Exploiting poor randomness: Collecting data

Nadia Heninger

University of Pennsylvania

October 14, 2014

What do we use randomness for anyway?

Cryptographic usage:

- Encrypted hard disks
- Communications: TLS/SSL, SSH, PGP, OTR, TextSecure
- Money: ATM networks, EMV/chip and pin, Bitcoin
- Identity/Authentication: Access cards

Security:

- ASLR (address space layout randomization)
- TCP sequence numbers
- DNS source port randomization
- password hash salting

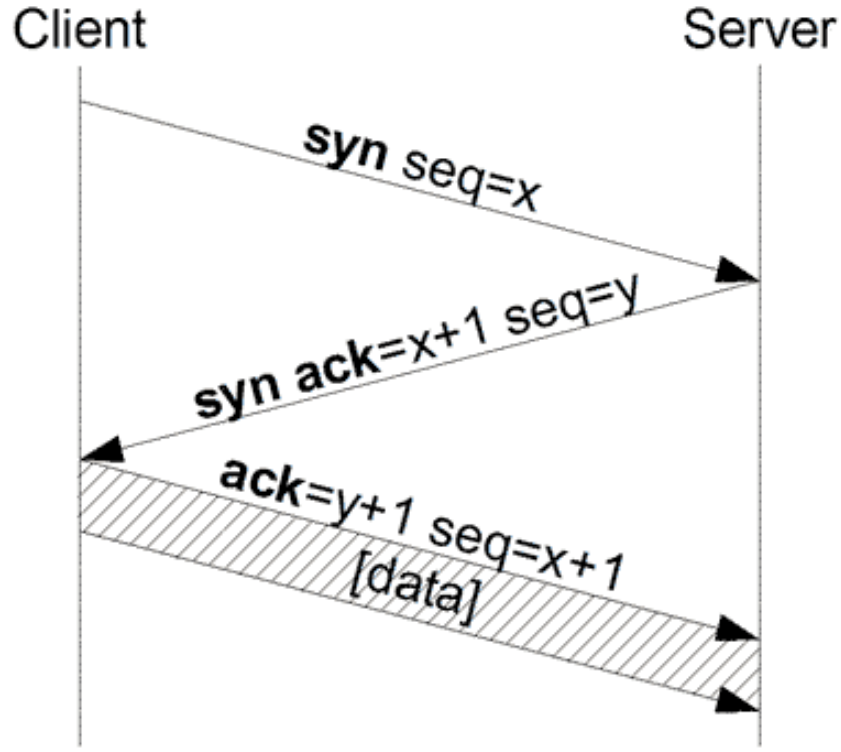
Let’s scan the internet for public keys!

Networking

OSI Model

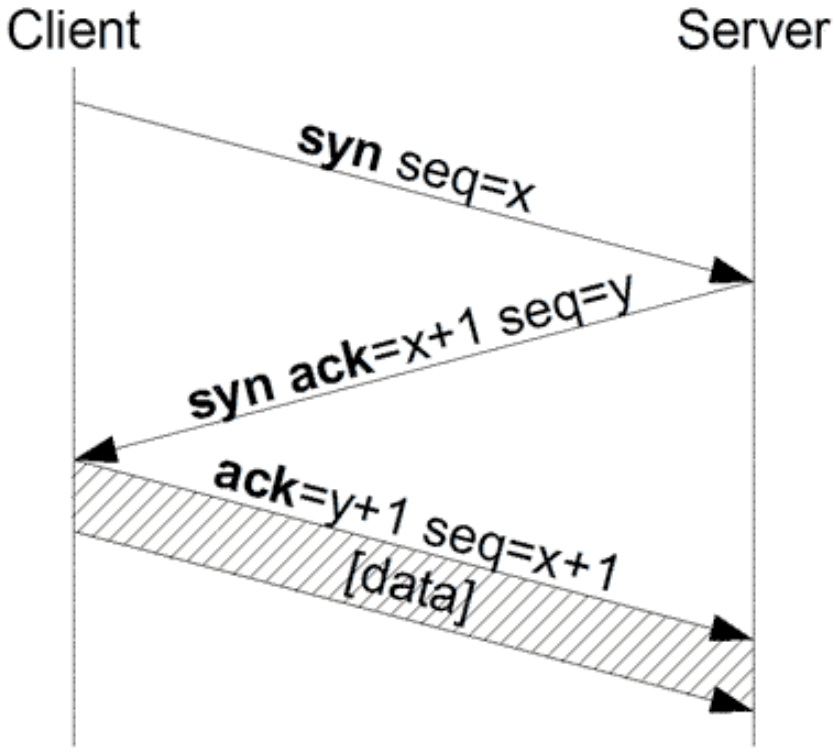
Application	HTTP
Presentation	TLS
Transport	TCP
Network	IP
Datalink	Ethernet, WiFi
Physical	Copper, Fiber

TCP Handshake



Wikipedia

Methodology: SYN Scanning



Wikipedia

1. Send syn packets to many addresses+port.
2. If syn ack received, address is listening on port.
3. Initiate further connection if desired.

Nmap

- Open source port scanner.
- Optimized for scanning all ports on a single host.


```
nadiah$ nmap attackschool.di.uminho.pt

Starting Nmap 6.46 ( http://nmap.org ) at 2014-10-13 11:53 WEST
Nmap scan report for attackschool.di.uminho.pt (193.136.19.20)
Host is up (0.041s latency).
rDNS record for 193.136.19.20: www.di.uminho.pt
Not shown: 995 filtered ports
PORT     STATE SERVICE
53/tcp   open  domain
80/tcp   open  http
443/tcp  open  https
3000/tcp open  ppp
9000/tcp open  cslistener
```

Internet Scanning, a brief history

- Census and Survey of the Visible Internet (2008)
  - 3 months, 2200 CPU-hours

EFF SSL Observatory (2010)

- 3 months on 3 Linux desktop machines (6500 CPU-hours)

- Mining your Ps and Qs (2012)
  - 25 hours across 25 Amazon EC2 instances (625 CPU-hours)

- Internet Census (2012)
  - 420,000-node botnet of unsecured devices scanning for 5 months

Internet Scanning, a brief history

- Census and Survey of the Visible Internet (2008)
  - 3 months, 2200 CPU-hours

EFF SSL Observatory (2010)

- 3 months on 3 Linux desktop machines (6500 CPU-hours)

- Mining your Ps and Qs (2012)
  - 25 hours across 25 Amazon EC2 instances (625 CPU-hours)

- Internet Census (2012)
  - 420,000-node botnet of unsecured devices scanning for 5 months

Problem: Nmap not optimized for broad scanning.

Durumeric, Wustrow, Halderman Usenix Security 2013

- Scanner optimized for large scans.
- Scan all of IPv4 on one port in 45 minutes (1GBPS uplink) or 4.5 minutes (10GBPS uplink).
- Increased speed from avoiding kernel TCP stack.
- Scan addresses in random order to avoid DOSing single networks.

Scanning: Legal and Ethical Issues

2013 EU directive on attacks against information systems recommends criminializing:

- *Illegal access to information systems*
  "access without right, to the whole or to any part of an information system,"
- *Illegal system interference*
  "seriously hindering or interrupting the functioning of an information system by inputting computer data, by transmitting, damaging, . . . , or by rendering such data inaccessible"
- *Illegal interception*
  "intercepting, by technical means, non-public transmissions of computer data to, from or within an information system, including electromagnetic emissions from an information system carrying such computer data"

Scanning: Legal and Ethical Issues

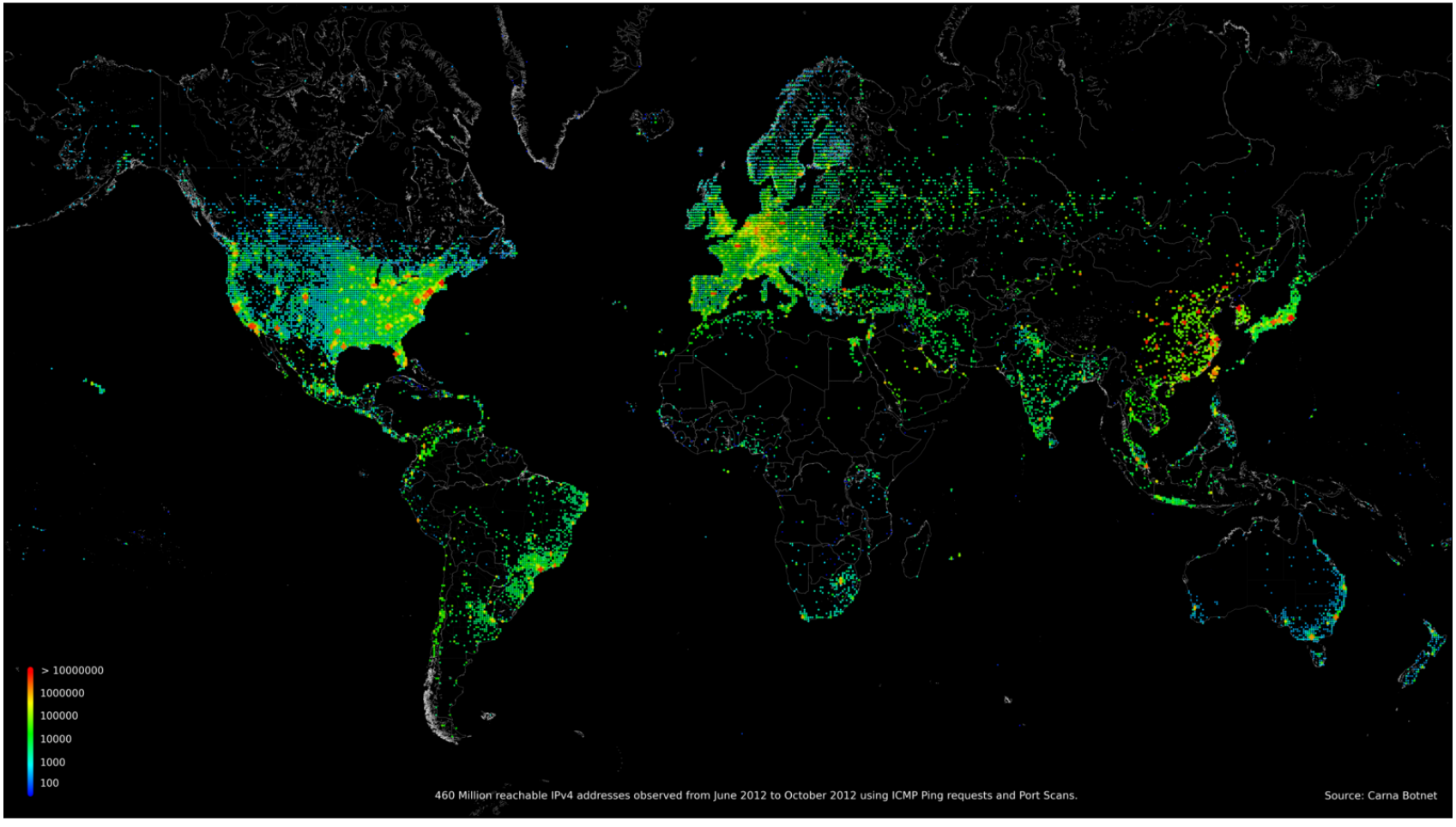
- Scanning is a common tool used by academic and industry security researchers.
- Initiating a normal connection with a public IP address on a public port is not circumventing any access control.

Ethical principles:
Want to scan in such a way that do not cause problems for hosts or provider.

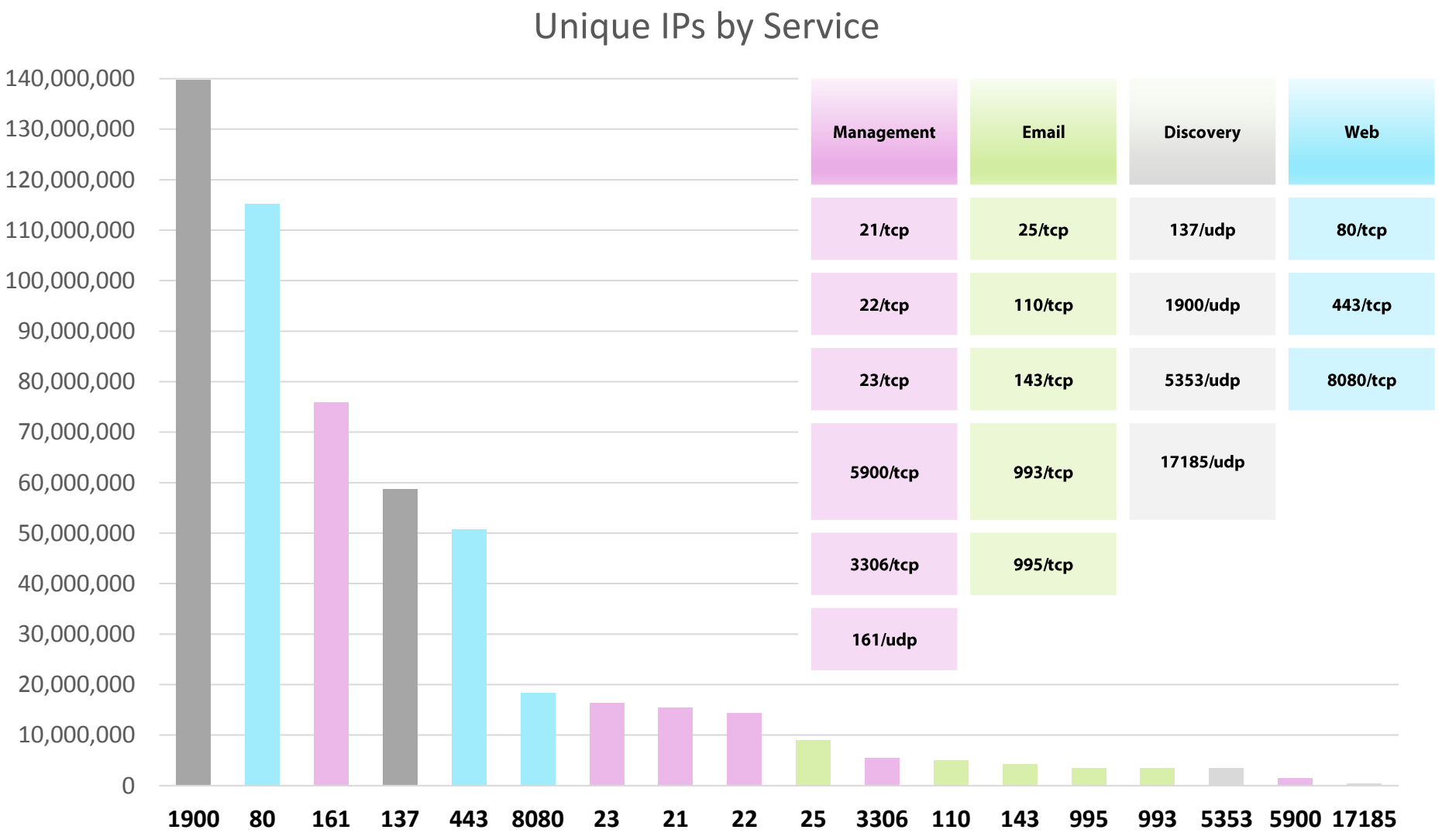
- Randomize and slow scan to not overwhelm end hosts.
- Signal benign scan in DNS entry.
- Let hosts opt out.

Internet Scan Results

Geolocation map from Internet Census



Identified Network Services

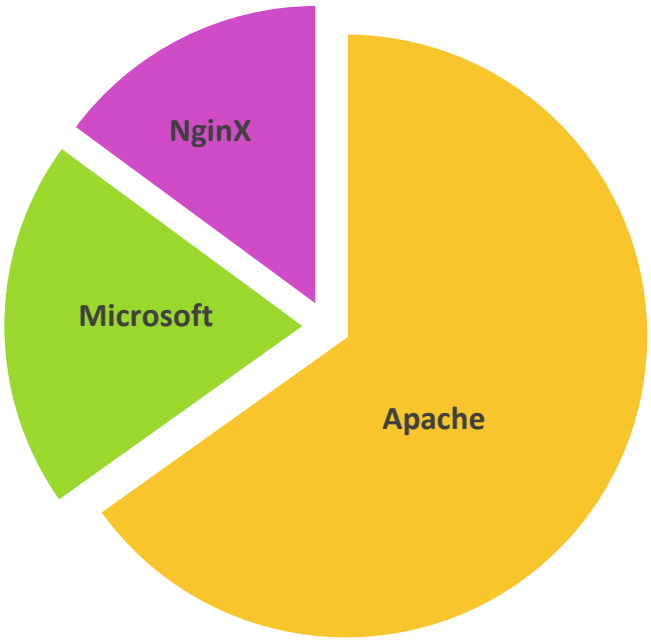


RAPID7

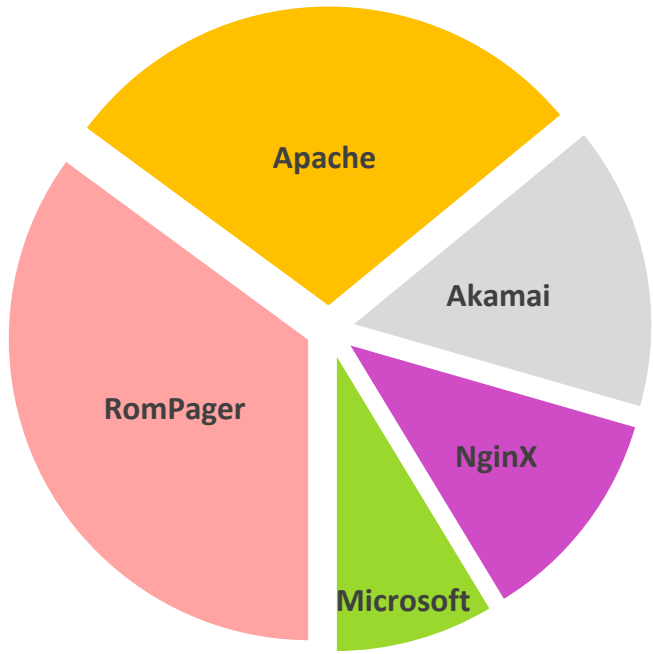
Web: Software

- The top web servers are not the most common
  - Netcraft reports web servers by domain, not by IP address
  - Embedded web servers outnumber Apache & IIS

Netcraft - January 2013



Critical.IO - January 2013



RAPID7

The Internet of Things is already here

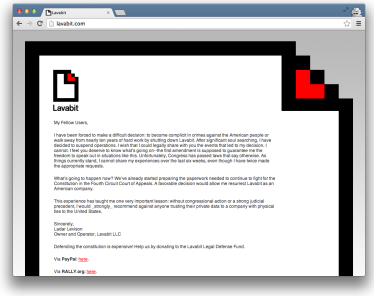
Internet Census 2012

Product	Count	Percent
Apache	14208112	20.1
Allegro RomPager	13116974	18.5
	8881082	12.5
Microsoft IIS httpd	6071267	8.6
AkamaiGHost	4064402	5.7
nginx	4045993	5.7
micro_httpd	1991840	2.8
Arris cable modem	1610036	2.3

TLS RSA Key Exchange

hello, 28 byte client random, 4 byte time

list of cipher suites



TLS RSA Key Exchange

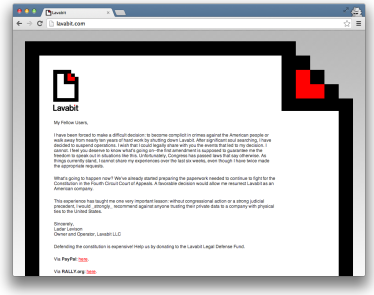
hello, 28 byte client random, 4 byte time

list of cipher suites

hello, 28 byte server random, 4 byte time

certificate = public RSA key + CA signatures

chosen cipher suite



TLS RSA Key Exchange

hello, 28 byte client random, 4 byte time →

list of cipher suites →

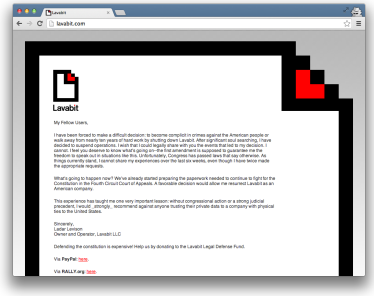
← hello, 28 byte server random, 4 byte time

← certificate = public RSA key + CA signatures

← chosen cipher suite

← $\text{RSAEnc}_{RSAkey}(\text{PMS})$

→



F(PMS, randoms) → keys

F(PMS, randoms) → keys

TLS RSA Key Exchange

F(PMS, randoms) → keys

hello, 28 byte client random, 4 byte time →

list of cipher suites →

← hello, 28 byte server random, 4 byte time

← certificate = public RSA key + CA signatures

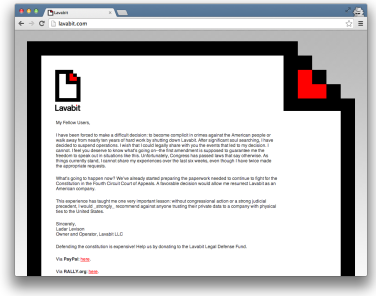
← chosen cipher suite

RSAEnc$_{RSAkey}$(PMS) →

MAC(dialog) →

← MAC(dialog)

← Enc(website contents)



F(PMS, randoms) → keys

TLS RSA Key Exchange

hello, 28 byte client random, 4 byte time →

list of cipher suites →

← hello, 28 byte server random, 4 byte time

← certificate = public RSA key + CA signatures

← chosen cipher suite

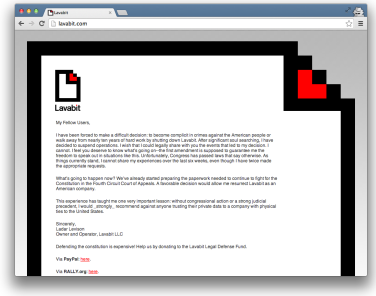
RSAEnc$_{RSAkey}$(PMS) →

MAC(dialog) →

← MAC(dialog)

← Enc(website contents)

F(PMS, randoms) → keys



F(PMS, randoms) → keys

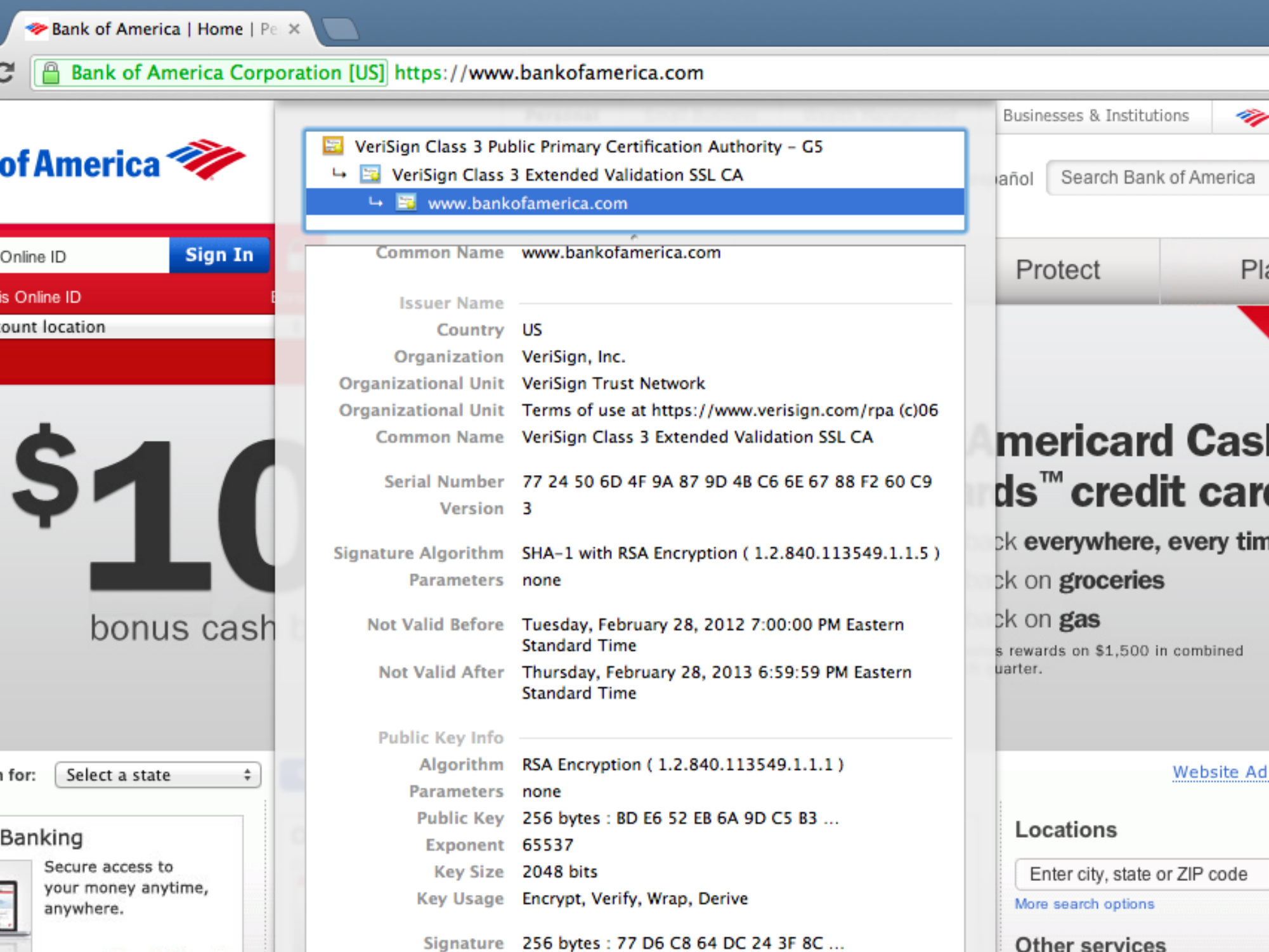
TLS Public Keys

Public keys are used to:

- bind domain names to certificates
- sign certificates (for self-signed certificates or CA keys)
- RSA-encrypt session key information

A compromised key could be used to:

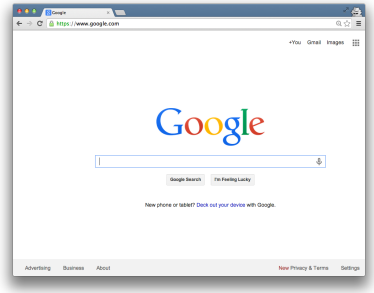
- Man in the middle connections.
- Decrypt passively collected session traffic if connection uses RSA key exchange
- Sign new trusted certificates (if it’s a CA key).


Bank of America | Home
Bank of America Corporation [US] https://www.bankofamerica.com
VeriSign Class 3 Public Primary Certification Authority - G5
VeriSign Class 3 Extended Validation SSL CA
www.bankofamerica.com
Common Name www.bankofamerica.com
Issuer Name
Country US
Organization VeriSign, Inc.
Organizational Unit VeriSign Trust Network
Organizational Unit Terms of use at https://www.verisign.com/rpa (c)06
Common Name VeriSign Class 3 Extended Validation SSL CA
Serial Number 77 24 50 6D 4F 9A 87 9D 4B C6 6E 67 88 F2 60 C9
Version 3
Signature Algorithm SHA-1 with RSA Encryption (1.2.840.113549.1.1.5)
Parameters none
Not Valid Before Tuesday, February 28, 2012 7:00:00 PM Eastern Standard Time
Not Valid After Thursday, February 28, 2013 6:59:59 PM Eastern Standard Time
Public Key Info
Algorithm RSA Encryption (1.2.840.113549.1.1.1)
Parameters none
Public Key 256 bytes : BD E6 52 EB 6A 9D C5 B3 ...
Exponent 65537
Key Size 2048 bits
Key Usage Encrypt, Verify, Wrap, Derive
Signature 256 bytes : 77 D6 C8 64 DC 24 3F 8C ...
Businesses & Institutions
Search Bank of America
Sign In
Protect
bonus cash
Select a state
Banking
Secure access to your money anytime, anywhere.
Locations
Enter city, state or ZIP code
More search options
Other services

TLS Diffie-Hellman Key Exchange

hello, 28 byte client random, 4 byte time

list of cipher suites



TLS Diffie-Hellman Key Exchange

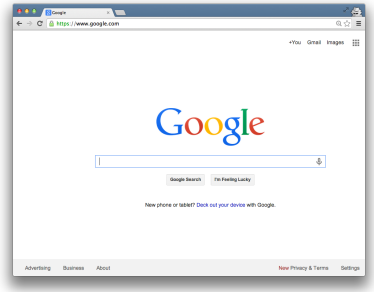
hello, 28 byte client random, 4 byte time →

list of cipher suites →

← hello, 28 byte server random, 4 byte time

← certificate = public RSA key + CA signatures

← chosen cipher suite, g^a, $\text{Sign}_{\text{RSAkey}}(g^a)$



TLS Diffie-Hellman Key Exchange

hello, 28 byte client random, 4 byte time →

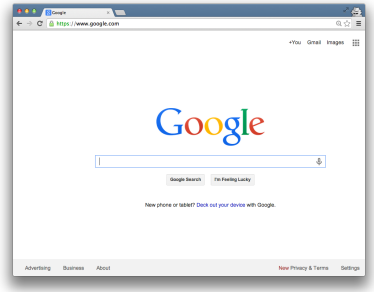
list of cipher suites →

← hello, 28 byte server random, 4 byte time

← certificate = public RSA key + CA signatures

← chosen cipher suite, g^a, $\text{Sign}_{\text{RSAkey}}(g^a)$

g^b →



F(g^{ab}, randoms) → keys

F(g^{ab}, randoms) → keys

TLS Diffie-Hellman Key Exchange

hello, 28 byte client random, 4 byte time →

list of cipher suites →

← hello, 28 byte server random, 4 byte time

← certificate = public RSA key + CA signatures

← chosen cipher suite, g^a, $\text{Sign}_{\text{RSAkey}}(g^a)$

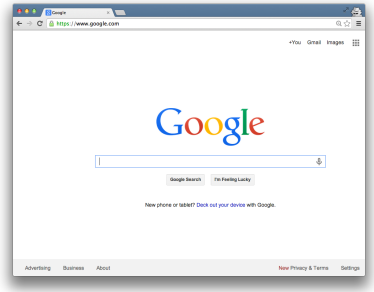
g^b →

MAC(dialog) →

← MAC(dialog)

← Enc(website contents)

F(g^{ab}, randoms) → keys



F(g^{ab}, randoms) → keys

TLS Diffie-Hellman Key Exchange

hello, 28 byte client random, 4 byte time

list of cipher suites

hello, 28 byte server random, 4 byte time

certificate = public RSA key + CA signatures

chosen cipher suite, g^a, $\text{Sign}_{\text{RSAkey}}(g^a)$

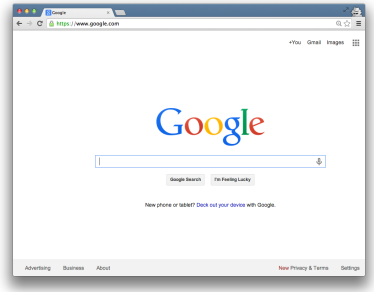
g^b

F(g^{ab}, randoms) → keys

MAC(dialog)

MAC(dialog)

Enc(website contents)



F(g^{ab}, randoms) → keys

TLS Public Keys with Diffie-Hellman key exchange

Public keys are used to:

- bind domain names to certificates
- sign certificates (for self-signed certificates or CA keys)

A compromised key could be used to:

- Man in the middle connections.
- Sign new trusted certificates (if it’s a CA key).

TLS Diffie-Hellman key exchange is used

- to negotiate session encryption.

HTTPS Scans

Eckersley Burns 2010
Heninger Durumeric Wustrow Halderman 2012
Durumeric Kasten Bailey Halderman 2013
Bos Halderman Heninger Moore Naehrig Wustrow 2014

Methodology:

- Use Nmap or Zmap to find hosts with port 443 open.
- Send client hello.
- Receive certificate/handshake in response.

Scan	Date	443 Open	Handshake	Certs	Trusted
EFF	12/2010	16.2 M	7.7M	4.0 M	1.46 M
Mining	10/2011	28.9 M	12.8 M	5.8 M	1.96 M
Zmap	06/2012	31.8 M	19.0 M	7.8 M	2.95 M
Zmap	05/2013	34.5 M	22.8 M	8.6 M	3.27 M
Zmap ECC*	10/2013	30.2 M	2.2 M	*	*

Certificate Key types

All Certificates:

Date	RSA	DSA	ECDSA	GOST
10/2011	5.6 M	6,000	8	200

Trusted Certificates (03/2013)

Type	Trusted	Valid
RSA $\leq$ 512	2600	16
RSA 768	73	0
RSA 1024	340 K	170 K
RSA 2048	2.8 M	2.5 M
RSA 4096	74 K	65 K
RSA $>$ 4096	230	190
DSA	17	7
ECDSA	0	0

"I only speak ECC" TLS negotiated cipher suites

Bos Halderman Heninger Moore Naehrig Wustrow FC 14

Scan 10/2013:

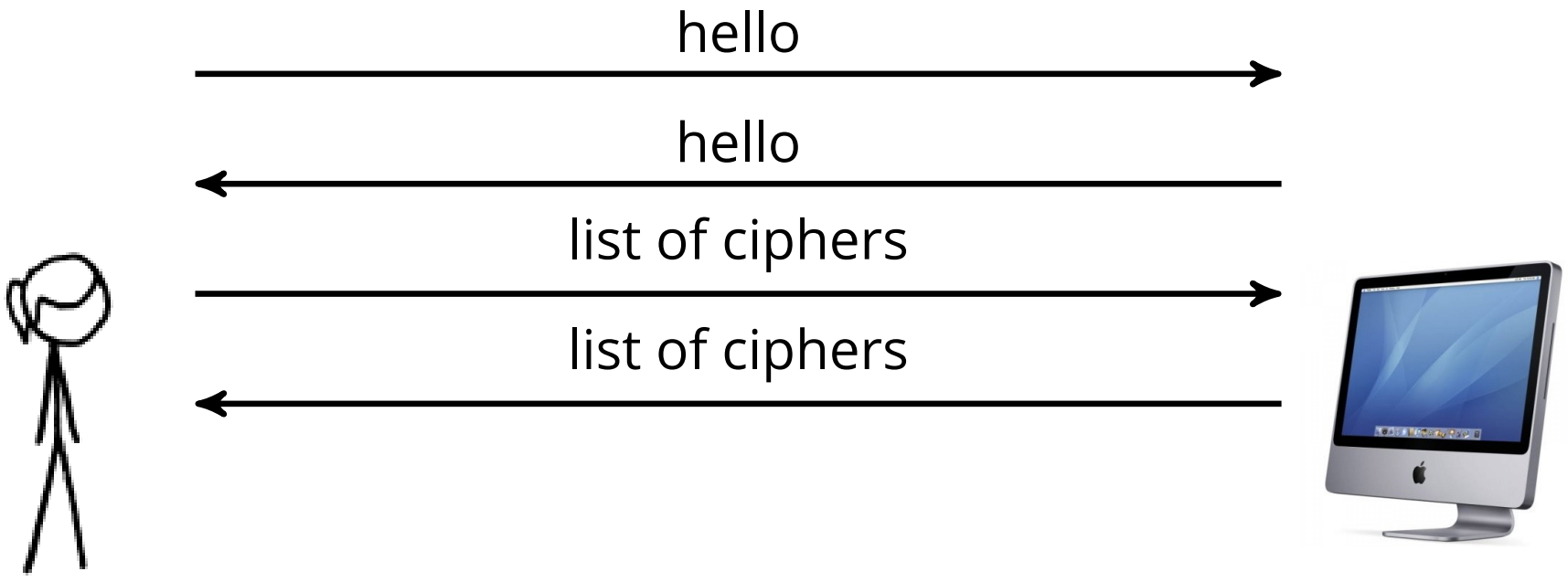
27742879	TLS_NULL_WITH_NULL_NULL
1760684	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
325787	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
167261	TLS_RSA_WITH_RC4_128_MD5
107836	TLS_ECDHE_RSA_WITH_RC4_128_SHA
23950	TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
22611	TLS_RSA_WITH_AES_256_CBC_SHA
14311	TLS_RSA_WITH_RC4_128_SHA
2273	TLS_RSA_WITH_AES_128_CBC_SHA
1785	TLS_RSA_EXPORT_WITH_RC4_40_MD5
1567	TLS_RSA_WITH_3DES_EDE_CBC_SHA
1330	TLS_DH_anon_EXPORT_WITH_RC4_40_MD5
715	TLS_DH_RSA_WITH_AES_128_CBC_SHA
370	TLS_ECDH_RSA_WITH_AES_256_CBC_SHA
332	TLS_RSA_WITH_NULL_SHA

SSH Handshake

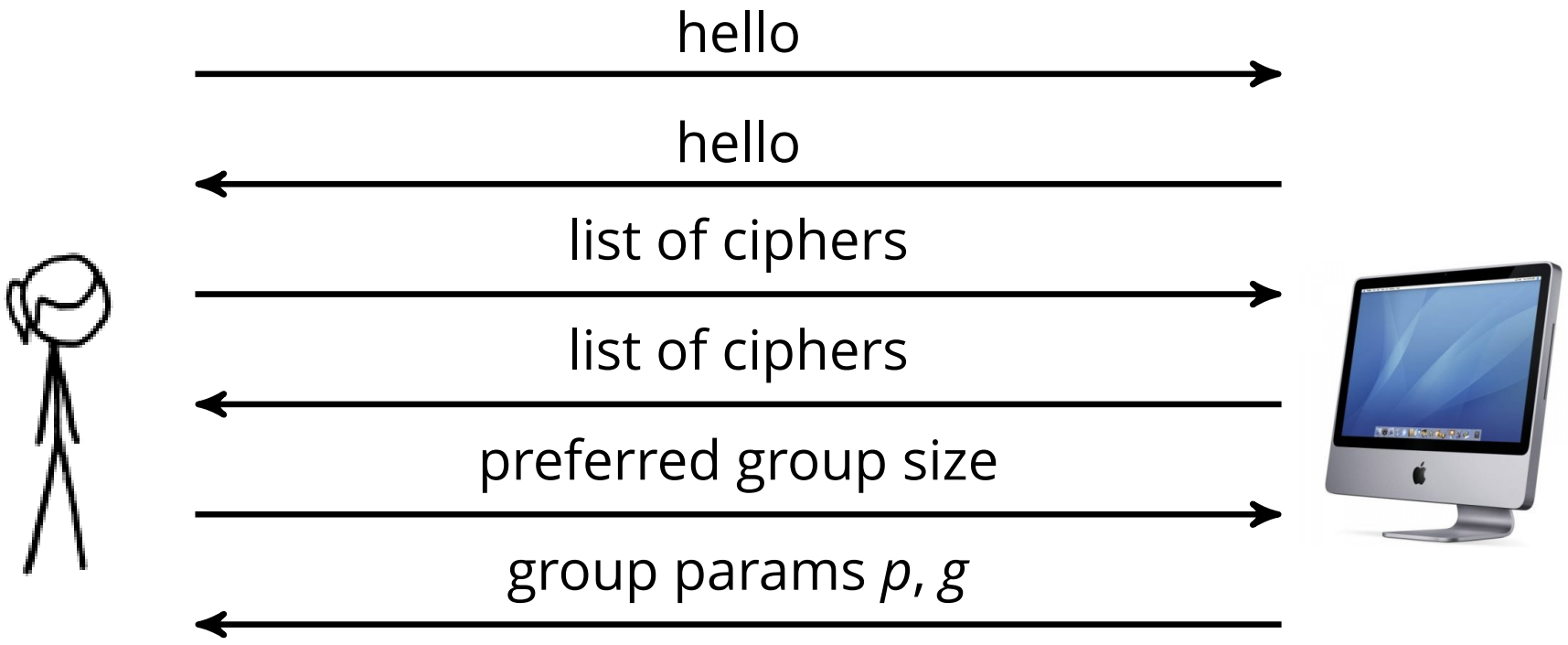
hello

hello

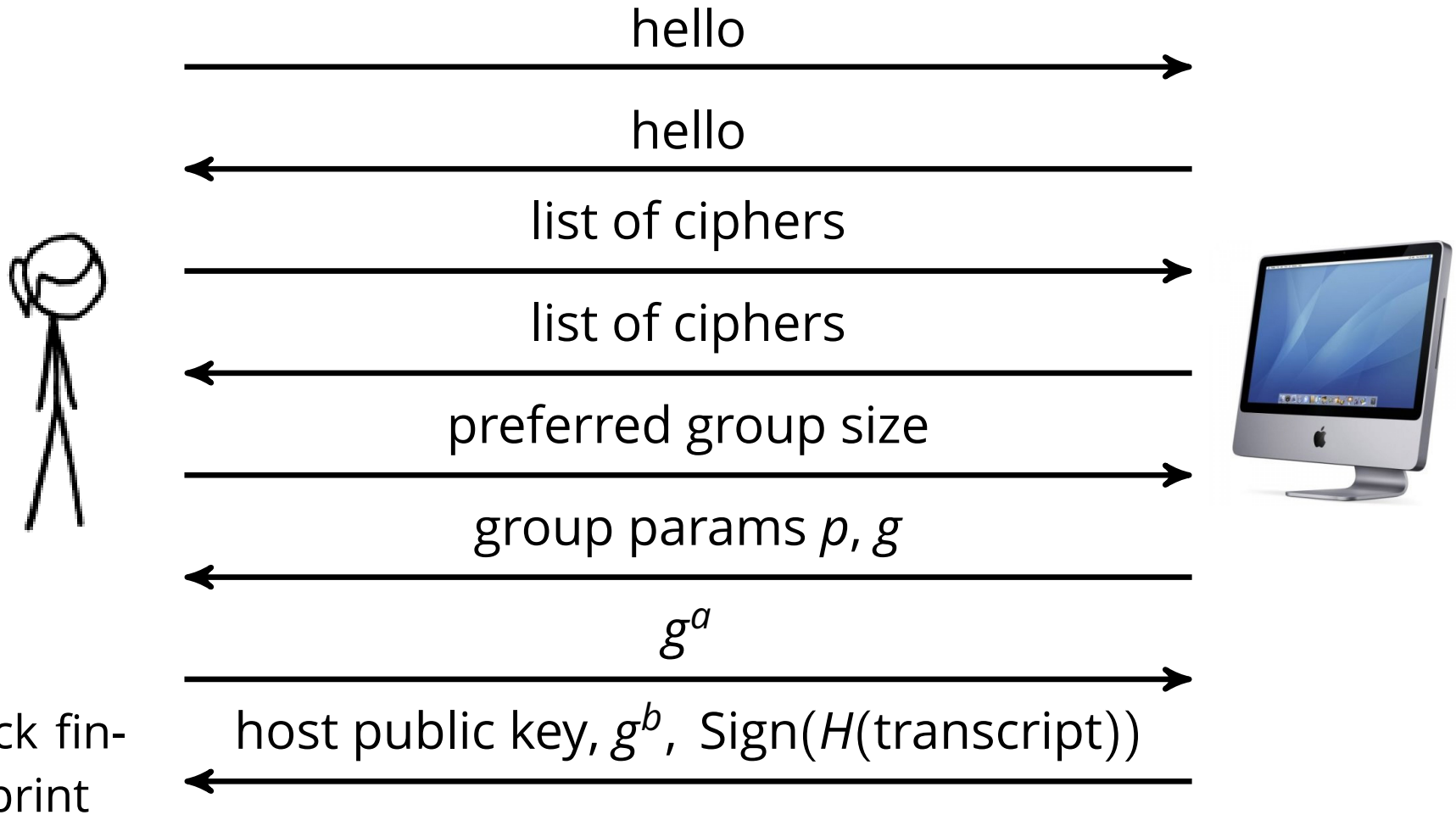
SSH Handshake



SSH Handshake



SSH Handshake



SSH Handshake

hello

hello

list of ciphers

list of ciphers

preferred group size

group params p, g

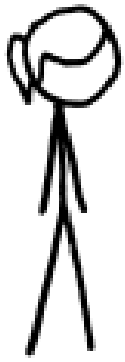
g^a

host public key, g^b, Sign(H(transcript))

Enc(password)

check fingerprint

SSH Handshake



hello

hello

list of ciphers

list of ciphers

preferred group size

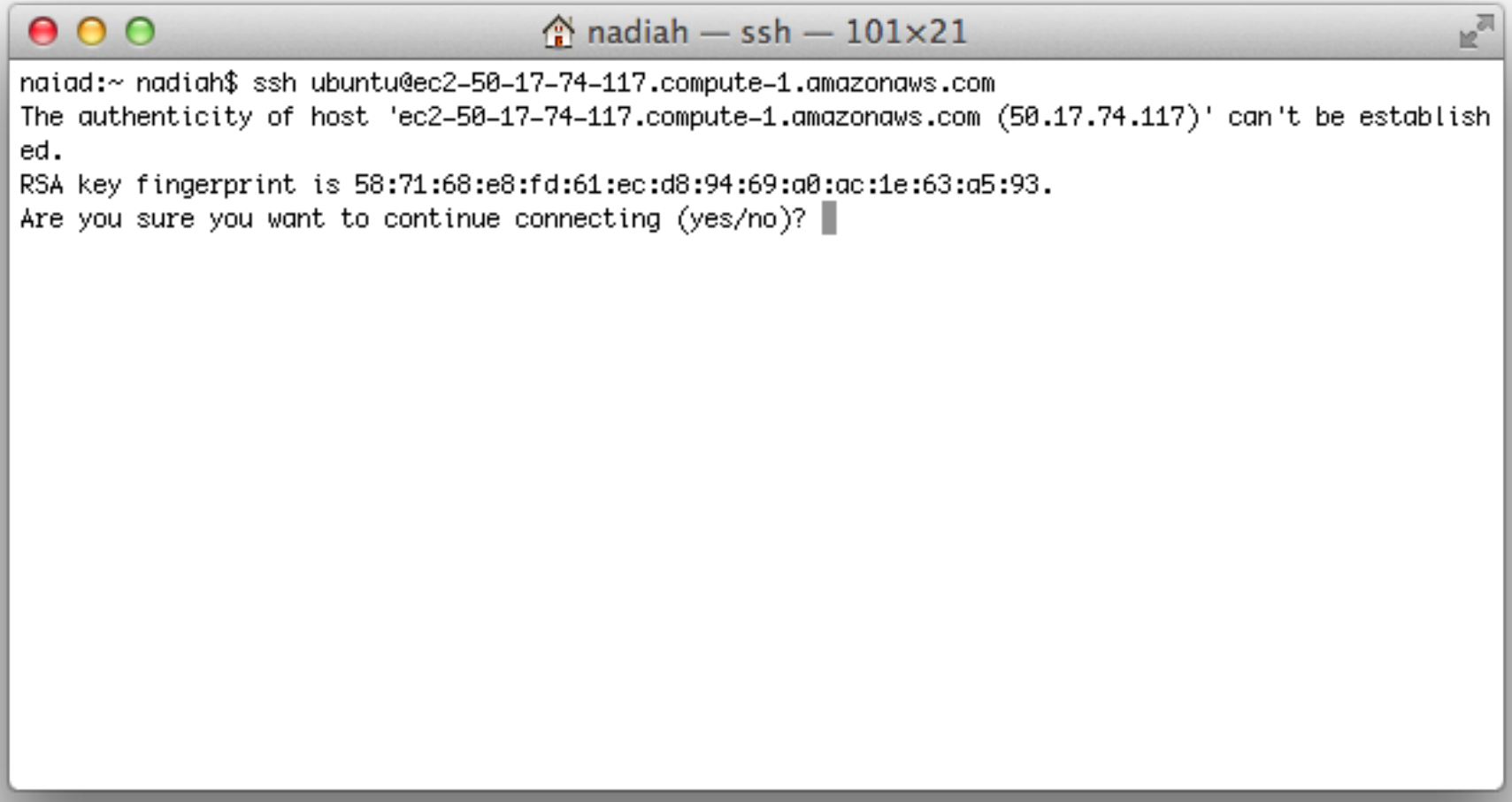
group params p, g

g^a

host public key, g^b, Sign(H(transcript))

Enc(password)

check fingerprint


nadiah — ssh — 101×21
naiad:~ nadiah$ ssh ubuntu@ec2-50-17-74-117.compute-1.amazonaws.com
The authenticity of host 'ec2-50-17-74-117.compute-1.amazonaws.com (50.17.74.117)' can't be establish
ed.
RSA key fingerprint is 58:71:68:e8:fd:61:ec:d8:94:69:a0:ac:1e:63:a5:93.
Are you sure you want to continue connecting (yes/no)?

SSH public keys

SSH host keys are used

- to authenticate hosts to clients.

A compromised SSH host key could be used

- to man-in-the-middle connections (for SSHv2, most common case)

SSH Diffie-Hellman key exchange is used

- to negotiate session encryption.

SSH Key Statistics

Heninger Durumeric Wustrow Halderman 2012; Bos Halderman Heninger Moore Naehrig Wustrow 2014

Scan	Date	Port 22	Handshake
Mining	03/2012	23 M	12 M
Zmap ECC*	10/2013	?	12 M

Host public key types:

Scan	Date	Hosts	RSA	DSA	ECDSA	GOST
Mining	02/2012	10.4 M	9.1 M	8.8 M	320K	8
Zmap ECC*	10/2013	12.1 M	10.9 M	9.9 M	1.2 M	114

Host supported key exchange algorithms:

Scan	Diffie-Hellman	ECDH	RSA1024	ECDH-GOST
Zmap ECC*	12.1 M	1.7 M	16 K	10

PGP

PGP keys are used to

- sign and encrypt email messages.

A compromised key could be used to

- decrypt messages intended for that person
- sign messages or other keys as that person.

XKCD

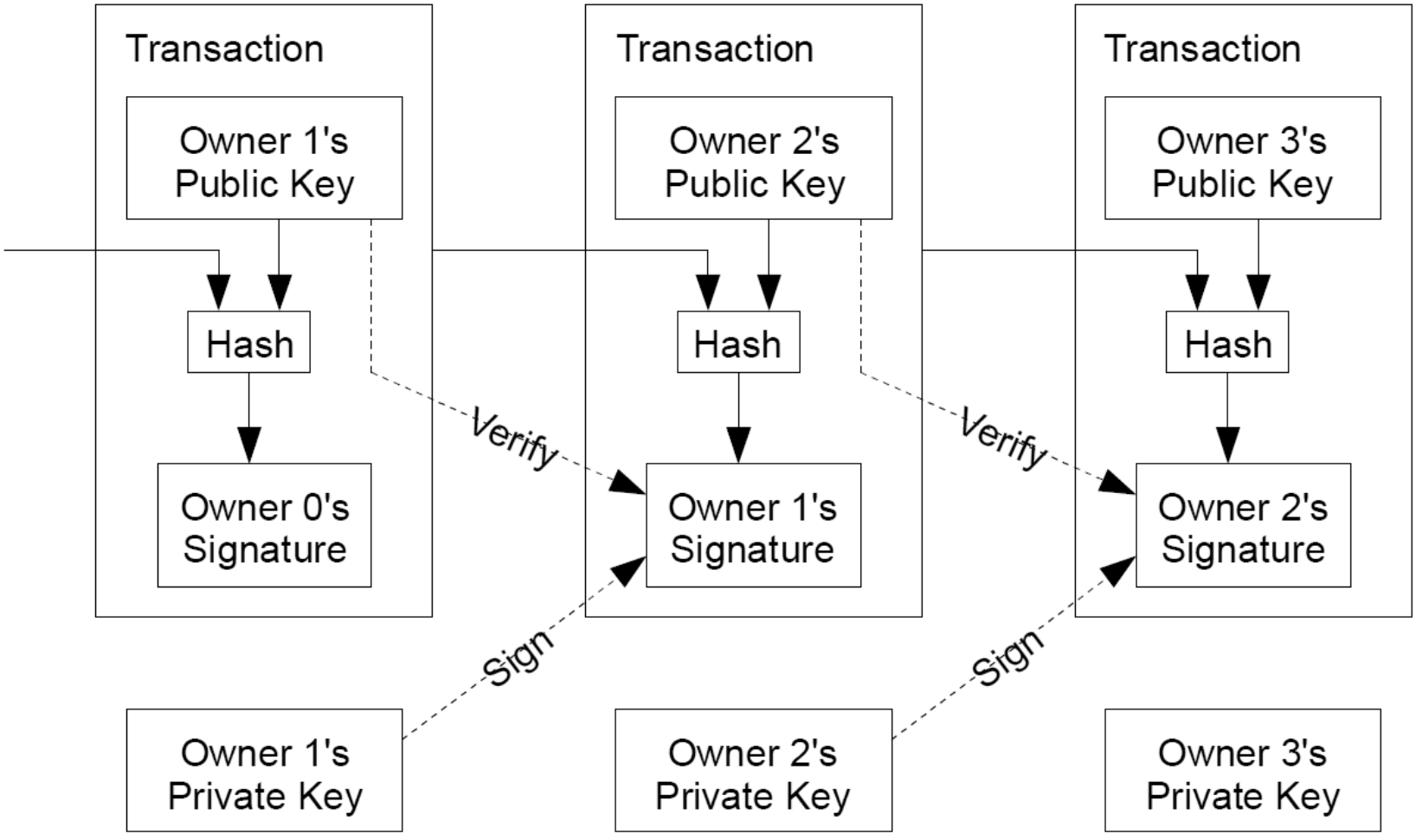
Obtaining PGP public keys

Methodology:

- Download PGP key repository dump containing public keys, signatures.
  (e.g. `http://keyserver.borgnet.us/dump/`)

RSA keys	DSA keys	ElGamal keys
700 K	2.1 M	2.1 M

Bitcoin


Transaction
Owner 1's Public Key
Hash
Owner 0's Signature
Verify
Transaction
Owner 2's Public Key
Hash
Owner 1's Signature
Verify
Transaction
Owner 3's Public Key
Hash
Owner 2's Signature
Sign
Owner 1's Private Key
Sign
Owner 2's Private Key
Owner 3's Private Key

Bitcoin

Bitcoin uses ECDSA.

A compromised bitcoin key could be used to

- transfer bitcoins from the compromised account.

Bitcoin

Addresses are (sort of) ECDSA public keys, and block chains contain signatures.

Unusual curve choice: `sec256k1`

Block chain is transferred to clients connecting to the network.

Can also be downloaded in bulk.

August 2013:

transactions	public keys	signatures
22 M	46 M	46 M