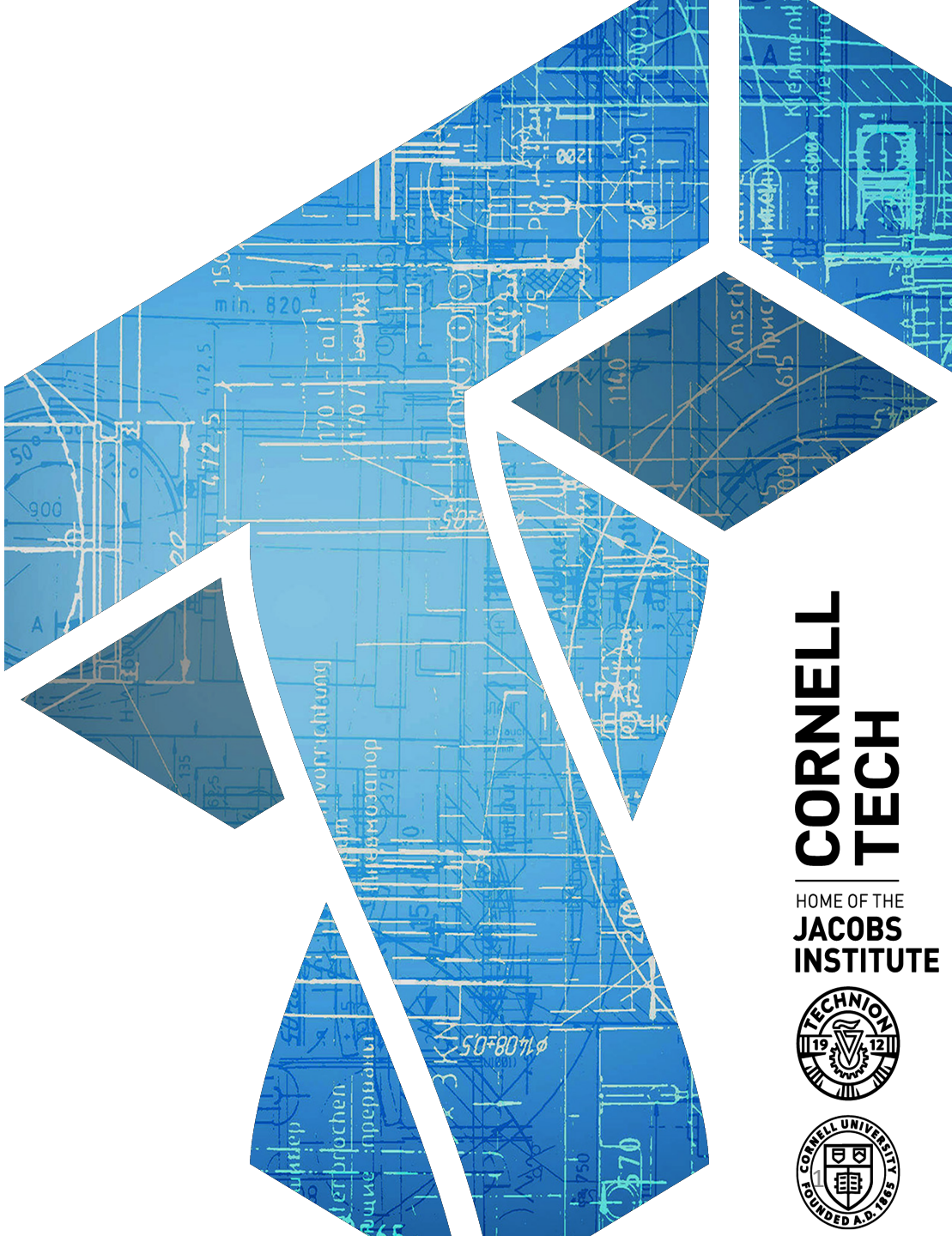
# CS 6431: Internet Measurement

Instructor: Tom Ristenpart

https://github.com/tomrist/cs6431-fall2021



CORNELL TECH
HOME OF THE JACOBS INSTITUTE

# Basic research questions on attacks

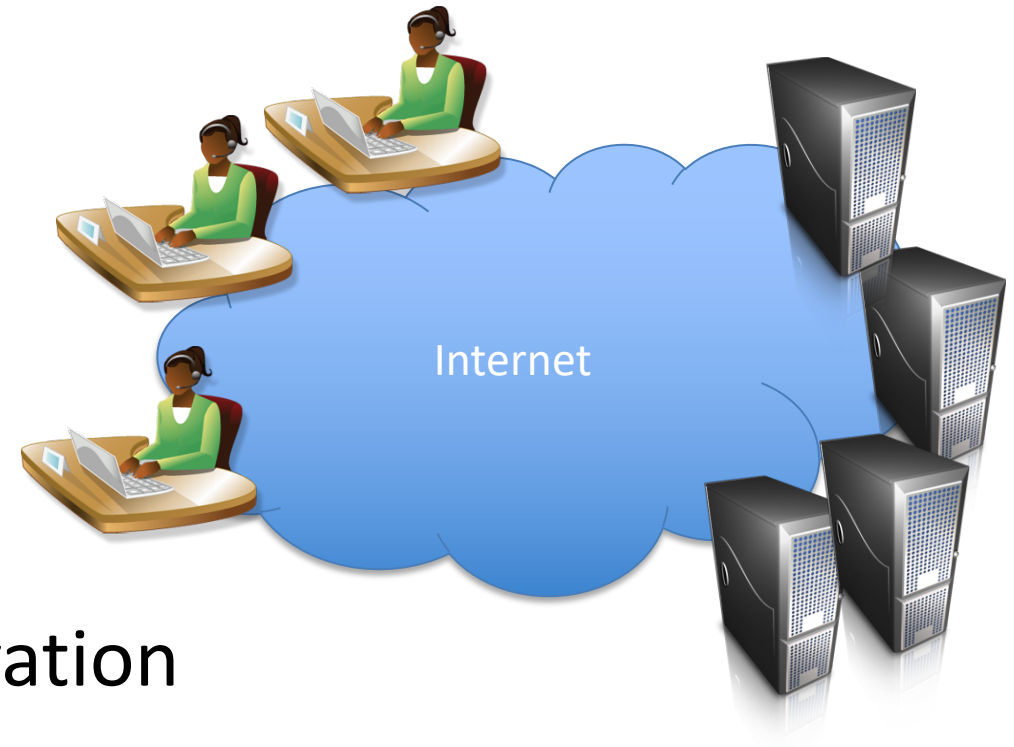
- Padding oracle and other attacks demonstrated experimentally
- What are interesting empirical questions related to vulnerabilities?
  - How many servers on the internet are currently vulnerable?
  - How many attacks are occurring?

# Padding oracle measurements

- Experimentally confirm TLS 1.2 with certain ciphersuites vulnerable to attack
  - Any ciphersuite matching TLS_*_WITH_AES_*_CBC_SHA*
  - TLS_RSA_WITH_AES_128_CBC_SHA256
- How can we infer which servers support this ciphersuite?
- How can we infer fraction of sessions using this ciphersuite?

# Measurement observation points

1. Client-side
2. Server-side
3. On-path




What questions would each observation point potentially help answer?

Plusses/minuses of different approaches?

# TLS 1.2 handshake for RSA transport

Client

Server

Pick random Nc

ClientHello, MaxVer, Nc, Ciphers/CompMethods →

Server has RSA key pair (pk,sk)

Pick random Ns

← ServerHello, Ver, Ns, SessionID, Cipher/CompMethod

Check CERT using CA public verification key

← CERT = (pk, signature over it by CA)

Pick random PMS
C <- Enc(pk,PMS)

C →

PMS <- Dec(sk,C)

ChangeCipherSpec,
{ Finished, PRF(MS, “Client finished” || H(transcript)) } →

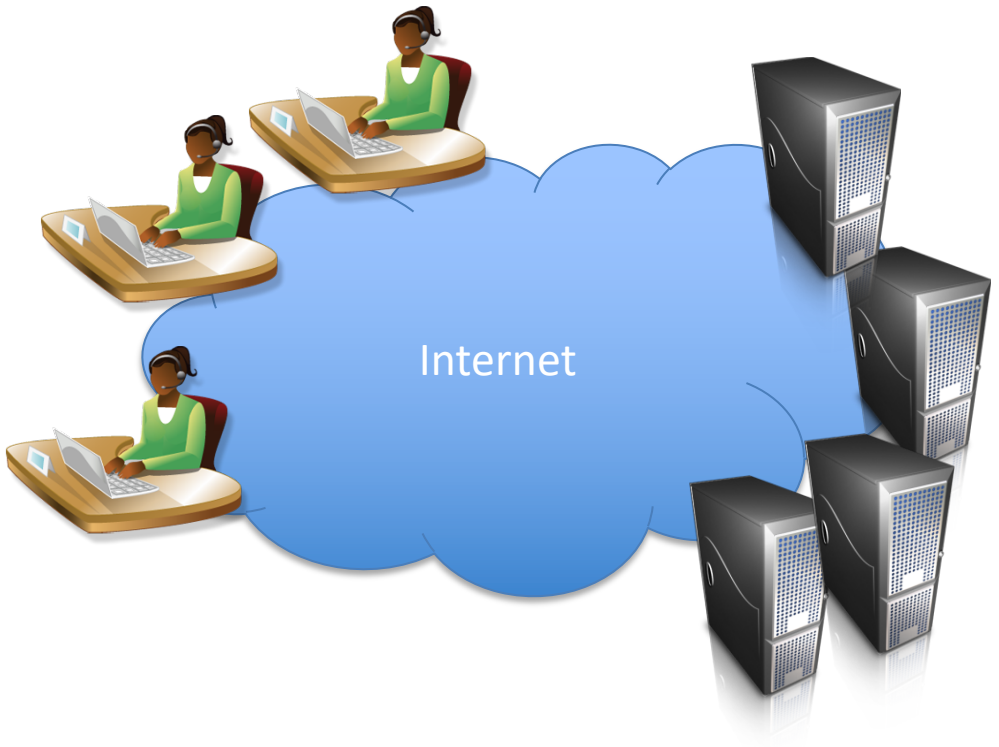
ChangeCipherSpec,
{ Finished, PRF(MS, “Server finished” || H(transcript’)) } ←

Bracket notation means contents encrypted

MS <- PRF(PMS, “master secret” || Nc || Ns )

# Measurement case study: P's and Q's paper

- Want to understand strength of RSA keys used on Internet
- Hypothesis:
  - Some hosts have weak keys due to bad random number generators
  - Some kinds of weak pairs of keys are easily factorable



# RSA math

Let N be a positive number

Looking ahead: N = pq for large primes p,q

N will be called the modulus

$\mathbf{Z}_N = \{0,1,2,3,\ldots, N-1\}$

$\mathbf{Z}_N^* = \{ i \mid \gcd(i,N) = 1 \text{ and } i < N\}$

gcd(X,Y) = 1 if greatest common divisor of X,Y is 1

# RSA math

$\mathbf{Z}_N^* = \{ i \mid \gcd(i,N) = 1 \}$

N = 13 $\quad \mathbf{Z}_{13}^* = \{ 1,2,3,4,5,6,7,8,9,10,11,12 \}$

N = 15 $\quad \mathbf{Z}_{15}^* = \{ 1,2,4,7,8,11,13,14 \}$

The size of a set S is denoted by $|S|$

Def. $\phi(N) = |\mathbf{Z}_N^*|$ (This is Euler's totient function)

$\phi(13) = 12$

$\phi(15) = 8$

$\mathbf{Z}_{\phi(15)}^* = \mathbf{Z}_8^* = \{ 1,3,5,7 \}$

# RSA math

$\mathbf{Z}_N^* = \{ i \mid \gcd(i,N) = 1 \}$

Fact. For any a,N with N > 0, there exists unique q,r such that

$$a = Nq + r \qquad \text{and} \qquad 0 \le r < N$$

$$17 \bmod 15 = 2 \qquad 105 \bmod 15 = 0$$

Def. $a \bmod N = r \in \mathbf{Z}_N$

Def. $a \equiv b \pmod N$ iff $(a \bmod N) = (b \bmod N)$

Operations work in natural way:

$$a \bullet b \bmod N \qquad a+b \bmod N$$

# RSA math

$\mathbf{Z}_N^* = \{ i \mid \gcd(i,N) = 1 \}$

$(\mathbf{Z}_N^* , \bullet)$ is a **group** where $\bullet$ denotes multiplication mod N

Group $(G,\bullet)$ is a set G and operator $\bullet$ that satisfy:

1. *Closure*: for all $a,b \in G$ it holds that $a \bullet b \in G$
2. *Associativity*: for all $a,b,c \in G$ it holds that $a \bullet (b \bullet c) = (a \bullet b) \bullet c$
3. *Identity*: Exists $I \in G$ s.t. for all $a \in G$ $a \bullet I = a$
4. *Inverses*: for $a \in G$ there exists $a^{-1} \in G$ s.t. $a \bullet a^{-1} = I$

Abelian group is additionally commutative:

for all $a,b \in G$ it holds that $a \bullet b = b \bullet a$

# RSA math

$\mathbf{Z}_N^* = \{ i \mid \gcd(i,N) = 1 \}$

$(\mathbf{Z}_N^* , \bullet)$ is a **group**

$$\mathbf{Z}_{15}^* = \{ 1,2,4,7,8,11,13,14 \}$$

$2 \bullet 7 \equiv 14 \pmod{15}$

$4 \bullet 8 \equiv 2 \pmod{15}$

Closure: for any $a,b \in \mathbf{Z}_N^*$ $\quad a \bullet b \bmod N \in \mathbf{Z}_N^*$

Def. $a^i \bmod N = \underbrace{a \bullet a \bullet a \bullet \ldots \bullet a}_{i \text{ times}} \bmod N$

# Some needed algorithms

| **Algorithm** | **Running time (n = log N)** |
|---|---|
| Modular multiplication<br>ab mod N | $O(n^2)$ |
| Modular exponentation<br>$a^i$ mod N | $O(n^3)$ |
| Modular inverse<br>$a^{-1}$ mod N | $O(n^2)$ |

# RSA math

$$\mathbf{Z}_N^* = \{ i \mid \gcd(i,N) = 1 \} \qquad \phi(N) = |\mathbf{Z}_N^*|$$

Claim: Suppose $e,d \in \mathbf{Z}_{\phi(N)}^*$ satisfying $ed \bmod \phi(N) = 1$
then for any $x \in \mathbf{Z}_N^*$ we have that

$$(x^e)^d \bmod N = x$$

$$\begin{aligned} x^{ed} \bmod N &= x^{1 + k\,\phi(N)} \bmod N \\ &= x^1 x^{k\,\phi(N)} \bmod N \\ &= x \bmod N \end{aligned}$$

By Euler's Theorem:
$x^{\phi(N)} = 1 \pmod N$

# RSA math

$\mathbf{Z}_N^* = \{ i \mid \gcd(i,N) = 1 \}$ $\quad\quad$ $\phi(N) = |\mathbf{Z}_N|$

Claim: Suppose $e,d \in \mathbf{Z}_{\phi(N)}^*$ satisfying $ed \bmod \phi(N) = 1$
then for any $x \in \mathbf{Z}_N^*$ we have that

$$(x^e)^d \bmod N = x$$

$\mathbf{Z}_{15}^* = \{ 1,2,4,7,8,11,13,14 \}$ $\quad\quad$ $\mathbf{Z}_{\phi(15)}^* = \{ 1,3,5,7 \}$

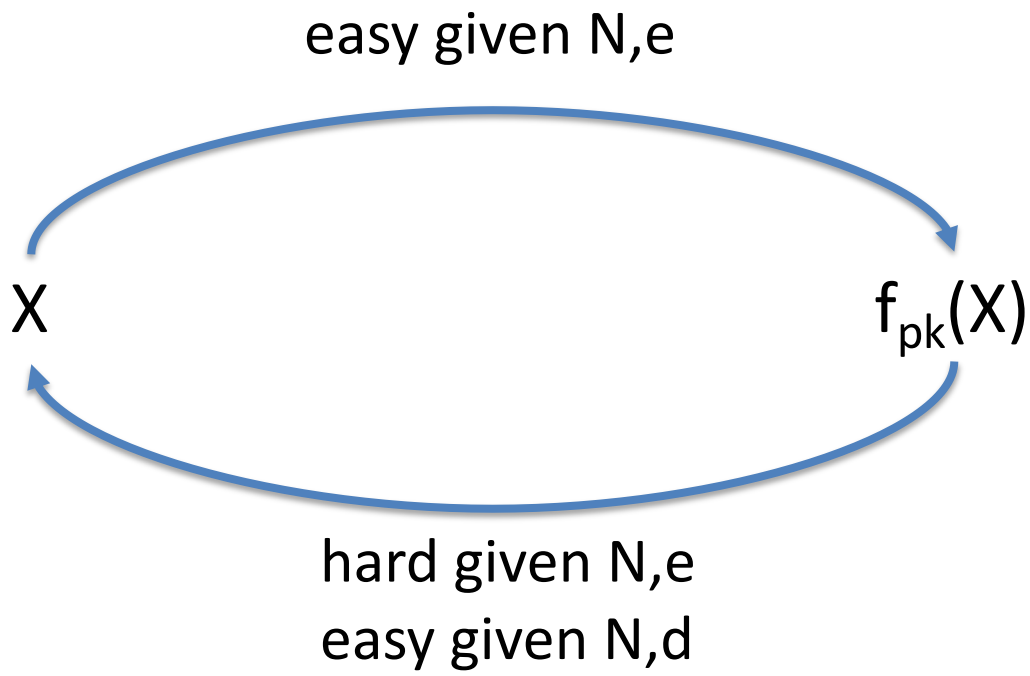
$e = 3$ , $d = 3$ gives $ed \bmod 8 = 1$

| x | 1 | 2 | 4 | 7 | 8 | 11 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|
| $x^3 \bmod 15$ | 1 | 8 | 4 | 13 | 2 | 11 | 7 | 14 |
| $y^3 \bmod 15$ | 1 | 2 | 4 | 7 | 8 | 11 | 13 | 14 |

# The RSA trapdoor permutation

pk = (N,e)  sk = (N,d)  with ed mod $\phi(N) = 1$

$f_{N,e}(x) = x^e \bmod N$  $g_{N,d}(y) = y^d \bmod N$



# The RSA trapdoor permutation

pk = (N,e)  sk = (N,d)  with ed mod $\phi(N) = 1$

$f_{N,e}(x) = x^e \bmod N$  $g_{N,d}(y) = y^d \bmod N$

But how do we find suitable N,e,d ?

If p,q distinct primes and N = pq then $\phi(N) = (p-1)(q-1)$

Why?

$\phi(N) = |\{1,\ldots,N-1\}| - |\{ip : 1 \le i \le q-1\}| - |\{iq : 1 \le i \le p-1\}|$

$= N-1 - (q-1) - (p-1)$

$= pq - p - q + 1$

$= (p-1)(q-1)$

# The RSA trapdoor permutation

pk = (N,e)  sk = (N,d)  with $ed \bmod \phi(N) = 1$

$f_{N,e}(x) = x^e \bmod N$  $g_{N,d}(y) = y^d \bmod N$

But how do we find suitable N,e,d ?

If p,q distinct primes and N = pq then $\phi(N) = (p-1)(q-1)$

Given $\phi(N)$, choose $e \in \mathbf{Z}^*_{\phi(N)}$ and calculate

$d = e^{-1} \bmod \phi(N)$

How to find suitable p,q prime?

Choose random numbers and test primality (Miller-Rabin testing)

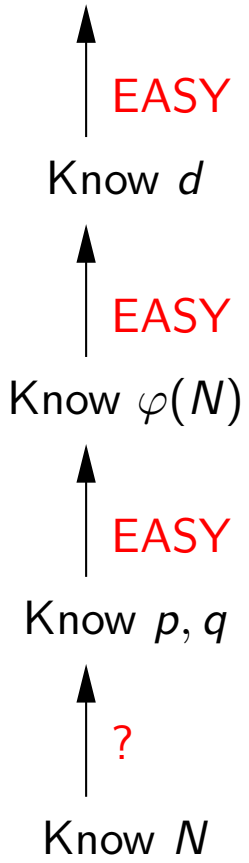
https://eprint.iacr.org/2018/749.pdf

# Summary

- Find 2 large primes p, q . Let N = pq
  - random integers + primality testing
- Choose e (usually 65,537)
  - Compute d using $\phi(N) = (p-1)(q-1)$
- pk = (N,e) and sk = (N,d)
  - Often store p,q with sk to use Chinese Remainder Theorem
- Don't use "raw RSA" to do encryption. Instead:
  - PKCS#1 v.1.5 padding of message before applying RSA
  - OAEP algorithm

# How hard is inverting RSA?

Inverting RSA : given $N, e, y$ find $x$ such that $x^e \equiv y \pmod{N}$




because $f^{-1}(y) = y^d \bmod N$

because $d = e^{-1} \bmod \varphi(N)$

because $\varphi(N) = (p-1)(q-1)$

Learning p,q from N is the factoring problem

We don't know if inverse is true, whether inverting RSA implies ability to factor

All this assumes p,q generated properly!

# Weak RSA keys

[Heninger et al. 2012]

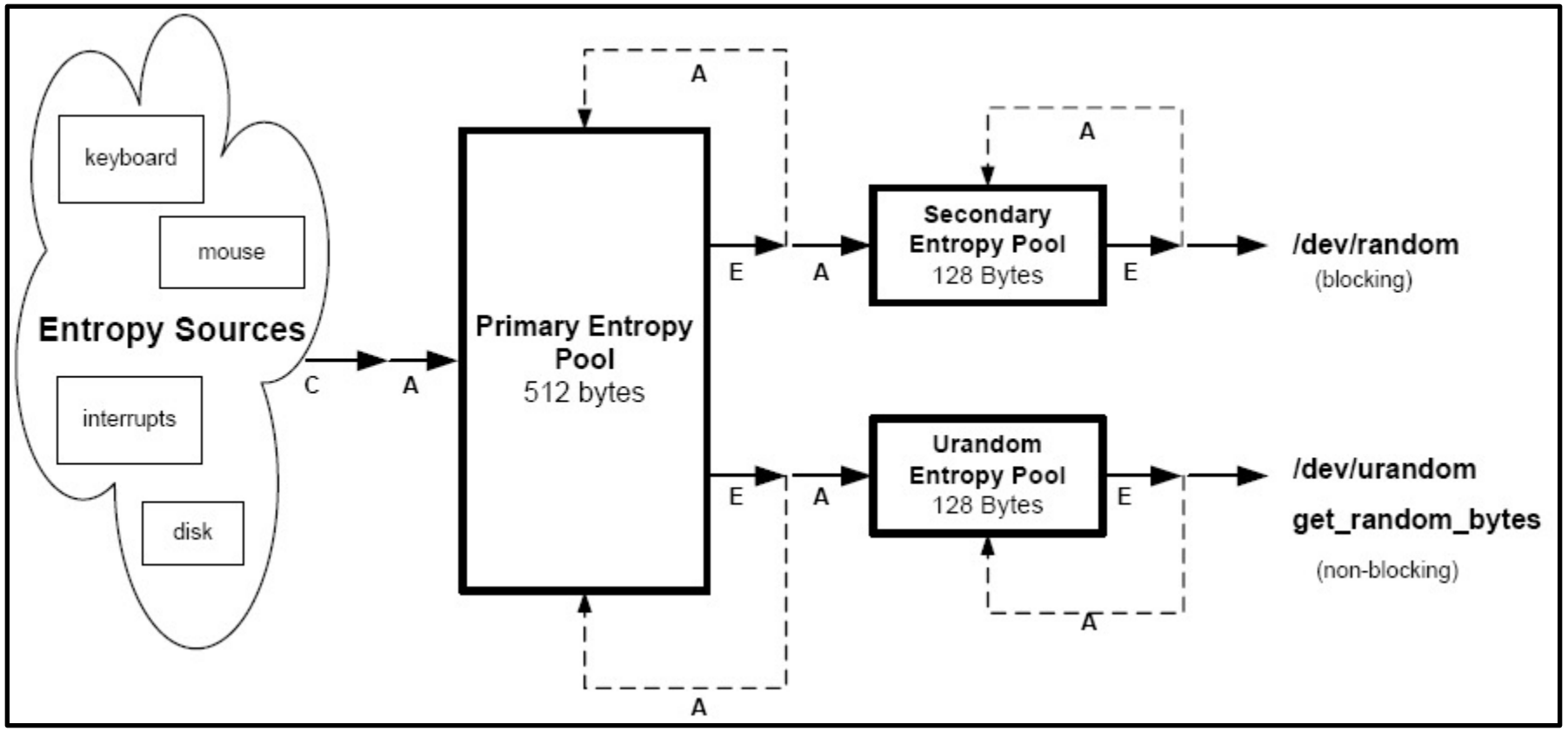
- Recall: (pk,sk) = ((N,e),(N,d))  for N = pq
- Repeat keys (may or may not be problem)

- Factoring is hard for large key sizes (>=1024)
  - But: bad randomness causes bad p,q generation
    - Ex:  N1 = p * q  N2 = p * q'
    - Compute GCD of large integers in milliseconds
    - Use Bernstein's all-pairs GCD to scale up

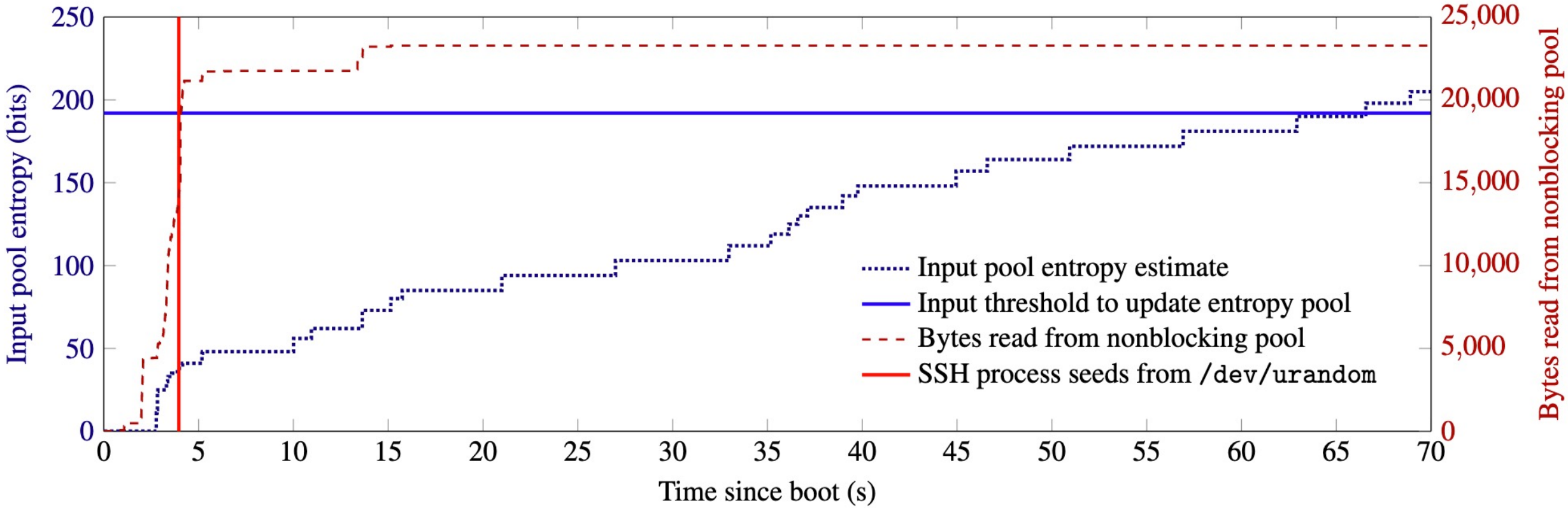
# Linux /dev/(u)random

Linux random number generator (2500 lines of undocumented code)

Diagram from [Gutterman, Pinkas, Reinman 2006]



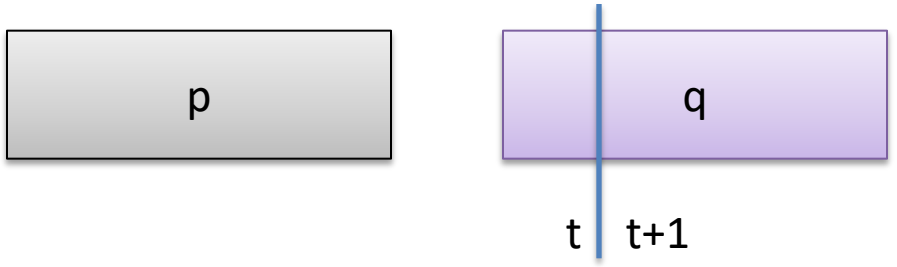

Primary entropy pool feeds into other entropy pools only when 192 bits of entropy are estimated. Favors /dev/random


Input pool entropy (bits)
250
200
150
100
50
0
0
5
10
15
20
25
30
35
40
45
50
55
60
65
70
Time since boot (s)
25,000
20,000
15,000
10,000
5,000
0
Bytes read from nonblocking pool
Input pool entropy estimate
Input threshold to update entropy pool
Bytes read from nonblocking pool
SSH process seeds from /dev/urandom

# RNGs and RSA key generation


Both primes the same


Both primes different




Shared prime

# Internet scanning

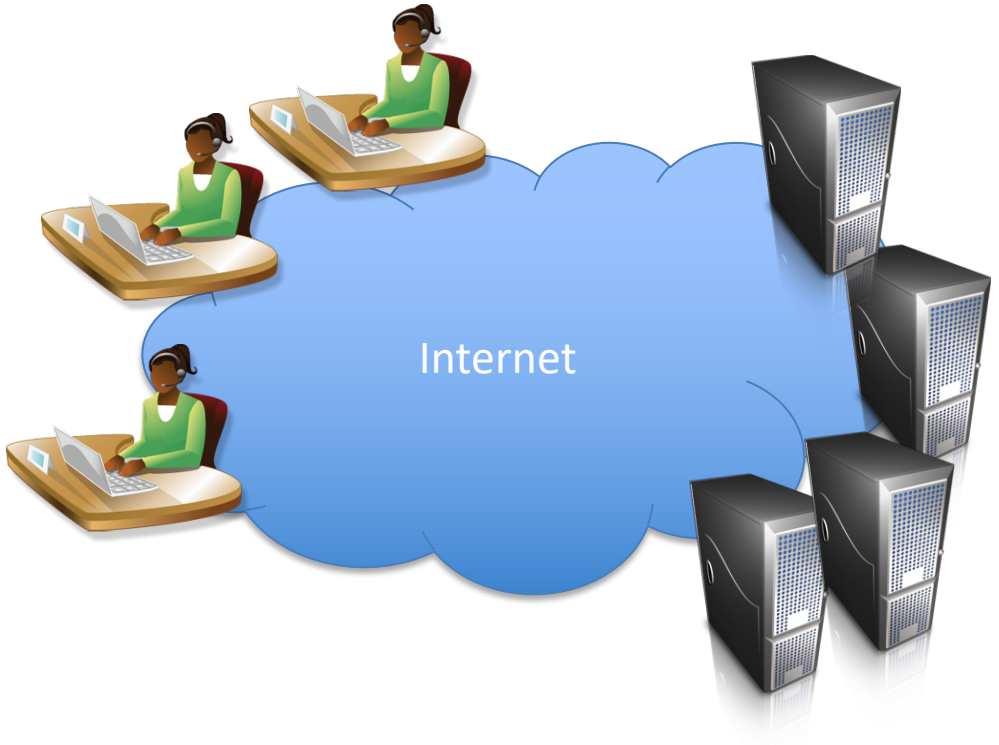
How do we survey the Internet to infer vulnerability status of systems?

Send network packets to IP address, see what response looks like

Example: TCP SYN scan

Horizontal scan: many IPs on 1 port

Vertical scan: many ports on one IP



# Basics of network reconnaissance

- Host discovery
  - Narrow broad swath of potential IPs to ones that have hosts associated with them
- Service discovery
  - For a particular host, identify running services
  - E.g., is it accepting SSH connections (22) or HTTP (80)?
- OS fingerprinting
  - Identify the OS software version running
  - E.g., Windows vs Linux?
- Application fingerprinting
  - same at higher level
  - Apache version 1.3 or 2.0+?

# Scanners

- NMAP (network map tool)
  - De-facto standard for network reconnaissance, testing
  - Numerous built in scanning methods
- Can use to scan *entire* IPv4 address space

```
[10/26/21 10:27:55 ~/Dropbox/work/teaching/cs6431-fall2021/slides-staging$ nmap -sT 192.168.1.0/28
Starting Nmap 7.92 ( https://nmap.org ) at 2021-10-26 10:28 EDT
Nmap scan report for Fios_Quantum_Gateway.fios-router.home (192.168.1.1)
Host is up (0.0024s latency).
Not shown: 992 closed tcp ports (conn-refused)
PORT     STATE    SERVICE
22/tcp   filtered ssh
53/tcp   open     domain
80/tcp   open     http
443/tcp  open     https
4567/tcp filtered tram
8022/tcp filtered oa-system
8080/tcp open     http-proxy
8443/tcp open     https-alt

Nmap done: 16 IP addresses (1 host up) scanned in 18.93 seconds
10/26/21 10:28:42 ~/Dropbox/work/teaching/cs6431-fall2021/slides-staging$
```

# Internet-scale scans

Fewer than $2^{32}$ IP addresses: ~3,706,452,992
Some ranges can be excluded:

- Private addresses (e.g., 192.168.0.0/16)
- Multicast addresses
- …

Nmap is not particularly fast for horizontal scans:

Nmap –sT on target IP:port takes ~1 seconds

117 years for horizontal scan of all addresses? Need to speed up

“Insane” custom configuration: 116 days

# Internet-scale scans

Nmap can nevertheless be scaled up

EFF SSL observatory: 2010 scan of ports 443
3 servers and 3 months of time

P’s and Q’s paper: 2011 scan of port 443 & 22:
25 hours from 25  EC2 Micro instances

# Weak keys

| | Our TLS Scan | |
|---|---|---|
| Number of live hosts | 12,828,613 | (100.00%) |
| . . . using repeated keys | 7,770,232 | (60.50%) |
| . . . using vulnerable repeated keys | 714,243 | (5.57%) |
| . . . using default certificates or default keys | 670,391 | (5.23%) |
| . . . using low-entropy repeated keys | 43,852 | (0.34%) |
| . . . using RSA keys we could factor | 64,081 | (0.50%) |
| . . . using DSA keys we could compromise | | |
| . . . using Debian weak keys | 4,147 | (0.03%) |
| . . . using 512-bit RSA keys | 123,038 | (0.96%) |
| . . . identified as a vulnerable device model | 985,031 | (7.68%) |
| . . . model using low-entropy repeated keys | 314,640 | (2.45%) |

From [Heninger et al. 2012]

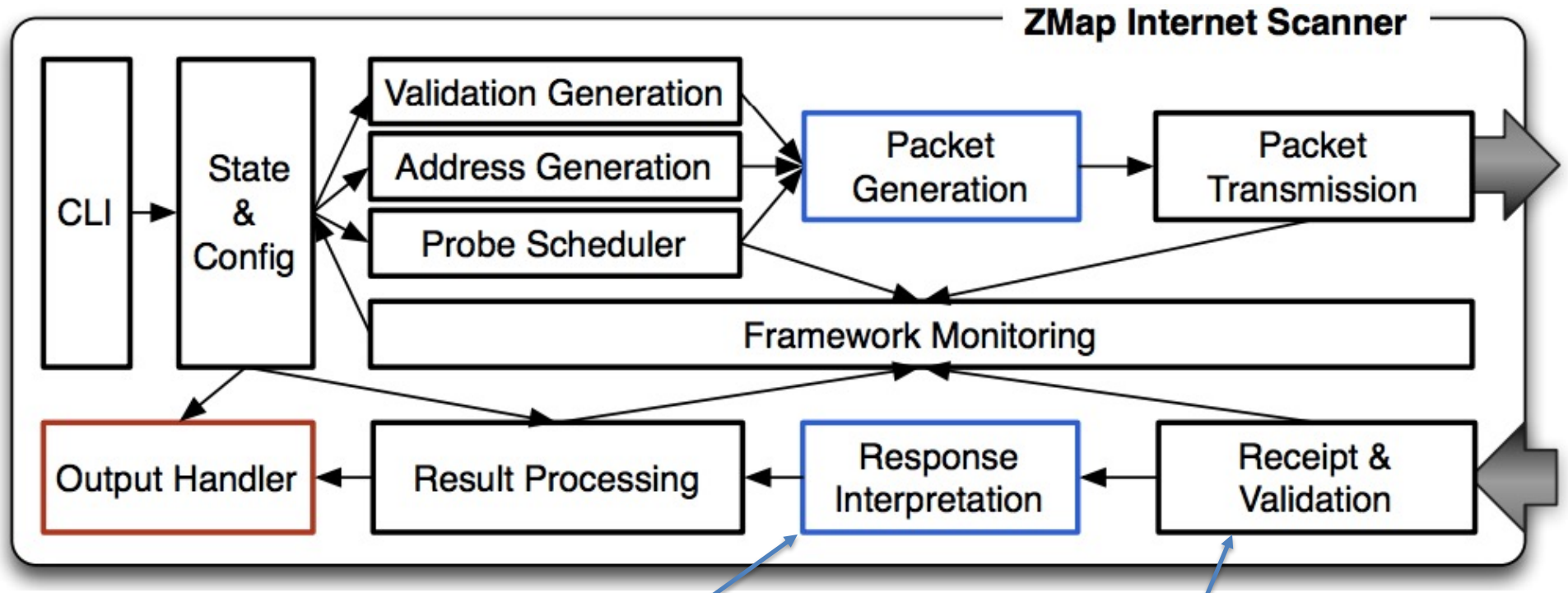
# Internet-scale scans

- How to scale up scans and make them faster/cheaper?
- Use a “horizontal scanner”
- Leonard & Loguinov 2010:  24 hour horizontal scans
  - Requires kernel modifications (on Windows)
- ZMap tool improved to 45 minute IPv4 scans

# ZMap's design

- Random target ordering
  - What would be a problem with serial order?
  - Choose random generator g in $\mathbf{Z}_p$ for $p = 2^{32} + 15$, and walk the cycle $g, g^1, g^2, \ldots$ . Can stop when one hits g again
- No per-connection state
- No retransmission
  - Send fixed number of probes (usually 1)
  - Don't try again
- Send as quickly as CPU/link allow
  - No Kernel mods necessary
  - Use raw socket and send ethernet frames directly

From [Durumeric et al. 2013]



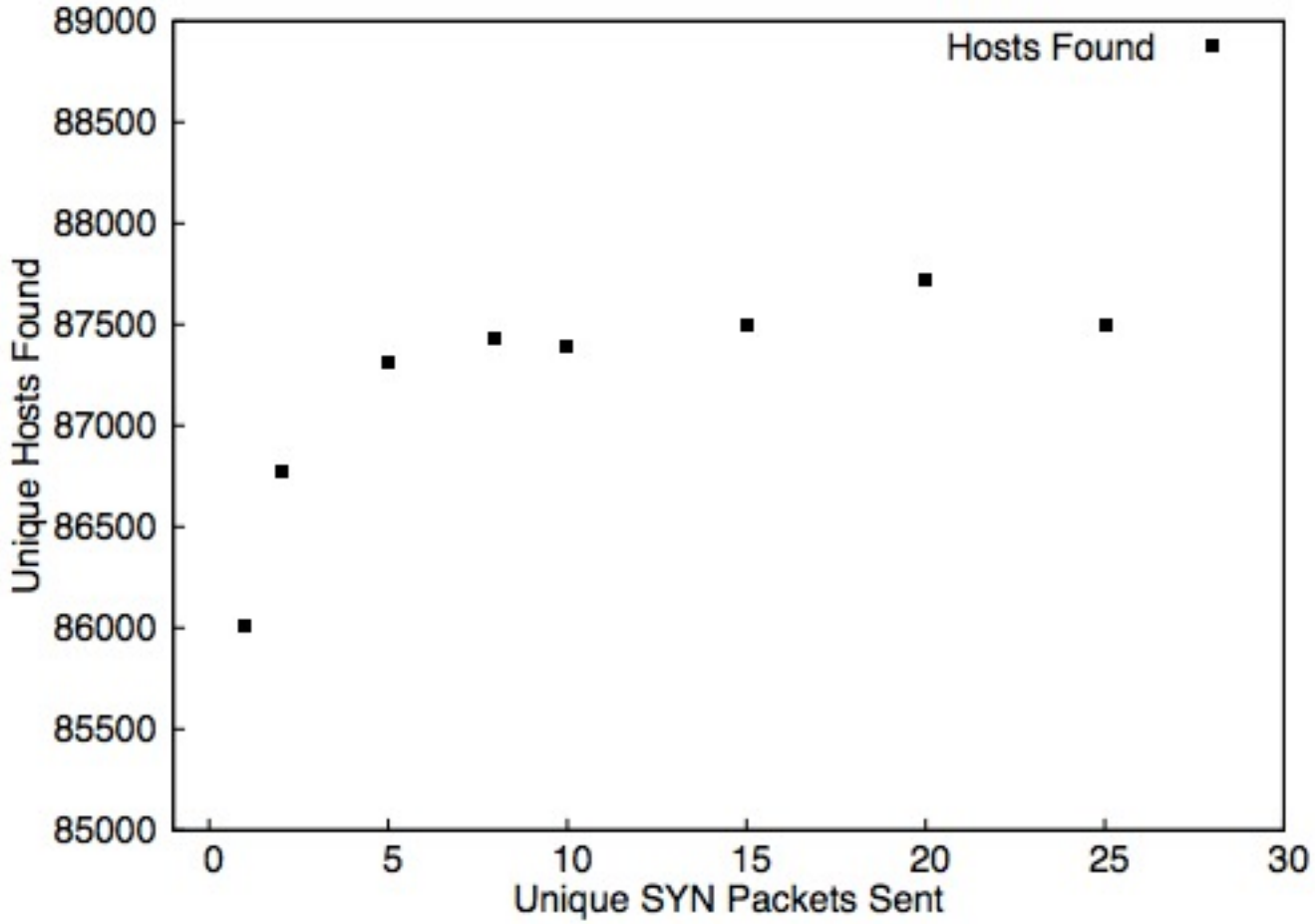

Validate response:
Check cryptographic MAC within response packet

Pcap-based response handler
Not synchronized with transmission

# Performance

- Full horizontal IPv4 scan in 45 minutes with entry-level server and gigabit ethernet
- Scan rate up to 1.4 million packets / sec doesn’t affect hit rate
- Coverage of ~98% with just one probe:
  - For smaller sample size, ramp up # of probes, look for plateau
- Much better than Nmap (97% coverage at 117 days)

# Coverage estimation



# What to do with a fast scanner?

- Track protocol use over time (e.g., uptick in HTTPS)
- Find vulnerable hosts
  - Custom probe module to look for remotely exploitable vulnerabilities
  - Empirical analysis of cryptographic weaknesses
- Build a search engine (Censys follow-up paper)
  - https://censys.io/

# Cryptographic vulnerabilities

- Perform TLS handshakes with entire IPv4 address space
- Investigate security of hosts based on responses

# Scanning now a widely used tool

- DROWN attacks (Bleichanbacher downgrade against SSLv2):
  - 30% of TLS servers on IPv4 are vulnerable
- Dual EC fingerprinting
  - Found 720 servers using BSAFE-Java
- LogJam attacks (Downgrade to export-level crypto in TLS)
  - 7% of Alexa Top Million
- Many further measurement studies: TLS CA ecosystem, SMTP server ecosystem, SCADA control systems, FTP servers, …

# The future

- Improvements over Zmap?
- Application layer scanning?
- IPv6 will cause horizontal scan challenges
- Think about methodology, not just results
  - Zmap paper has had a lot of impact
  - “Cottage industry” for Zmap authors, but many others also using it

# Textbook exponentiation

Let $G$ be a group.
How do we compute $h^x$ for any $h \in G$?

```
Exp(h,x)
X' = h
For i = 2 to x  do
      X' = X'•h
Return X'
```

Requires time O(|G|) in worst case.

```
SqrAndMulExp(h,x)
b_k,…,b_0 = x
f = 1
For i = k down to 0 do
          f = f^2
          If b_i = 1 then
                    f = f•h
Return f
```

Requires time O(k) multiplies and squares in worst case.

SqrAndMulExp(h,x)

$b_k,\ldots,b_0 = x$

f = 1

For i = k down to 0 do

    f = $f^2$

    If $b_i$ = 1 then

        f = f*h

Return f

$$x = \sum_{b_i \neq 0} 2^i$$

$$h^x = h^{\sum_{b_i \neq 0} 2^i} = \prod_{b_i \neq 0} h^{2^i}$$

$h^{11} = h^{8+2+1} = h^8 \bullet h^2 \bullet h$

$b_3 = 1$ $f_3 = 1 \bullet h$

$b_2 = 0$ $f_2 = h^2$

$b_1 = 1$ $f_1 = (h^2)^2 \bullet h$

$b_0 = 1$ $f_0 = (h^4 \bullet h)^2 \bullet h = h^8 \bullet h^2 \bullet h$

Don't implement this algorithm: side-channel attacks