CSE 484 / CSE M 584: Computer Security and Privacy

The End of Software Security

(and some Cryptography)

Spring 2016

Ada (Adam) Lerner
lerner@cs.washington.edu

Thanks to Franzi Roesner, Dan Boneh, Dieter Gollmann, Dan Halperin, Yoshi Kohno, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...

Buy ALL the Lottery Tickets

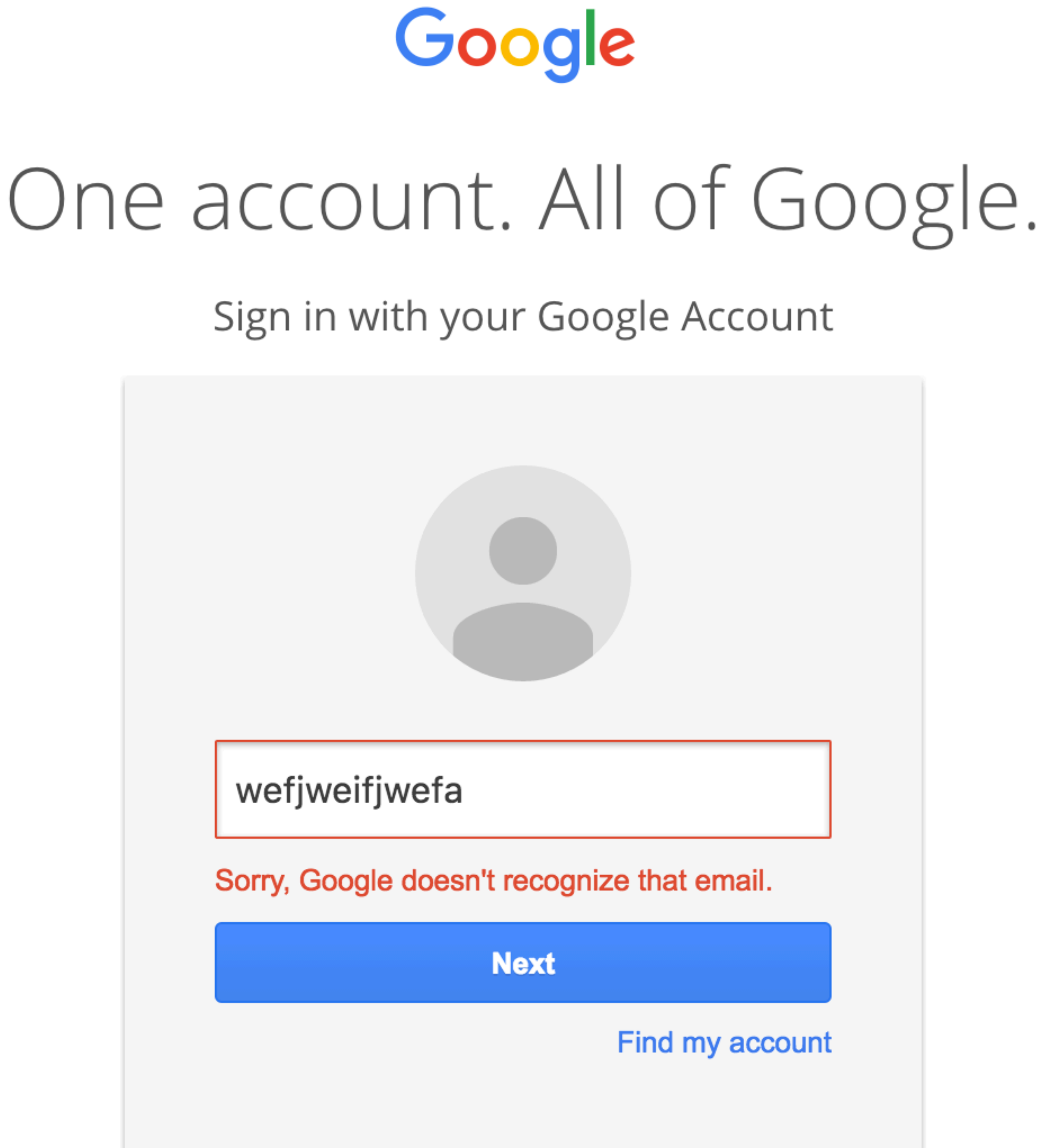
- Some MIT students won \$3.5M over seven years in the Mass. State lottery
- In 1992, a group bought 5M out of 7M possible lottery tickets in Virginia

Side Channel Attacks

```
PwdCheck(RealPwd, CandidatePwd)  // both 8 chars
   for i = 1 to 8 do
      if (RealPwd[i] != CandidatePwd[i]) then
         return FALSE
   return TRUE
```

Side Channel Attacks

- Timing
- David mentioned telescope + camera to read bits off **modem lights**
- Power usage
- Sound
- Error messages
- Facial expressions, tone of voice


Google
One account. All of Google.
Sign in with your Google Account
wefjweifjwefa
Sorry, Google doesn't recognize that email.
Next
Find my account

Side Channel Attacks

Randomness Issues

- Many applications (especially security ones) require randomness

- If you use predictable randomness, bad things can happen

Randomness Issues

- Ma … y sec … mness

- If y … mness, bac

Randomness Issues

– Generate cryptographic keys
– Generate passwords for new users
– Shuffle the order of votes (in an electronic voting machine)
– Shuffle cards (for an online gambling site)

C's rand() Function

- C has a built-in random function: rand()

```
unsigned long int next = 1;
/* rand:  return pseudo-random integer on 0..32767 */
int rand(void) {
    next = next * 1103515245 + 12345;
    return (unsigned int)(next/65536) % 32768;
}
/* srand:  set seed for rand() */
void srand(unsigned int seed) {
    next = seed;
}
```

- Problem: don't use rand() for security-critical applications!
  - Given a few sample outputs, you can predict subsequent ones

Problems in Practice

- One institution used (something like) rand() to generate passwords for new users
  - Given your password, you could predict the passwords of other users

Problems in Practice

- Kerberos (1988 - 1996)
  - Random number generator improperly seeded
  - Possible to trivially break into machines that rely upon Kerberos for authentication

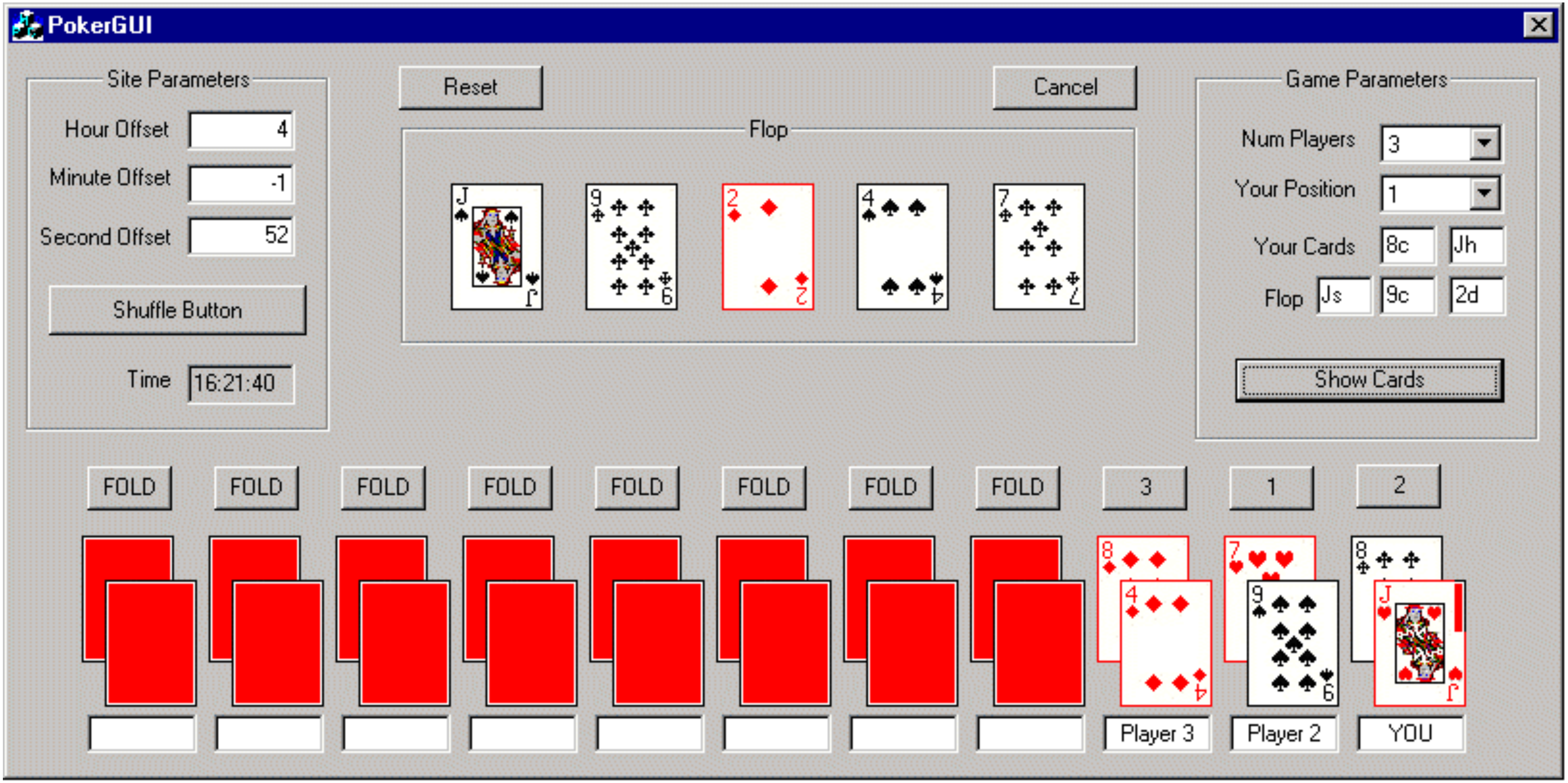
Problems in Practice

- Debian Linux (2006-2008)
  - OpenSSL key generator seeded using only process ID.
  - Only ~32,000 choices for key…

Problems in Practice

- Online gambling websites
  – Random numbers to shuffle cards
  – Real money at stake
  – But what if poor choice of random numbers?


A World of Action!
Help
Suggestions
Fold in Turn
$0
jonnyboy
brute
Fold
Check
Bet ($5)
mamajoe
RickG
Options
Sit Out
Leave
584330
mamajoe: Hey guys, Big B is in!
Dealer Text



More details: "How We Learned to Cheat at Online Poker: A Study in Software Security"
http://www.cigital.com/papers/download/developer_gambling.php


A World of Action!
Help
Suggestions
Fold in Turn
$2
jonnyboy
brute
Fold
Show
mamajoe
RickG
Options
Sit Out
Leave
584330
mamajoe: Hey guys, Big B is in!
Dealer Text

PS3 and Randomness

Hackers obtain PS3 private cryptography key due to epic programming fail? (update)

http://www.engadget.com/2010/12/29/hackers-obtain-ps3-private-cryptography-key-due-to-epic-programm/

- 2010/2011: Hackers found/released private root key for Sony's PS3
- Key used to sign software – now can load any software on PS3 and it will execute as "trusted"
- Due to bad random number: same "random" value used to sign all system updates

PS3 and Randomness

- Example Current Event report from a past iteration of 484
  - https://catalyst.uw.edu/gopost/conversation/kohno/452868

Posted Jan 3, 2011 5:01 PM Quote

PS3 Exploit
Today, January 3rd, George “Geohot” Hotz found and released the private root key for Sony's Playstation 3 (PS3) video game console (http://www.geohot.com/). What this means is that homebrew software enthusiasts, scientists, and software pirates can now load arbitrary software on the PS3 and sign it using this key, and the system will execute it as trusted code. Legitimately, this allows Linux and other operating systems to take advantage of the PS3's cell processor architecture; however, it also opens up avenues of software piracy previously impossible on Sony's system without requiring any hardware modifications to the system (previous access of this kind required a USB hardware dongle)

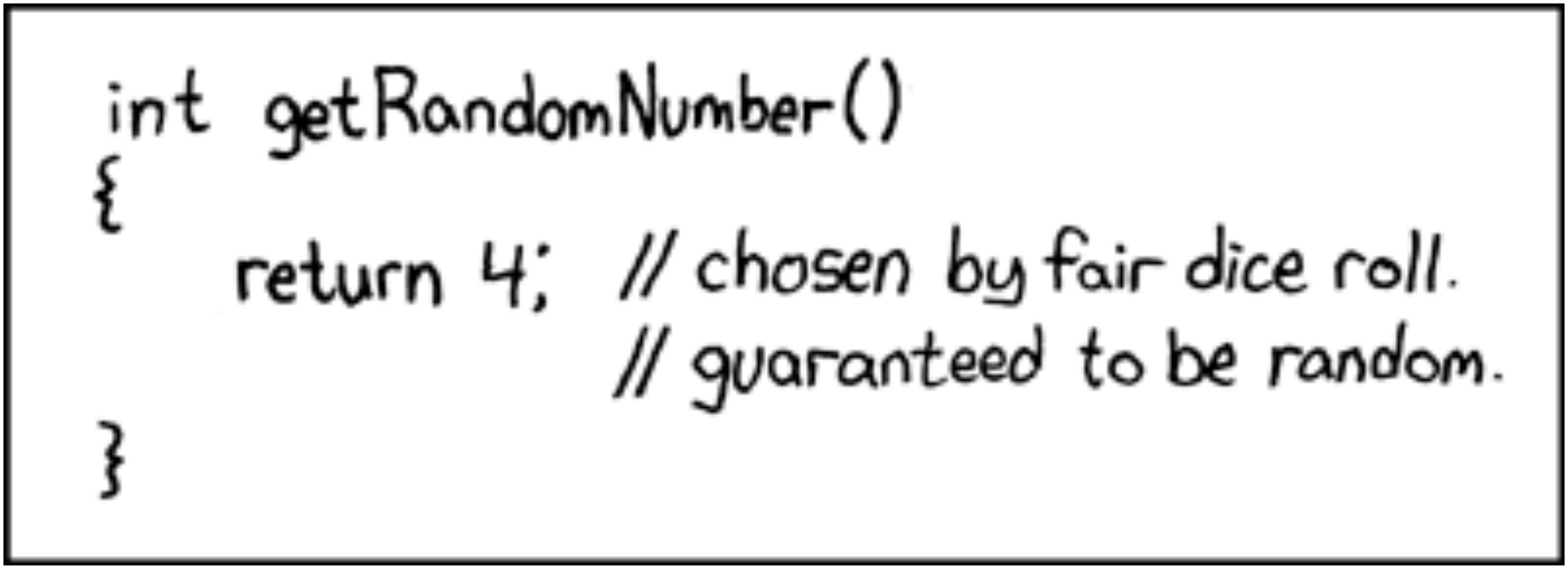
How it Was Done
This was enabled by a cryptographic error by Sony developers in their update process. In the DSA signature algorithm, a number k is chosen from a supposedly random source for each signed message. So long as the numbers are unique, the system is secure, but duplicating a random number between messages can expose the private key to an untrusted party using simple mathematics (http://rdist.root.org/2010/11/19/dsa-requirements-for-random-k-value/). Sony used the exact same “random value” k for all updates pushed to the system, making the signature scheme worthless.

The Most Secure
After Sony removed the “other OS” functionality of the PS3, greater scrutiny was placed on the PS3. Since it's release in 2006, the Playstation 3 was considered the most secure of the three major video game consoles, as it was the only console without a “root” compromise in the four years since release (there were vulnerabilities limited to specific firmware or that required specialized hardware, but nothing that provided unfettered access). By comparison, Microsoft's Xbox 360 was cracked over 4 years ago (http://www.theregister.co.uk/2007/03/01/xbox_hack), and the Wii was cracked over 2 years ago (http://wiibrew.org/wiki/Index.php).

Cullen Walsh
Mark Jordan
Peter Lipay

Other Problems

- Key generation
  - Ubuntu removed the randomness from SSL, creating vulnerable keys for thousands of users/servers
  - Undetected for 2 years (2006-2008)
- Live CDs, diskless clients
  - May boot up in same state every time
- Virtual Machines
  - Save state: Opportunity for attacker to inspect the pseudorandom number generator's state
  - Restart: May use same "psuedorandom" value more than once



https://xkcd.com/221/

Obtaining Pseudorandom Numbers

- For security applications, want "cryptographically secure pseudorandom numbers"
- Libraries include cryptographically secure pseudorandom number generators
- Linux:
  - /dev/random
  - /dev/urandom - nonblocking, possibly less entropy
- Internally:
  - Entropy pool gathered from multiple sources

Where do (good) random numbers come from?

- Humans: keyboard, mouse input
- Timing: interrupt firing, arrival of packets on the network interface
- Physical processes: unpredictable physical phenomena

Software Security: Defenses in Summary

Buffer Overflow Defense Catalog

- Execute bit off on heap/stack
- StackGuard (canaries)
- PointGuard (encrypted pointers)
- ASLR
- str**n**cpy vs strcpy
- Static analysis, dynamic analysis
- Type safe languages (e.g., Java)

Fuzz Testing

- Generate "random" inputs to program
  - Sometimes conforming to input structures (file formats, etc.)
- See if program crashes
  - If crashes, found a bug
  - Bug may be exploitable
- Surprisingly effective

- Now standard part of development lifecycle

General Principles

- Check inputs

Shellshock

- Example: Shellshock (September 2014)
  - Vulnerable servers processed input from web requests, passed (user-provided) environment variables (like user agent, cookies…) to CGI scripts
  - Maliciously crafted environment variables exploited a bug in bash to execute arbitrary code

```
env x='() { :;}; echo OOPS' bash -c :
```

Software Security Principles

- Check/sanitize inputs
- Check all return values
- Least privilege
- Securely clear memory (passwords, keys, etc.)
- Failsafe defaults
- Defense in depth
  - Also: prevent, detect, respond

- NOT: security through obscurity

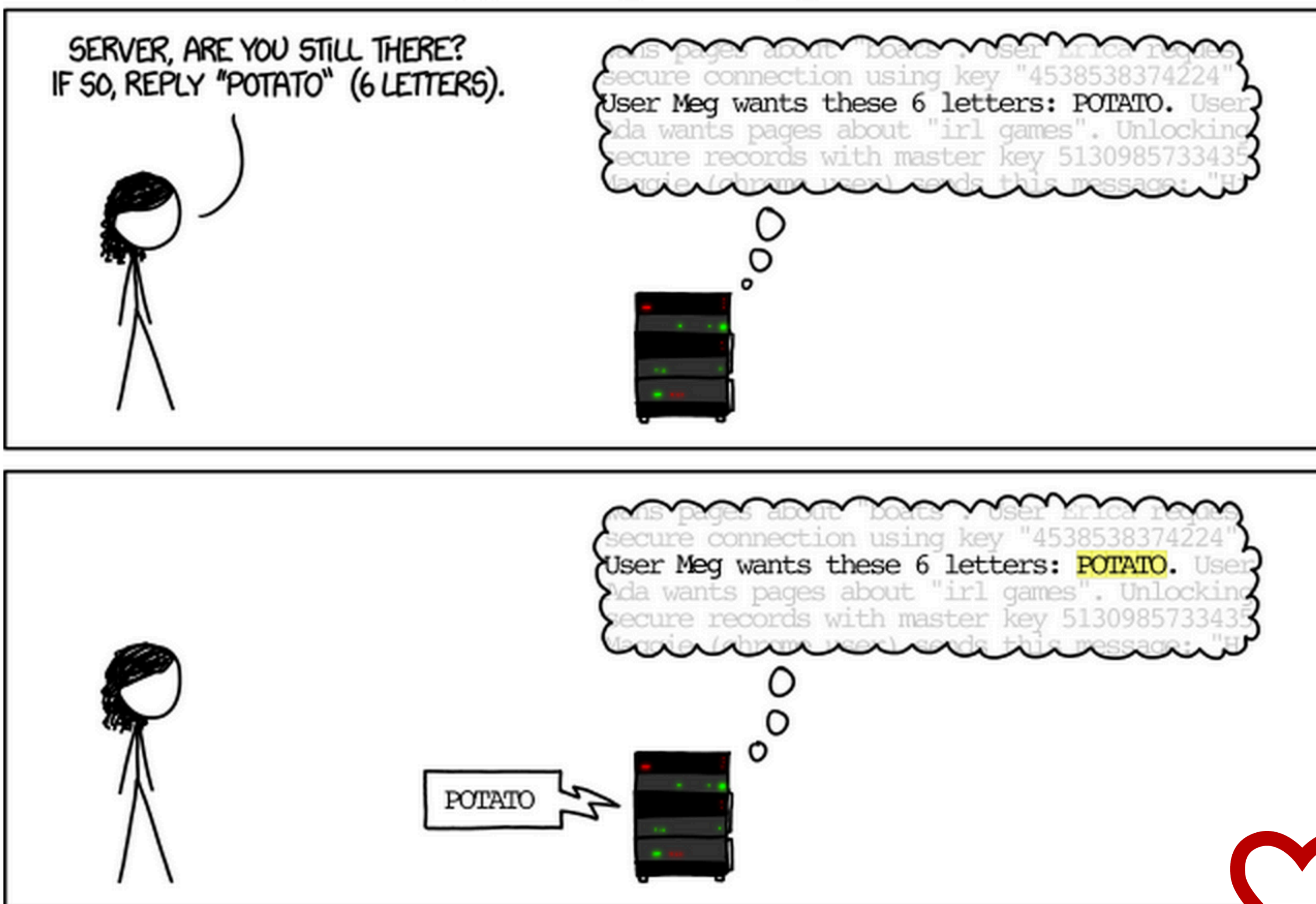
General Principles

- Reduce size of trusted computing base (TCB)
- Simplicity, modularity
  - But: Be careful at interface boundaries!
- Minimize attack surface
- Use vetted component
- Security by design
  - But: tension between security and other goals
- Open design? Open source? Closed source?
  - Different perspectives

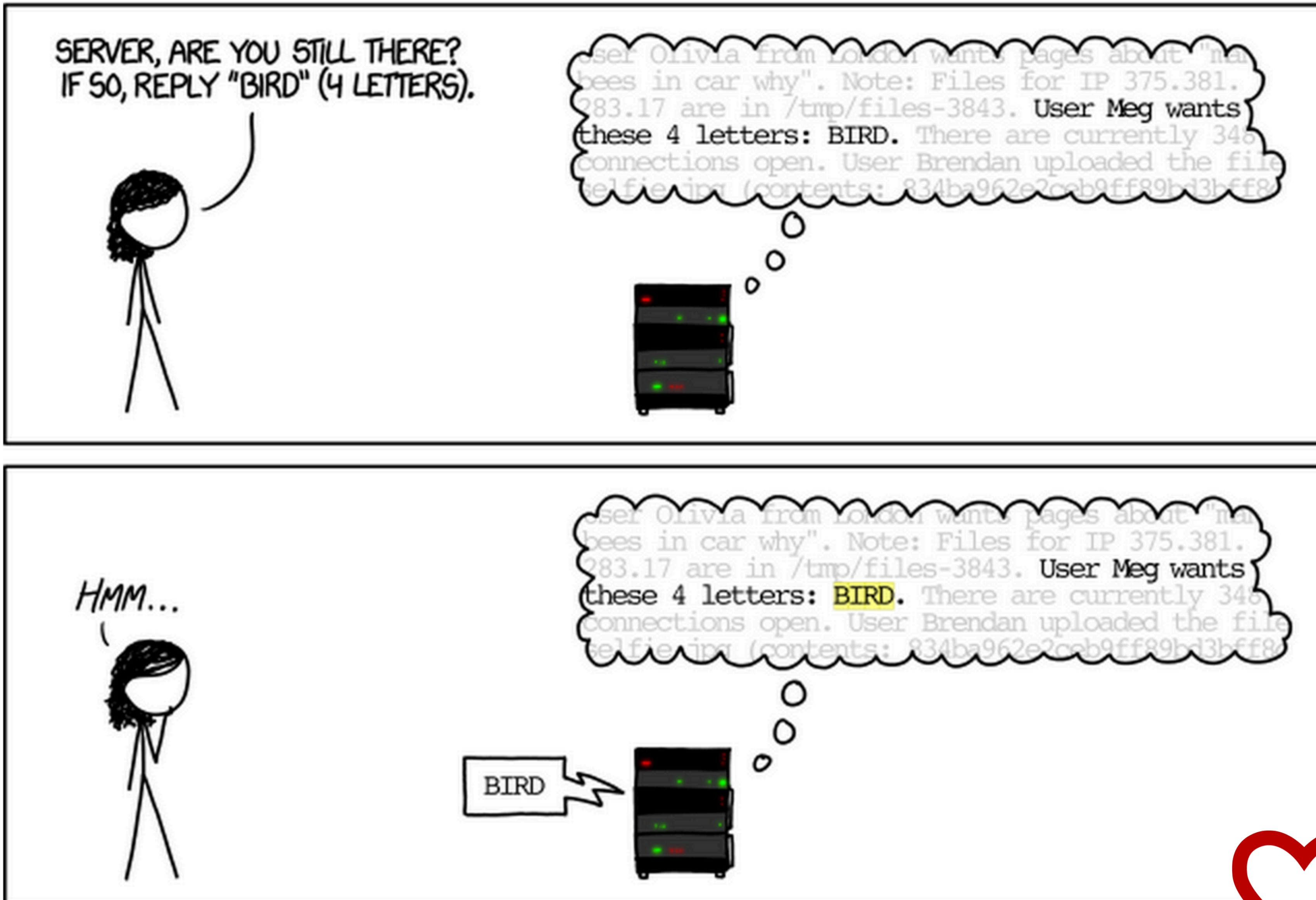
Does Open Source Help?

- Different perspectives…
- Happy example:
  - Linux kernel backdoor attempt thwarted (2003) (http://www.freedom-to-tinker.com/?p=472)
- Sad example:
  - Heartbleed (2014)
    - Vulnerability in OpenSSL that allowed attackers to read arbitrary memory from vulnerable servers (including private keys)

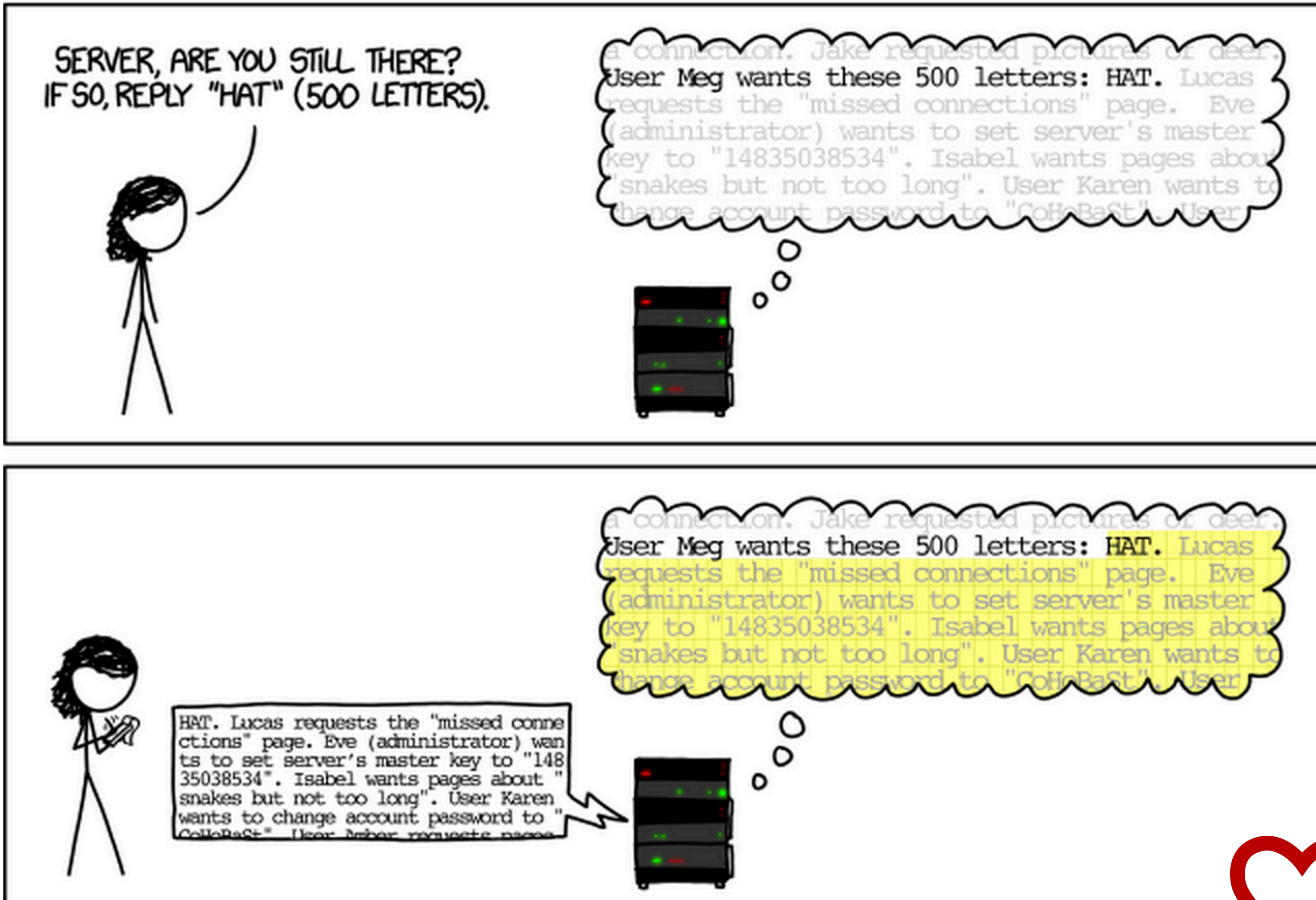
http://xkcd.com/1354/



http://xkcd.com/1354/



http://xkcd.com/1354/



Responsible Disclosure

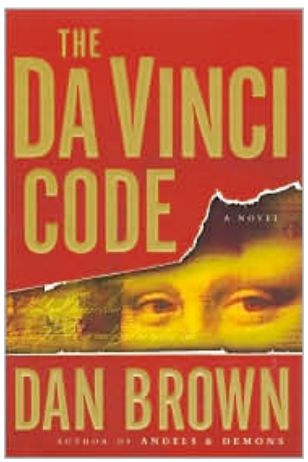
- What do you do if you've found a security problem in a real system?
- Say
  - A commercial website?
  - UW grade database?
  - Boeing 787?
  - TSA procedures?

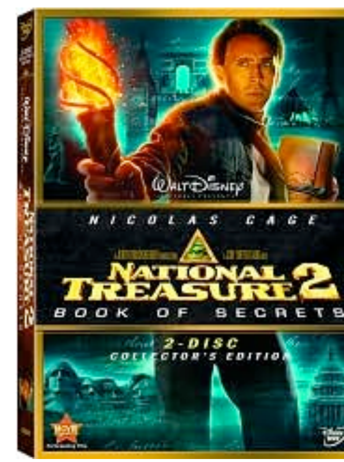
Abj sbe fbzr pelcgbtencul!

Now for some cryptography!

Cryptography and Security

- Art and science of *protecting* our *information*.
  - Keeping it **private,** if we want privacy.
  - Protecting its **integrity**, if we want to avoid forgeries.

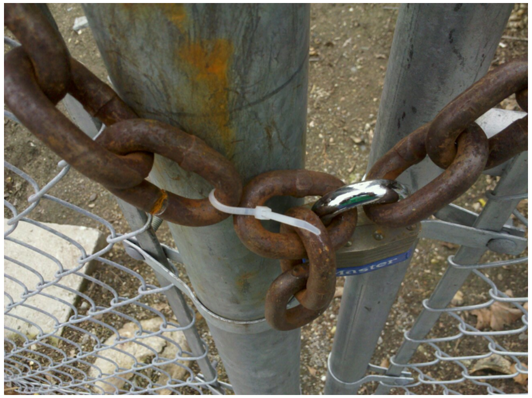




Images from Wikipedia and Barnes & Noble

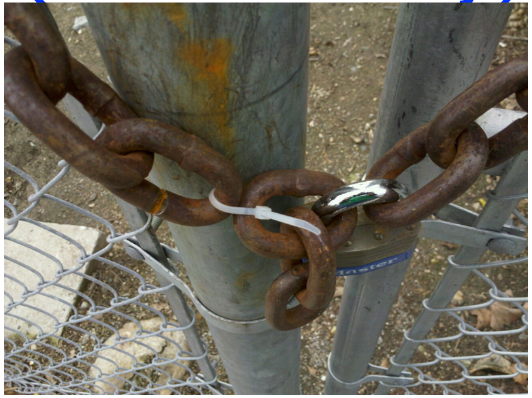
Some Thoughts About Cryptography

- Cryptography only one small piece of a larger system
- Must protect entire system
  - Physical security
  - Operating system security
  - Network security
  - Users
  - Cryptography



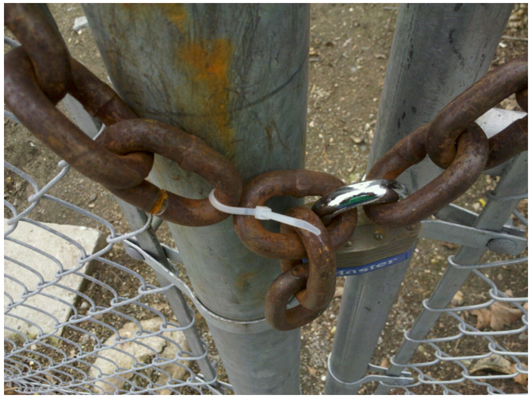
Some Thoughts About Cryptography

- "Security only as strong as the weakest link"
  - Need to secure weak links
  - But not always clear what the weakest link is (different adversaries and resources, different adversarial goals)
  - Crypto failures may not be (immediately) detected



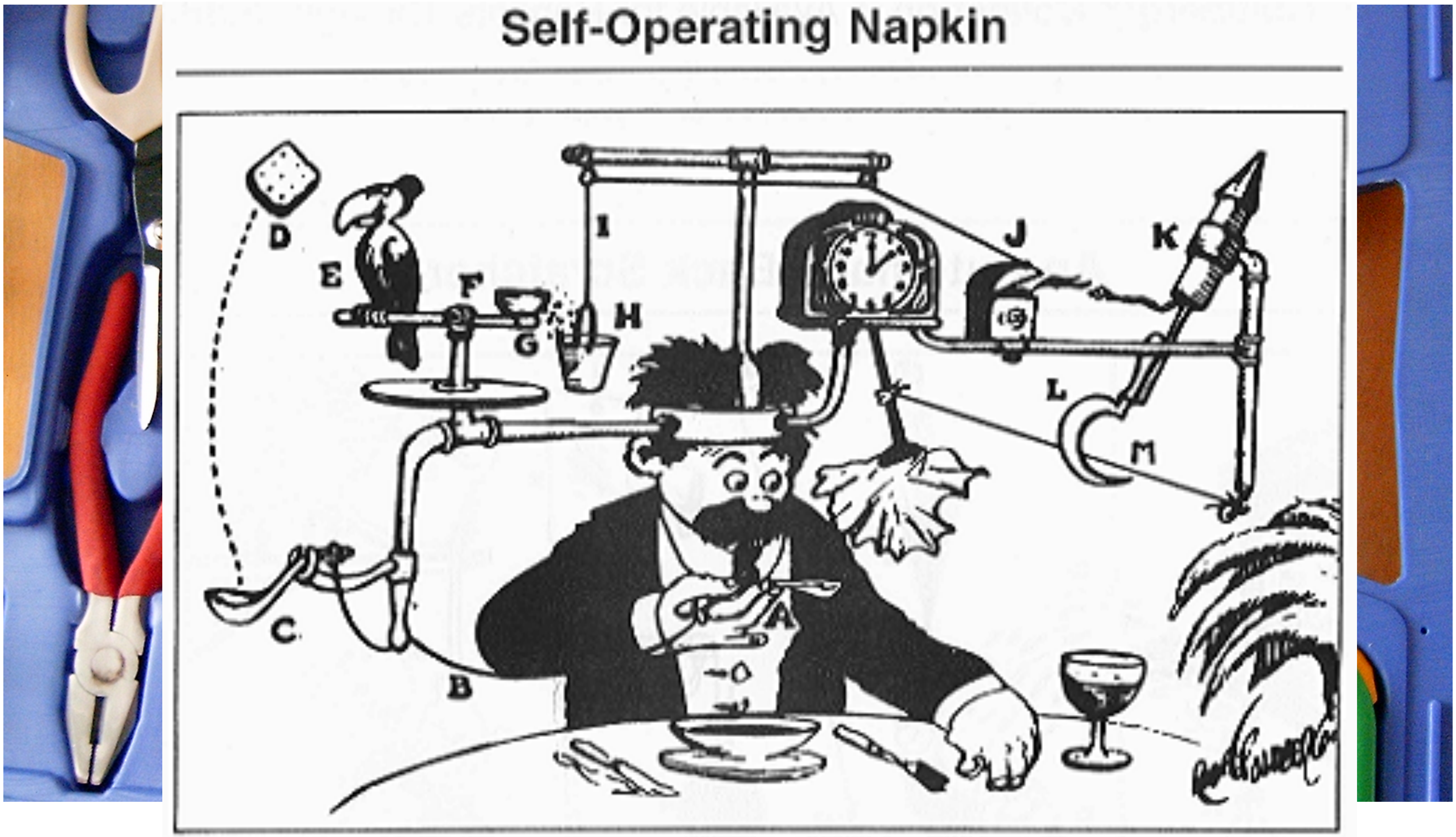
Some Thoughts About Cryptography

- Cryptography helps after you've identified your threat model and goals
  - Famous quote: "Those who think that cryptography can solve their problems don't understand cryptography and don't understand their problems."



Think of Cryptography as a Tool in your Toolbox

Think of Cryptography as a Tool in your Toolbox



Improved Security, Increased Risk

- RFIDs in car keys:
  - RFIDs in car keys make it harder to hotwire a car
  - Result: Car jackings increased

Improved Se

- RFIDs in car keys:
  - RFIDs in car keys
  - Result: Car jackin

Biometric car lock defeated by cutting off owner's finger

POSTED BY CORY DOCTOROW, MARCH 31, 2005 7:53 AM | PERMALINK

Andrei sez, "'Malaysia car thieves steal finger.' This is what security visionaries Bruce Schneier and Ross Anderson have been warning about for a long time. Protect your $75,000 Mercedes with biometrics and you risk losing whatever body part is required by the biometric mechanism."

> ...[H]aving stripped the car, the thieves became frustrated when they wanted to restart it. They found they again could not bypass the immobiliser, which needs the owner's fingerprint to disarm it.
>
> They stripped Mr Kumaran naked and left him by the side of the road - but not before cutting off the end of his index finger with a machete.

Key Entry Pad (4-digit PIN)

Image from profmason.com

- This is the key pad on my office safe.
- Inside my safe is a copy of final exam.
- How long would it take you to break in?

- Answer (combinatorics):
  - 10^4 tries *maximum*
  - 10^4 / 2 tries on *average*
- Answer (unit conversion):
  - 3 seconds per try --> 4 hours and 10 minutes on average

Key Entry Pad (4-digit PIN)

Image from profmason.com

- Now assume the safe automatically calls police after 3 failed attempts.
- What is the probability that you will guess the PIN within 3 tries? (Assume no repeat tries.)

- Answer (combinatorics)
  - 10000 choose 3 possible choices for the 3 guesses
  - 1 · (9999 choose 2) possible choices contain the correct PIN
  - So success probability is 3 / 10000

Key Entry Pad (4-digit PIN)

Image from profmason.com

- Could you do better at guessing the PIN?
- Answer (*chemical* combinatorics):
  - Put different chemical on each key (NaCl, KCl, LiCl, …)

Idea from http://eprint.iacr.org/2003/217.ps

Key Entry Pad (4-digit PIN)

Image from profmason.com

- Could you do better at guessing the PIN?
- Answer (*chemical* combinatorics):
  - Put different chemical on each key (NaCl, KCl, LiCl, ...)
  - Observe residual patterns after I access safe

Idea from http://eprint.iacr.org/2003/217.ps

Key Entry Pad (4-digit PIN)

Image from profmason.com

- Could you do better at guessing the PIN?
- Answer (*chemical* combinatorics):
  - Put different chemical on each key (NaCl, KCl, LiCl, ...)
  - Observe residual patterns after I access safe

Idea from http://eprint.iacr.org/2003/217.ps

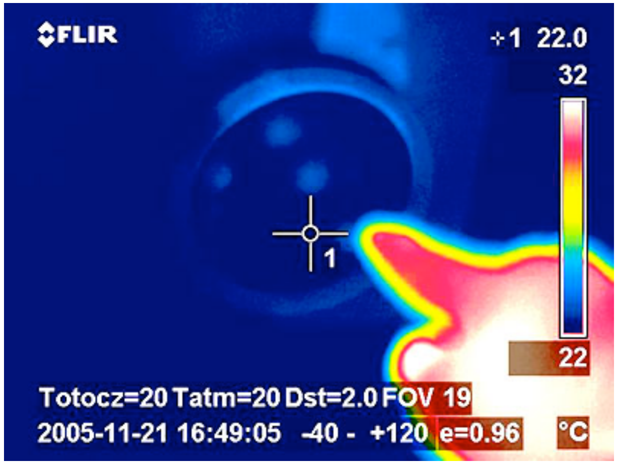
Key Entry Pad (4-digit PIN)

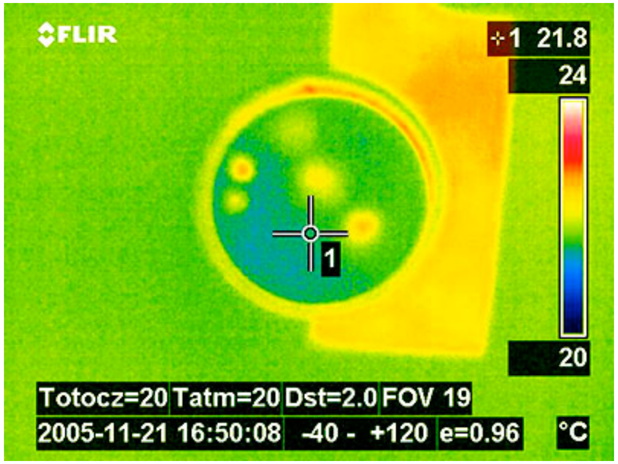
Image from profmason.com

- Could you do better at guessing the PIN?
- Answer (*chemical* combinatorics):
  - Put different chemical on each key (NaCl, KCl, LiCl, ...)
  - Observe residual patterns after I access safe
- **Lesson:** Consider the complete system, physical security, etc.
- **Lesson:** Think outside the box

Idea from http://eprint.iacr.org/2003/217.ps

Thermal Patterns



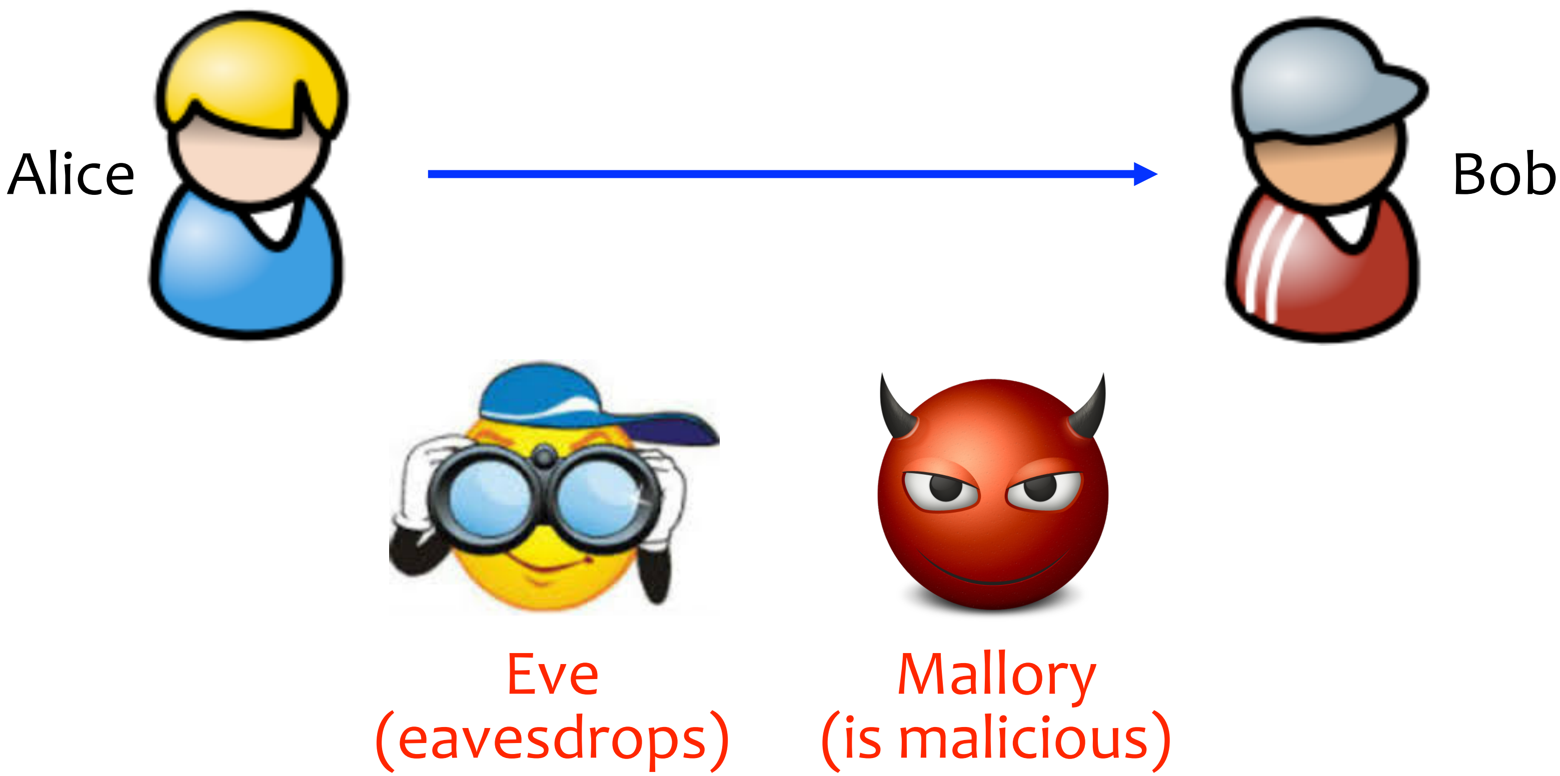


Images from http://lcamtuf.coredump.cx/tsafe/

Cryptography: Terminology, Patterns, and Principles

Alice and Bob

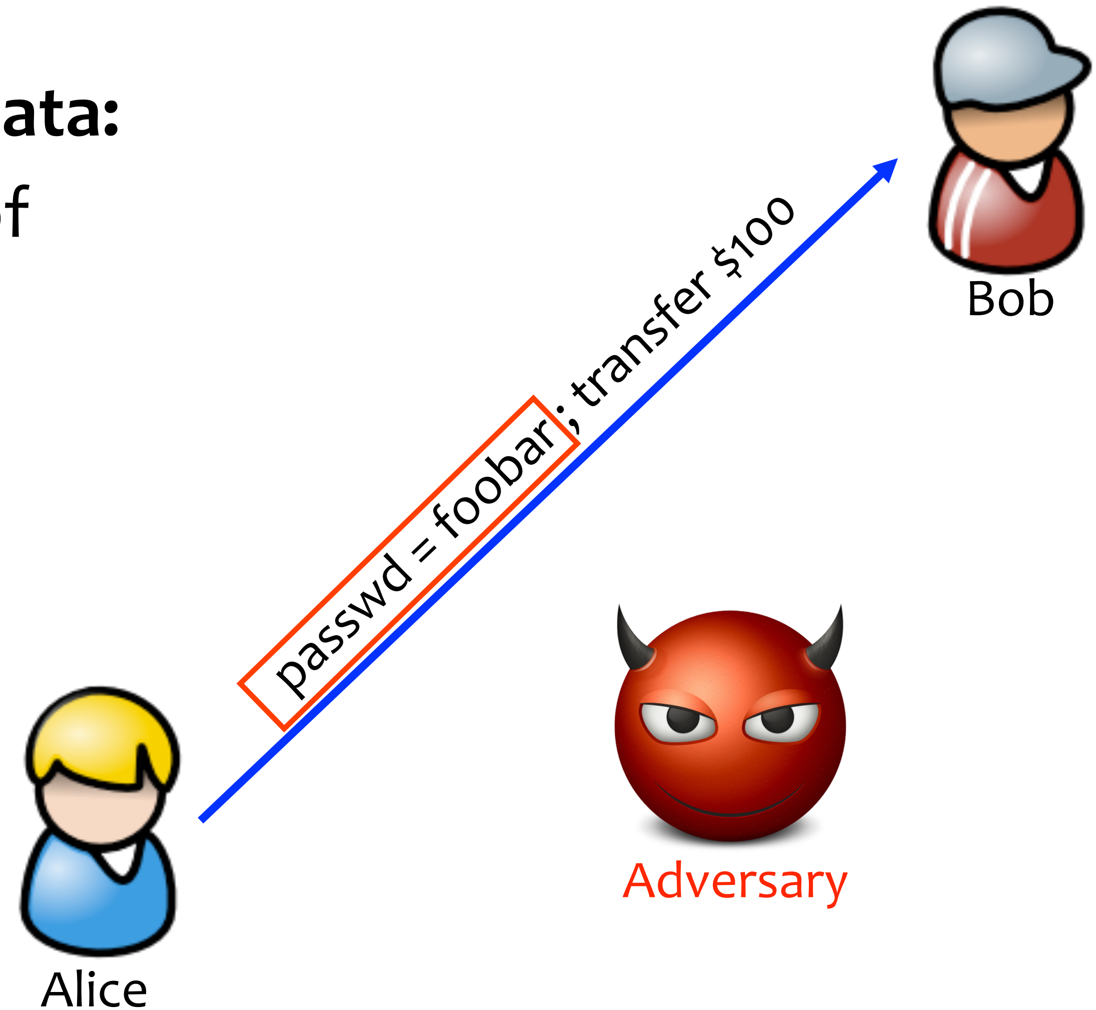
- Archetypical characters



Common Communication Security Goals

Confidentiality of data:

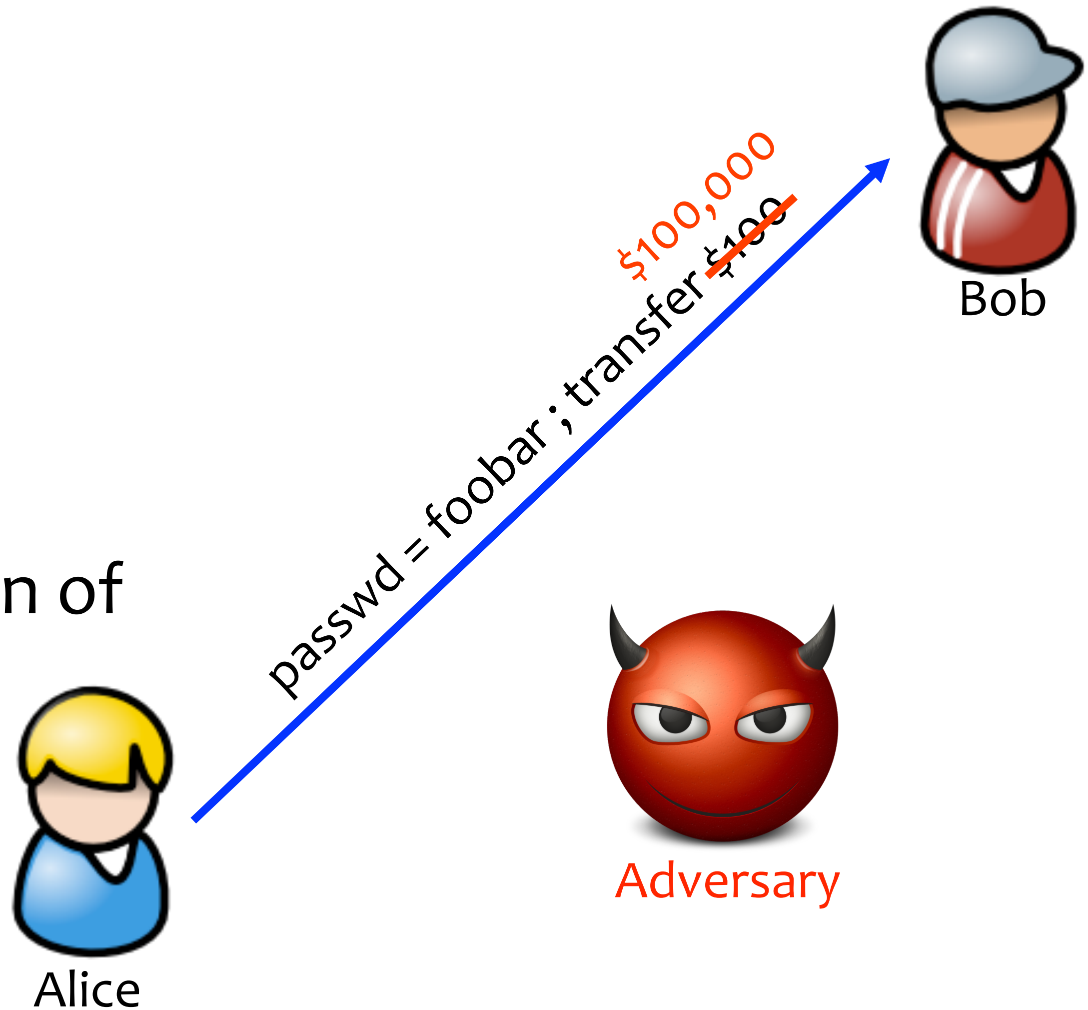
Prevent exposure of information



Common Communication Security Goals

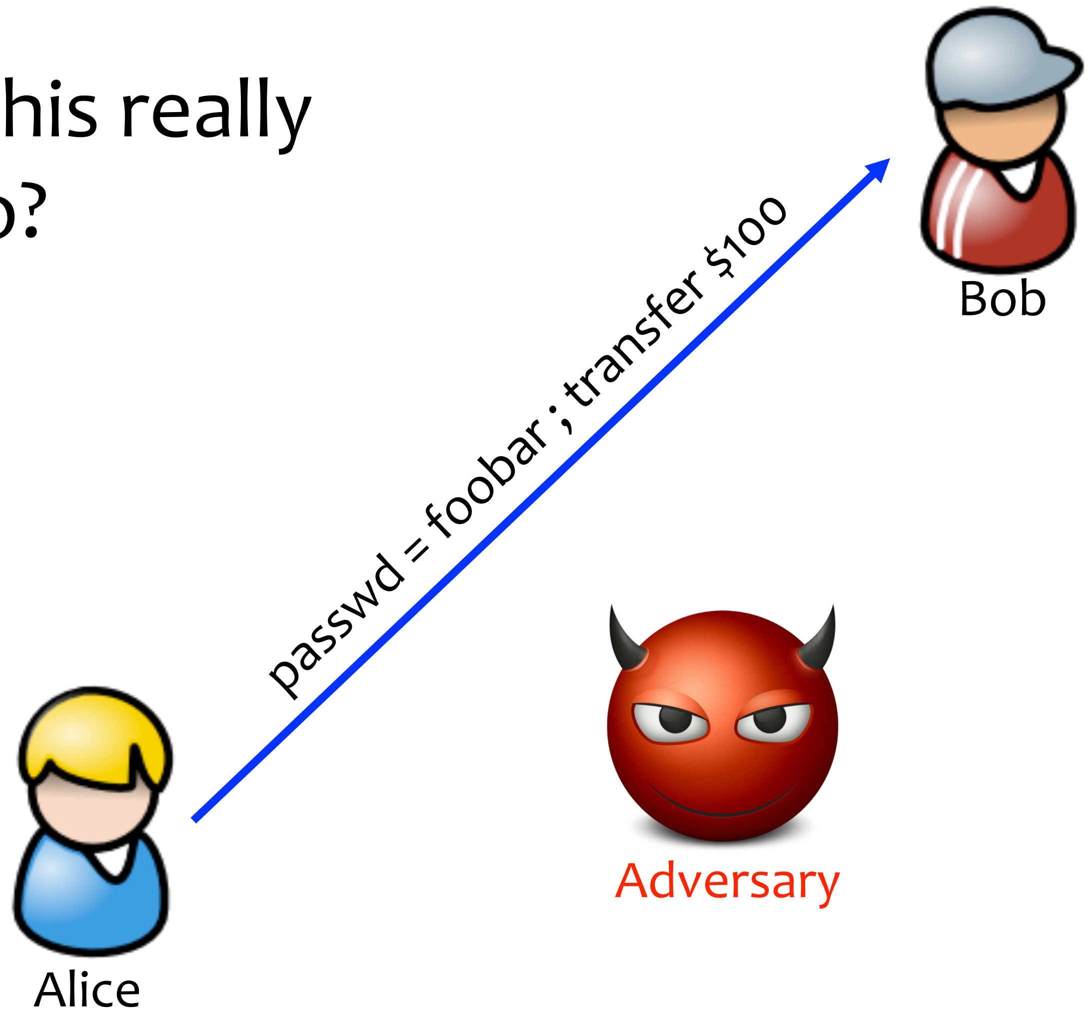
Integrity of data:

Prevent modification of information



Common Communication Security Goals

Authenticity : Is this really Bob I'm talking to?

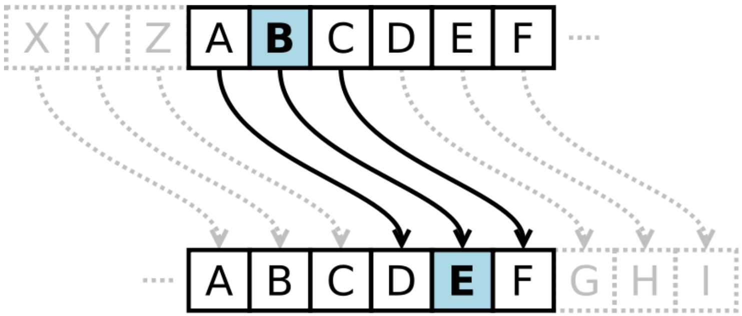


History

- Substitution Ciphers
  - Caesar Cipher
- Transposition Ciphers
- Codebooks
- Machines

- Recommended Reading: **The Codebreakers** by David Kahn and **The Code Book** by Simon Singh.

History: Caesar Cipher (Shift Cipher)



- Plaintext letters are replaced with letters a fixed shift away in the alphabet.
- Example:
  - Plaintext: `The quick brown fox jumps over the lazy dog`
  - Key: Shift 3

    `ABCDEFGHIJKLMNOPQRSTUVWXYZ`

    `DEFGHIJKLMNOPQRSTUVWXYZABC`

  - Ciphertext: `WKHTX LFNEU RZQIR AMXPS VRYHU WKHOD CBGRJ`

History: Caesar Cipher (Shift Cipher)

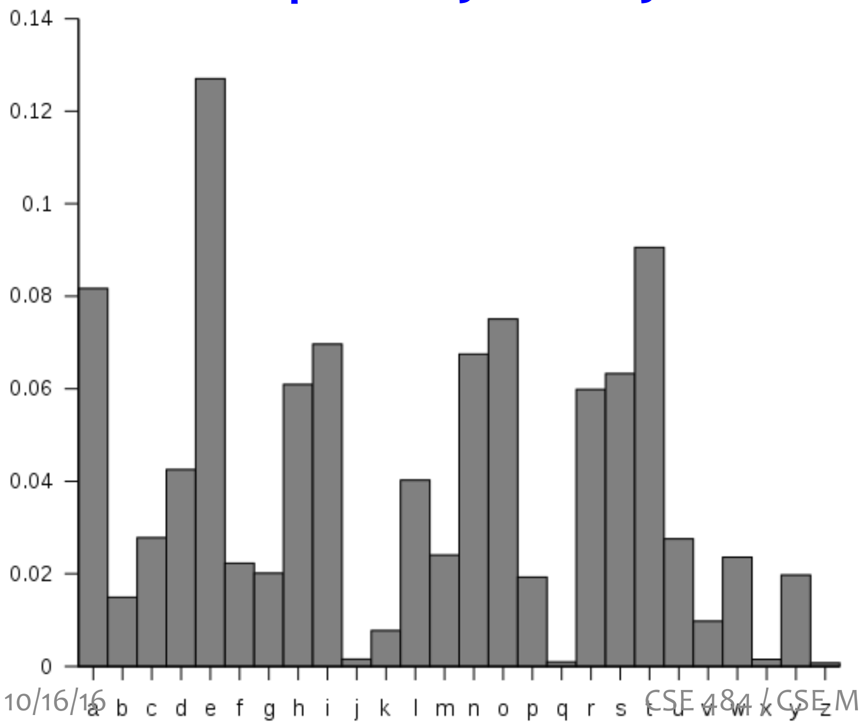
- ROT13: shift 13 (encryption and decryption are symmetric)
- What is the key space?
  – 26 possible shifts.
- How to attack shift ciphers?
  – Brute force.

History: Substitution Cipher

- Superset of shift ciphers: each letter is substituted for another one.
- Add a secret key
- Example:
  - Plaintext: `ABCDEFGHIJKLMNOPQRSTUVWXYZ`
  - Cipher: `ZEBRASCDFGHIJKLMNOPQTUVWXY`
- "State of the art" for thousands of years

History: Substitution Cipher

- What is the key space? 26! ~= 2^88
- How to attack?
  - Frequency analysis.



Bigrams:

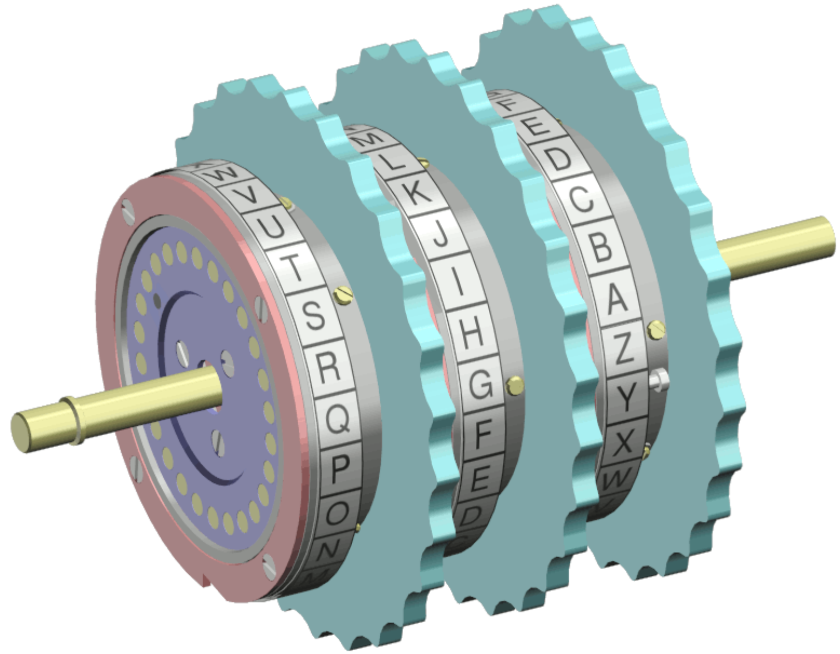
th 1.52%	en 0.55%	ng 0.18%
he 1.28%	ed 0.53%	of 0.16%
in 0.94%	to 0.52%	al 0.09%
er 0.94%	it 0.50%	de 0.09%
an 0.82%	ou 0.50%	se 0.08%
re 0.68%	ea 0.47%	le 0.08%
nd 0.63%	hi 0.46%	sa 0.06%
at 0.59%	is 0.46%	si 0.05%
on 0.57%	or 0.43%	ar 0.04%
nt 0.56%	ti 0.34%	ve 0.04%
ha 0.56%	as 0.33%	ra 0.04%
es 0.56%	te 0.27%	ld 0.02%
st 0.55%	et 0.19%	ur 0.02%

Trigrams:

1. the	6. ion	11. nce
2. and	7. tio	12. edt
3. tha	8. for	13. tis
4. ent	9. nde	14. oft
5. ing	10. has	15. sth

History: Enigma Machine

Uses rotors (substitution cipher) that change position after each key.



Key = initial setting of rotors

Key space?

26^n for n rotors

Kerckhoff's Principle

- Don't rely on secrecy of your algorithms for the security of your cryptography

Kerckhoff's Principle

- Security of a cryptographic object should depend only on **the secrecy of the secret key.**

Kerckhoff's Principle

Secret Key "K"

Alice

K

Everyone knows
cryptographic algorithm A

Adversary

K?

Bob

K

How Cryptosystems Work Today

- Layered approach:
  - Cryptographic primitives, like block ciphers, stream ciphers, hash functions, and one-way trapdoor permutations
  - Cryptographic protocols, like CBC mode encryption, CTR mode encryption, HMAC message authentication
- Public algorithms (Kerckhoff's Principle)
- Security proofs based on assumptions (not this course)

- Don't roll your own!

Flavors of Cryptography

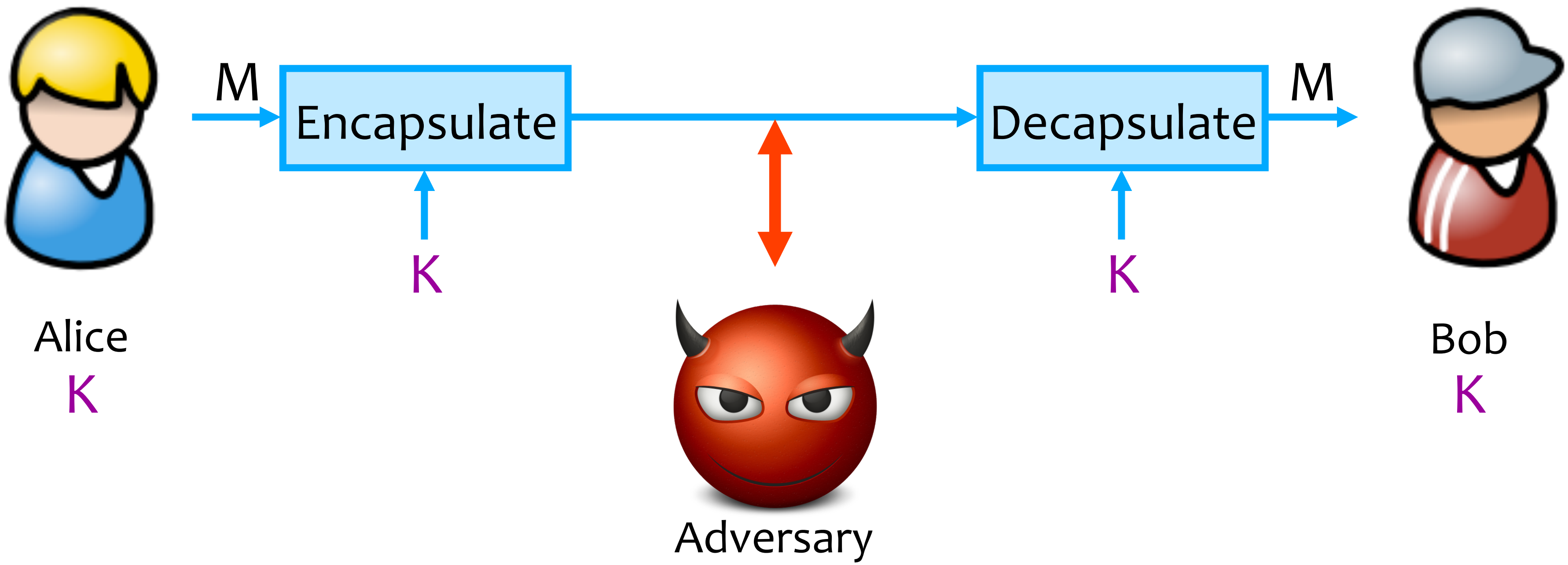
- Symmetric cryptography
  - Both communicating parties have access to a shared random string K, called the key.
- Asymmetric cryptography
  - Each party creates a public key pk and a secret key sk.

Flavors of Cryptography

- Symmetric cryptography
  - Both communicating parties have access to a shared random string K, called the key.
  - Challenge: How do you privately share a key?
- Asymmetric cryptography
  - Each party creates a public key pk and a secret key sk.
  - Challenge: How do you validate a public key?

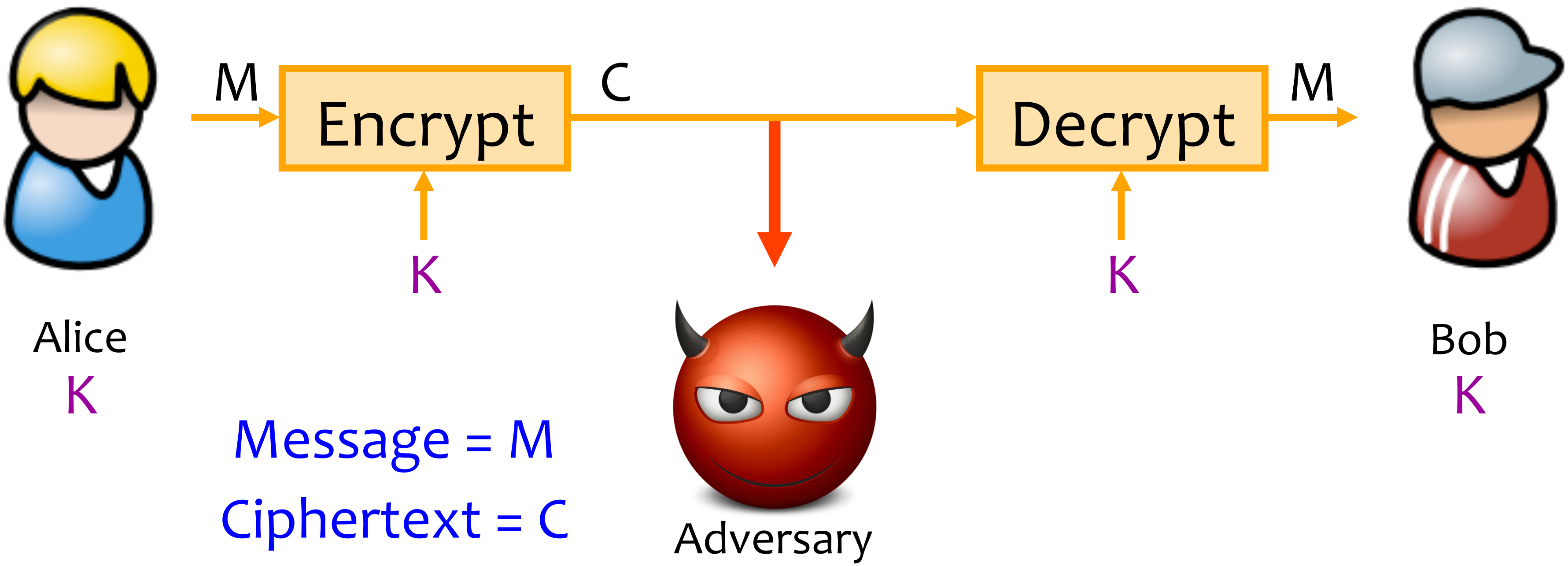
Symmetric Setting

Both communicating parties have access to a shared random string K, called the key.



Achieving Privacy (Symmetric)

Encryption schemes: A tool for protecting privacy.



Message = M

Ciphertext = C

Flavors of Cryptography

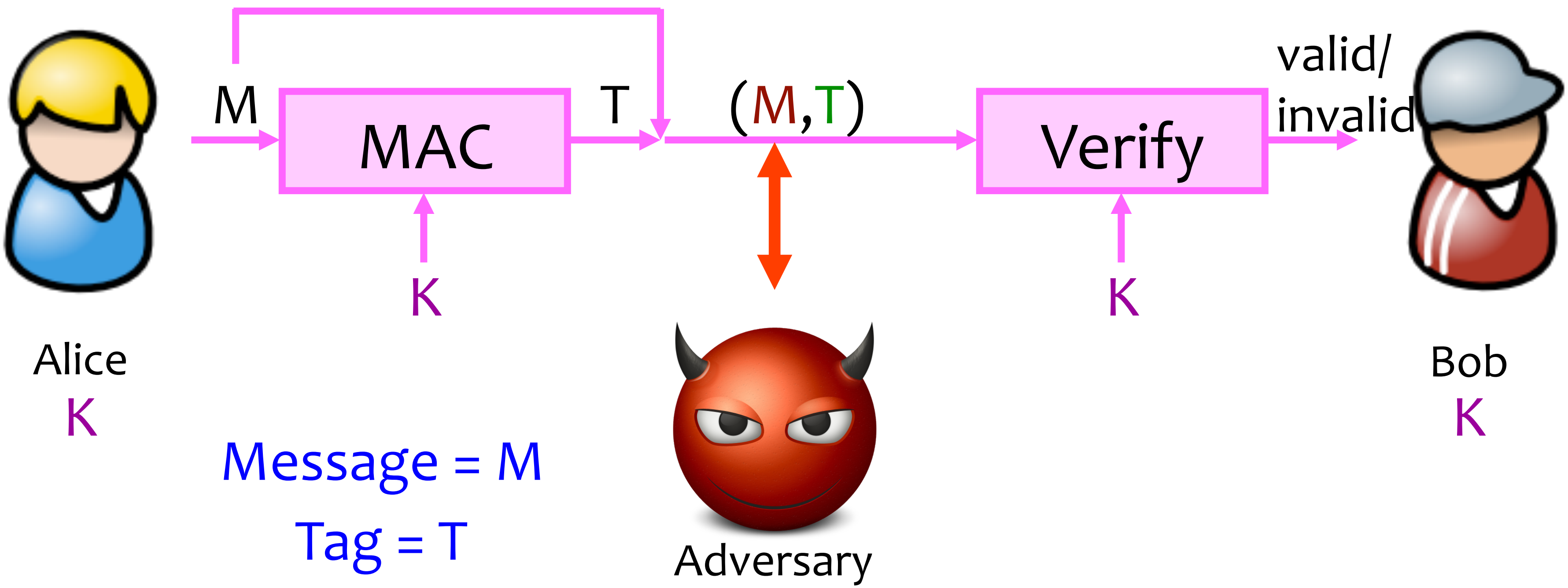
- Symmetric cryptography
  - Both communicating parties have access to a shared random string K, called the key.

- Asymmetric cryptography
  - Each party creates a public key pk and a secret key sk.

Achieving Integrity (Symmetric)

Message authentication schemes: A tool for protecting integrity.

(Also called message authentication codes or MACs.)

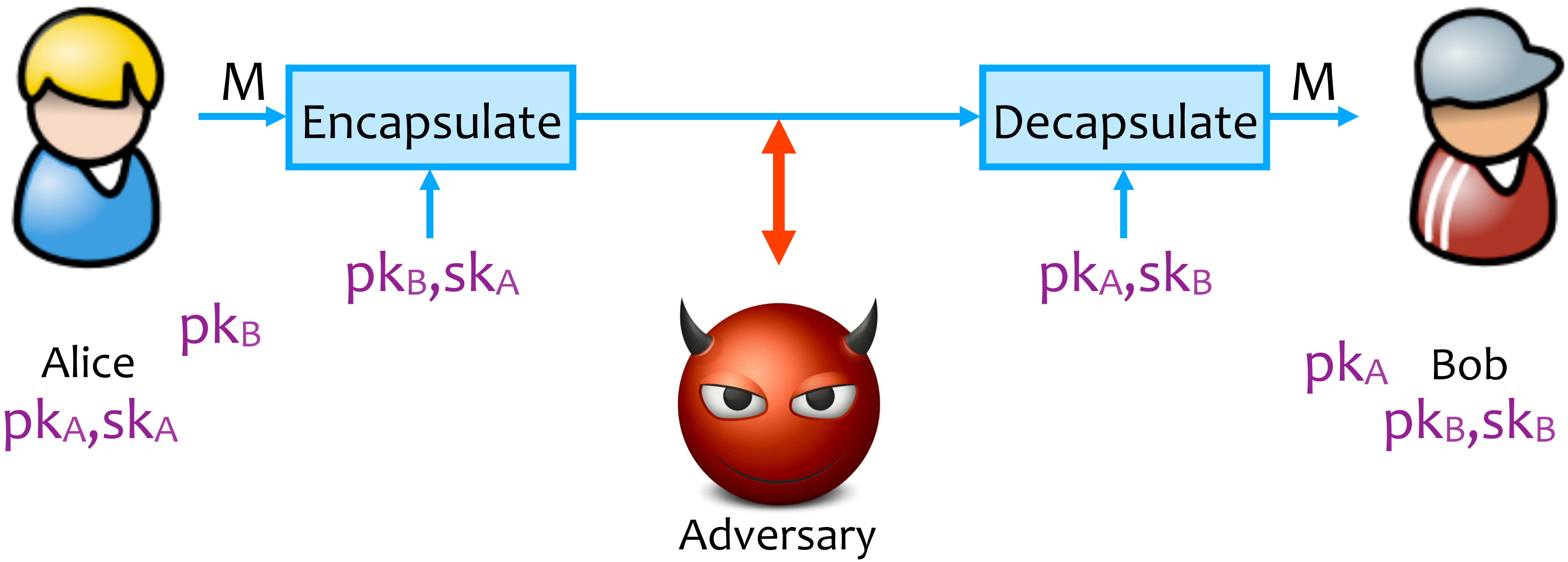


Message = M

Tag = T

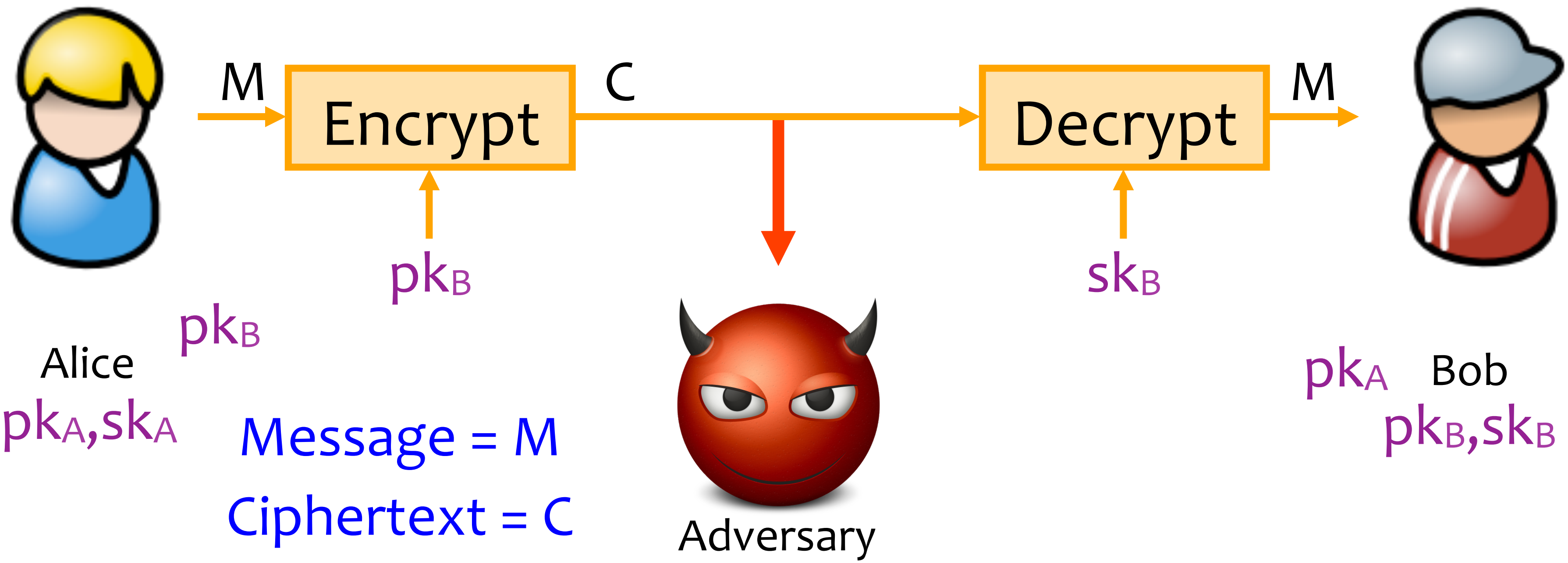
Asymmetric Setting

Each party creates a public key pk and a secret key sk.



Achieving Privacy (Asymmetric)

Encryption schemes: A tool for protecting privacy.

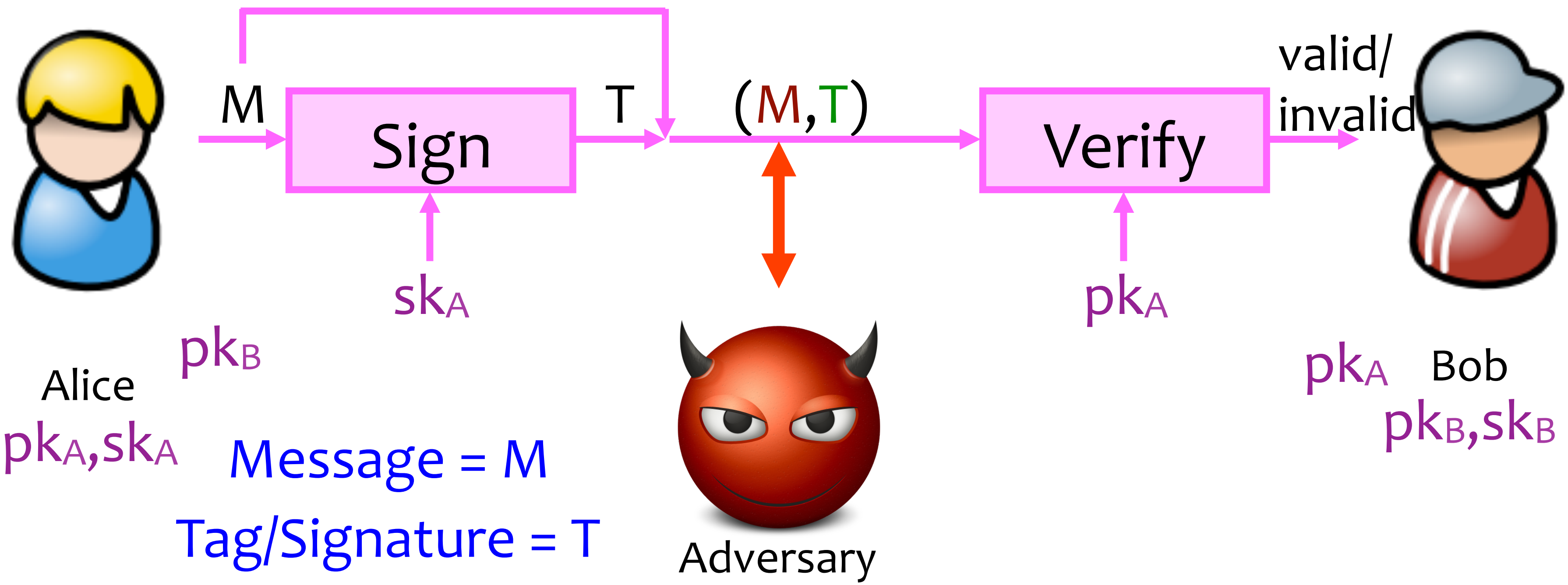


Message = M

Ciphertext = C

Achieving Integrity (Asymmetric)

Digital signature schemes: A tool for protecting integrity and authenticity.



Message = M

Tag/Signature = T