Using Large-Scale Empirical Methods to Understand Fragile Cryptographic Ecosystems

DAVID ADRIAN

PhD Defense
December 5th, 2018
High Noon

UNIVERSITY OF
MICHIGAN

Cryptography Research

Cryptography is a traditionally **formal** field born from mathematics.

- Cryptographic Primitives
  - Key Exchange
  - Signatures
  - Encryption
- Proofs of security under different assumptions
  - Random oracle model
  - Computationally-bounded adversaries
- Formal modeling
  - Symbolic verification of protocols

Targeted Case Studies vs Large-Scale Ecosystems

Case Studies

- Bugs in implementations
  - Why Cryptostems Fail [Andersoon 1994], Lessons Learned [Gutmann 2002], Most Dangerous Code [Georgiev 2012]
- Usability problems
  - Why Johnny Can’t Encrypt [Whitten 1999]

Ecosystems

- How do bugs affect the security of the Internet?
  - Matter of Heartbleed [Durumeric 2013],
- How are we failing to use cryptography to secure the Internet?
  - Alice in Warningland [Akhawe 2013], Imperfect Forward Secrecy [Adrian 2015]

Real World Cryptography

The real world is not formal.

- How do we coalesce a formal field with real world systems?
  - What problems exist in the real world that don't on paper?
- How do we design systems cannot function insecurely?
  - Make cryptographic failures result in functional failure, not silent insecurity
- How do we confidently instantiate theoretical constructs?
  - Formal proofs of implementation correctness

The field is heading in this direction, but it's **not there yet**.

Where are we now?

Fragile, Very Fragile

Cryptography deployed on the Internet is often *fragile*, but not because the math is wrong.

Complicated

- Difficult to correctly implement primitives and protocols
- Difficult to correctly deploy cryptographic solutions
- Difficult to map between proofs and production

Fragile

- Single mistake can have devastating effects
- Systems with a cryptographic failure often remain functional, but lose security
- Leads to *insecurity*

Fragile Ecosystems

The *ecosystem* of cryptography deployed on the Internet is *fragile*.

Internet

- Large-scale
- Interconnected and distributed
- Diverse

Cryptography

- Many different protocols, implementations, configurations
- “Cryptographic agility” differentiates otherwise identical software
- Anything *could* fail

Empirical Methods in Cryptography

Empirical evidence is gathered through **observation** and **experimentation**

- How do you observe and experiment in a formal field?

Understand how cryptography is being used to secure the Internet, and where it fails

- Characterize the fragility

Provide *empirical* evidence, rather than *anecdotal* evidence of problems

- Prioritize and inform solutions

Empiricism and Fragility

Cryptography that is difficult to implement correctly is **fragile**.

The more **parameters** in a cryptographic construct, the more ways for it to fail.

These parameters are **measurable**.

<u>Corollary</u>: *Fragile cryptographic ecosystems* lend themselves to be studied empirically using Internet-wide scanning.

Thesis Statement

Large-scale empirical methods allow us to observe fragility in how cryptography is being used on the Internet, identify new vulnerabilities, and better secure the Internet in the future.

Show how empirical measurements collected using Internet-wide scanning provides insight into how cryptography is used to secure the Internet.

Outline

1. **Improving Measurement**
   How do we use and improve Internet-wide scanning to understand cryptography?

2. **Measuring Export Cryptography**
   How does understanding obsolete cryptography improve modern cryptography?

3. **Beyond Internet-Wide Scanning**
   TLS 1.3 was standardized. What comes next?

TRUST THE PROCESS

ZMap: Fast Internet-Wide Scanning

Capable of performing a TCP SYN scan in **45 minutes** on a 1Gbps connection, 1300x faster than nmap [Durumeric 2013]

Many Applications

- Measuring protocol adoption
- Visibility into distributed systems
- High-speed vulnerability scanning
- *Measure cryptographic configuration*

Internet-Wide Scanning

Multi-Stage Process

1. Identify Hosts
   - IP and Transport Layer (L3/L4)
   - 4 billion IPv4, but only 50M hosts with port 443
2. Measure Individual Hosts
   - Application Layer (L5-L7)
   - 50M hosts with port 443, but only 40M HTTPS servers
3. Answer Questions
   - What percentage of hosts support *X* at time *T*?
   - Terabytes of Data

Applicability

Internet-wide scanning provides an *aggregate* understanding of the Internet over time.

- Daily / weekly measurement intervals

Can we reach a global understanding of *individual* hosts?

- Hourly / real-time measurement
- Requires improvements at all stages of the funnel

Is 45 minutes fast enough?

Scientific Method and Internet-Wide Scanning

A scan is an experiment

- Filtering lets you run your experiment on what matters?

Fix everything except one variable.

- Did I complete a handshake?
- What parameters were used for the handshake?

ZIPPIER ZMAP

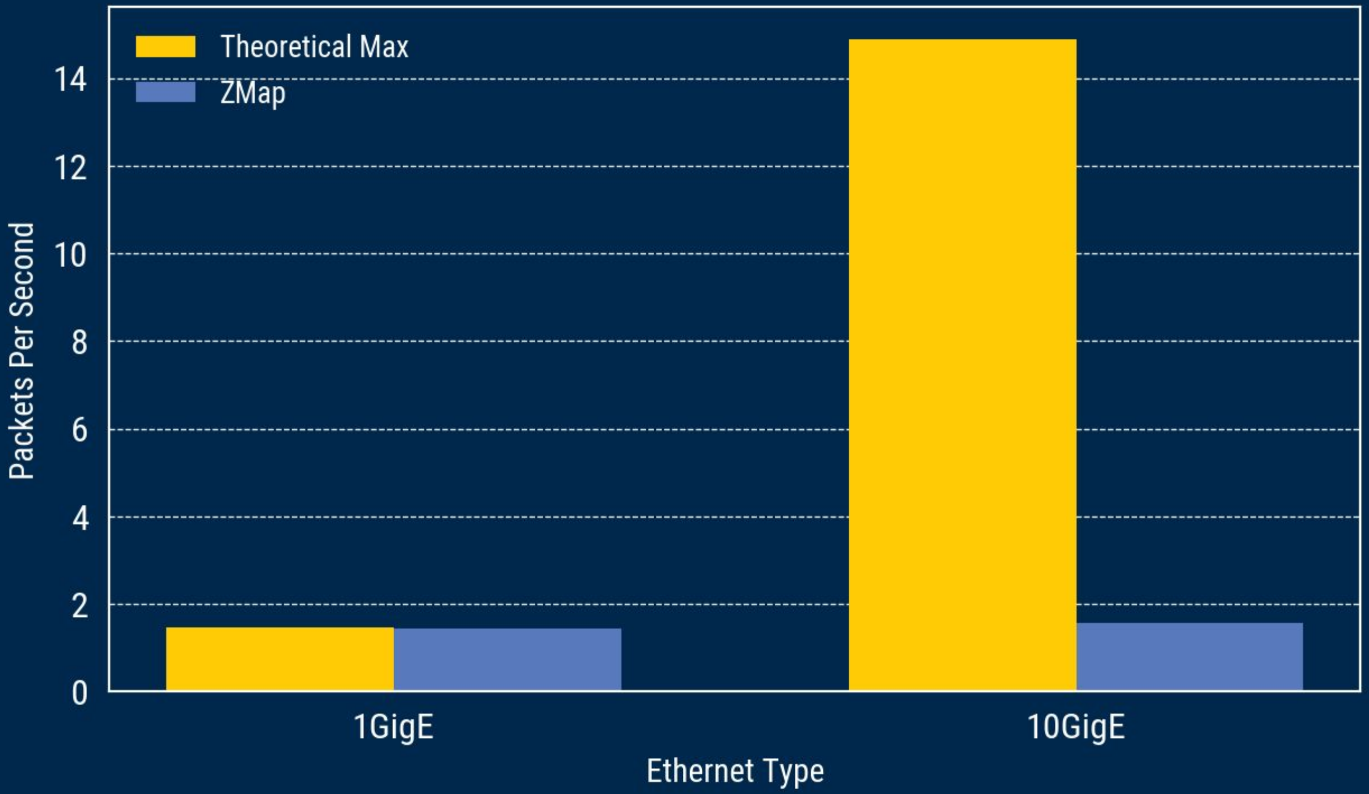
Scanning Speeds

At 45 minutes per port, it would take **5.6 years to scan IPv4 on all 65,535 ports**, or 1 month to scan 1000 ports.

Network hardware is getting faster, with 10Gbps Ethernet now common.

Can we work towards a better global understanding of individual hosts by running ZMap at faster speeds?

- Measure vulnerabilities at time of disclosure
- Correlating results across multiple ports/protocols
- Decrease moving camera effect



ZMap Performance vs Theoretical Max

UNIVERSITY OF MICHIGAN

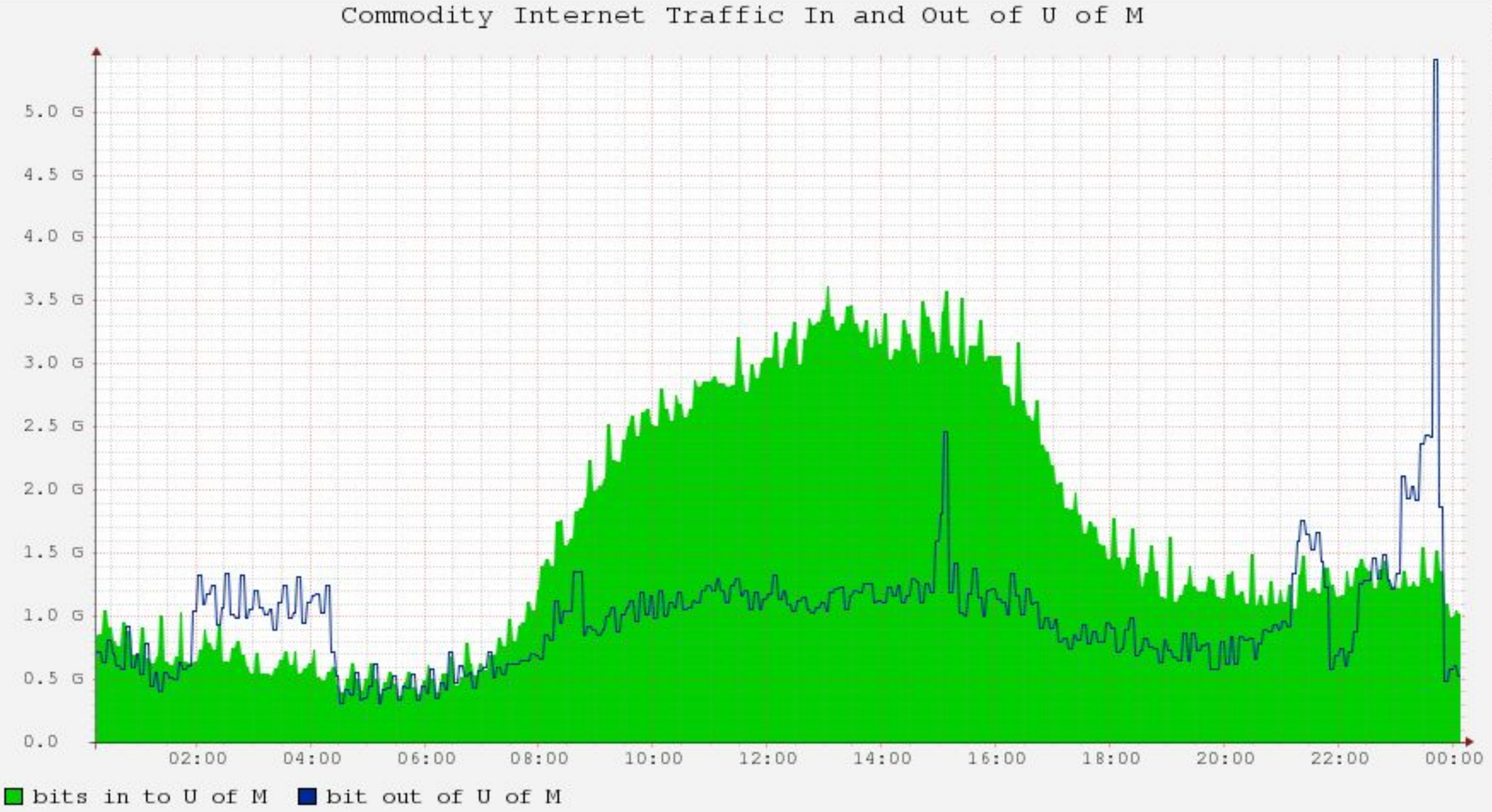
Performance Enhancements

Parallelized Address Generation

- Extend multiplicative cyclic group iteration to multiple threads
- Remove global “next address” lock

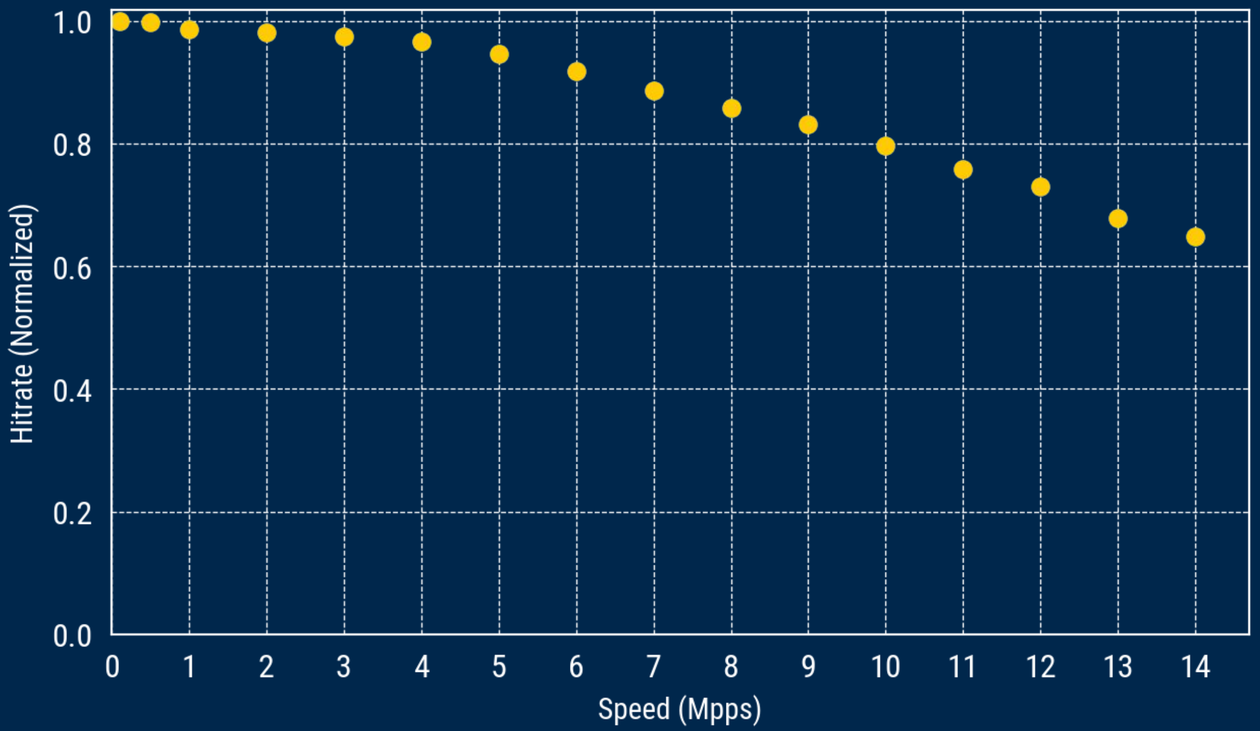
Zero-Copy Packet Transmission

- Use PF_RING to bypass kernel during packet `send`
- Removes memcpy and context switch to kernel

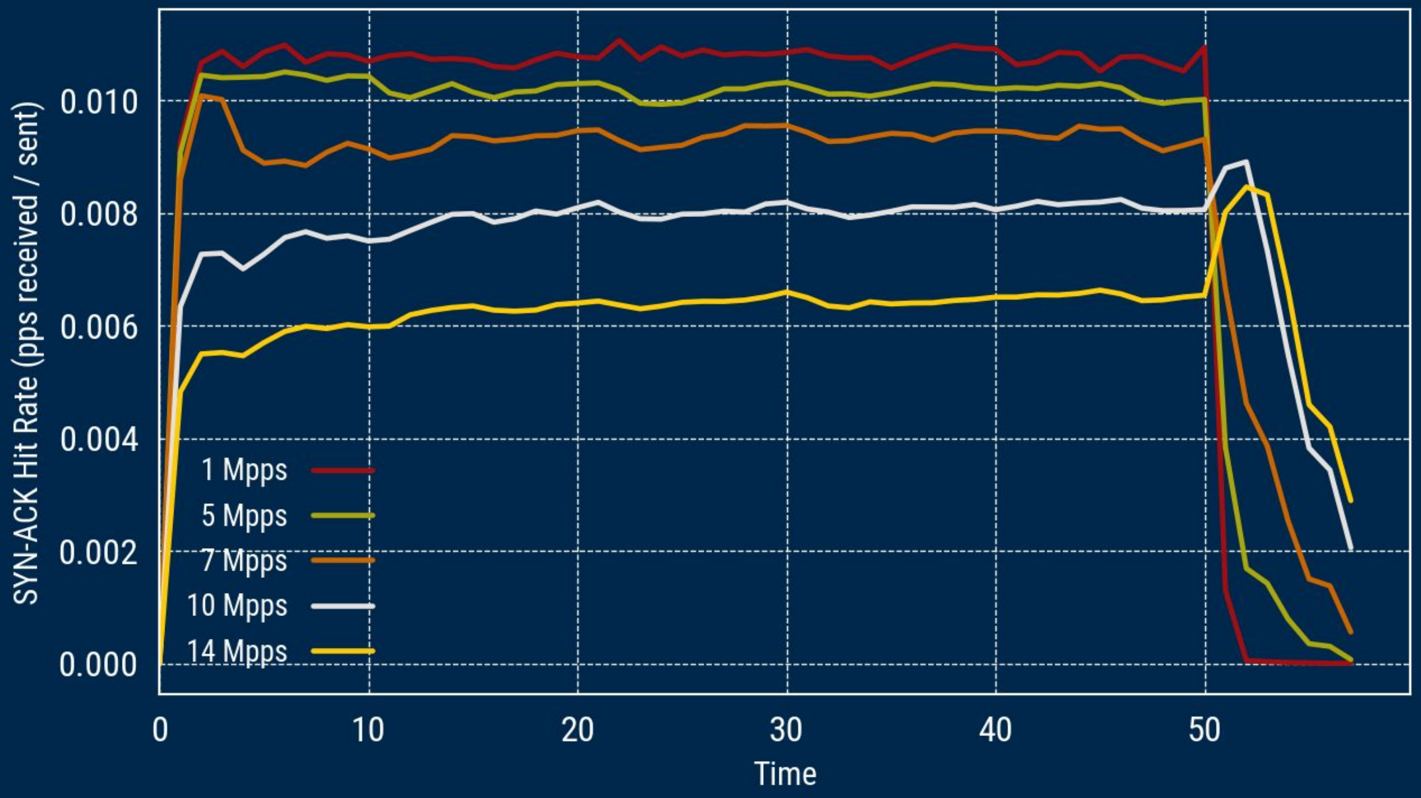

Commodity Internet Traffic In and Out of U of M
5.0 G
4.5 G
4.0 G
3.5 G
3.0 G
2.5 G
2.0 G
1.5 G
1.0 G
0.5 G
0.0
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
18:00
20:00
22:00
00:00
🟩 bits in to U of M
🟦 bit out of U of M

Scan Rate	Hit Rate	Duration
1.44 Mpps (≈1GigE)	1.00	42:08
3.00 Mpps	0.99	20:47
4.00 Mpps	0.97	15:38
14.23 Mpps (≈10GigE)	0.63	4:29

Performance of Complete Internet-Wide Scans, Port 443



Efficacy of Faster Scanning



Response rate during 50-second scans with 8 second cooldown

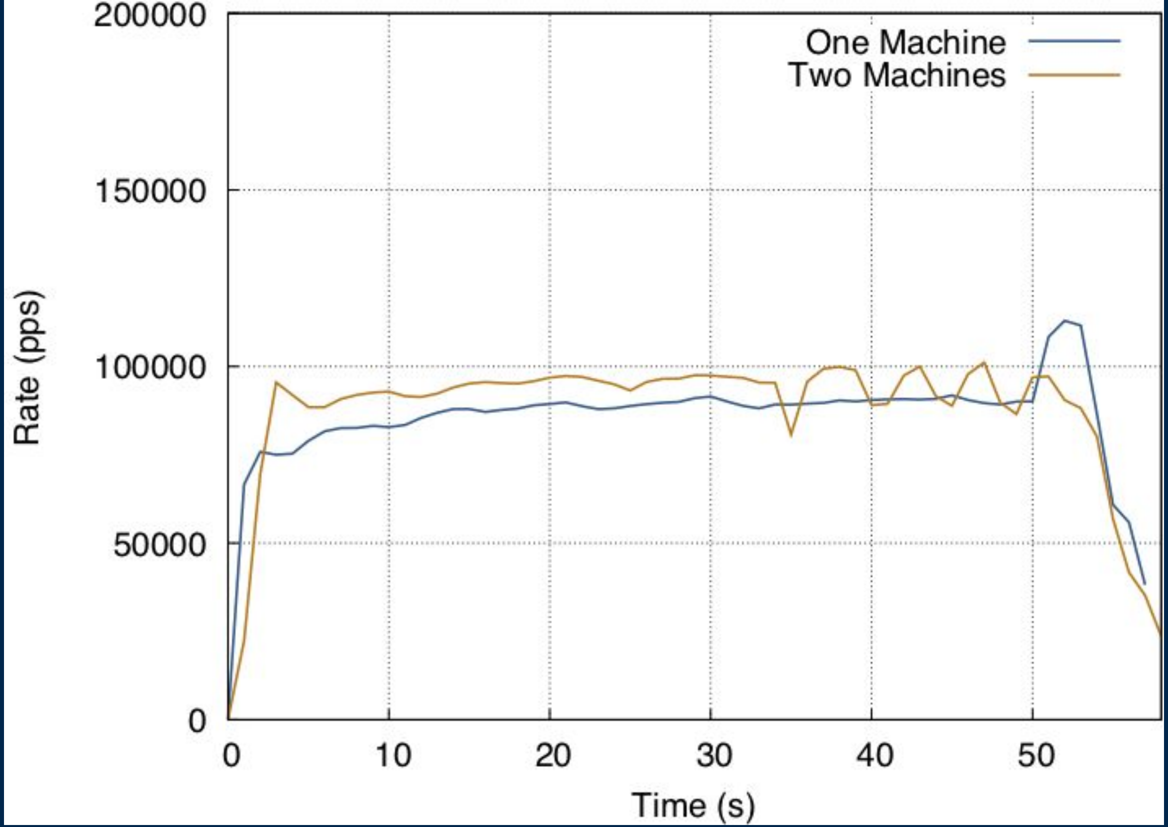
What Happened?

Local Drop?: Saw same behavior when splitting send and receive into two machines

Blocked AS?: Drop larger than any one AS, top ASes were the same

Dropped In Transit?: Results from Censys (internally, 2018) suggest drop when transiting through certain ASes.

Future Work: Look for commonality among “missing” hosts AS path



Two Machines

What Instead?

Step towards continuously understanding the behavior of individual hosts at global scale.

Current Approach

- Sub-1Gbps scanning distributed across a cluster of hosts
- Bound by application layer scanning (ZGrab)
- Still provides aggregate perspective (tens of ports on a weekly basis)

Open Problems

- Packet Drop
- Faster Application Layer Scanning
- Non-scanning approaches

MEASURING SECURE CHANNELS

UNIVERSITY OF MICHIGAN

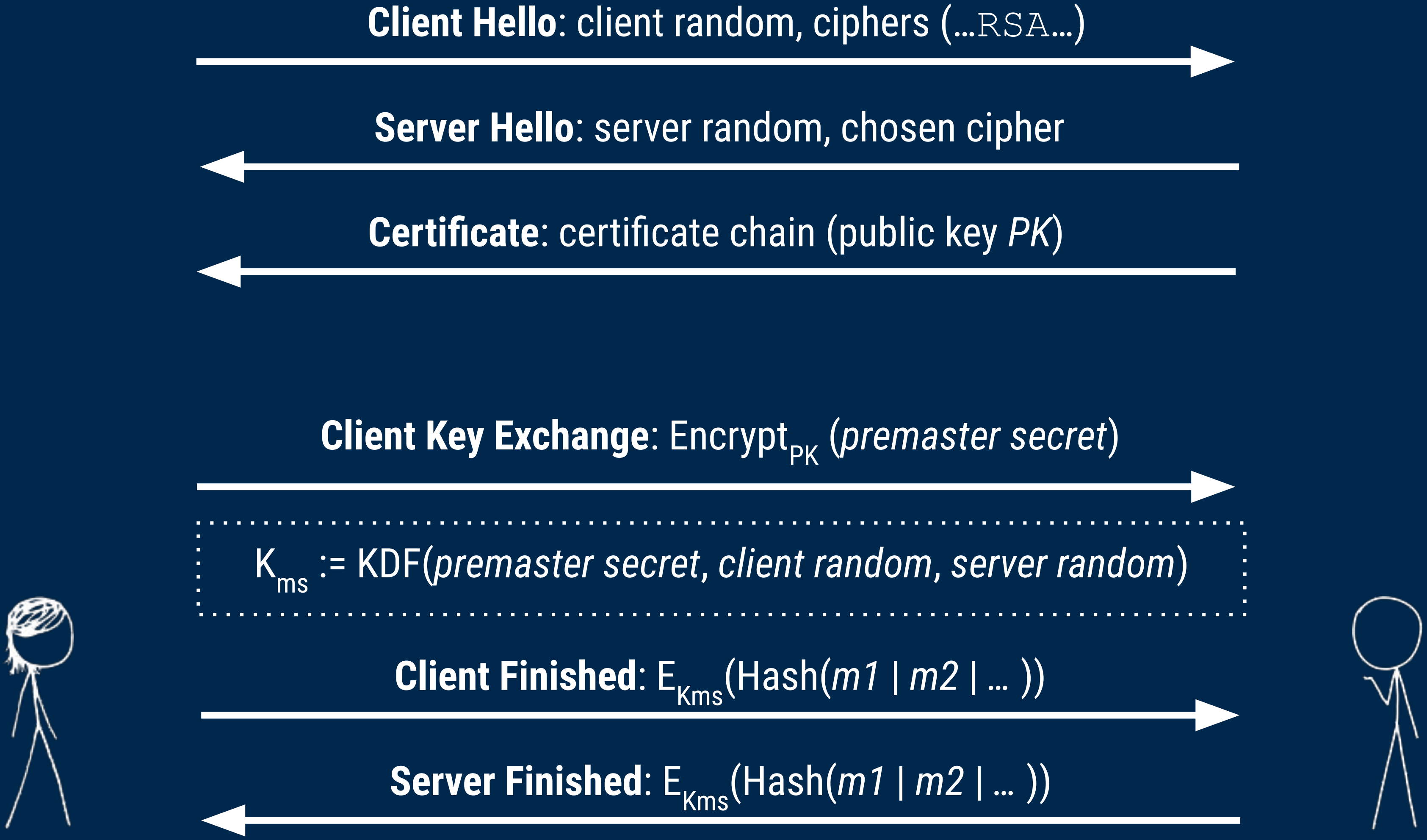
HTTPS and TLS

TLS is the protocol that provides the secure channel for HTTPS and is the most widely deployed cryptographic protocol

Billions of users per day use HTTPS/TLS in their web browser

TLS was originally built for ecommerce, but is required to have a fair and equitable web

TLS 1.2 was standardized in 2008 and was the most current version until August, 2018



Client Hello: client random, ciphers (…`RSA`…)

Server Hello: server random, chosen cipher

Certificate: certificate chain (public key *PK*)

Client Key Exchange: Encrypt$_{PK}$ (*premaster secret*)

K$_{ms}$:= KDF(*premaster secret, client random, server random*)

Client Finished: E$_{Kms}$(Hash(*m1* | *m2* | …))

Server Finished: E$_{Kms}$(Hash(*m1* | *m2* | …))

Measuring TLS

What is measurable?

- All parameterizable or optional aspects of the protocol
- “Cryptographic Agility”

What should we measure?

- Adoption / existence
- Verify assumptions
- Determine impact

MEASURING EXPORT CRYPTOGRAPHY

Export Cryptography

In the 1990s, “export” of cryptography by US persons was regulated.

International Traffic in Arms Regulations (ITAR), then Export Administration Regulations (EAR)

Ban on exporting code with a printed material exception

Export Key Length Restrictions

Regulations applied to communication with non-US entities

Public-key Cryptography: Max 512-bit public keys

- Finite Field Diffie-Hellman (key exchange)
- RSA (key exchange, encryption)

Symmetric Cryptography: Max 40-bit keys

- Block ciphers (DES)
- Stream ciphers (RC4)

Signatures and Message Authentication Codes were *unregulated*

Meanwhile...

1995 – SSLv2 designed, deployed, and deprecated

1996 – SSLv3 replaces SSLv2, forms the basis for modern TLS

1999 – TLSv1.0 standardized by the IETF

1999 – Export regulations are lifted after extensive litigation [*Bernstein v United States*]

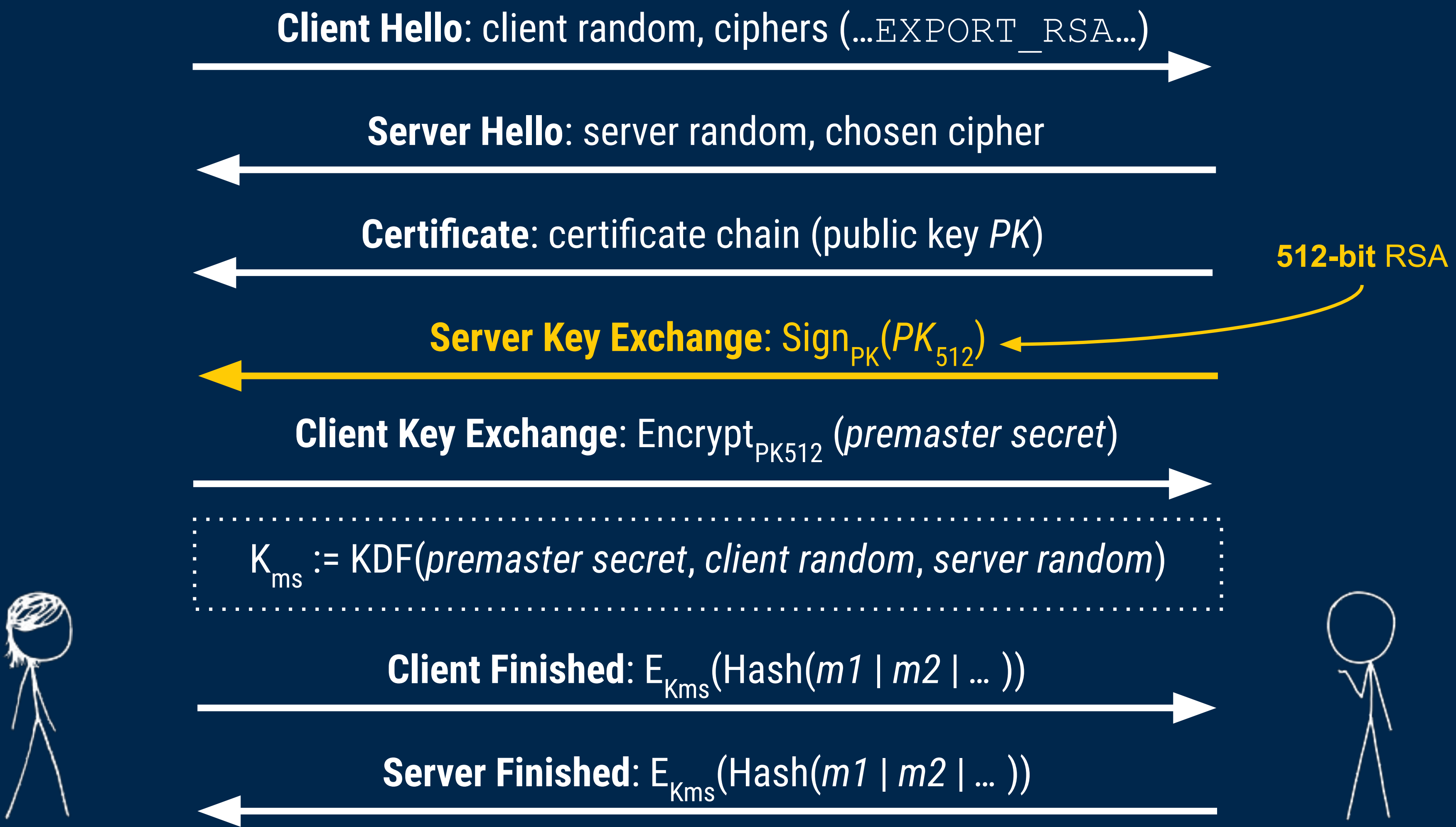
Impact on TLS

TLS 1.0 and earlier were designed with compliance mechanisms for export regulations

TLS certificates contain >512-bit RSA keys

- OK for authentication!
- *Literally Illegal* for key exchange!

Solution: “Export” Ciphers



Client Hello: client random, ciphers (…`EXPORT_RSA`…)

Server Hello: server random, chosen cipher

Certificate: certificate chain (public key *PK*)

Server Key Exchange: $\text{Sign}_{PK}(PK_{512})$

512-bit RSA

Client Key Exchange: Encrypt_{PK512} (*premaster secret*)

K_{ms} := KDF(*premaster secret*, *client random*, *server random*)

Client Finished: $E_{Kms}(\text{Hash}(m1 \mid m2 \mid \ldots))$

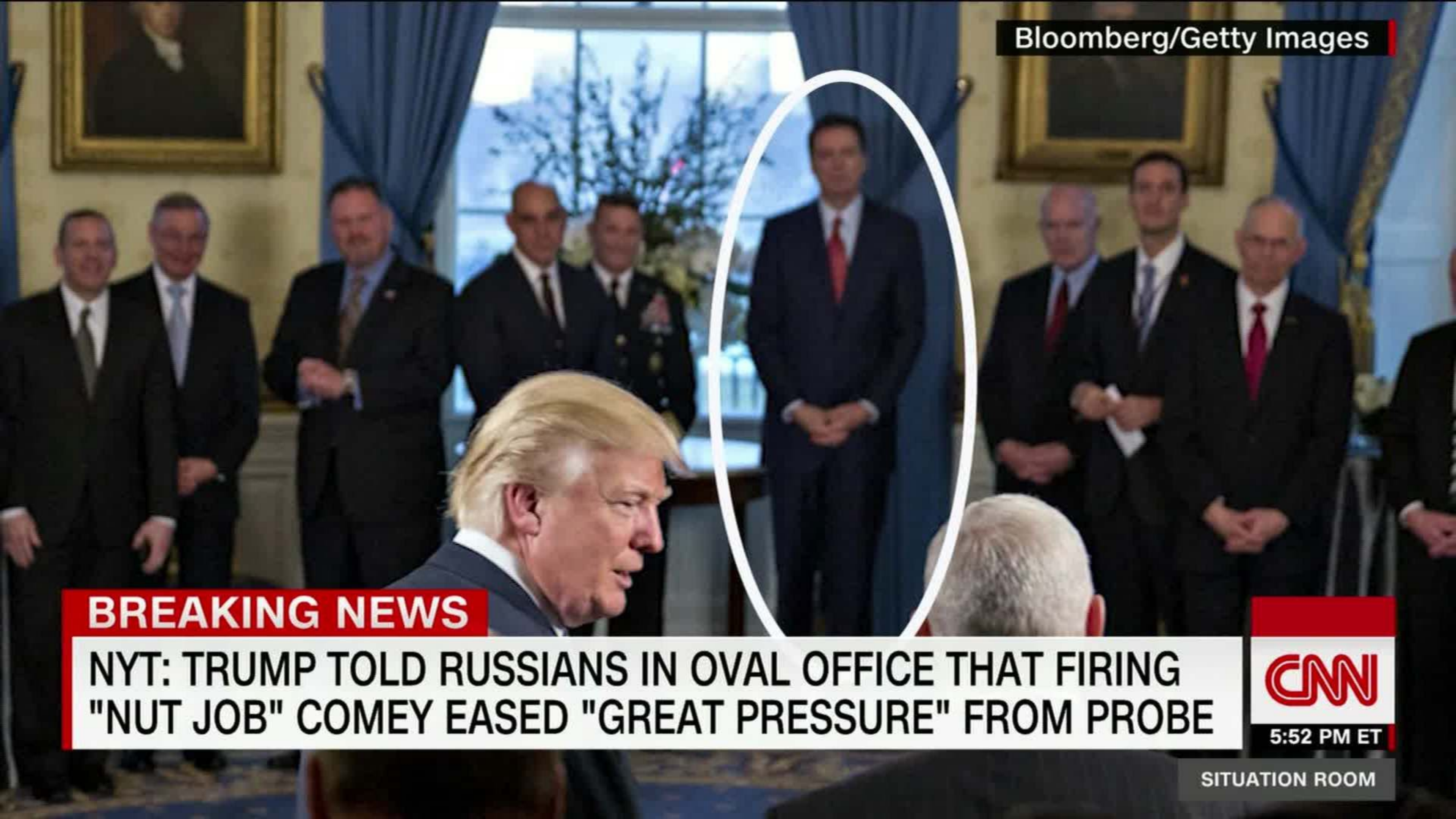
Server Finished: $E_{Kms}(\text{Hash}(m1 \mid m2 \mid \ldots))$

Factoring

Security of RSA relies and the computational hardness of factoring the public key

Computers are faster now than in 1998, how does this impact factoring?

512-bit keys can be factored in **1-2 hours for ~$100** [Valenta 2015]



Bloomberg/Getty Images

BREAKING NEWS

NYT: TRUMP TOLD RUSSIANS IN OVAL OFFICE THAT FIRING "NUT JOB" COMEY EASED "GREAT PRESSURE" FROM PROBE

“Going Dark”

Lawful access mechanisms?

“Secure golden key” for law enforcement?

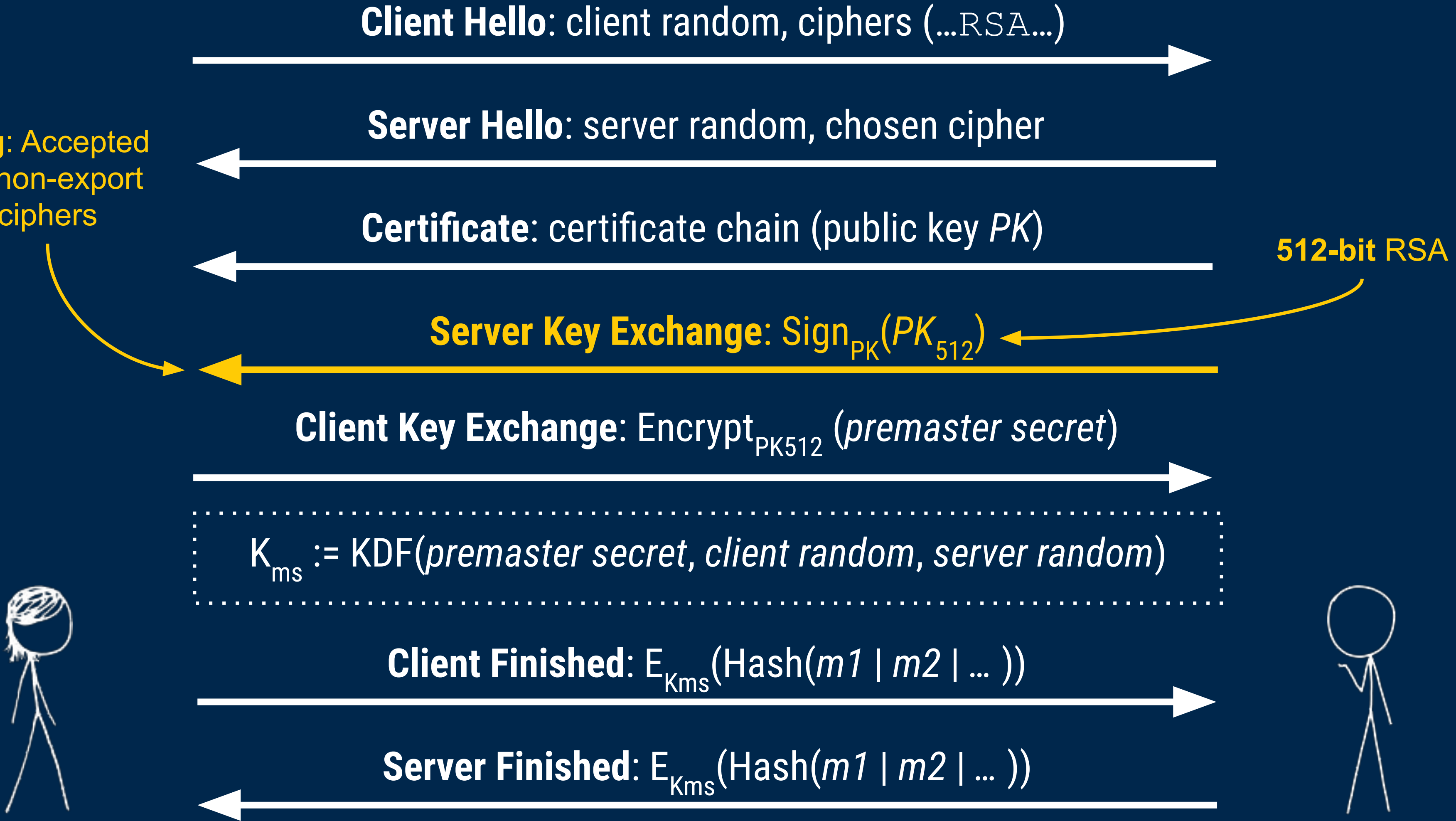
Shouldn’t the academy be able to come up with a “secure” solution for “lawful access?”, or is this fundamentally opposed to the goals of cryptography?

What happened last time and were there any lasting effects on either the protocol or the Internet?

FREAK Attack

Discovered by researchers at INRIA attempting to formally model TLS implementation state machines.

FREAK attack allows an attacker who can factor 512-bit RSA keys to man-in-the-middle TLS connections to servers that support export-grade RSA.



Client Hello: client random, ciphers (…RSA…)

Server Hello: server random, chosen cipher

Certificate: certificate chain (public key *PK*)

Server Key Exchange: $\text{Sign}_{PK}(PK_{512})$

Client Key Exchange: Encrypt_{PK512} (*premaster secret*)

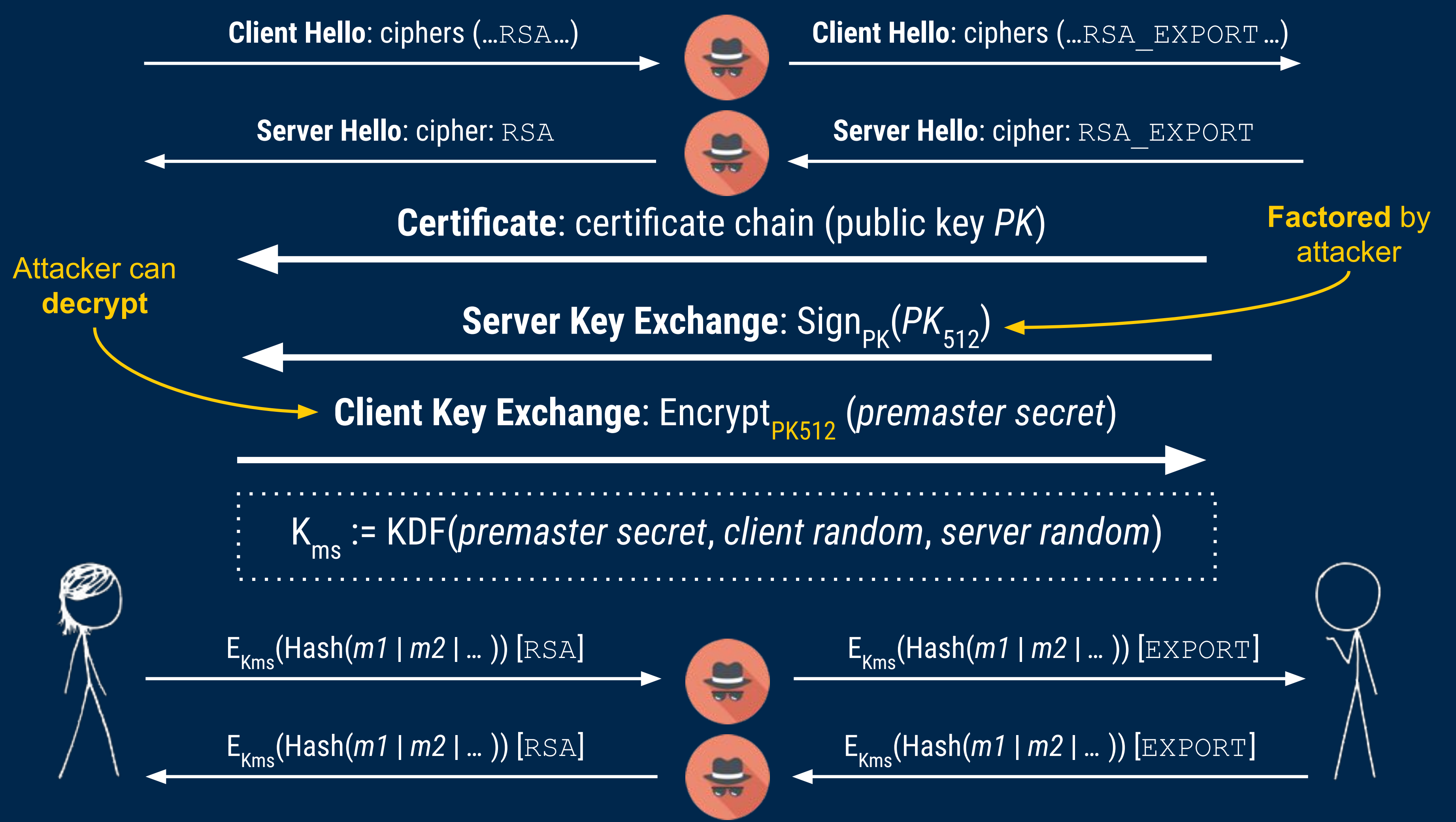
K_{ms} := KDF(*premaster secret*, *client random*, *server random*)

Client Finished: $E_{Kms}(\text{Hash}(m1 \mid m2 \mid \ldots))$

Server Finished: $E_{Kms}(\text{Hash}(m1 \mid m2 \mid \ldots))$

Bug: Accepted on non-export ciphers

512-bit RSA



Client Hello: ciphers (…RSA…)

Client Hello: ciphers (…RSA_EXPORT…)

Server Hello: cipher: RSA

Server Hello: cipher: RSA_EXPORT

Certificate: certificate chain (public key *PK*)

Factored by attacker

Attacker can **decrypt**

Server Key Exchange: $\text{Sign}_{PK}(PK_{512})$

Client Key Exchange: Encrypt_{PK512} (*premaster secret*)

K_{ms} := KDF(*premaster secret, client random, server random*)

E_{Kms}(Hash(*m1* | *m2* | …)) [RSA]

E_{Kms}(Hash(*m1* | *m2* | …)) [EXPORT]

E_{Kms}(Hash(*m1* | *m2* | …)) [RSA]

E_{Kms}(Hash(*m1* | *m2* | …)) [EXPORT]

FREAK Impact

FREAK is an **implementation bug** stemming from complexity around compliance mechanisms for export regulations

- OpenSSL (Chrome)
- Microsoft SChannel (Internet Explorer)
- Apple SecureTransport (Safari)

Modern clients are vulnerable, but FREAK *only has impact if* servers support export-grade RSA ciphers.

Measure support for export-grade RSA!

FREAK Measurements

The FREAK attack is possible when a vulnerable browser connects to a susceptible web server—a server that accepts “export-grade” encryption.

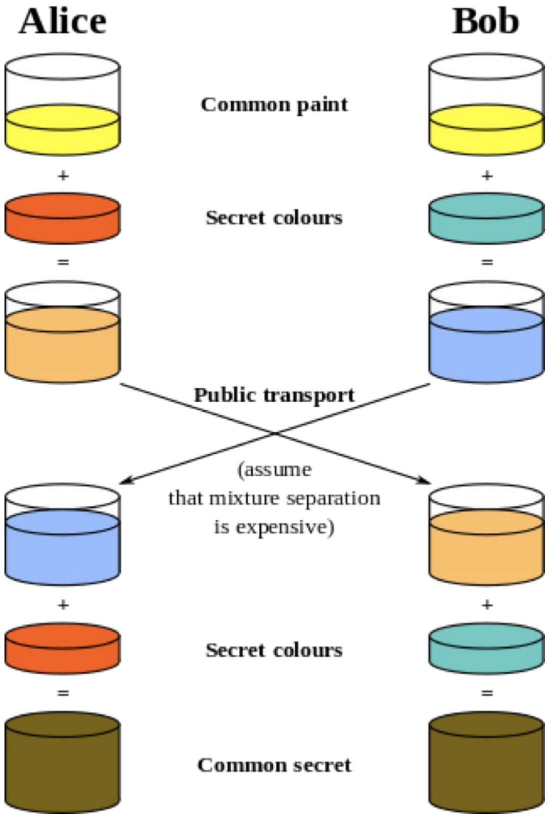
	Vulnerable at Disclosure (March 3, 2015)	**Vulnerable One Week Later (March 10, 2015)**
HTTPS Servers at Alexa Top 1M Domains	9.6%	8.5%
HTTPS servers with browser-trusted certificates	36.7%	6.5%
All HTTPS servers	26.3%	11.8%

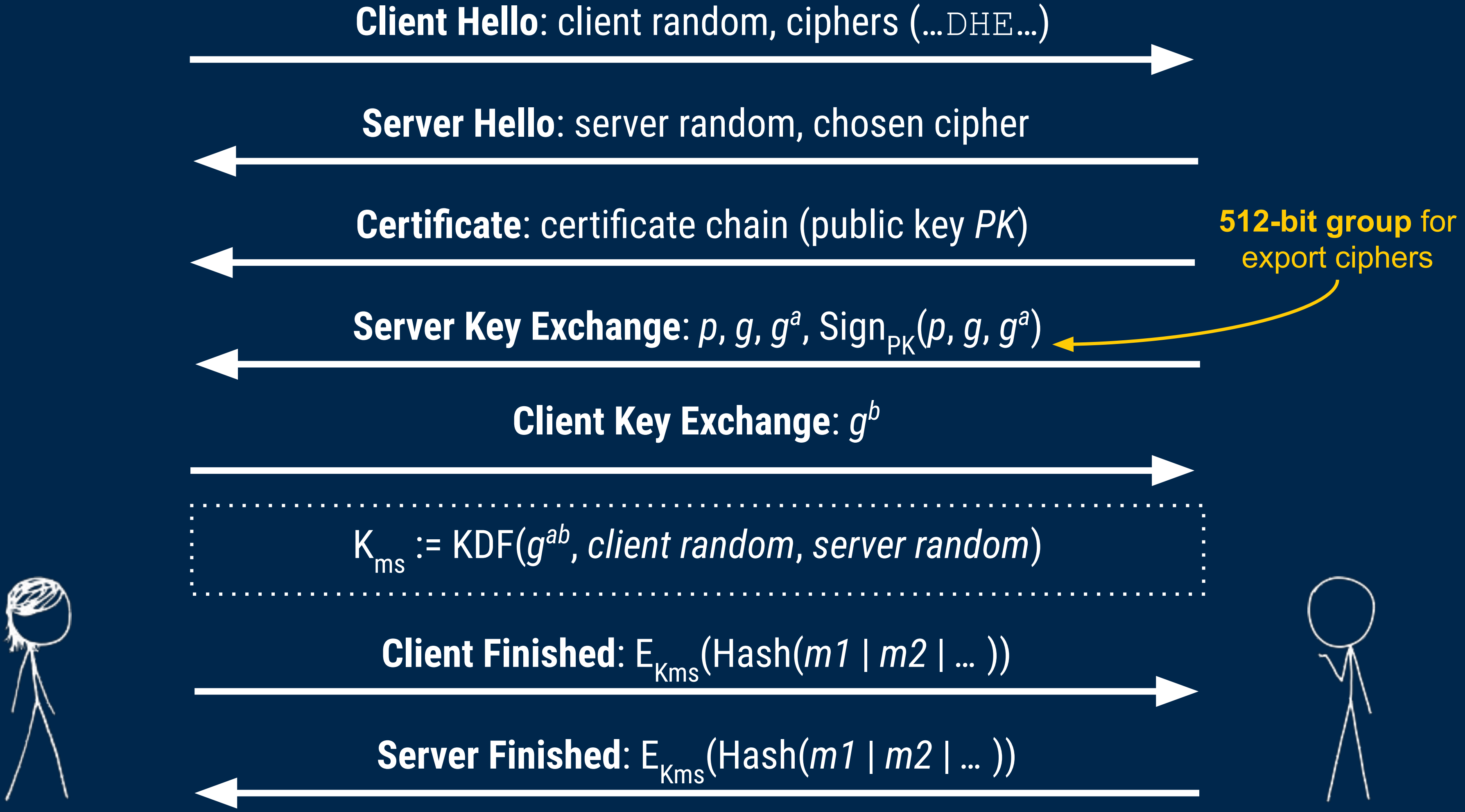
UNIVERSITY OF MICHIGAN

What About Other Export-Grade Key Exchange

TLS also contains export-grade Diffie-Hellman ciphers.

How do these ciphers work, and how hard is it to break 512-bit Diffie-Hellman today?





Client Hello: client random, ciphers (…`DHE`…)

Server Hello: server random, chosen cipher

Certificate: certificate chain (public key *PK*)

Server Key Exchange: p, g, g^a, $\text{Sign}_{PK}(p, g, g^a)$

Client Key Exchange: g^b

K_{ms} := KDF(g^{ab}, *client random*, *server random*)

Client Finished: $E_{K_{ms}}(\text{Hash}(m1 \mid m2 \mid \ldots))$

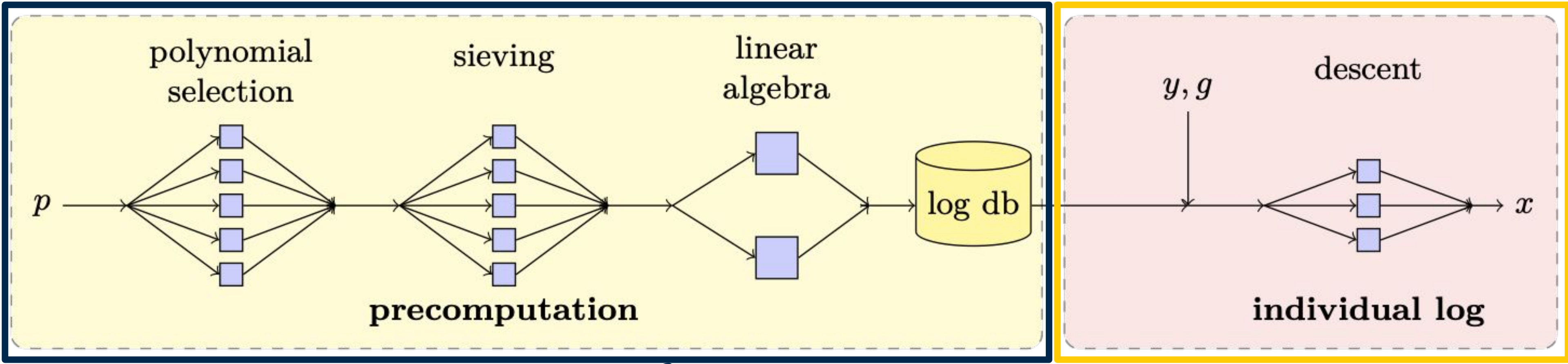
Server Finished: $E_{K_{ms}}(\text{Hash}(m1 \mid m2 \mid \ldots))$

512-bit group for export ciphers

Discrete Log

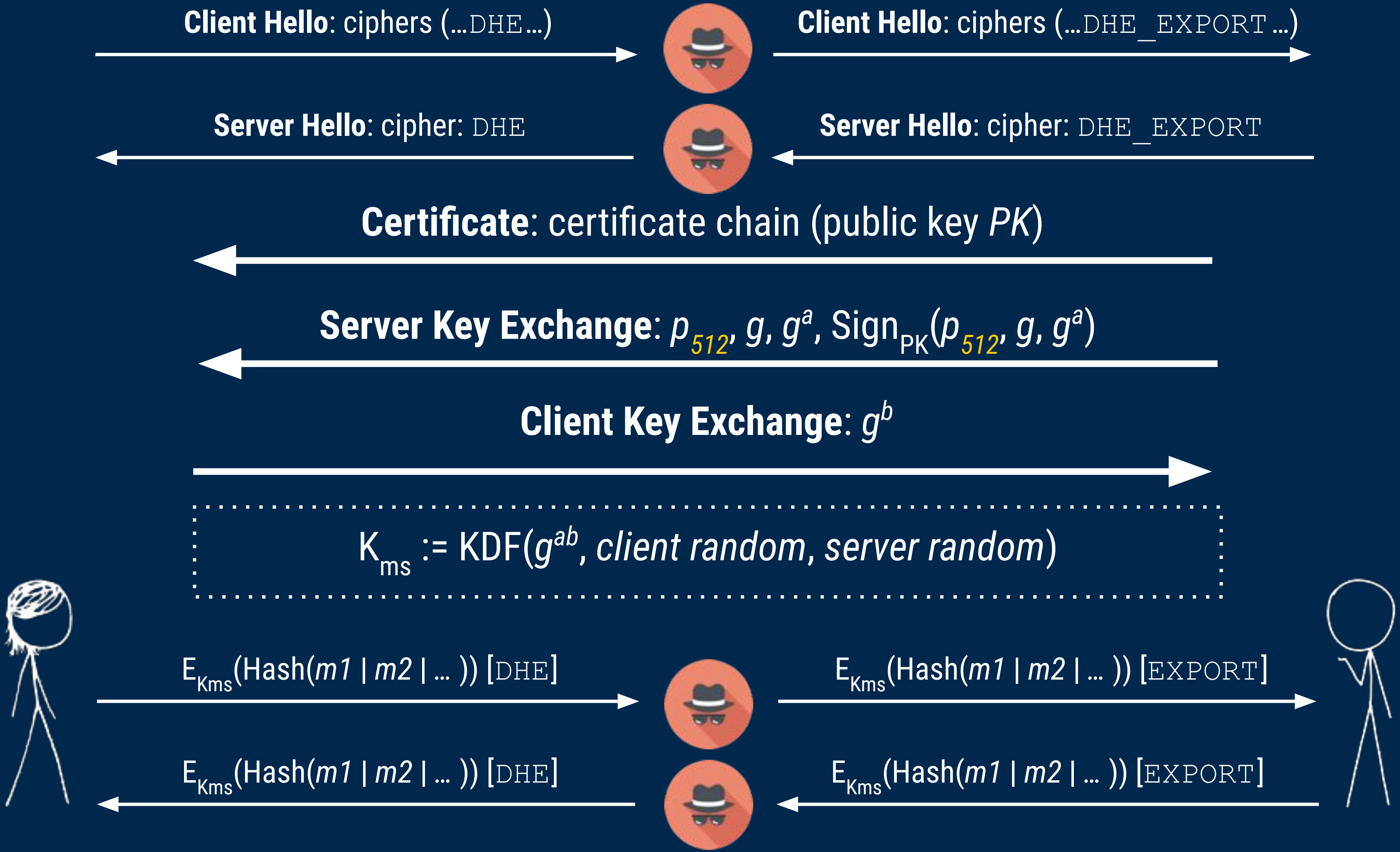
The security of Diffie-Hellman relies on the computational hardness of computing **discrete logs**, e.g. given *g*, *p*, and g^x mod *p*, calculate *x*.

The **number-field sieve** is the fastest-known algorithm for computing discrete logs.



Depends only on p, 1 week for 512-bit p

Fast



Client Hello: ciphers (…`DHE`…)

Client Hello: ciphers (…`DHE_EXPORT`…)

Server Hello: cipher: `DHE`

Server Hello: cipher: `DHE_EXPORT`

Certificate: certificate chain (public key *PK*)

Server Key Exchange: p_{512}, g, g^a, Sign$_{PK}(p_{512}, g, g^a)$

Client Key Exchange: g^b

K_{ms} := KDF(g^{ab}, *client random*, *server random*)

E_{Kms}(Hash(*m1* | *m2* | …)) [`DHE`]

E_{Kms}(Hash(*m1* | *m2* | …)) [`EXPORT`]

E_{Kms}(Hash(*m1* | *m2* | …)) [`DHE`]

E_{Kms}(Hash(*m1* | *m2* | …)) [`EXPORT`]

Logjam Empirical Questions

Do real-world servers support export Diffie-Hellman?

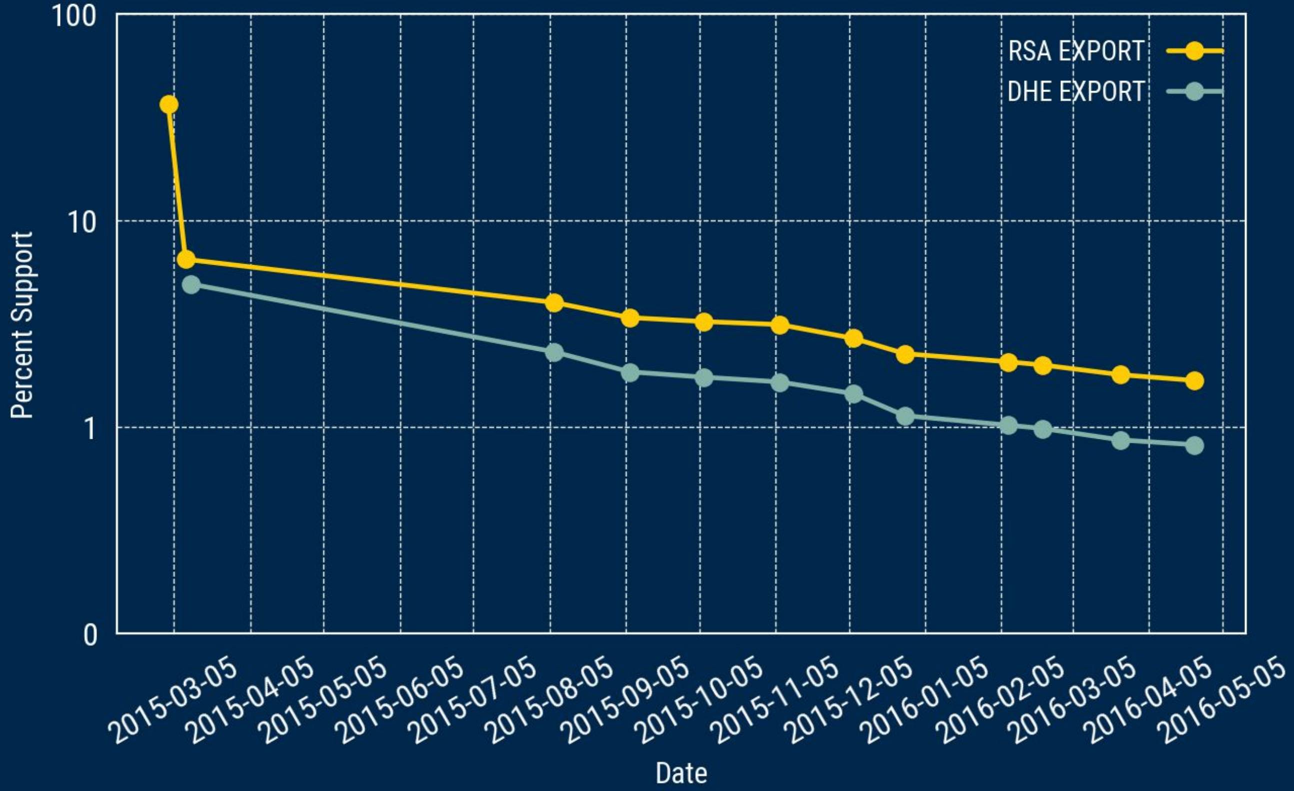
- How many trusted HTTPS hosts support export DHE? Alexa Top 1M?
- Did people disable export DHE when disabling export RSA?

Precomputation takes ~1 week and is not feasible for many unique *p*.

- How many unique 512-bit primes are used by trusted servers?
- Do implementations regenerate primes?

Answer these questions with Internet-Wide Scanning

- Implement support for Diffie-Hellman in ZGrab
- Parse out selected Diffie-Hellman parameters



Export Cipher Support Among Servers with Trusted Certificates

UNIVERSITY OF MICHIGAN

Top 1M Support

8.5% of the Alexa Top 1M supported DHE_EXPORT
3.4% of trusted IPv4 supported DHE_EXPORT

Prime	**Popularity Among Top 1M**
Apache mod_ssl	82%
nginx	10%
Other (463 primes)	8%

Measuring Export Cryptography

FREAK is an implementation vulnerability surrounding export-grade cryptography, which was only possible to exploit because of empirical properties of the Internet.

Logjam is a protocol vulnerability where the difficulty to exploit relies on empirical properties of the Internet.

Where else is there export cryptography, and does it have similar vulnerabilities?

SSLv2: Cryptography from the Deep

SSLv2 is already know to be broken.

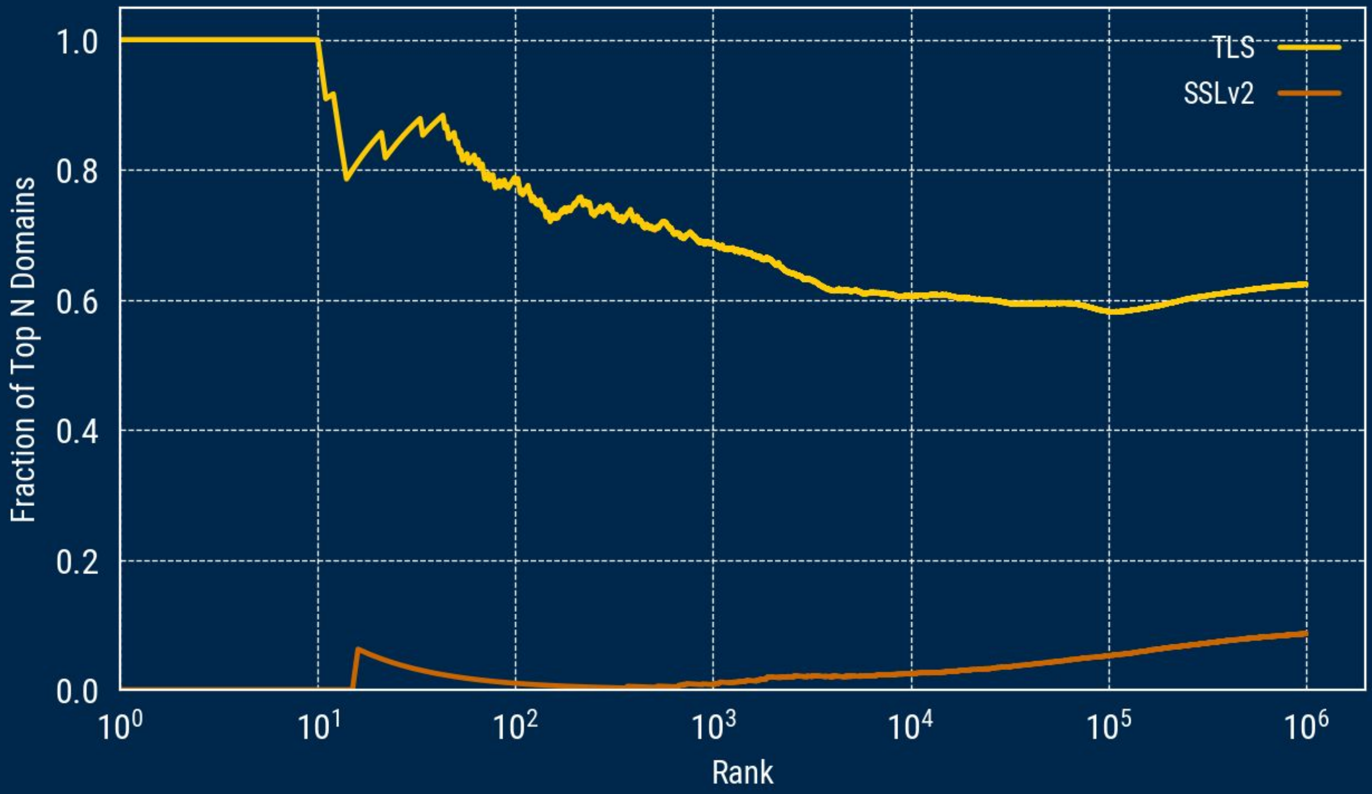
- Does not authenticate handshake
- Only used for one year (1995), officially deprecated in 2011

FREAK and Logjam show harms of supporting obsolete cryptography.

- Conventional wisdom for servers was to support all ciphers for compatibility
- This advice appears to be actively harmful

Is SSLv2 a harmless vestige, or can it be used to attack modern TLS?

- Do servers still support SSLv2? Are people actually using SSLv2?
- SSLv2 has export ciphers, does this affect modern TLS?



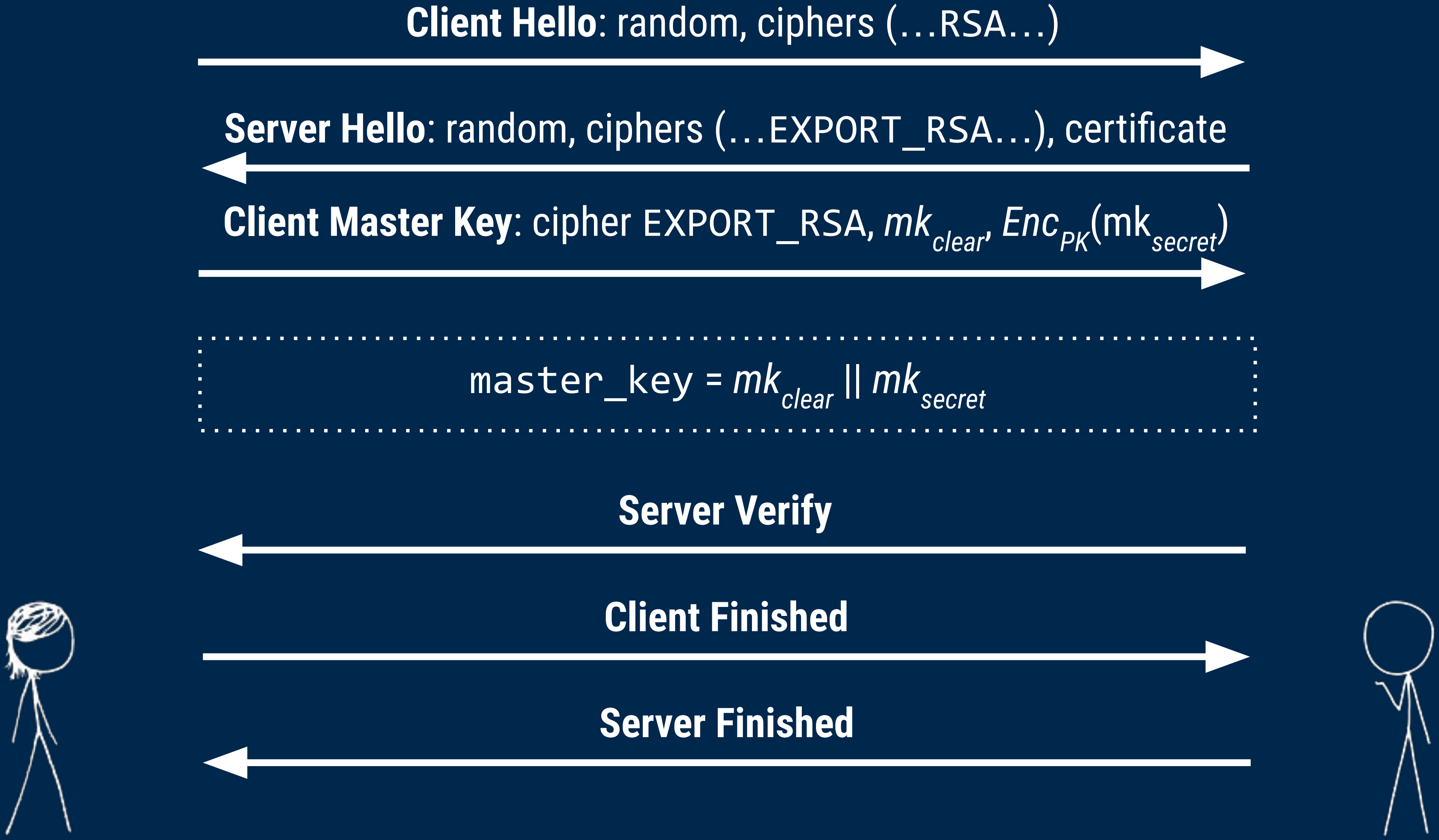
SSLv2 / TLS Support Among Top 1M Domains

		All Certificates		Trusted Certificates	
Protocol	**Port**	**TLS**	**SSLv2**	**TLS**	**SSLv2**
SMTP	25	**3,357 K**	**936 K (28%)**	**1,083 K**	**190 K (18%)**
POP3	110	4,193 K	404 K (10%)	1,787 K	230 K (13%)
IMAP	143	4,202 K	473 K (11%)	1,781 K	223 K (13%)
HTTPS	443	34,727 K	5,975 K (17%)	17,490 K	1,749 K (10%)
SMTPS	465	3,596 K	291 K (8%)	1,641 K	40 K (2%)
SMTP	587	3,507 K	423 K (12%)	1,657 K	133 K (8%)
IMAPS	993	4,315 K	853 K (20%)	1,909 K	260 K (14%)
POP3S	995	4,322 K	884 K (20%)	1,974 K	304 K (15%)

SSLv2 Support in Non-HTTPS Protocols

SSLv2 Export Ciphers

Send all but 5 bytes of the key in plaintext.



Client Hello: random, ciphers (`...RSA...`)

Server Hello: random, ciphers (`...EXPORT_RSA...`), certificate

Client Master Key: cipher `EXPORT_RSA`, mk_{clear}, $Enc_{PK}(\mathrm{mk}_{secret})$

`master_key` = $mk_{clear} \,||\, mk_{secret}$

Server Verify

Client Finished

Server Finished

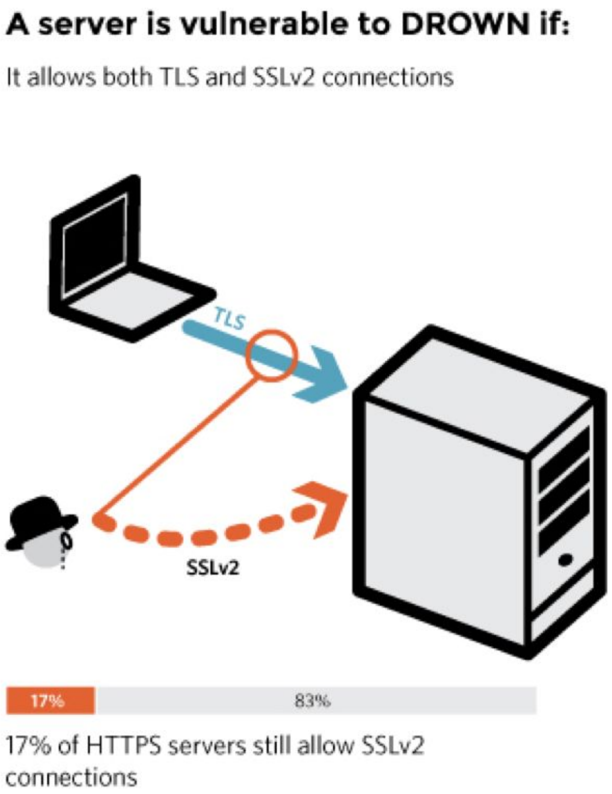
SSLv2 Oracle

Use SSLv2 as a Bleichenbacher oracle by malleating TLS ciphertexts into SSLv2 cipher texts

Options for Oracle Query:

1. Brute-force the five-byte export key
2. Exploit “extra-clear” bug in OpenSSL
3. Exploit “leaky export” bug in OpenSSL

DROWN Vulnerability



A server is vulnerable to DROWN if:

It allows both TLS and SSLv2 connections

17% of HTTPS servers still allow SSLv2 connections

		All Certificates			Trusted Certificates		
Protocol	Port	TLS	SSLv2	Vulnerable Key	TLS	SSLv2	Vulnerable Key
SMTP	25	3,347K	936K (28%)	1,666K (50%)	1,083K	190K (18%)	686K
POP3	110	4,193K	404K (10%)	1,764K (42%)	1,787K	230K (13%)	1,031K (58%)
IMAP	143	4,202K	473 K (11%)	1,759K (59%)	1,781K	223K (13%)	1,022K (58%)
HTTPS	443	34,727K	5,075K (17%)	**11,444K (33%)**	17,400K	1,749K(10%)	**3,931K (22%)**
SMTPS	465	3,596K	291K (8%)	1,439K (40%)	1,641K	40K (2%)	949K (58%)
SMTP	587	3,507K	423K (12%)	1,464K (40%)	1,657K	133K (8%)	986K (59%)
IMAPS	993	4,315K	853K (20%)	1,835K (43%)	1,909K	260K (14%)	1,119K (59%)
POP3S	995	4,322K	884K (20%)	1,919K (44%)	1,974K	304K (15%)	1,191K (60%)

Impact of Key Reuse

UNIVERSITY OF
MICHIGAN

Mitigations and Lessons

Fully disable SSLv2

- Don’t only disable export ciphers
- If only ciphers are disabled, make sure they’re actually disabled (CVE-2015-3197)

Have single-use keys

- Usually discussed in the context of signatures vs. encryption
- Prudent to use different keys across different protocol versions

Authenticate the client before sending secret-derived data

- DROWN is possible because of the early `ServerVerify` message
- Design protocols to check the client has knowledge of the secret first

All types of export cryptography have led to attacks against modern cryptography.

Empirical results were key to all export-related attacks on TLS.

Generalizing DROWN

Reusing a key across multiple TLS protocols increases attack surface

- HTTPS, SMTPS, IMAPS…
- X.509 keys are not bound to any particular port or protocol

A key compromised in one protocol can be used to attack another protocol

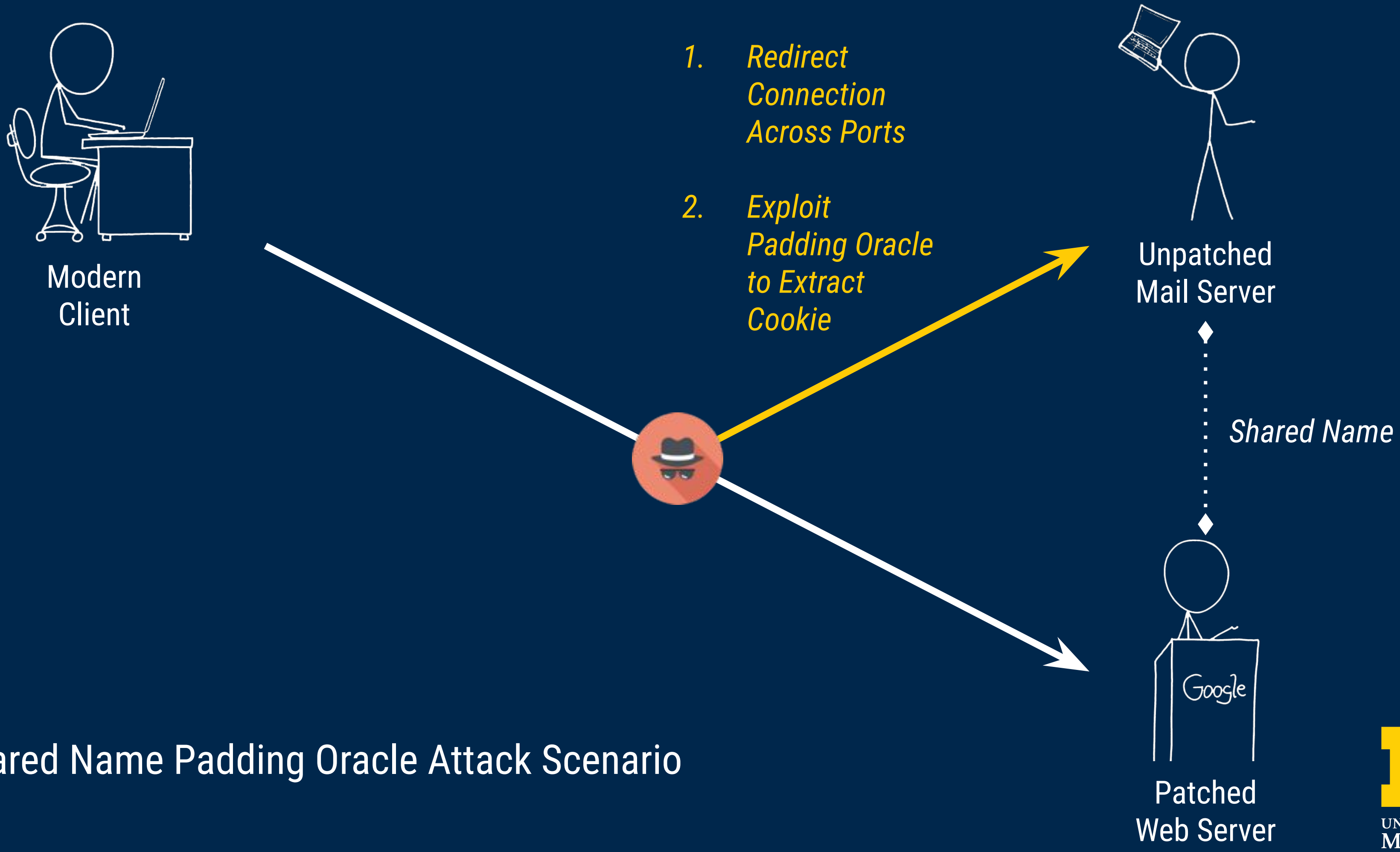
- Compromise does not need to be a TLS protocol vulnerability
- Compromise could be as simple as an RCE on a mail server

An active attacker can leverage any TLS vulnerability in one protocol to attack another

- Perform any necessary application-layer protocol handshake
- Rewrite victims client to another port

Cross-Protocol Attack Types

		Can be used to attack hosts the share a...	
Attack Type	**Example**	**...key but not name**	**...name but not key**
Key Compromise	*RCE*	Yes	Yes
Passive Attack on Handshake	*DROWN*	Yes	No
Active Attack on Handshake	*Logjam* *FREAK* *DROWN*	Likely	Yes
Post-Handshake Attack	*Padding Oracle*	No	Yes



Modern Client
1. Redirect Connection Across Ports
2. Exploit Padding Oracle to Extract Cookie
Unpatched Mail Server
Shared Name
Google
Patched Web Server

Shared Name Padding Oracle Attack Scenario

UNIVERSITY OF MICHIGAN

Future Work in Cross Protocol Attacks

The increase in attack surface is measurable.

1. Identify hosts vulnerable to TLS attacks on non-web ports
   AES-NI Padding Oracle, ROBOT, Logjam, FREAK

2. Identify HTTPS hosts
   Basic Internet-wide Scanning

3. Map between the two datasets, accounting for preconditions
   Hosts that meet preconditions for cross-protocol attack

LOOKING FORWARD

TLS 1.3

Using empirical cryptography to inform future protocol design

- Standardized in August 2018
- Informed by the last 10 years of cryptography research
- Can it avoid the pitfalls of the past?
- New “version”, but with a very new shape

TLS 1.3 Design

Named Groups

- Unify parameter selection for (EC)DHE
- Select from preset list of curves/groups, each following best practices

Limiting RSA

- No RSA key exchange
- Replace PKCS #1.5 with PSS to avoid Bleichenbacher failures

Explicit Verification

- Server signs handshake transcript with certificate key
- Prevents downgrade attacks leveraging weak session keys

TLS 1.3 Risks

Server sends secret-derived data before verifying client has knowledge of secret

- Necessary condition for DROWN-like attacks
- May lead to new types of server oracles

0-RTT mode *explicitly* does not prevent replay attacks

- Is this an application layer or protocol concern?
- Can it be leveraged for protocol-level attacks?

If you can measure it,
then it can fail.

Wireguard

Wireguard is a VPN protocol with only two cryptography-related settings

- Which `ed25519` key to use?
- Which `ed25519` keys to trust?

Handshake is not configurable, what is there to measure?

Empirical techniques let us characterize problems, but they are no substitute for avoiding problems before they happen.

Cryptography Measurement Beyond Secure Channels

Monitoring cryptocurrency peer-to-peer networks has already been used to deanonymize transactions

Post-quantum cryptography is being standardized, can we measure it as it’s being deployed?

Certificate Transparency requires a gossip protocol, can we measure its effectiveness?

Are websites that use Javascript (client-side) cryptography doing so safely?

Is the cryptography we’re using working as well as the math says that it should?

Network Measurement Beyond Cryptography

How do everyday users take advantage of all this data? Is cryptography data relevant to sysadmins?

- Measuring attack surface
- Quantifying risk
- Identifying assets

You can't protect what you don't know you own.

Can we index the devices and configurations of devices on the Internet in the same way Google indexes the content? Can we measure the Internet at the same rate it's changing? What about IPv6?

Keep your denominators consistent.

Acknowledgements

Margaret, Andrew, and Alex Adrian

Alex Halderman, Peter Honeyman, Chris Peikert, Florian Schaub, Prabal Dutta, Brian Noble

Zakir Durumeric, Pat Pannuto, Bradford Campbell, Noah Klugman, Kyle Lady

Nimrod Aviram, Bennett Black, David Bild, Matt Bernhard, Charlotte Campbell, Dave Corcoran, Robert Dick, Chris Dzombak, Adrienne Porter Felt, Branden Ghena, Adam Goodman, Alex Holland, Will Huang, James Kasten, Brian Kelly, Deepak Kumar, Zach Lanier, Elson Liu, Yue Liu, Veronica Long, Allison McDonald, Ariana Mirian, Michael Moore, Deepika Natarajan, Jon Oberheide, Lane Powell, Thomas Ptacek, Andrew Sardone, Kyle Schiller, Ryan Sleevi, Dug Song, Drew Springall, Emily Stark, Ethan Stark, George Tankersley, Ben VanderSloot, Tristan Weber, Eric Wustrow

Everyone at Censys, and many other known friends and trusted agents