22. Computer Security Basics – Building Secure Hardware (and Software)

EECS 370 – Introduction to Computer Organization - Winter 2016

Profs. Valeria Bertacco & Reetu Das

EECS Department
University of Michigan in Ann Arbor, USA

Announcements

- Last Homework due Tuesday

- THURSDAY: last lecture! Final review (Prof. Das)

Outline for Today’s Lecture

- Why Building Secure Hardware?
- Security Basics
- Security Exploits in Hardware
- System Security Protections

Why is Security Important?
(to Architects and Compiler Designers)

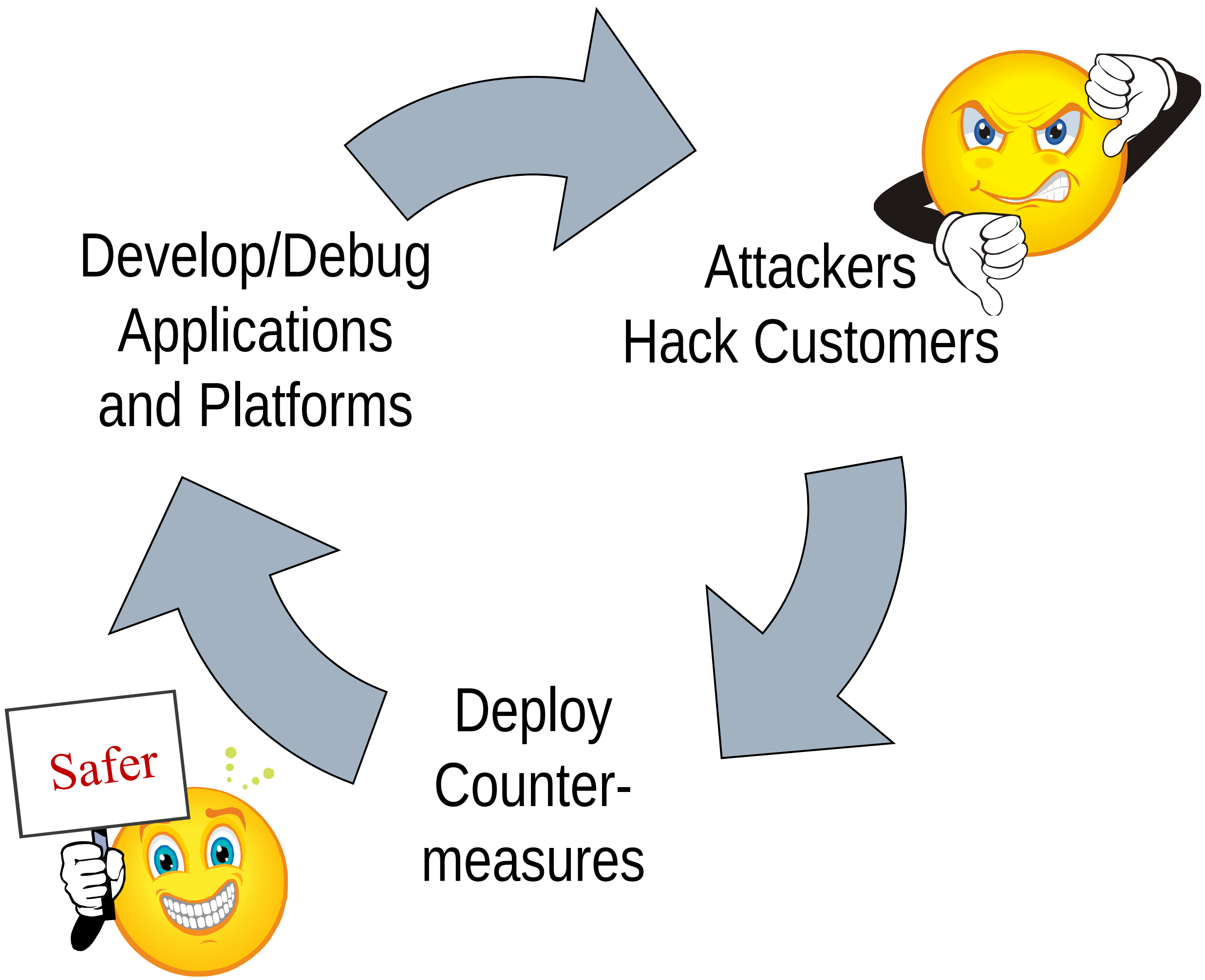
OPPORTUNITY: Hardware and system-level solutions are needed to protect software and intellectual property (IP)

- Hardware and low-level software support improves speed and quality of cryptography

- Hardware and system-level software support can most effectively seal up security vulnerabilities

COST: Hardware and system-level software vulnerabilities enable security attacks

The Security Arms Race

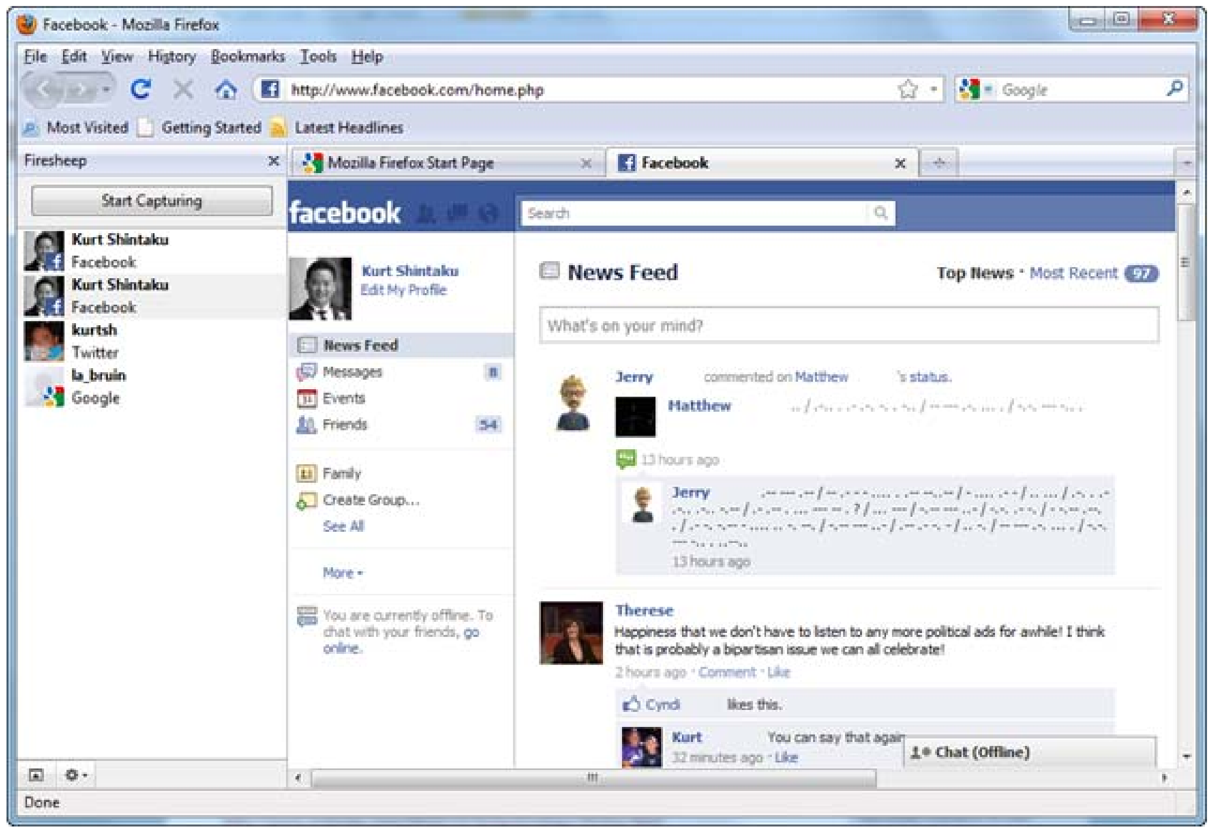


Why Do Attackers Attack?

- To gain access to private information, e.g., credit card numbers
- To punish/embarrass individuals and institutions, e.g., Sony
- To gain control of machines, e.g., BotNets

Why Do Attackers Attack?(cont)

- To educate and advocate, e.g., FireSheep
- To earn reputation in the attacker community, e.g., hackers vs. script kiddies
- etc…

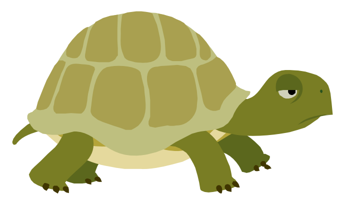


The Ultimate Goal of the Designer

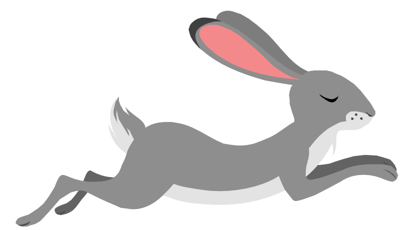
- Win the bear race…

Attackers

Someone more valuable

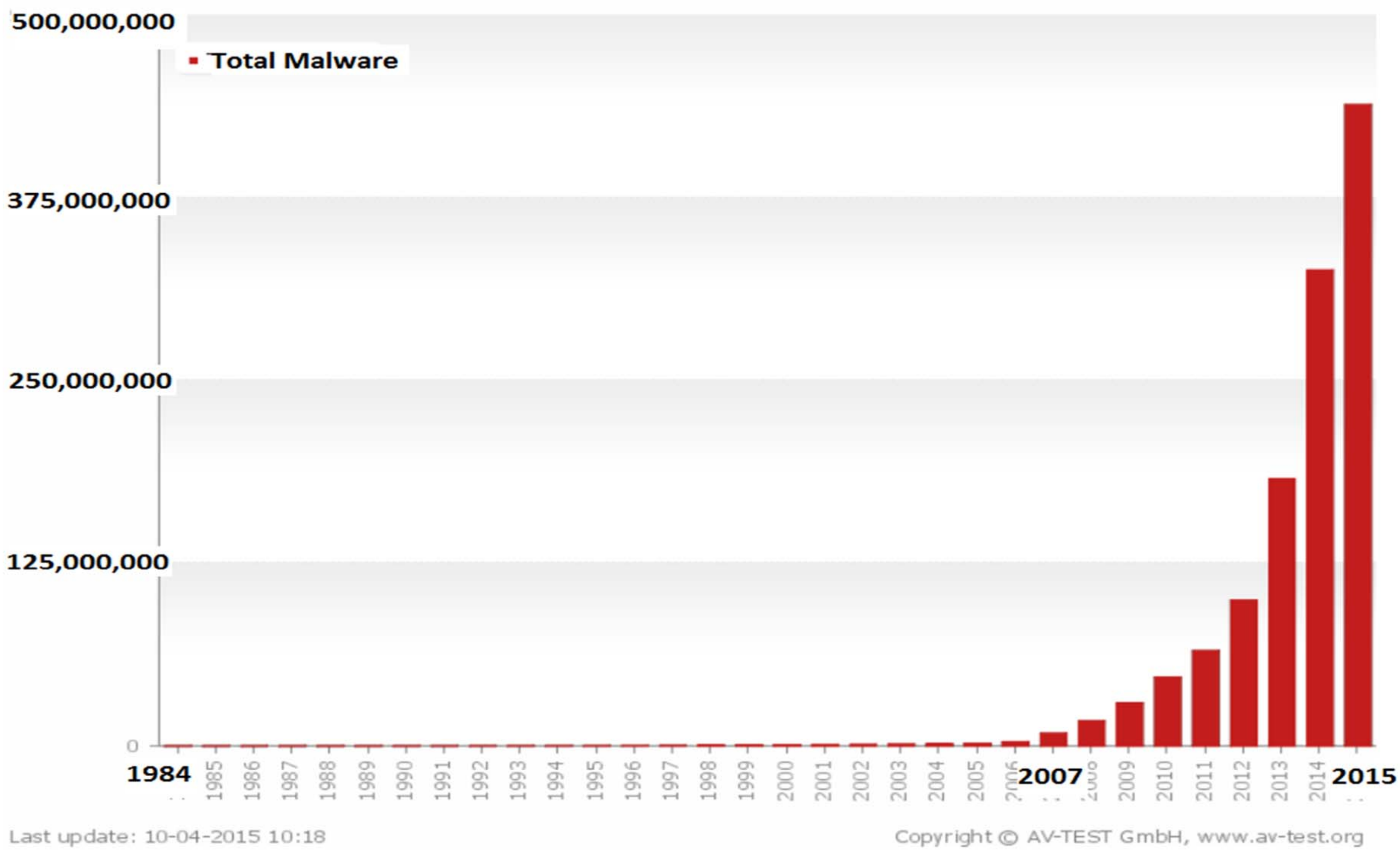


You

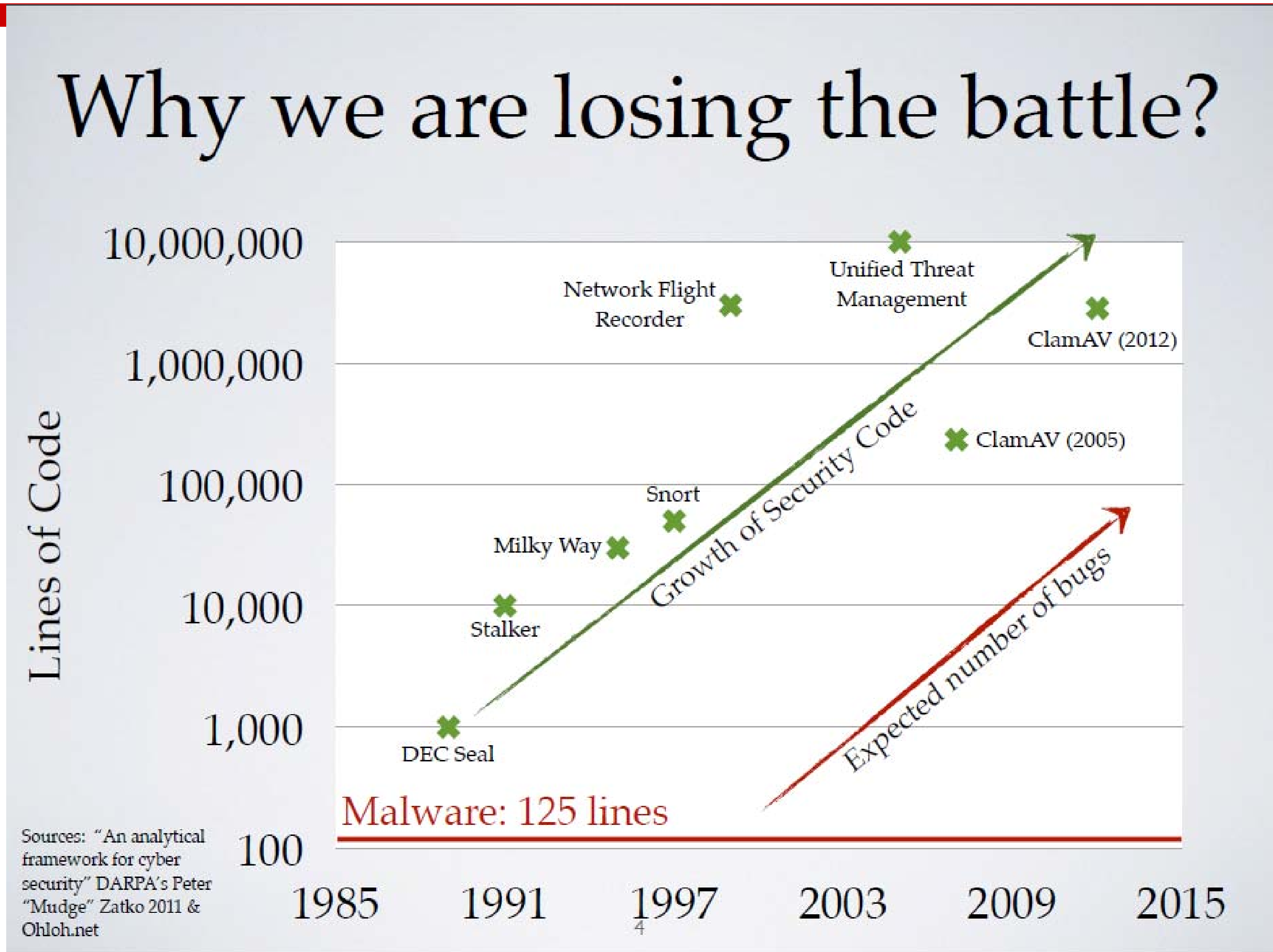


- Value = f(easy of attack, population, loot therein, goodwill, etc…)

Flood of Malware



Flood of Malware



"Hardware Malware Detectors" Demme, et al.

Recent developments: hw-based attacks

- 2008: Kris Kapersky announced the discovery of an OS-independent remote code execution exploit based on an Intel CPU bug (not disclosed)
- 2008: UIUC researcher Sam King demonstrate that 1400 additional gates added to a Leon SPARC processor creates an effective Linux backdoor
- 2008: Princeton researcher Ed Felten demonstrates that disk encryption keys can be extraction after system shutdown from frozen DRAM chips
- 2010: Christopher Tarnovsky announced a successful hardware exploit of an Infineon TPM chip
- 2011: Sturton/Hicks develop non-stealthy malicious circuits, provide plausible deniability to rogue designers
- 2014: Rowhammer bug demonstrated, able to flip DRAM bits in adjacent rows even without access permission

Security Basics

- Cryptography
  - Symmetric key cryptography
  - Asymmetric key cryptography
  - Secure sockets layer (SSL) overview
  - Cryptographic Hashes

Value of Cryptography

The Security Division of EMC

$2.1 billion

1,300 employees

by Symantec

$5.7 billion

1,000 employees

$39 billion

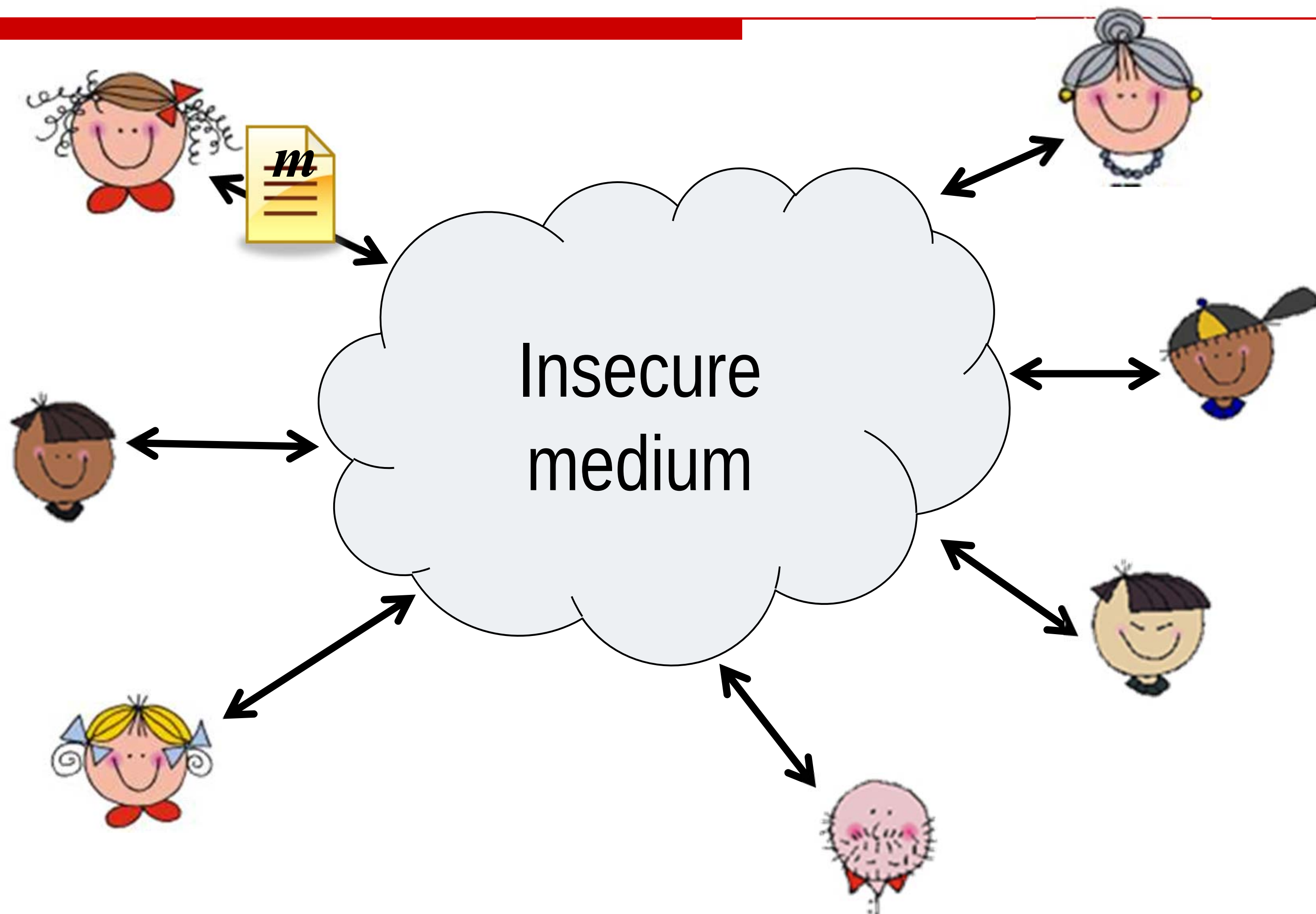
18,000 employees

$82 billion

34,000 employees

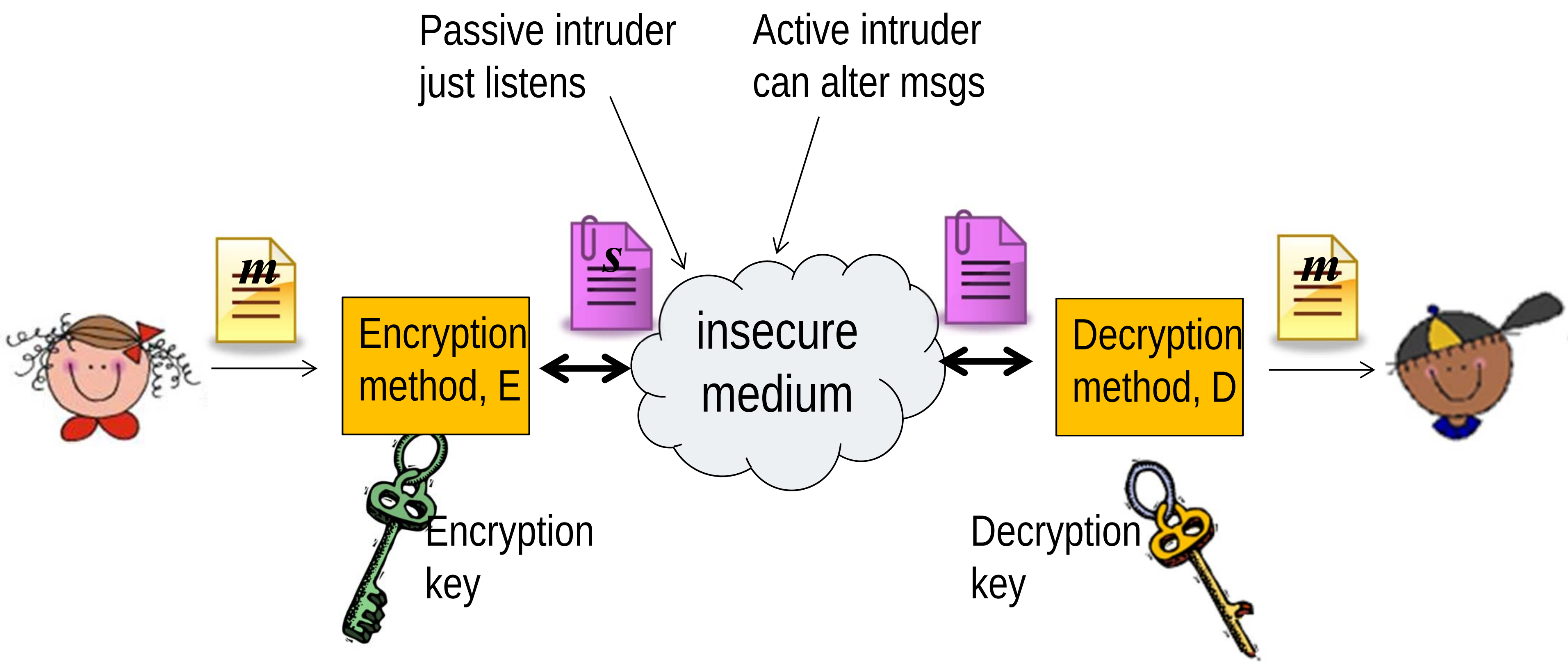
Source: Wiki & NASDAQ

What is Authenticated Communication?

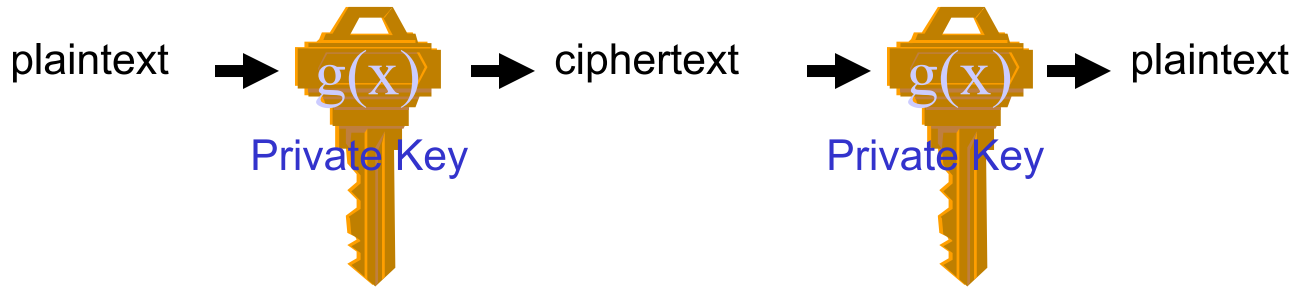


How do we enable authenticated communication?

Terminology

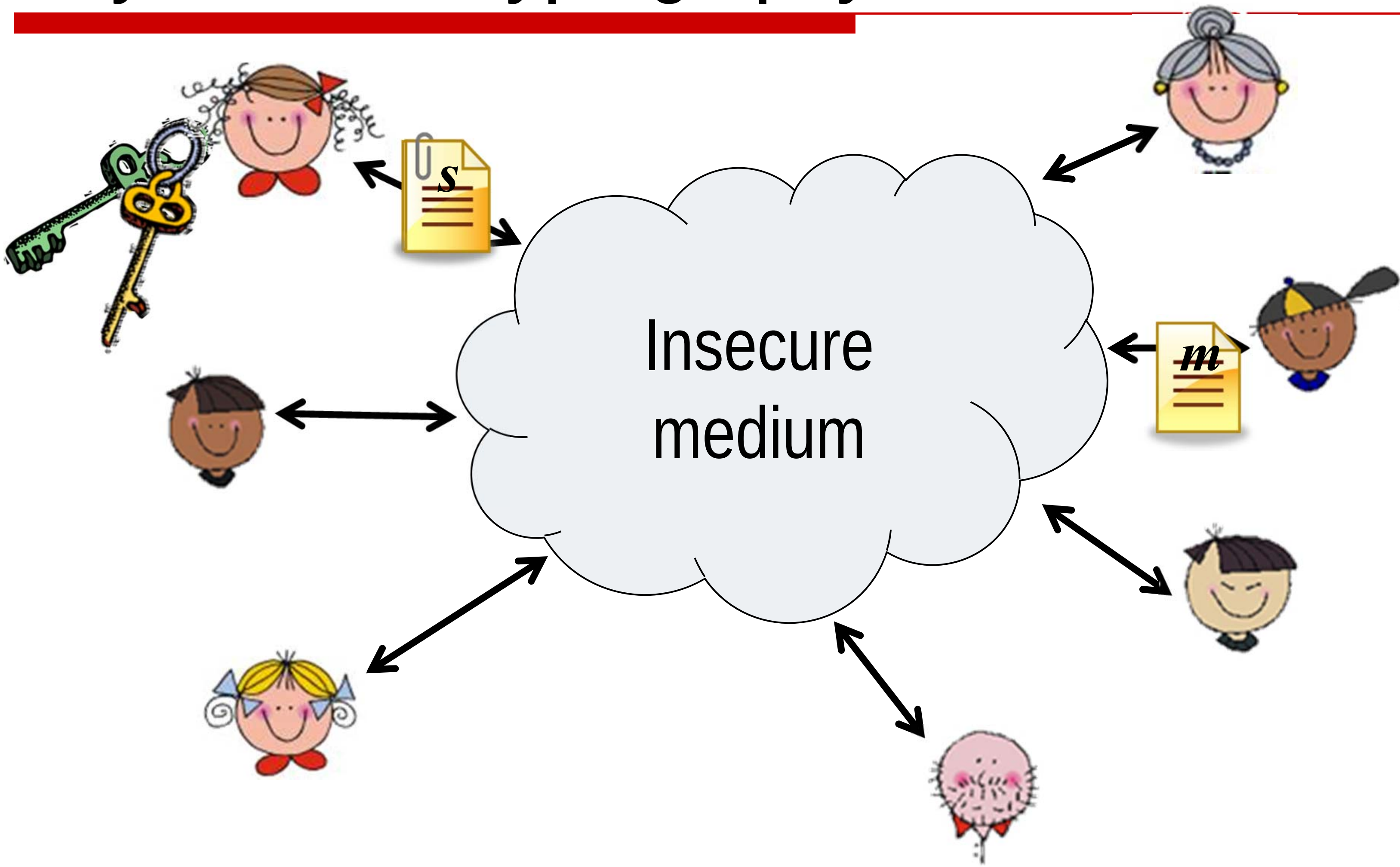


Symmetric Key Cryptography

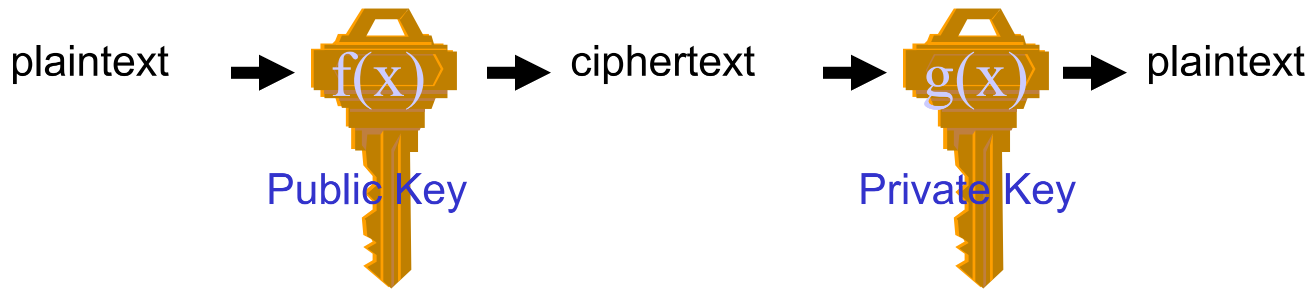


- Sender and receiver share the same private key
- Key has to be exchanged by some other mean
- Anyone who knows the private key can listen in
- Has to be REVERSIBLE
- Often called a “private-key cipher”
- Examples: AES, DES, Blowfish

Asymmetric Cryptography



Asymmetric Key Cryptography

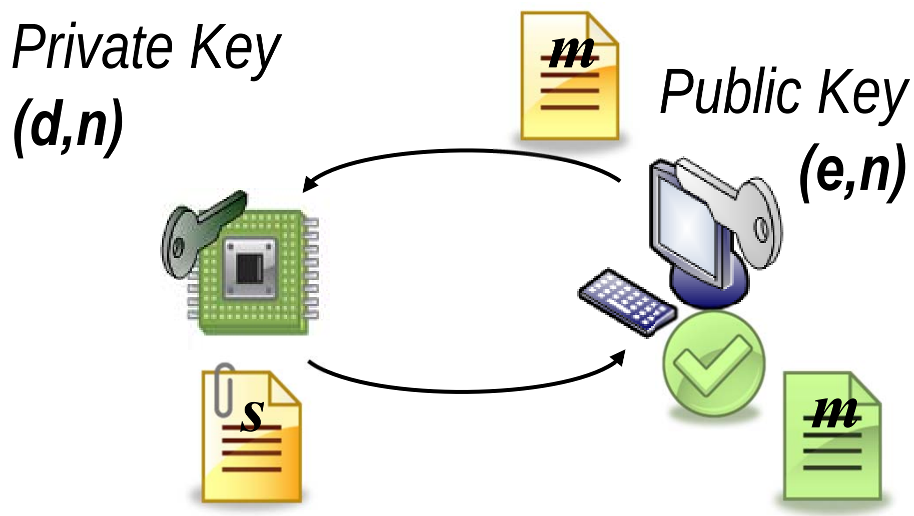


- Sender has the receiver's public key, receiver has the private key
- Anyone can encrypt a message with the public key, only the holder of the private key can decrypt the message
  - Allows sharing of private information with no initial shared secret
- The reverse path also works: everyone can decrypt a message that was encrypted by the holder of the private key
- Often called a "public-key cipher"
- Examples: RSA, Diffie-Hellman

RSA Authentication

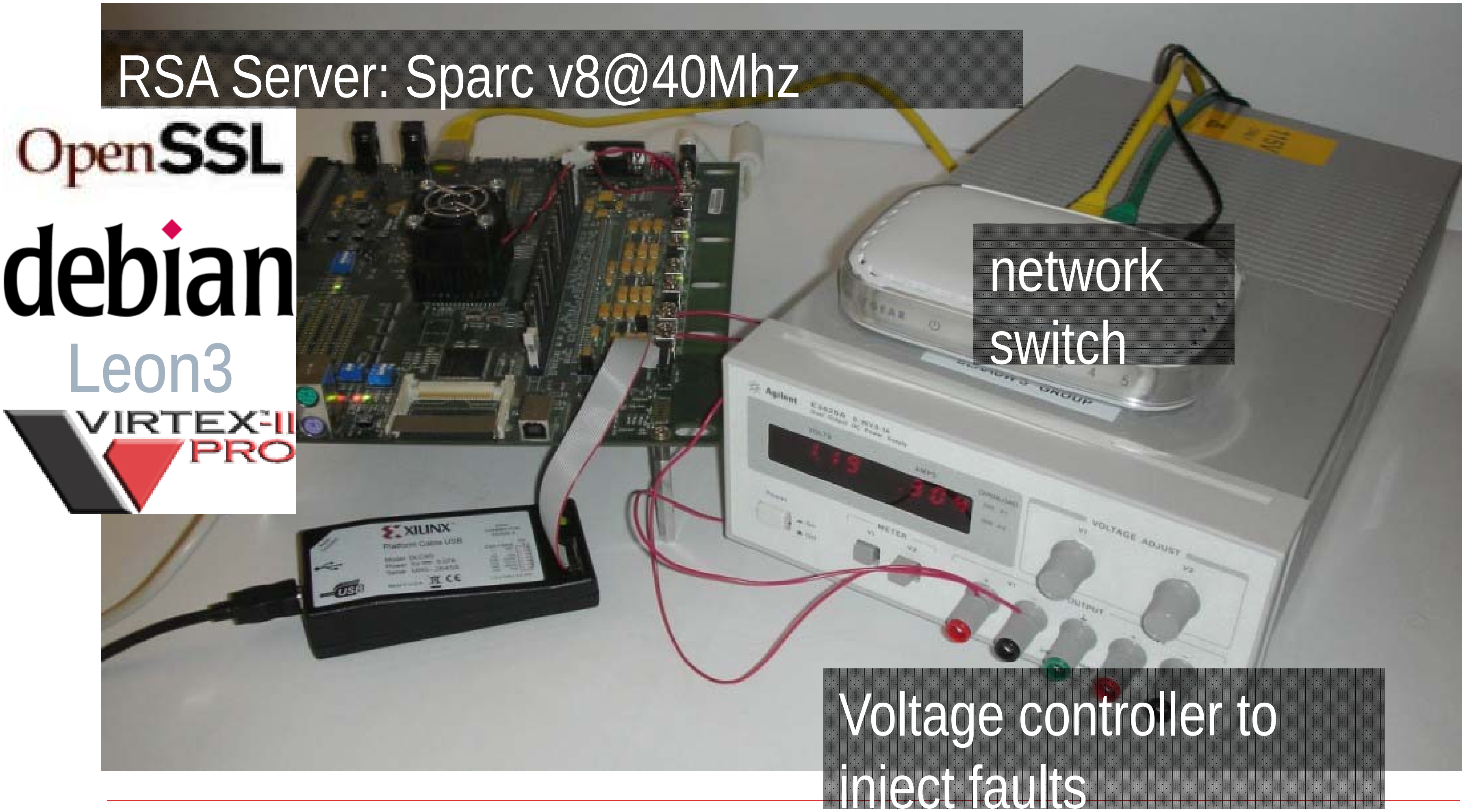
- Client sends a unique message to server
- Server encrypts unique message with private key
- Client decrypts the message with public key and verifies it is the same

- ***Authentication***: only server could return private-key encrypted unique message



Attack on RSA

Bertacco, et al.



Faulty RSA Authentication

Correct Authentication:

- Server challenge:

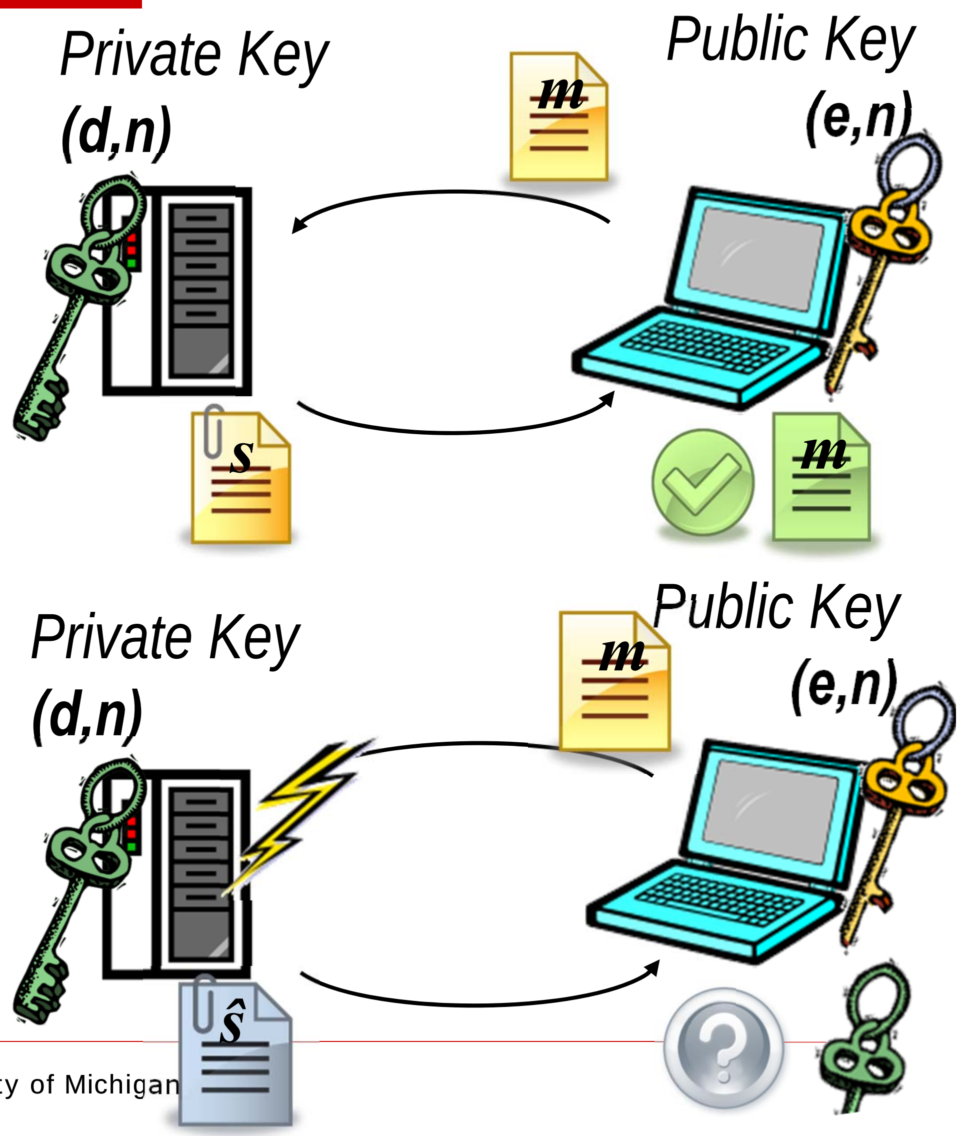
  $s = m^d \bmod n$

- Client verifies:

  $m = s^e \bmod n$

Faulty Server:

$\hat{s} \mathrel{!=} m^d \bmod n$



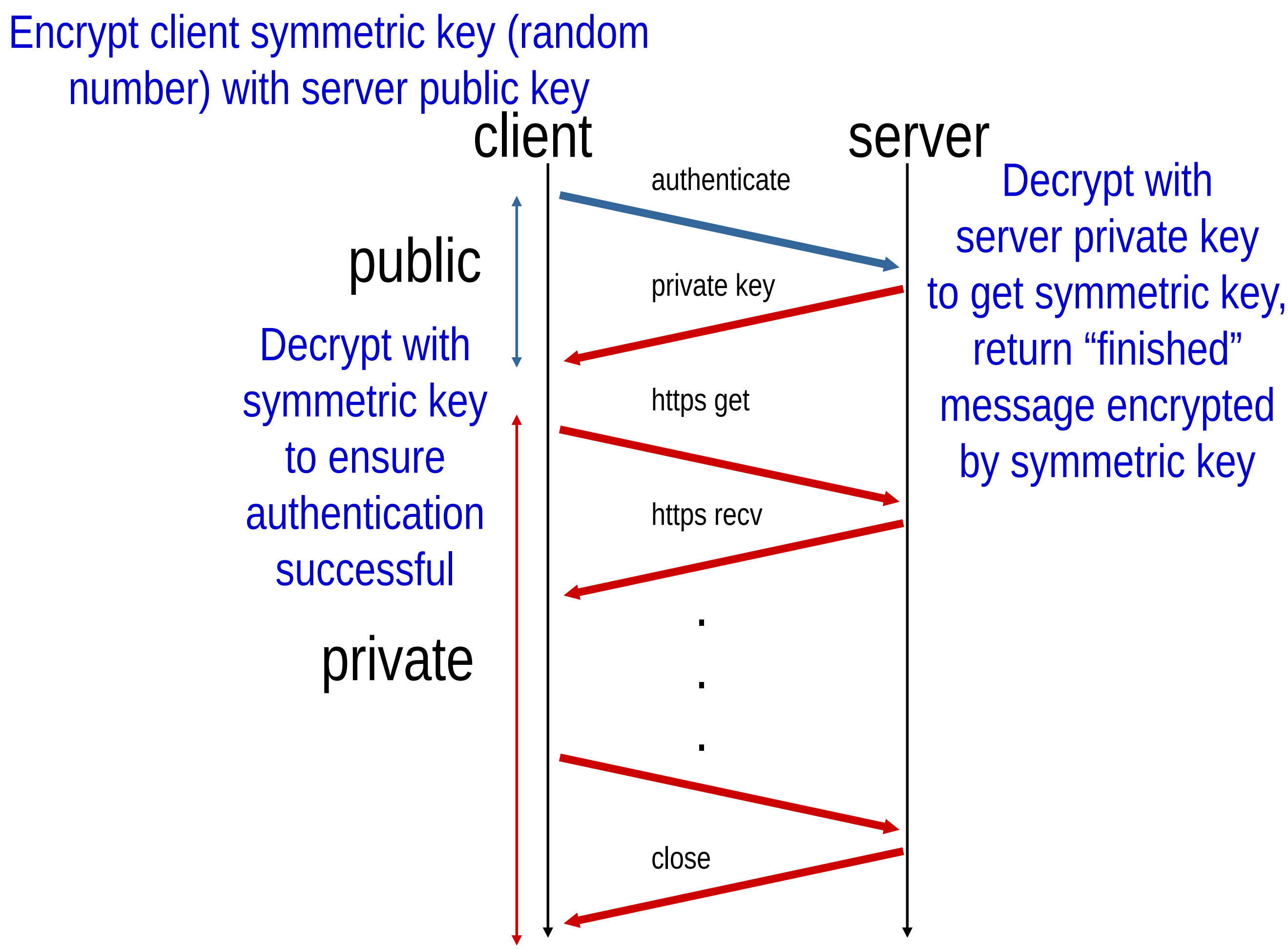
Symmetric vs. Asymmetric Ciphers

- Symmetric Ciphers
  - Fast to compute
  - Require prior shared knowledge to establish private communication

- Asymmetric Ciphers
  - Orders of magnitude slower to compute
  - No shared secrets required to establish private communication

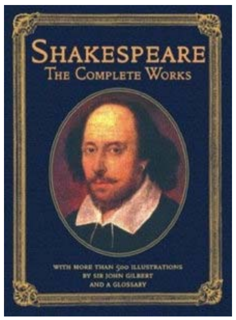
- Individual benefits create a need for both types of cryptography

Secure Sockets Layer (SSL) Overview



Verifying Integrity: Hash Functions

Arbitrary-length message *m*



0xdeadbeefbaadf00d

Cryptographic hash Function, *h*

Fixed-length message digest *y*

- Goal: provide a (nearly) unique "fingerprint" of the message
- Hash function for L-bit hash must demonstrate three properties:
  1. Fast to compute y from m.
  2. One-way: given y = h(m), can't find m' satisfying h(m') = y without $O(2^L)$ search
  3. *Strongly* collision-free: For m_1 != m_2, we find $h(m_1)=h(m_2)$ with probability $1/2^L$
- Widely useful tool, e.g., Has this web page changed?
- Examples: MD5 (cryptographically broken), SHA-1, SHA-2

Hash Application: Password Storage

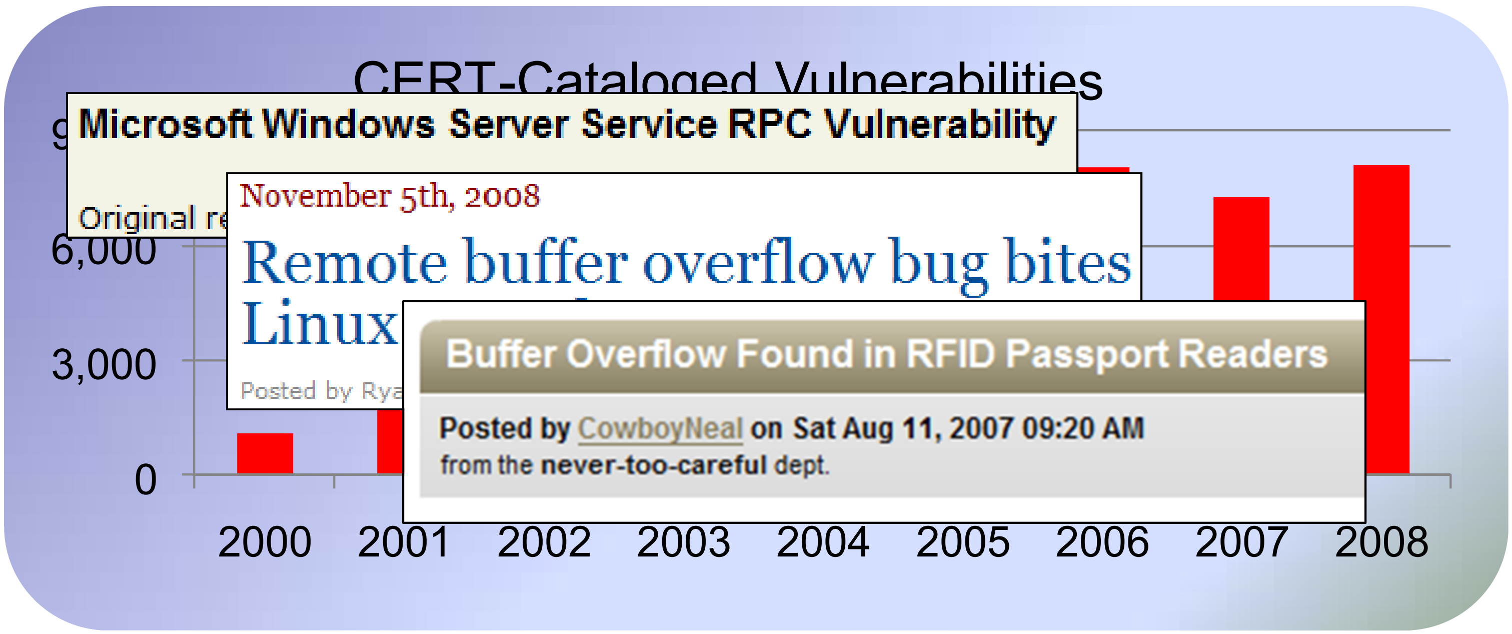
- Never store passwords as plain text
  - If your machine is compromised, so too are all the user passwords
  - E.g., Gawker.com attack in 2010
- Why protect passwords on a compromised machine?
- Instead, store a cryptographic hash of the password
  - Even with a compromised password file, the passwords are still unknown
  - Use “salt” to eliminate the use of “rainbow tables”

Security Basics

Attack		Defense
Buffer overflow attacks	->	No-Execute (NX) Stacks
Heap spray attacks	->	Address Space Layout Randomization (ASLR)
Return-oriented programming attacks	->	Stack Canaries
Rowhammer attacks		
Cold boot attacks		

Security Vulnerabilities are Everywhere

- Most often born out of software bugs
- NIST estimates that S/W bugs cost U.S. $60B/year
- Many of these errors create security vulnerabilities

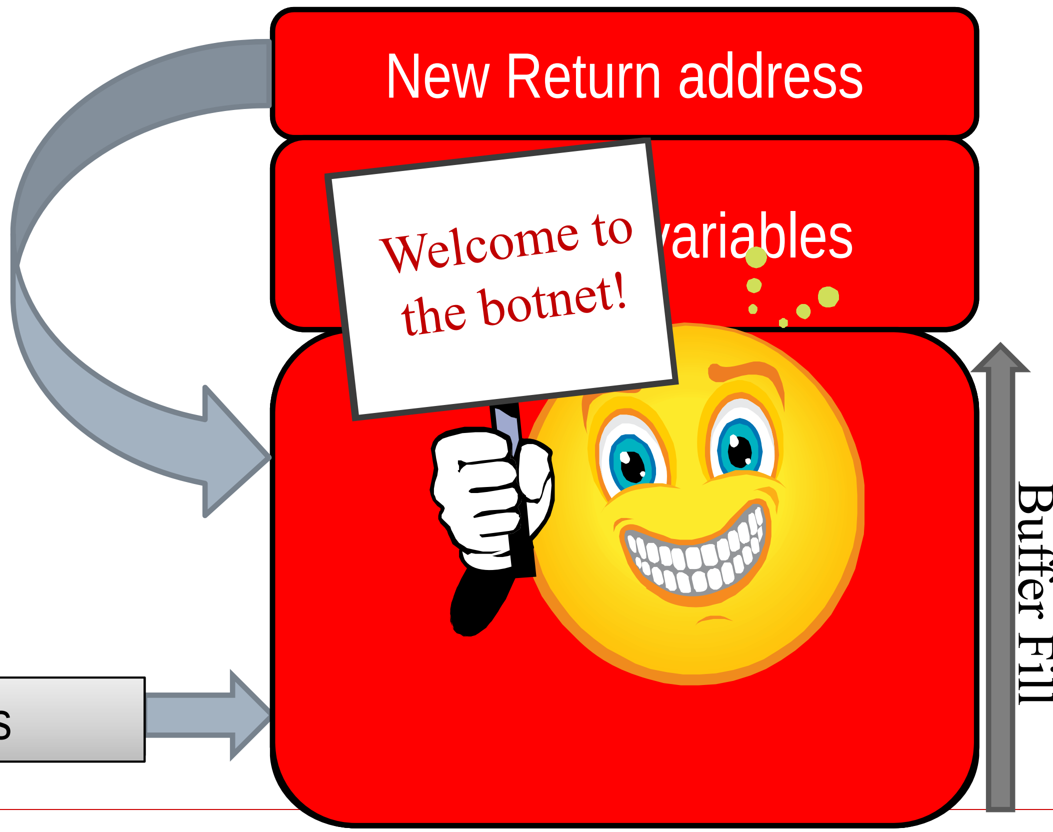


Buffer Overflow Attack

- Buffer overflows constitute a large class of security vulnerabilities
- Goal: inject code into an unsuspecting machine, and redirect control

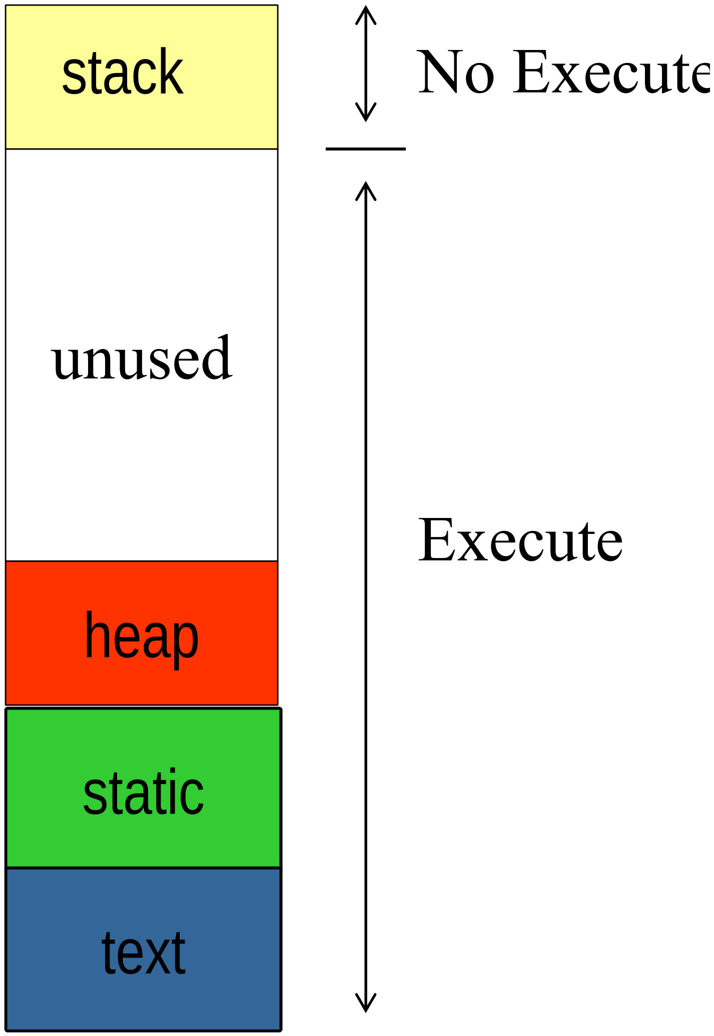
```
void foo()
{
  int local_variables;
  int buffer[256];
  …
  buffer = read_input();
  …
  return;
}
```

If read_input() reads >256 ints



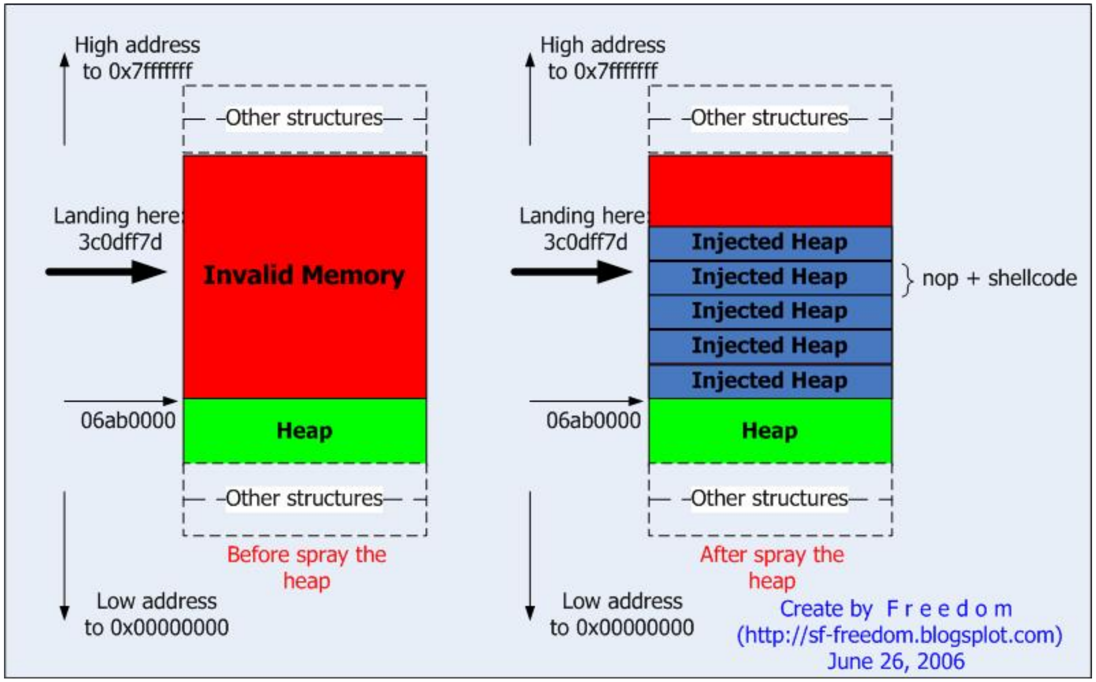
No-Execute (NX) Stacks

- Eliminate stack code injection by preventing code execution on stack
- Can be a problem for some safe programs, e.g., JITs
- NX bit in newer x86 PTEs indicates no-execute permission for pages



Escalate: No code allowed on stack

- Use a ***heap-spray attack***

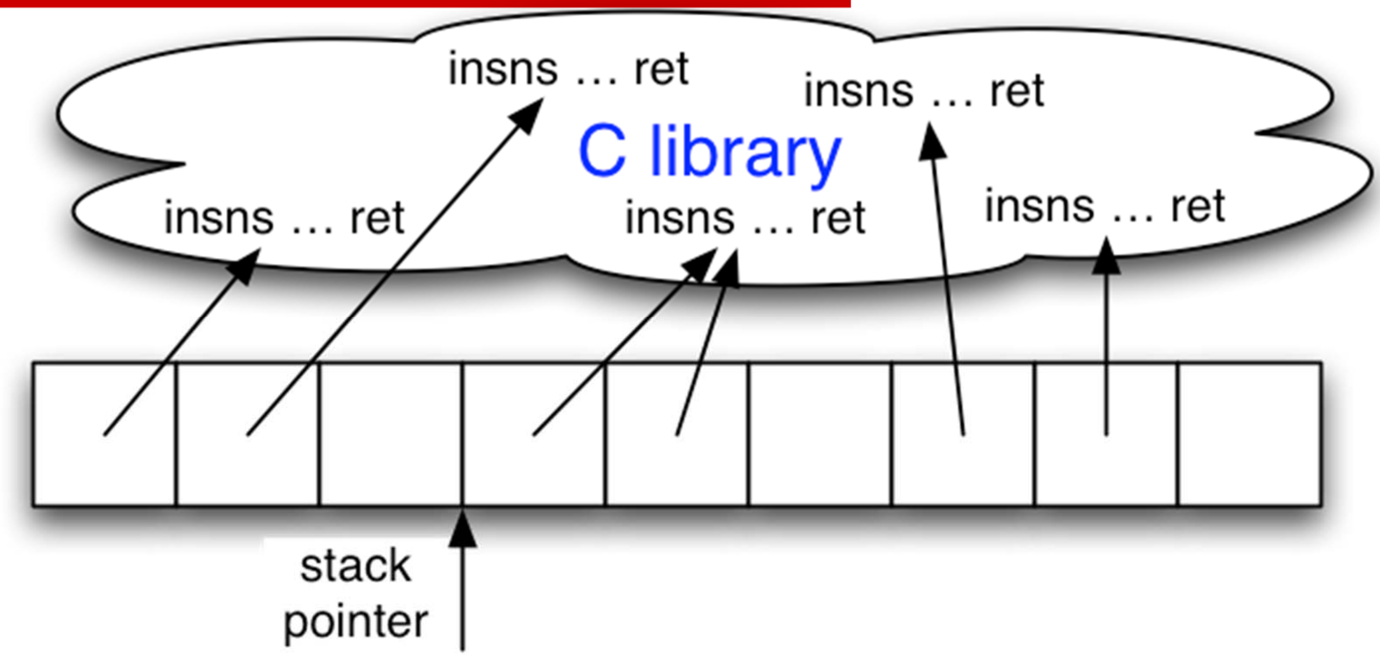


- Inject executable data into heap, then perform random stack smash
  - Example, generate many strings in Javascript that are also real code
- Generous heap sprays will likely be found by stack smash attack

Address Space Layout Randomization (ASLR)

- At load time, insert random-sized padding before all code, data, stack sections of the program
- Successfully implementing a buffer overflow code injection requires guessing the padding geometry ***on the first try***
- Implemented in recent Windows, Linux and MacOS kernels

Escalate: No new code allowed at all

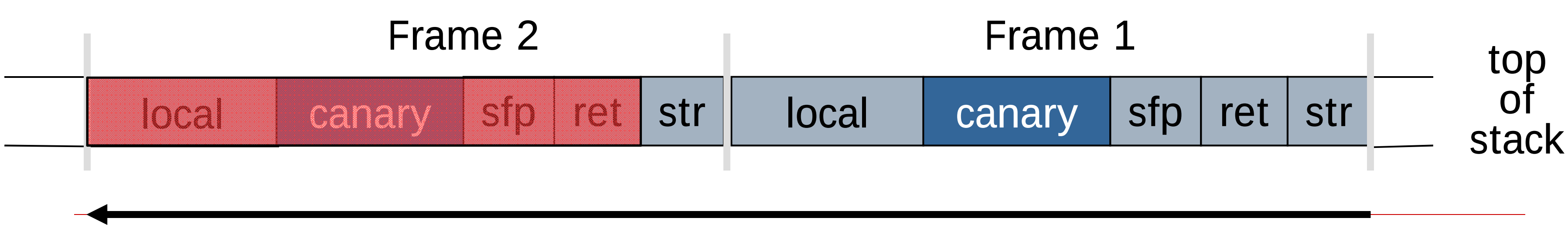


- Use ***return-oriented programming*** to attack…
  - "RET" instruction transfers control to address on top of stack.
  - Return-oriented programming introduces no new instructions, just carefully craft injected stack returns to link existed function tails
  - New program is formed from sequence of selected function tails composed from existing code

Stack Canaries with StackGuard

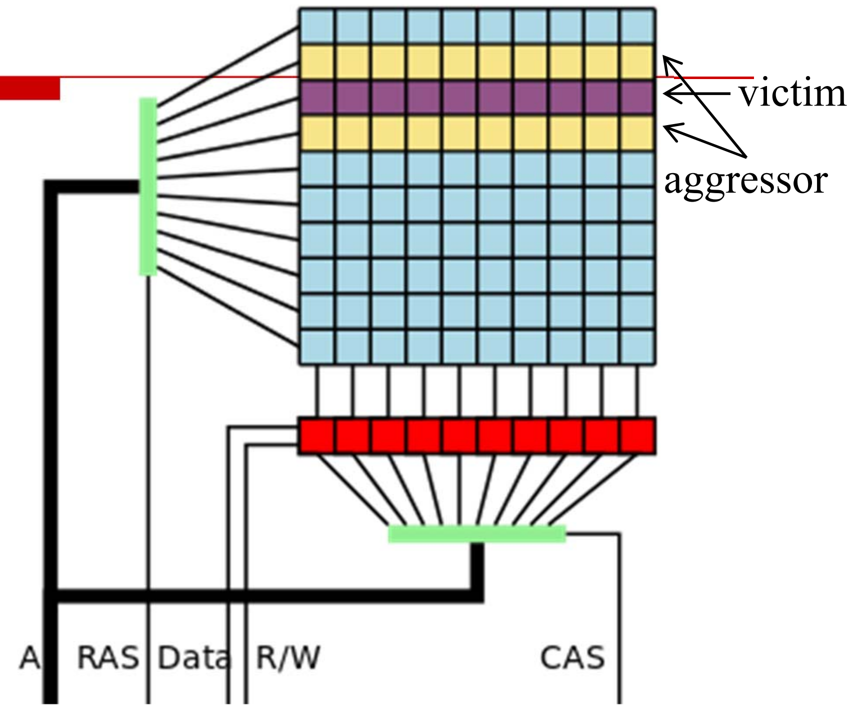
- Implemented in compiler (GCC), runtime check of stack integrity
- Embed “canaries” in stack frame before the return address, in function prologue, verify their integrity in function epilogue

- Canary is a per-instance random value that attacker must guess ***on the first try*** for a successful attack
- About 10% overhead for typical programs
- Can be thwarted with overflow attacks on function pointers

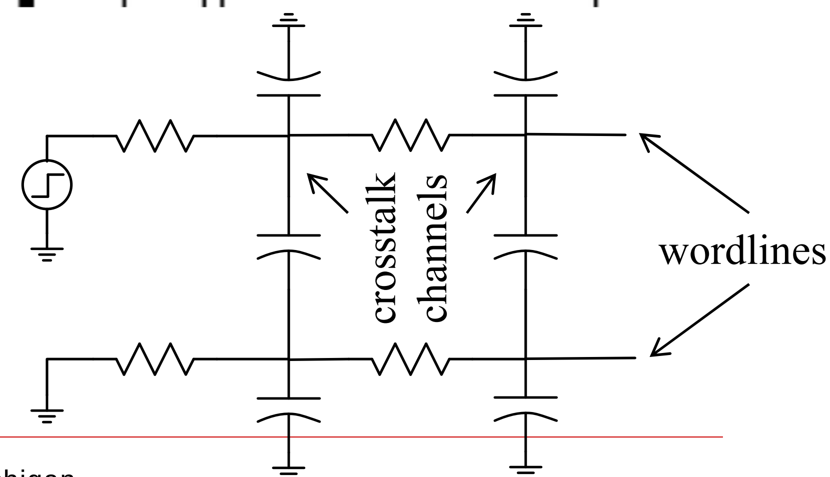


Row Hammer Attack

- ❑ Attack flips bits in victim DRAM row, without permission to access
  - Result of wordline crosstalk
  - Creates small pulses on adjacent wordlines, increases bitcell leakage
  - Hammer enough times (~400k) in one refresh cycle (~64ms) and bits will flip in victim row

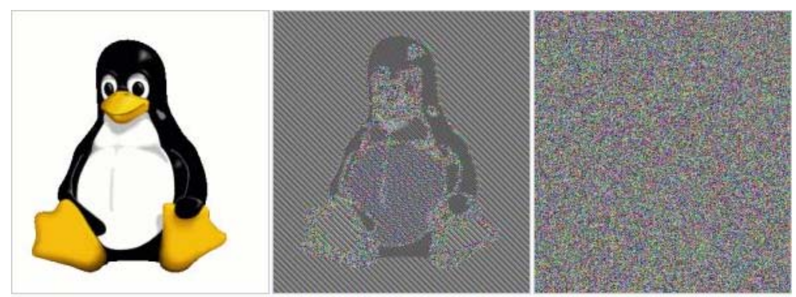


- ❑ Typical protection requires doubling the refresh rate
- ❑ Why doesn't this happen all the time?



Cold-Boot Attacks

- Cold-boot attacks steal encryption keys
  - Super-cool DRAM, rip it from running machine
  - Analyze it in a second machine without security
  - Circa 2007
- Many modern DDR3+ interfaces utilize memory scrambling
  - Data to DRAM is encrypted with per-boot key
  - Non-chained cipher, only 48 key expansions



- Recently, Michigan PhD students cold-boot attacked a DDR3 interface
  - Used known plaintext to identify key expansions
  - Located TrueCrypt AES keytable, regen'ed key

Things to remember

- Don't make yourself an easy security target!
  - Change passwords
  - Use different passwords for different sites
- Don't invent your own cryptographic algorithm, unless that is your only career goal.
- Think of security from the start as you design new systems, software or hardware
- This is just the beginning -- security is a growing and rewarding challenge

See you all at the final!