden on users)



E-voting analysis

2. Identify the assets / goals of the system

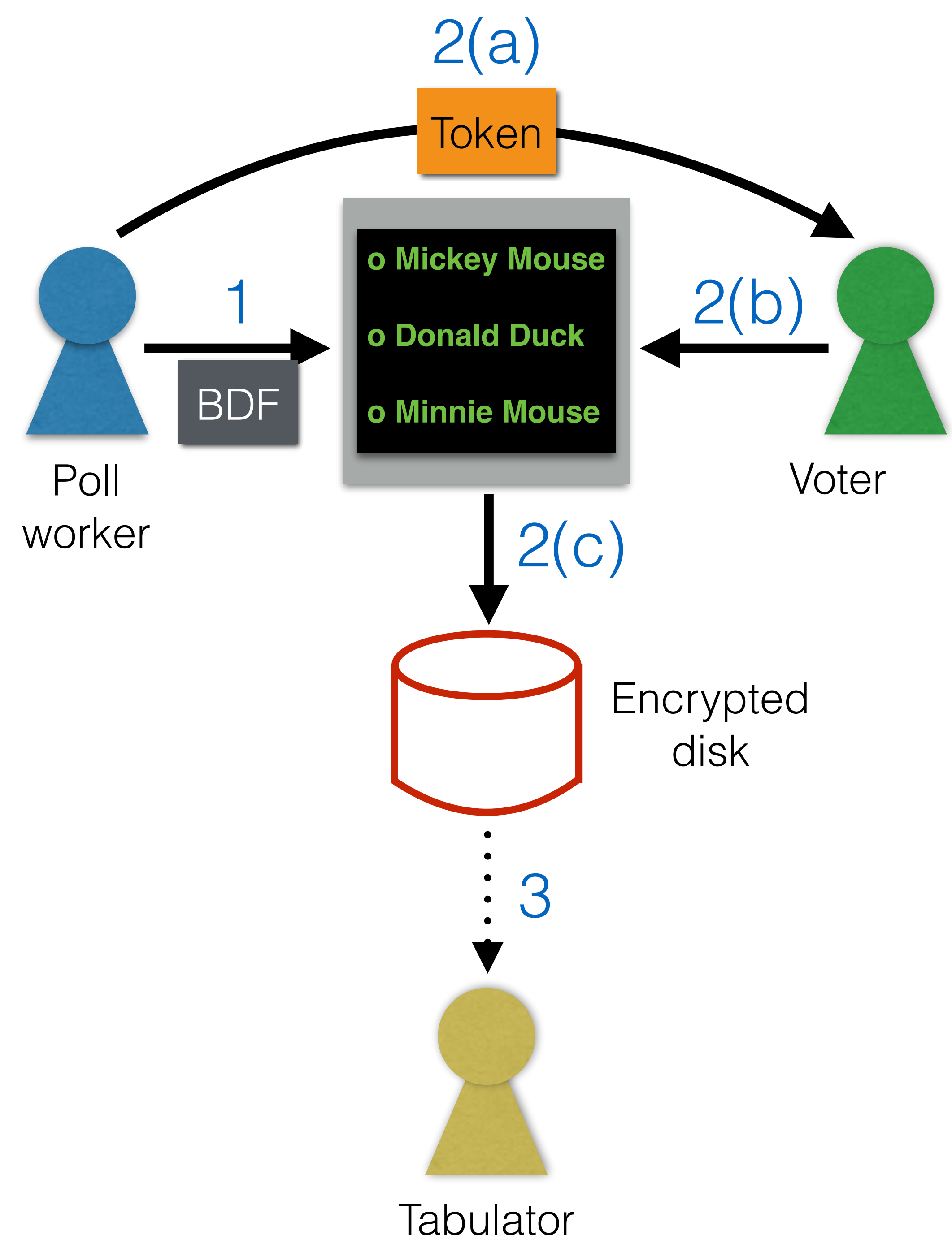
- Confidentiality (can't steal data)
  - No one knows for whom any given voter voted (except for the voter)
- Integrity (can't modify data)
- Availability (system stays up)
- Usability (no undue burden on users)



E-voting analysis

2. Identify the assets / goals of the system

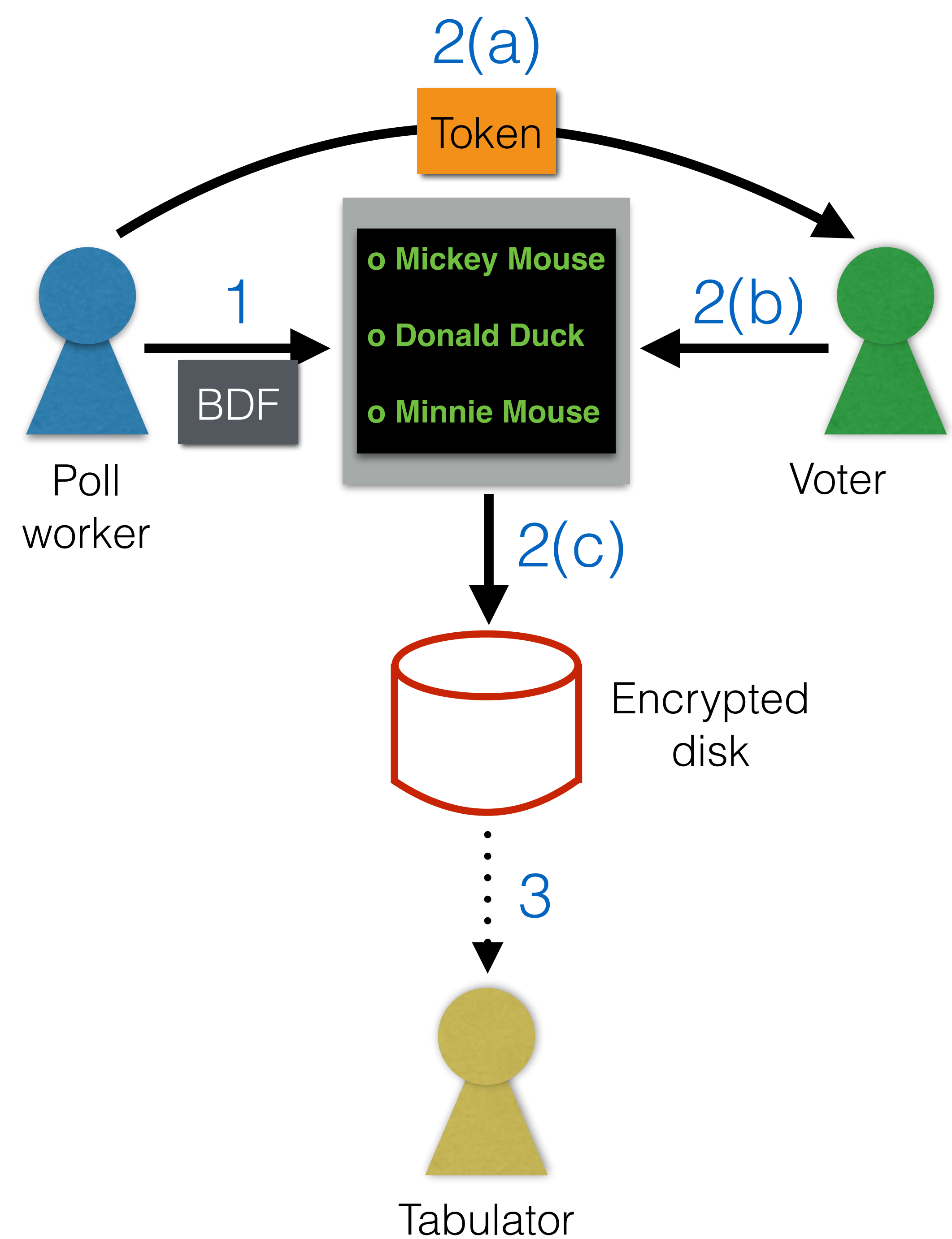
- Confidentiality (can't steal data)
  - No one knows for whom any given voter voted (except for the voter)
- Integrity (can't modify data)
  - Every voter's vote counted *once*
  - No voter's vote changed
- Availability (system stays up)
- Usability (no undue burden on users)



E-voting analysis

2. Identify the assets / goals of the system

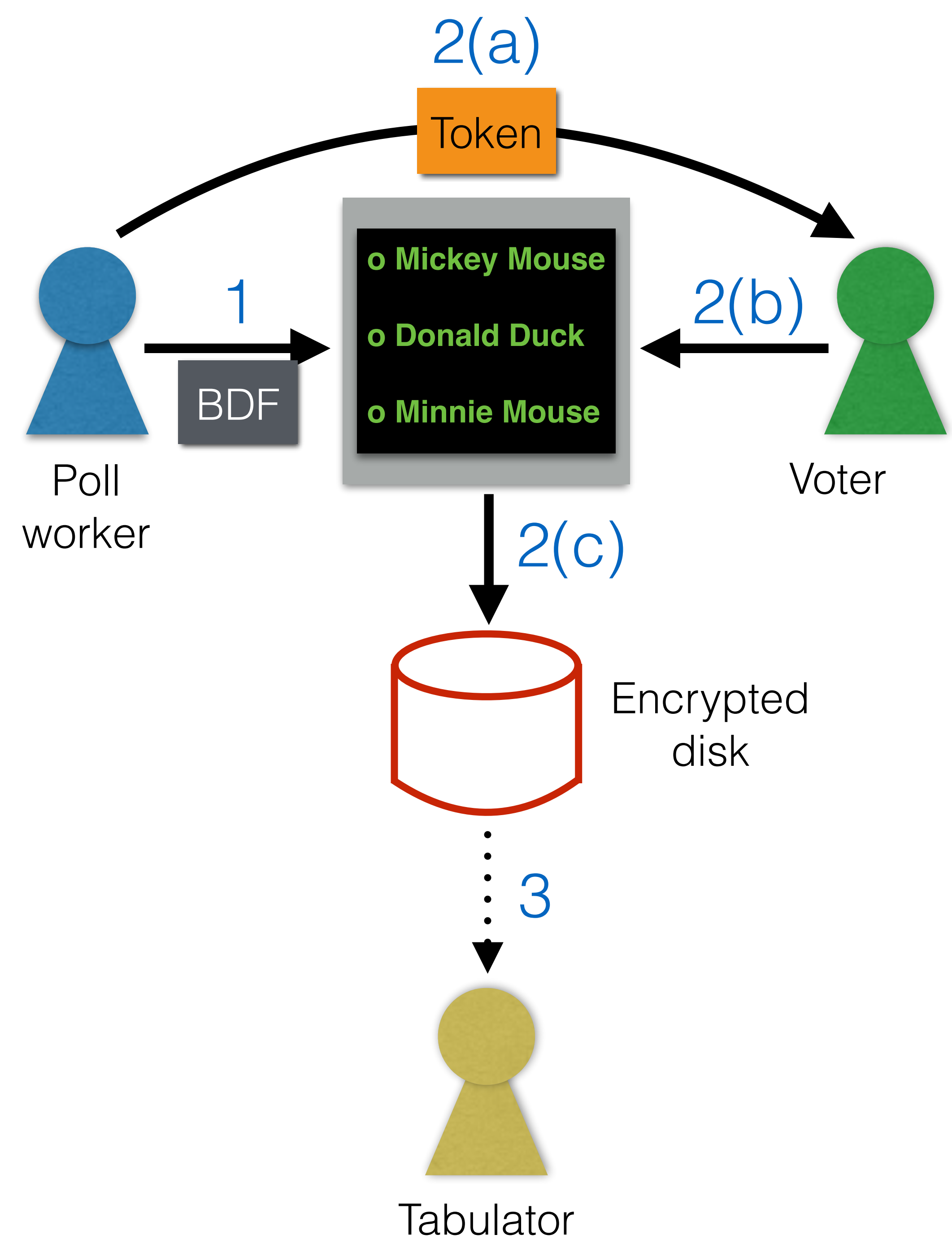
- Confidentiality (can't steal data)
  - No one knows for whom any given voter voted (except for the voter)
- Integrity (can't modify data)
  - Every voter's vote counted *once*
  - No voter's vote changed
- Availability (system stays up)
  - Everyone has the ability to cast their vote
- Usability (no undue burden on users)



E-voting analysis

2. Identify the assets / goals of the system

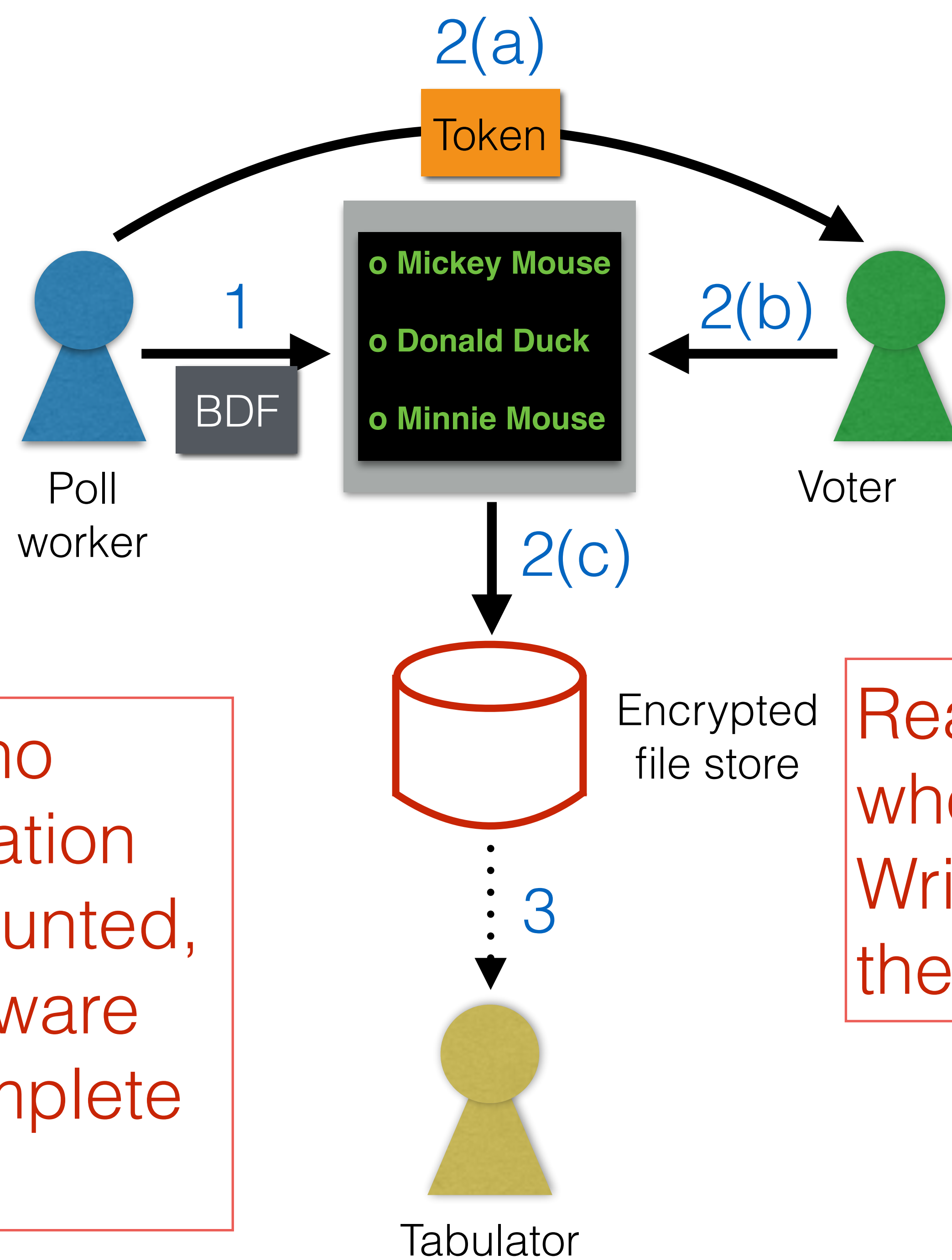
- Confidentiality (can't steal data)
  - No one knows for whom any given voter voted (except for the voter)
- Integrity (can't modify data)
  - Every voter's vote counted *once*
  - No voter's vote changed
- Availability (system stays up)
  - Everyone has the ability to cast their vote
- Usability (no undue burden on users)
  - Easy for the voter to vote (correct language, good UI)
  - Easy for the tabulator to count votes



E-voting analysis

3. Identify the adversaries and threats

Poll worker could set BDF to print “Mickey Mouse” but record as “Minnie Mouse”



Voter could attempt to generate their own tokens & get ≥2 votes

Because there is no end-to-end verification that a vote was counted, modifying the software could result in complete control

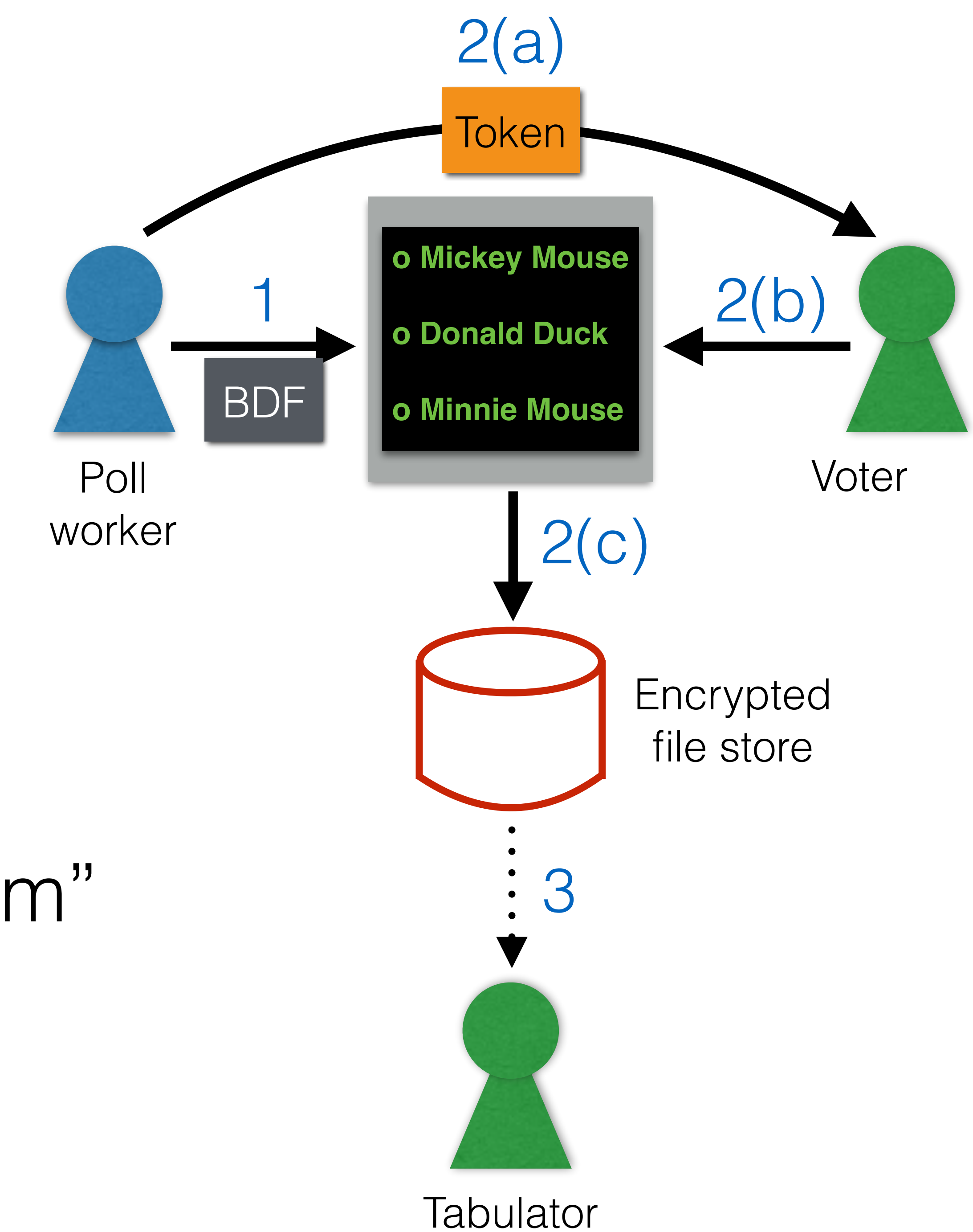
Reading this could reveal who voted for whom. Writing it could change the outcome altogether

E-voting analysis

4. Identify the vulnerabilities

Not a theoretical system:

Diebold AccuVote-TS
Used in 37 states at time of study

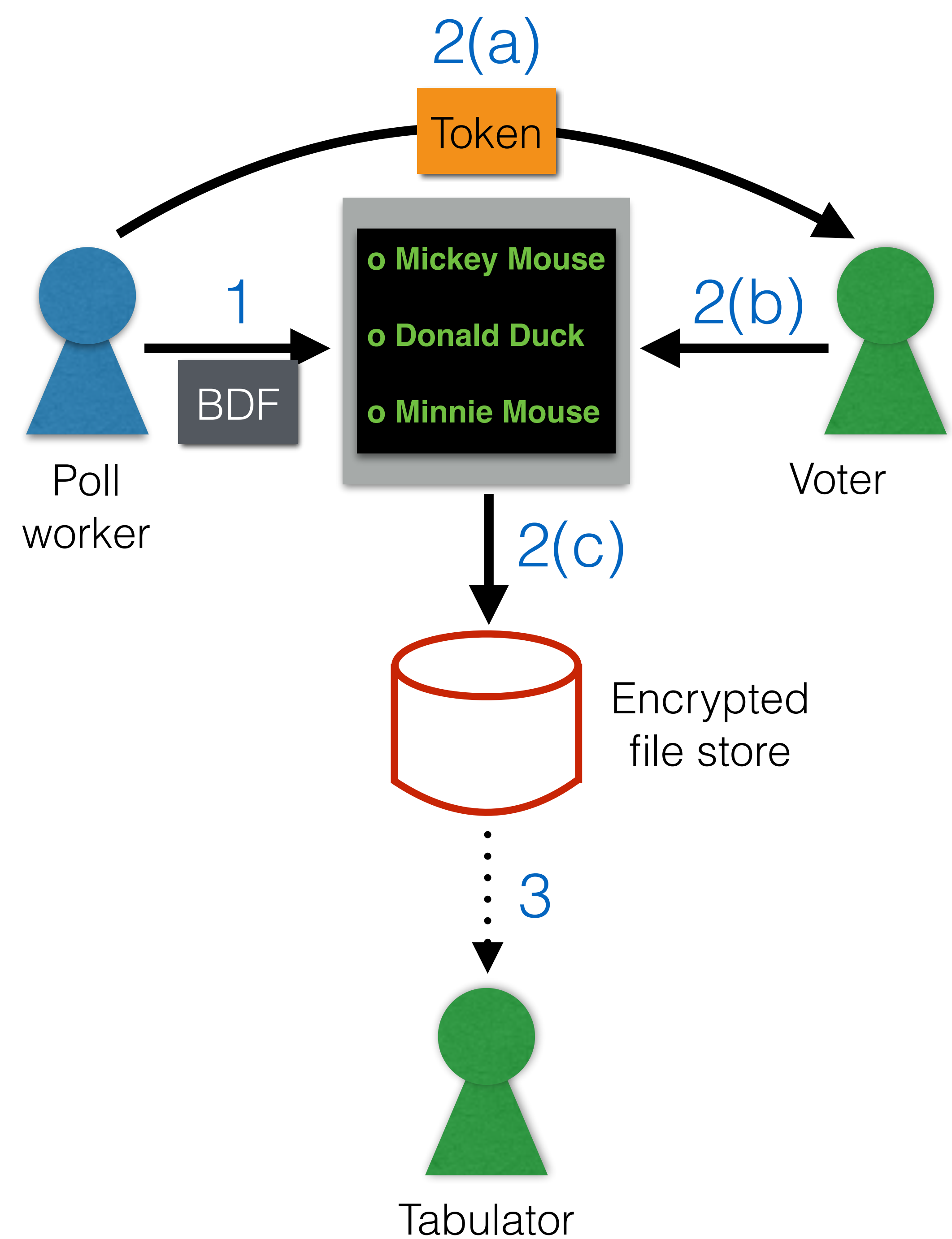


Optional reading
"Analysis of an Electronic Voting System"
Kohno et al.

E-voting analysis

4. Identify the vulnerabilities

- Ballot definition files are not authenticated
  - How do we know they're from the election board?
  - Can redefine "Candidate A" as "Candidate B"
  - Viruses
- Smartcards are not authenticated
  - How do we know they're not user-generated?
  - Possible to make your own and vote multiple times.
- Specific software vulnerabilities
  - Every machine has the same encryption key!
  - Break one, and they all fall
- Votes are shipped unencrypted!
- Votes are stored in the order cast
  - If one can view the data unencrypted, this violates our confidentiality goal



Analyzing security

Takeaway points

- Analyzing security requires a whole-systems view
  - Hardware, software, users, economics, ….
- Security is only as strong as the weakest link
  - May have been difficult to break into the building
  - But if the data is sent unencrypted…
- Securing a system can be difficult
  - Interdisciplinary (software, hardware, UI design)
  - Humans are in the loop
- Security through obscurity does not work
  - Especially for high-value assets
  - It's only a matter of time until someone finds out

The Goals of CMSC 414

Be able to eliminate bugs and design flaws
and/or make them harder to exploit.

Be able to think like attackers.

Develop a foundation for deeply understanding
the systems we use and build.

Software	**Hardware**	**Protocols**
Users	**Law**	**Economics**

Next time

We will begin our 1st section:
Software Security

By investigating
Buffer overflows
and other memory safety vulnerabilities

To prepare: you may want to brush up on your C

Particularly if this seems foreign to you:

```
char buf[32];
unsigned *ptr = (unsigned*) (buf + 12);
*ptr += 0x1a;
```