Were It So Easy
Using TLS in the Real World

David Adrian
@davidcadrian
Who am I?

I’m David Adrian, a graduate student at the University of Michigan, advised by Professor J. Alex Halderman.

One of those god damn academics.
What am I talking about?

What is TLS and why study it?
What happens when TLS fails?
What can we do to prevent TLS failing in the future?
TLS provides a **secure channel** for communication that other protocols can build on top of.
Does not provide application security!
Confidentiality
Attacker cannot read messages

Integrity
Attacker cannot modify or replay messages

Authentication
Attacker cannot impersonate the recipient
TLS, SSL, HTTPS, oh my…

**SSL**: Secure Socket Layer
- Originally developed by Netscape in the 90s
- It didn’t catch on until the third version (SSLv3)

**TLS**: Transport Layer Security
- Successor to SSL (TLS v1.0 = SSLv3.1)

**HTTPS**: HTTP Secure
- HTTP wrapped in TLS
What happens when TLS fails?
In April 2014, OpenSSL disclosed a bug in their implementation of the TLS Heartbeat Extension. Vulnerability allowed attackers to dump private cryptographic keys, logins, and other private user data. Potentially affects any service that used OpenSSL for TLS — including web, mail, messaging and database servers. An estimated 24-55% of HTTPS websites were initially vulnerable.
How the Heartbleed Bug Works:

Server, are you still there?
If so, reply "potato" (6 letters).

User Meg wants these 6 letters: POTATO. User
Meg wants pages about "I'M games". Unlocking
server records with master key: 51309857334334.

Ham...

User Meg wants these 4 letters: HIRD. These are currently
connections open. User Brendan uploaded the file:

Server, are you still there?
If so, reply "hat" (500 letters).

User Meg wants these 500 letters: HAT. User
requests the "closed connections" page. User
administrator wants to set server's master key to: 1493592854. User
wants pages about "smokes but not too long".

User Meg wants these 500 letters: HAT. User
requests the "closed connections" page. User
administrator wants to set server's master key to: 14935928544. User
wants pages about "smokes but not too long".

User Meg wants these 500 letters: HAT. User
requests the "closed connections" page. User
administrator wants to set server's master key to: 14935928544. User
wants pages about "smokes but not too long".

User Meg wants these 500 letters: HAT. User
requests the "closed connections" page. User
administrator wants to set server's master key to: 14935928544. User
wants pages about "smokes but not too long".

User Meg wants these 500 letters: HAT. User
requests the "closed connections" page. User
administrator wants to set server's master key to: 14935928544. User
wants pages about "smokes but not too long".
What can we **learn** about TLS from Heartbleed?

How can we **measure the impact** of Heartbleed on TLS a whole?
Network port scanner capable of completing an Internet-wide TCP SYN scan, in **under five minutes** on a 10G uplink, and under forty-five minutes on a 1G uplink, from a single machine.

We use ZMap to identify and connect to **all HTTPS servers** on the Internet.
Data Collection
- Began scanning 48 hours after public disclosure
- Scanned Alexa Top 1 Million and 1% samples of IPv4 every 8 hours

Scanning for Heartbleed
- Instead of exploiting the vulnerability, we checked for non-compliant behavior of vulnerable OpenSSL version

https://zmap.io/heartbleed
Vulnerable Percentage of HTTPS Hosts

Date

Alexa Top 1 Million Sites

Public IPv4 Address Space
First Evidence of Attacks
What happens if you email everybody on the Internet who is vulnerable and tell them to patch?
What happens if you email everybody on the Internet who is vulnerable and tell them to patch?

*Only two people threaten to sue you!*
47% increase in patching

April 28th Notifications (Group A)
May 7th Notifications (Group B)
How do we **better defend** TLS in the future? Can we use ZMap and measurement to proactively identify new ways that TLS can fail in practice?
How do we **better defend** TLS in the future? Can we use ZMap and measurement to proactively identify new ways that TLS can fail in practice?

**Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice**

David Adrian\(^*\) Karthikeyan Bhargavan\(^*\) Zakir Durumeric\(^*\) Pierrick Gaudry\(^{\dagger}\) Matthew Green\(^{\dagger}\) J. Alex Halderman\(^{\dagger}\) Nadia Heninger\(^{\ddagger}\) Drew Springall\(^{\dagger}\) Emmanuel Thomé\(^{\dagger}\) Luke Valenta\(^{\dagger}\) Benjamin VanderSloot\(^{\dagger}\) Eric Wustrow\(^{\dagger}\) Santiago Zanella-Béguelin\(^{\ddagger}\) Paul Zimmermann\(^{\dagger}\)

\(^*\) INRIA Paris-Rocquencourt  \(^{\dagger}\) INRIA Nancy-Grand Est, CNRS and Université de Lorraine  \(^{\ddagger}\) Johns Hopkins  \(^{\#}\) University of Michigan

For additional materials and contact information, visit WeakDH.org.

**ABSTRACT**

We investigate the security of Diffie-Hellman key exchange as used in popular Internet protocols and find it to be less secure than widely believed. First, we present a novel flaw in TLS that allows a man-in-the-middle to downgrade connections to “export-grade” Diffie-Hellman. To carry out this attack, we implement the number field sieve discrete log algorithm. After a week-long precomputation for a specified 512-bit group, we can compute arbitrary discrete logs in this group in minutes. We find that 92% of vulnerable servers use a single 512-bit group, allowing us to compromise connections to 7% of Alexa Top Million HTTPS sites. In response, major browsers are being changed to reject short groups.

We go on to consider Diffie-Hellman with 768- and 1024-bit logs in that group, amortizing the cost over all targets that share this parameter. The algorithm can be tuned to reduce individual log cost even further. Although this fact is well known among mathematical cryptographers, it seems to have been lost among practitioners deploying cryptosystems. We exploit it to obtain the following results:

*Active attacks on export ciphers in TLS.* We identify a new attack on TLS, in which a man-in-the-middle attacker can downgrade a connection to export-grade cryptography. This attack is reminiscent of the FREAK attack [6], but applies to the ephemeral Diffie-Hellman ciphersuites and is a TLS protocol flaw rather than an implementation vulnerability.

We present measurements that show that this attack applies to 8.4% of Alexa Top Million HTTPS sites and 3.4% of all Alexa sites.
Diffie-Hellman

Two parties agree on a secure, secret key over an insecure channel.

Only share public data, agree on shared private secret.

Eavesdropper cannot determine the secret key.

Leverages a “hard” mathematical problem known as discrete log.
Alice and Bob agree on two numbers, $p$ and $g$, where $p$ is prime. Alice picks a secret $a$, and Bob picks $b$. 

![Diagram showing Alice and Bob](image-url)
Alice and Bob agree on two numbers, $p$ and $g$, where $p$ is prime. Alice picks a secret $a$, and Bob picks $b$.

$$A = g^a \mod p$$
Alice and Bob agree on two numbers, $p$ and $g$, where $p$ is prime. Alice picks a secret $a$, and Bob picks $b$.

\[ A = g^a \mod p \]
\[ B = g^b \mod p \]
Alice and Bob agree on two numbers, \( p \) and \( g \), where \( p \) is prime. Alice picks a secret \( a \), and Bob picks \( b \).

\[
A = g^a \mod p \\
B = g^b \mod p \\
s = B^a \mod p \\
s = A^b \mod p
\]
Alice and Bob agree on two numbers, $p$ and $g$, where $p$ is prime. Alice picks a secret $a$, and Bob picks $b$. 

$$A = g^a \mod p$$

$$B = g^b \mod p$$

$$s = B^a \mod p$$

$$s = A^b \mod p$$

$$g^{ab} \mod p$$

$$g^{ab} \mod p$$
Why does it work?

To break Diffie-Hellman, you need to be able to calculate $g^{ab} \mod p$, given only $g^a \mod p$ and $g^b \mod p$

This is the **discrete log** problem. Mathematicians tell us this is hard and infeasible to compute, so long as $p$ is **sufficiently large**.

Recommended to use 2048-bit primes or higher.
You can use the **number-field sieve** algorithm to calculate discrete log and break Diffie-Hellman.

It turns out the algorithm almost entirely depends on $p$, not $a$ or $b$.

Feasible for academics to break **512-bit** primes.
Certificate

The identity of DUO SECURITY, INC. at Ann Arbor, Michigan US has been verified by DigiCert SHA2 Extended Validation Server CA and is publicly auditable.

Certificate Information

Your connection to www.duosecurity.com is encrypted with modern cryptography.

The connection uses TLS 1.2.

The connection is encrypted and authenticated using AES_128_GCM and uses ECDHE_RSA as the key exchange mechanism.

Site information

You have never visited this site before today.

What do these mean?

Learn More

What is Platform?
Certificate

Cipher Suite

The identity of DUO SECURITY, INC. at Ann Arbor, Michigan US has been verified by DigiCert SHA2 Extended Validation Server CA and is publicly auditable.

Certificate Information

Your connection to www.duosecurity.com is encrypted with modern cryptography.

The connection uses TLS 1.2.

The connection is encrypted and authenticated using AES_128_GCM and uses ECDHE_RSA as the key exchange mechanism.

Site information

You have never visited this site before today.

What do these mean?

Learn More

What is Platform?
Were It So Easy: Using TLS in the Real World ▪ David Adrian
Were It So Easy: Using TLS in the Real World

David Adrian

Client

Client Random
Client Cipher Suites

Client Hello

Server

Client Random
Client Cipher Suites

Server Random
Selected Cipher Suite

Server Certificate
Server Public Key
Were It So Easy: Using TLS in the Real World

David Adrian

Client

- Client Random
- Client Cipher Suites

Server Random
- Selected Cipher Suite
- Server Certificate
- Server Public Key

Server

- Client Hello
- Server Random
- Selected Cipher Suite
- Server Certificate
- Server Public Key

Client Hello

Server Hello

Server Certificate
Were It So Easy: Using TLS in the Real World

David Adrian

Client

Client Random
Client Cipher Suites

Server Random
Selected Cipher Suite

Server Certificate
Server Public Key

Server

Client Hello

Server Hello

Server Certificate

Server Random
Selected Cipher Suite

Server Certificate
Server Public Key

DH Params \( g, p \)

Server Public DH Value \( Y_s \)

Signature

Server Private DH Value \( X_s \)
Client

- Client Random
- Client Cipher Suites
- Server Random
- Selected Cipher Suite
- Server Certificate
- Server Public Key
- DH Params $g, p$
- Server Public DH Value $Y_s$
- Signature

Client Hello

Server

- Server Random
- Selected Cipher Suite
- Server Certificate
- Server Public Key
- DH Params $g, p$
- Server Public DH Value $Y_s$
- Signature
- Server Private DH Value $X_s$

Server Hello

Server Certificate

Server Key Exchange
Were It So Easy: Using TLS in the Real World

David Adrian

Client

- Client Random
- Client Cipher Suites

Server Random
- Selected Cipher Suite
- Server Certificate
- Server Public Key
- DH Params $g, p$
- Server Public DH Value $Y_s$
- Signature
- Client Private DH Value $X_c$

Server

- Client Hello
- Server Hello
- Server Certificate
- Server Key Exchange
- DH Params $g, p$
- Server Public DH Value $Y_s$
- Signature
- Server Private DH Value $X_s$
Client

- Client Random
- Client Cipher Suites
- Server Random
- Selected Cipher Suite
- Server Certificate
- Server Public Key
- DH Params $g, p$
- Server Public DH Value $Y_s$
- Signature
- Client Private DH Value $X_c$
- Client Public DH Value $Y_c$

Server

- Client Hello
- Server Hello
- Server Certificate
- Server Key Exchange
- DH Params $g, p$
- Server Public DH Value $Y_s$
- Signature
- Server Private DH Value $X_s$
- Client Public DH Value $Y_c$
Were It So Easy: Using TLS in the Real World

David Adrian

Client

Server

Client Random
Client Cipher Suites

Server Random
Selected Cipher Suite

Server Certificate
Server Public Key

DH Params $g, p$

Server Public DH Value $Y_s$

Signature

Client Private DH Value $X_c$

Client Public DH Value $Y_c$

Premaster Secret
Session Key

Encrypted Hash

Finished

Client Hello

Server Hello

Server Certificate

Server Certificate

Server Key Exchange

Client Key Exchange

Premaster Secret
Session Key

Encrypted Hash

Server Private DH Value $X_s$
“Surely, no one uses 512-bit primes?”
Export Ciphers

Remnant of the 90s “crypto wars”

It used to be illegal to export “strong crypto” outside of the United States

9th Circuit Court overturned the law in *Bernstein vs. United States of America*

TLS was designed before the law was overturned

Included weak (short-key) “export ciphers” for use outside of the United States, e.g. DHE_EXPORT
Good!

Bad!

Non-Ephemeral

Export

Ephemeral

Non-Export
Were It So Easy: Using TLS in the Real World

David Adrian

Non-Export

Non-Ephemeral

Export

Ephemeral

✗ Bad!

✗ Bad!
Logjam

Logjam is a **downgrade attack against TLS** that enables a man-in-the-middle to read and modify data passed over the connection.

Logjam affects any server that supports **DHE_EXPORT** ciphers.
Were It So Easy: Using TLS in the Real World

- David Adrian

**Diagram:**
- **Client**
- **MitM**
- **Server**
Were It So Easy: Using TLS in the Real World • David Adrian

Client

- Client Random
- Client Cipher Suites
- Server Random
- Selected Cipher Suite
- Server Certificate
- Server Public Key
- DH Params \( g, p \)
- Server Public DH Value \( Y_s \)
- Signature

MitM

- Client Hello
  - Non-export
- Server Hello
  - DHE

Server

- Client Random
- Client Cipher Suites
- Server Random
- Selected Cipher Suite
- Server Certificate
- Server Public Key
- DH Params \( g, p \)
- Server Public DH Value \( Y_s \)
- Signature
- Server Key Exchange
  - Export-Grade
- Server Private DH Value \( X_s \)
Client

- Client Random
- Client Cipher Suites
- Server Random Selected Cipher Suite
- Server Certificate Server Public Key
- DH Params $g, p$ Server Public DH Value $Y_s$ Signature
- Client Private DH Value $X_c$
- Client Public DH Value $Y_c$

MitM

- Client Hello Non-export
- Server Hello DHE

Server

- Server Random
- Server Cipher Suites
- Server Certificate Server Public Key
- Server Key Exchange Export-Grade
- Client Private DH Value $X_c$
- Client Public DH Value $Y_c$
Were It So Easy: Using TLS in the Real World

David Adrian

Client

- Client Random
- Client Cipher Suites

Server

- Server Random
- Selected Cipher Suite
- Server Certificate
- Server Public Key
- DH Params $g, p$
- Server Public DH Value $Y_s$
- Signature
- Server Private DH Value $X_s$
- Server Public DH Value $Y_c$
- Premaster Secret
- Session Key

MitM

- Client Hello
  - Non-export
- Server Hello
  - DHE
- Server Certificate
- Server Key Exchange
  - Export-Grade

Premaster Secret

Session Key
Were It So Easy: Using TLS in the Real World ▪︎ David Adrian

Client
- Client Random
- Client Cipher Suites
- Server Random
- Selected Cipher Suite
- Server Certificate
- Server Public Key
- DH Params \( g, p \)
- Server Public DH Value \( Y_s \)
- Signature
- Client Private DH Value \( X_c \)
- Client Public DH Value \( Y_c \)
- Premaster Secret
- Session Key
- Encrypted Hash

MitM
- Client Hello
  - Non-export
- Server Hello
  - DHE
- Server Hello
  - Export DHE

Server
- Server Random
- Selected Cipher Suite
- Server Certificate
- Server Public Key
- DH Params \( g, p \)
- Server Public DH Value \( Y_s \)
- Signature
- Server Private DH Value \( X_s \)
- Client Public DH Value \( Y_c \)
- Premaster Secret
- Session Key
- Encrypted Hash

Finished
- DHE
- Export DHE
“Surely, no one uses 512-bit primes?”

“Surely, no one uses export ciphers?”
In April 2015, 8.4% of the Top 1 Million HTTPS domains support DHE_EXPORT. Of these domains, 82% used the most common prime, and 10% used the second most common prime.

We carried out the precomputation on these primes in 7 days each, enabling us to break single connections in under two minutes.
“Surely, no one uses 512-bit primes?”

“Surely, no one uses export ciphers?”

“Surely, no one is actually exploiting this?”
What if we could break **1024-bit** Diffie-Hellman?

Regular, non-export connections already uses 1024-bit primes. No need to downgrade, just passively decrypt!
<table>
<thead>
<tr>
<th>Prime Length</th>
<th>Could Be Broken By</th>
<th>Precomputation Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>512 bits</td>
<td>Academics</td>
<td>7 days</td>
</tr>
<tr>
<td>768 bits</td>
<td>Academics</td>
<td>~1 month</td>
</tr>
<tr>
<td>1024 bits</td>
<td>Nation State Large Organization</td>
<td>~1 year ~$100-300 million</td>
</tr>
<tr>
<td>Prime Length</td>
<td>Could Be Broken By</td>
<td>Precomputation Time</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>512 bits</td>
<td>Academics</td>
<td>7 days</td>
</tr>
<tr>
<td>768 bits</td>
<td>Academics</td>
<td>~1 month</td>
</tr>
<tr>
<td>1024 bits</td>
<td>Nation State Large Organization</td>
<td>~1 year ~$100-300 million</td>
</tr>
</tbody>
</table>
"Surely, no one uses 512-bit primes?"

"Surely, no one uses export ciphers?"

"Surely, no one is actually exploiting this?"

"Surely, people are using more than one prime?"
<table>
<thead>
<tr>
<th></th>
<th>Top 1024-bit Prime</th>
<th>Top Ten 1024-bit Primes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HTTPS Top 1M</strong></td>
<td>205K (37.1%)</td>
<td>309K (56.1%)</td>
</tr>
<tr>
<td><strong>HTTPS All</strong></td>
<td>1.8M (12.8%)</td>
<td>3.4M (23.8%)</td>
</tr>
<tr>
<td><strong>SSH</strong></td>
<td>3.6M (25.7%)</td>
<td>3.6M (25.7%)</td>
</tr>
<tr>
<td><strong>IKE (VPN)</strong></td>
<td>1.7M (66.1%)</td>
<td>1.7M (66.1%)</td>
</tr>
<tr>
<td></td>
<td>Top 1024-bit Prime</td>
<td>Top Ten 1024-bit Primes</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td><strong>HTTPS Top 1M</strong></td>
<td>205K (37.1%)</td>
<td>309K (56.1%)</td>
</tr>
<tr>
<td><strong>HTTPS All</strong></td>
<td>1.8M (12.8%)</td>
<td>3.4M (23.8%)</td>
</tr>
<tr>
<td><strong>SSH</strong></td>
<td>3.6M (25.7%)</td>
<td>3.6M (25.7%)</td>
</tr>
<tr>
<td><strong>IKE (VPN)</strong></td>
<td>1.7M (66.1%)</td>
<td>1.7M (66.1%)</td>
</tr>
<tr>
<td>Protocol</td>
<td>Top 1024-bit Prime</td>
<td>Top Ten 1024-bit Primes</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>HTTPS Top 1M</td>
<td>205K (37.1%)</td>
<td>309K (56.1%)</td>
</tr>
<tr>
<td>HTTPS All</td>
<td>1.8M (12.8%)</td>
<td>3.4M (23.8%)</td>
</tr>
<tr>
<td>SSH</td>
<td>3.6M (25.7%)</td>
<td>3.6M (25.7%)</td>
</tr>
<tr>
<td>IKE (VPN)</td>
<td>1.7M (66.1%)</td>
<td>1.7M (66.1%)</td>
</tr>
</tbody>
</table>
“Also, we are investing in groundbreaking cryptanalytic capabilities to defeat adversarial cryptography and exploit Internet traffic”

–NSA, 2013
“Also, we are investing in groundbreaking cryptanalytic capabilities to defeat adversarial cryptography and exploit Internet traffic”

–NSA, 2013

Also, the US government has a $10 billion budget just to break crypto.
4. Communicate Results

Can we decrypt the VPN traffic?

- If the answer is “No” then explain how to turn it into a “YES!”
- If the answer is “YES!” then…
Happy Dance!!
Turn that Frown Upside Down! From “No” to “YES!”

- Depends on why we couldn’t decrypt it
- Find Pre-Shared Key
- Locate complete paired collect
- Locate both IKE and ESP traffic
- Have collection sites do surveys for the IP’s
- Find better quality collect with rich metadata
Passive decryption of VPN connections using a broken 1024-bit prime is consistent with Snowden documents.
Mitigations and Lessons

Transition to elliptic curve cryptography (ECC)

If ECC isn’t an option, use 2048-bit primes or larger

If 2048-bit isn’t an option, use a fresh 1024-bit prime

All major desktop browsers now reject 512-bit groups, and are sunsetting 768-bit and 1024-bit

Turn export ciphers off!
Questions?

I’m David Adrian, a graduate student at the University of Michigan

@davidcadrian

https://weakdh.org

https://zmap.io

https://davidadrian.org