

**15-410**

***“...”Windows NT is C2 Secure”...”***

**Security Overview**  
**Apr. 23, 2008**

**Dave Eckhardt**

**Roger Dannenberg**

# Synchronization

## Reminder...

- Don't forget to read *your partner's* P3 code
  - Suggestion: read it, then meet with questions

## P3 interview/feedback sessions

- Roughly 45 minutes
- Your reader will contact you to set up an appointment next week

# Synchronization

## Today

- Chapter 15, more or less

## Next time

- Fun stuff not in the text

# Overview

## Goals & Threats

## Technologies

- Scanning
- Hashes
- Random numbers
- Private-key/symmetric cryptography
- Public-key/asymmetric cryptography
- The mysterious nonce

## Next Time

- Applications
- Systems

# U.S. DoD “Orange Book” Security Classifications

**D –try again**

**C –authentication, controlled sharing**

**B –per-object sensitivity labels, user clearances**

**A –B-class system with formal spec, proofs**

## **Sub-levels**

- **C2 = C1 + ACLs, audit logs, anti-tamper OS, ...**

# **“Windows NT is C2 secure”**

**Windows NT is C2 secure**

**Wimpy old Unix is only C1**

**Use Windows, it's secure!**

# Windows NT is C2 secure

**Windows NT is C2 secure**

**Wimpy old Unix is only C1**

**Use Windows, it's secure!**

- *Melissa, Code Red, SQL Slammer, SoBig, ...*
- What's wrong with this picture?

**“Security Architecture” undermined by  
implementation (default login is superuser)**

**Physical security assumed in evaluation**

- Locked rooms, floppy booting disabled
- In practice, isolate from Internet!

# Goals & Threats

## **Goal: Authentication**

- Threat: impersonation

## **Goal: Secrecy**

- Threats: theft, eavesdropping, cipher breaking, ...

## **Goal: Integrity**

- Threat: cracking

## **Goal: Signature**

- Threats: impersonation, repudiation

...

# Goals & Threats

## Authentication

- Visitor/caller is Alice

## Threat: Impersonation

- Act/appear/behave like Alice
- Steal Alice's keys (or “keys”)

## Outcomes

- Maybe you can read Alice's secrets
- Maybe you can send Alice to jail

# Goals & Threats

## Secrecy

- Only Bob (or “Bob”) can read Bob's data

## Difficult secrecy threats

- Break a cipher (see below)
- Compromise a system (see below)
- Or...

## Eavesdropping –get data while it's unprotected!

- Wireless keyboard
- Keystroke logger
- TEMPEST

# TEMPEST

## Code name for electromagnetic security standard

- The *criteria document* is classified

## Problem

- Computers are *radios*
- Especially analog CRT monitors
  - ~150 MHz signal bandwidth (“dot clock”)
  - Nice sharp sync pulses
- Surveillance van can *read your screen* from 100 feet

# Goals & Threats

## Integrity

- Only *authorized personnel* can add bugs to a system
- Or edit bank account balances
- Or edit high school grades

## Threats

- Hijacking authorized accounts (impersonation)
- Bypassing authorization checks
  - Boot system in “administrator mode”?
  - Boot some other OS on the machine?
- Modifying hardware

# Goals & Threats

## Signature

- “Pay Bob \$5 for his program” was uttered by Alice

## Threats

- Alice repudiates message (after receiving program)
- Charlie signs “Pay Charlie \$500 for his program”
  - ... *with Bob's signature*

# Goals & Threats

## Anonymous communication

- “Whistle blowers”
- Secret agents

## Threat

- “Traffic analysis”
  - Observe repeated “coincidence”
    - » Node 11 sends a message, Nodes 1-10 attack
  - Which node is a good target?

# Goals & Threats

## Availability

- Web server is available to corporate customers
- Mailbox contains interesting mail

## Threat

- DoS –Denial of Service
  - Flood server with bogus data
  - “Buries” important data
  - SYN flooding, connection resetting

# Another DoS Attack

## Automated Flight Data Processing System

- Transfers flight arrival/departure data
  - ...between radar tower in Elgin, IL (where's that?)
  - ...and tower at *O'Hare International*

## Fallback system

- paper, pencil, telephone

# Another DoS Attack

## Automated Flight Data Processing System

- Transfers flight arrival/departure data
  - ...between radar tower in Elgin, IL (where's that?)
  - ...and tower at *O'Hare International*

## Fallback system

- paper, pencil, telephone

## Uh-oh...

- Chief engineer quit
  - after deleting *sole copy* of source code

# Now What?

**Police raided his house**

**Recovered code!**

- Encrypted
  - Cracked –after 6 months

**Summary**

- <http://archives.californiaaviation.org/airport/msg02974.html>

**Lesson?**

- People matter...

# Malicious Programs (“malware”)

**Buffer overflow**

**Virus/worm**

**Trojan horse**

**Trapdoor**

# Buffer overflow

# GET

```
/default.ida?XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
XXXXXX%u9090%u6858%ucbd3%u7801%u9090%u685  
8%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9  
090%u9090%u8190%u00c3%u0003%u8b00%u531b%  
u53ff%u0078%u0000%u00=a HTTP/1.0
```

# Virus/Worm

## Virus

- Program which cannot replicate itself
- Embedded in other programs, runs when they do
- Embeds self in other programs

## Worm

- Breaks into remote machine
- Launches remote copy
- May not reside permanently on disk

# Trojan, Trap Door

## Trojan Horse

- Program with two purposes
- Advertised – “Here is the new security update!”
- Actual – Here is a hard-disk-wipe program!

## Trap door

- login: anything
- Password: My hovercraft is full of eels!

**#insert <reflections\_on\_trusting\_trust>**

# Technologies

**Scanning/intrusion detection/auditing**

**Hashing**

**Encryption (1-time, private, public)**

**The mysterious nonce**

# Scanning

## Concept

- Check your system for vulnerabilities
  - Before somebody else does!

## Details

- Password scan
- Scan for privileged programs, extra programs
- Check for dangerous file permissions
- Check that program, config files have correct contents
- Are mysterious programs running?

# Intrusion Detection

## Concept

- Monitor system in secure state
- Summarize typical behavior
- Watch for disturbing variation

## Examples

- Sudden off-site traffic to/from a machine
- Change in system call mix
  - Gee, my web server doesn't *usually* exec("/bin/sh -i")...

## Issues –false positive, false negative

# Auditing

## Concept

- Estimate damage
  - What was taken?
- How to fix system?

## Approach

- Log system actions off-board
  - paper printer
  - disk with hardware roll-back

**Boring but useful *when* you're in trouble...**

# Hashing

## “One-way function”

- $h_1 = f(\text{message}_1)$
- Given  $h_1$  “infeasible” to find  $\text{message}_1$ 
  - Not so hard – “parity sum” is a one-way function

## “Collision resistant”

- Given  $h_1$ , “infeasible” to find  $\text{message}_2$  also hashing to  $h_1$
- “Infeasible” to find any two  $m_1, m_2$  hashing to  $h_x$

## Use

- Here is the MD5 hash of the OpenBSD CD-ROM image
  - And here is the list of mirror sites
- “Infeasible” to find/construct malware with that hash

# Hashing Issues

## Verify data?

- Compute hash function on data you have
- Compare to published official output of hash function run on the official data

## Say, what *is* the “official version hash”?

- Easy if you're in a room with the OpenBSD release coordinator
- Otherwise, not easy
- Preview of the *key distribution* problem

# Fate of Secure Hashes

## Secure hash functions don't last very long

- Some are “found weak” several years after proposal
- NIST SHA (now known as SHA-0) withdrawn almost immediately

## Status (Spring 2004)

- MD5 should be removed from service
- Code under development should use SHA-1

# Fate of Secure Hashes

## Status (Spring 2004)

- MD5 should be removed from service
- New projects should use SHA-1

## Status (Crypto2004, August)

- MD5 is “blown”
  - Team of Chinese researchers has a method to find collisions
    - » MD4, RIPEMD, HAVAL, MD5...uh-oh...
- SHA-1 is “on life support”
  - Collisions have been found in SHA-0
  - Collisions have been found in “reduced round” SHA-1
  - Collisions can be found in  $2^{69}$  attempts ( $\ll 2^{80}$ )
- Verdict: “schedule SHA-1 for replacement” -- with ...?

# Encryption

## Concept

**cipher** =  $E(\text{text}, K_1)$

**text** =  $D(\text{cipher}, K_2)$

## Algorithm E(),D()

- Should be *public*
  - Best known way to achieve strength

## Keys

- One (or maybe both) kept secret

# “Random” Numbers

## Three concepts

- Pseudo-random number generator (PRNG)
  - $\text{Next} = (\text{Previous} * L + I) \bmod M$
  - `srand()/random()`
  - Next “looks different” than Previous
  - Behaves *the same way every time* - not random *at all*
- Kind-of-random stuff
  - `srand(get_timer());`
  - Ok for games (where money isn't involved)
- Entropy pool
  - Genuinely random bits

# Entropy Pool

## Goal (for security) is unguessability

- aka unpredictability, true randomness, entropy

## Why “kind-of” doesn't work

- Netscape seeded SSL session key generator with
  - getpid(), getppid(), time of day
  - Time is a globally-known value
  - Process IDs occupy a small space
    - » ...especially if you are on the target's machine!

## Some things are genuinely random

- Which microsecond does the user press a key in?
- “Entropy Pool” is a queue of those events

# Encryption: One-Time Pad

## Key

- *Truly random* byte string
  - R K N Y Q T I D C E M W X ...

## Algorithm

- E(): XOR one key byte, one message byte
  - $M \oplus R = 1F$
  - $M E S S A G E \oplus R K N Y Q T I = 1F\ 0E\ 1D\ 0A\ 10\ 13\ 0C\ 0A$
- D(): same process –using the *same random string*
  - Recall
    - »  $random \oplus random = 0$
    - »  $msg \oplus 0 = msg$
  - So  $(msg \oplus random) \oplus random = msg$

# One-Time Pad

**Pad must be as long as message**

**Must be delivered securely**

***Never* re-use pads!!**

- $(m1 \oplus \text{pad}) \oplus (m2 \oplus \text{pad}) = (m1 \oplus m2)$
- Computationally *very* easy to see if a bit stream is text  $\oplus$ 'd with other text

# Private-Key Cryptography

**Concept:** *symmetric cipher*

*cipher* = E(*text*, Key)

*text* = E(*cipher*, Key)

**Good**

- Fast, intuitive (password-like), small keys

**Bad**

- Must share a key (*privately!*) before talking

**Applications**

- Bank ATM links, secure telephones

# Public-Key Cryptography

**Concept:** *asymmetric* cipher (aka “magic”)

**cipher** = E(**text**, Key1)

**text** = D(**cipher**, Key2)

**Keys are** *different*

- Generate *key pair*
  - Two very large bit strings
    - » Related to each other mathematically
    - » Work together
- Publish “public key”
- Keep “private key” *very* secret

# Public-Key Encryption

## **Sending secret mail**

- Locate receiver's public key
- Encrypt mail with it
- Nobody can read it
  - *Not even you!*

## **Receiving secret mail**

- Decrypt mail with your private key
  - No matter who sent it

# Public-Key Signatures

**Write a document**

**Encrypt it with your private key**

- Nobody else can do that

**Transmit plaintext *and ciphertext* of document**

**Anybody can decrypt with your public key**

- If they match, the sender knew your private key
  - ...sender was you, more or less

**(really: sign msg with  $E(\text{hash}(\text{msg}), K_p)$ )**

# Public Key Cryptography

## Good

- No need to privately exchange keys

## Bad

- Algorithms are slower than private-key
- Must trust key directory

## Applications

- Secret mail, signatures

# Comparison

## Private-key algorithms

- Fast crypto, small keys
- *Secret-key-distribution problem*

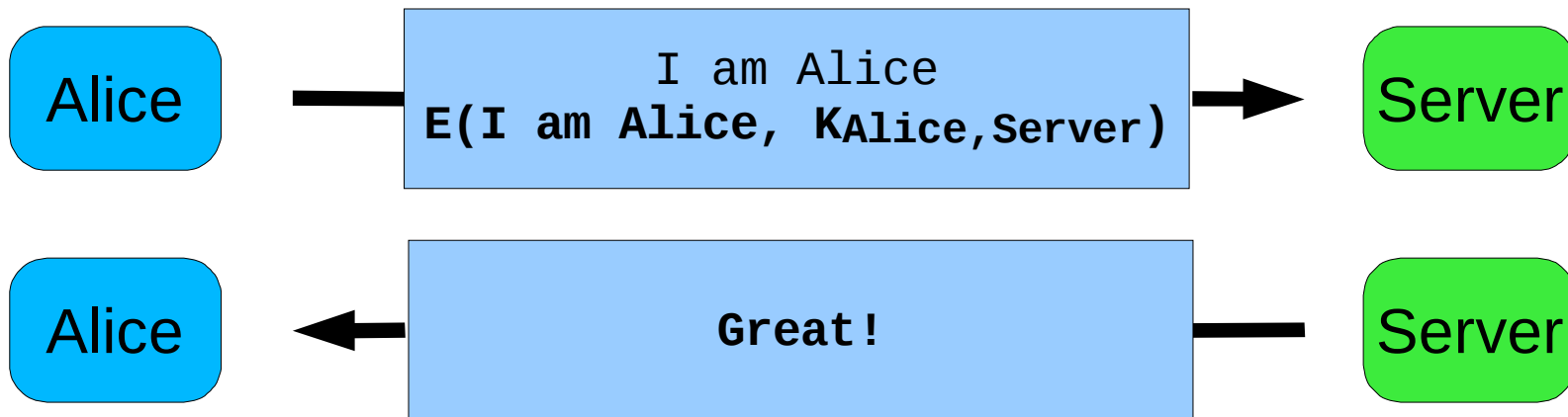
## Public-key algorithms

- “Telephone directory” key distribution
- Slow crypto, *keys too large to memorize*

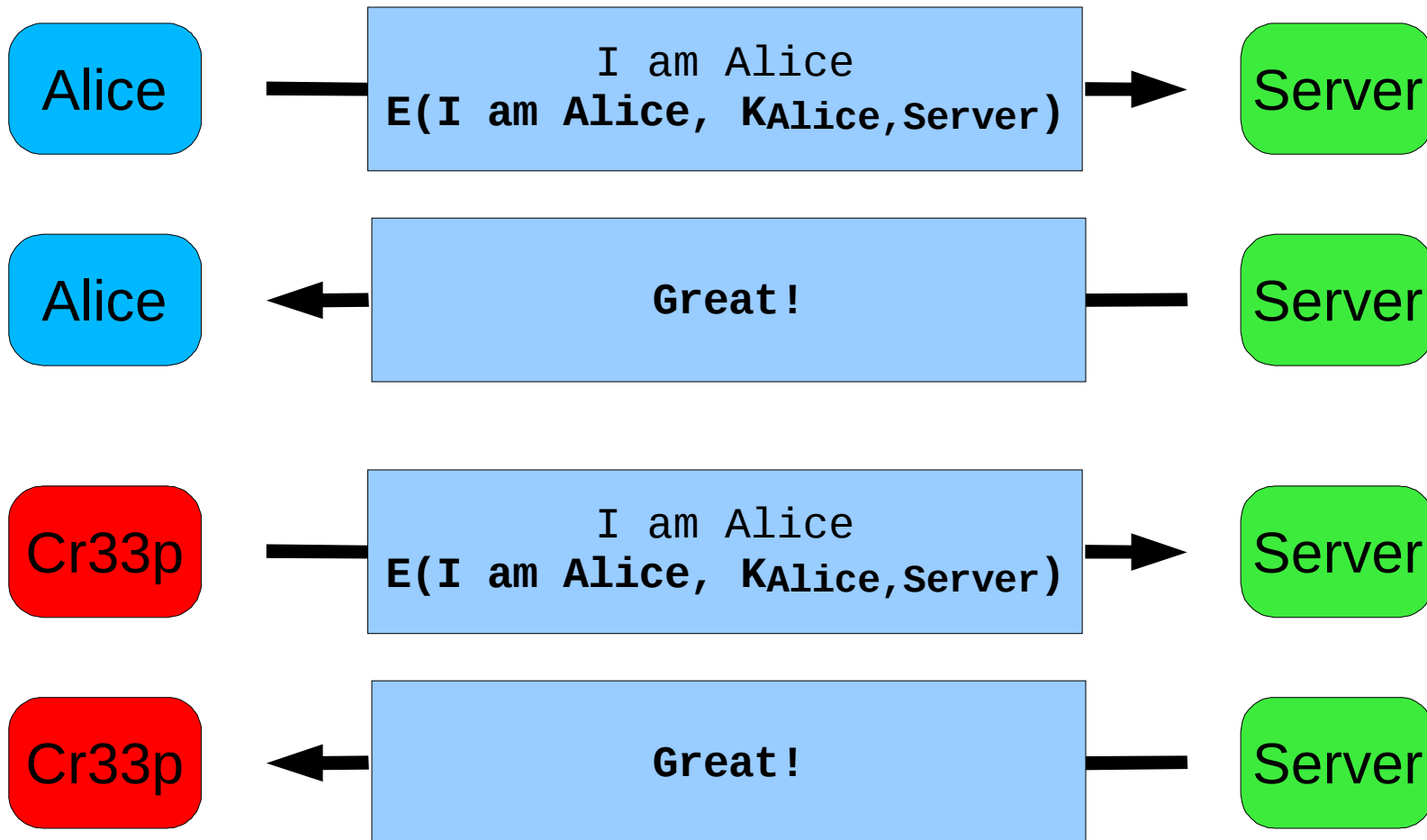
## Can we get the best of both?

- Next time!

# Secure Network Login



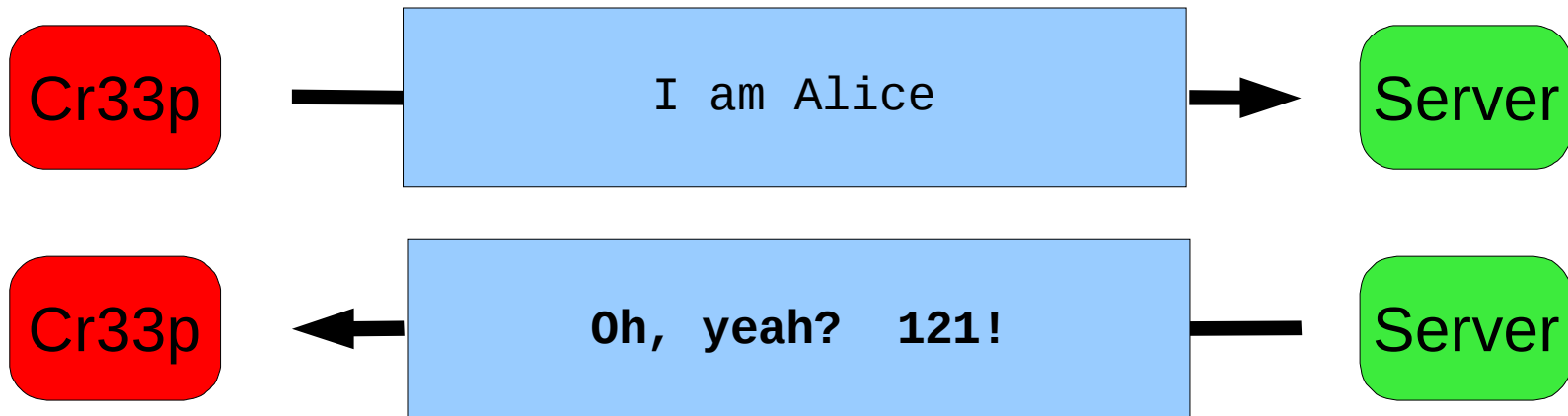
# Secure Network Login –Uh-oh...



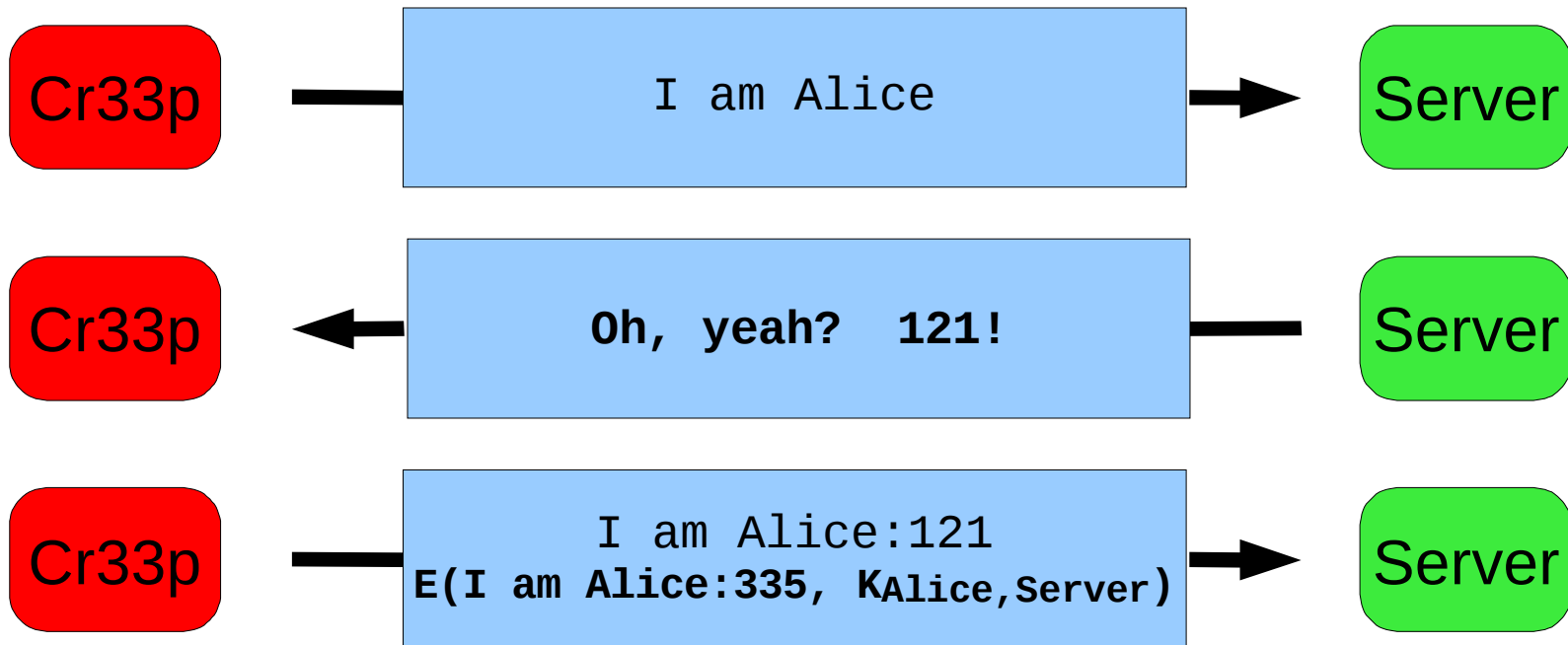
# Secure Network Login –Nonce



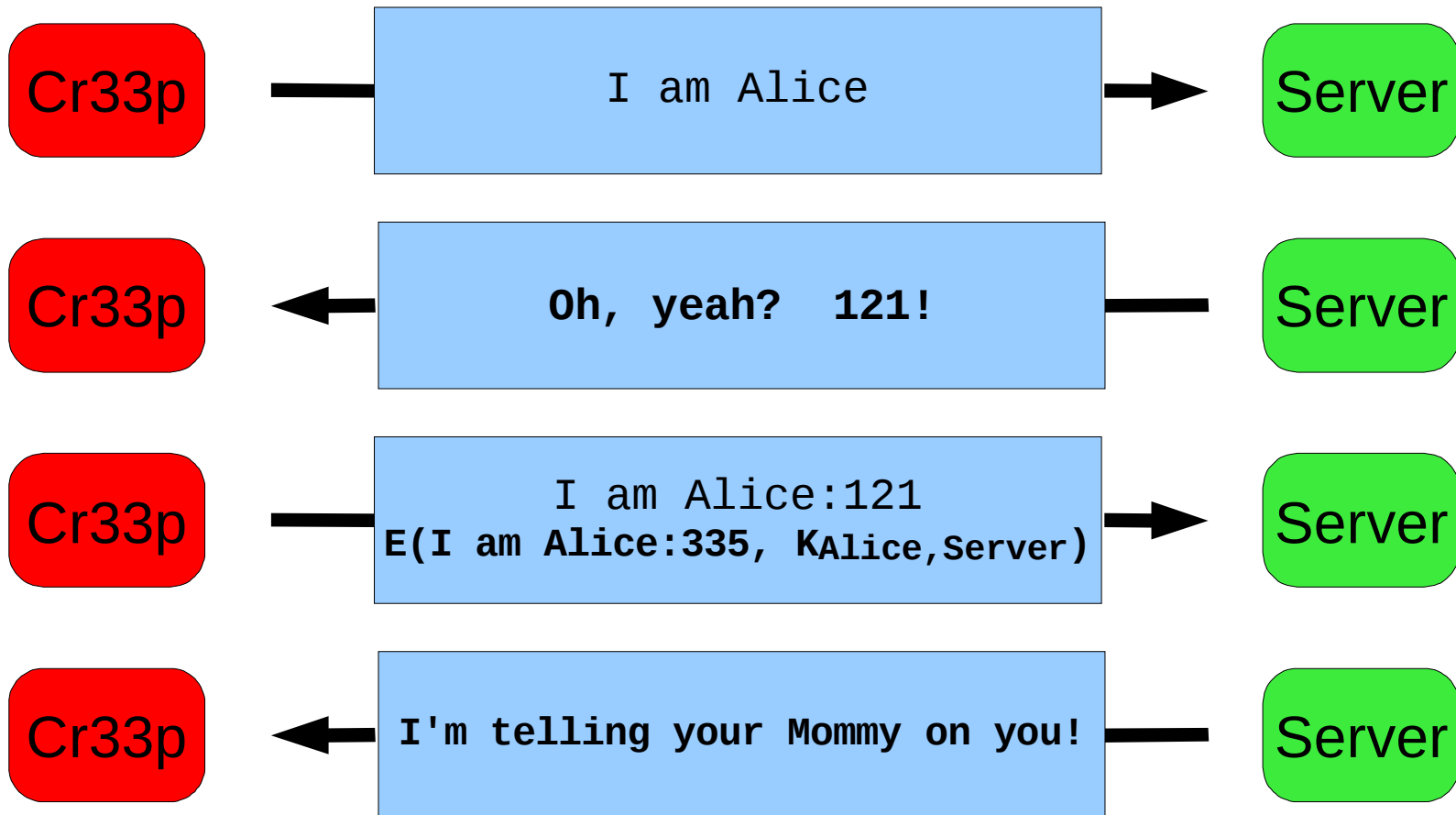
# Secure Network Login –Nonce



# Secure Network Login –Nonce



# Secure Network Login –Nonce



# Summary

**Many threats**

**Many techniques**

**“The devil is in the details”**

**Just because it “works” doesn't mean it's right!**

**Open algorithms, open source**

# Further Reading

## Status of secure hash functions

**MD5 is really dead (fast exploit code available)**

[http://www.schneier.com/blog/archives/2005/06/more\\_md5\\_collis.html](http://www.schneier.com/blog/archives/2005/06/more_md5_collis.html)

[http://www.schneier.com/blog/archives/2005/03/more\\_hash\\_funct.html](http://www.schneier.com/blog/archives/2005/03/more_hash_funct.html)

[http://cryptography.hyperlink.cz/md5/MD5\\_collisions.pdf](http://cryptography.hyperlink.cz/md5/MD5_collisions.pdf)

**SHA-1 has been seriously wounded**

[http://www.schneier.com/blog/archives/2005/02/cryptanalysis\\_o.html](http://www.schneier.com/blog/archives/2005/02/cryptanalysis_o.html)

[http://www.schneier.com/blog/archives/2005/02/sha1\\_broken.html](http://www.schneier.com/blog/archives/2005/02/sha1_broken.html)

[http://www.schneier.com/blog/archives/2005/08/new\\_cryptanalyt.html](http://www.schneier.com/blog/archives/2005/08/new_cryptanalyt.html)

Xiaoyun Wang's page

» <http://www.infosec.sdu.edu.cn/people/wangxiaoyun.htm>

# Further Reading

## **Soft Tempest: Hidden Data Transmission Using Electromagnetic Emanations**

- Markus Kuhn, Ross Anderson
- <http://www.cl.cam.ac.uk/~mgk25/ih98-tempest.pdf>

## **Optical Time-Domain Eavesdropping Risks of CRT Displays**

- Markus Kuhn
- <http://www.cl.cam.ac.uk/~mgk25/emsec/optical-faq.html>

## **Keyboard Acoustic Emanations Revisited**

- Zhuang, Zhou, Tygar
- [http://www.cs.berkeley.edu/~tygar/papers/Keyboard\\_Acoustic\\_Emanations\\_Revisited/ccs.pdf](http://www.cs.berkeley.edu/~tygar/papers/Keyboard_Acoustic_Emanations_Revisited/ccs.pdf)

# Further Reading

## Reflections on Trusting Trust

- Ken Thompson
- <http://www.acm.org/classics/sep96>

## Netscape random-number oops

- <http://www.cs.berkeley.edu/~daw/netscape-randomness.html>

## Lava-lamp random numbers

- <http://www.LavaRnd.org/>

## How to destroy somebody who uses a hash table

- [http://www.cs.rice.edu/~scrosby/hash/CrosbyWallach\\_UsenixSec2003/](http://www.cs.rice.edu/~scrosby/hash/CrosbyWallach_UsenixSec2003/)