HOW CRYPTO FAILS IN PRACTICE

GRAD SEC
OCT 31 2017
Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice

David Adrian
Karthikeyan Bhargavan
Zehra Durumeric
Pier Maria Gaudry
Matthew Green
J. Alex Halderman
Nadia Heninger
Drew Springall
Emmanuel Thomé
Luke Valenta
Benjamin VanderSloot
Eric Wustrow
Santiago Zanella-Béguelin
Paul Zwicky
NRAI Paris Roquetaillade
NRAI Nancy Grand Est, ONR, and Université de Lorraine
Microsoft Research
University of California, Berkeley
University of Michigan
For additional material and contact information, visit WeakDH.org

ABSTRACT

We investigate the security of the Diffie-Hellman key exchange as used in popular browser protocols and find it to be less secure than widely believed. First, we present a novel flaw in the way weak aleph parameters are used by the DHE-Diffie-Hellman protocol. To carry out this attack, we employ the number field sieve integer factorization algorithm. After a week-long pre-computation for a specific weak parameter, we were able to break DHE-Diffie-Hellman. To carry out this attack, we employ the number field sieve integer factorization algorithm and use a single $128$-bit group. Using a small key size, we demonstrate that the use of weak parameters is not a secure choice for key exchange. Our work highlights the need for improved algorithms for key exchange and the importance of selecting secure parameters. This work also has implications for other key exchange protocols, such as ECDHE-ECDSA and ECDHE-ECIES, which are vulnerable to similar attacks.

The current best practice in improving the security of DHE-Diffie-Hellman is to ensure that all parties are using a strong prime and a large field size. In this work, we describe how to carry out this attack and how to prevent it in practice. We also provide recommendations for selecting secure parameters for DHE-Diffie-Hellman and other key exchange protocols.

1. INTRODUCTION

Diffie-Hellman key exchange is widely used for establishing secure channels on the Internet. It is the main key exchange mechanism in SSL and TLS and a popular option in TCP. We examine how weak Diffie-Hellman is tosecondary key exchange and deployed with those protocols and find that, in practice, it is frequently used instead of its more secure alternatives.

There are two main reasons for this. First, a smaller number of servers use weak-aleph Diffie-Hellman parameters or certificates. Second, there is a lack of research and deployment of these protocols. In this paper, we demonstrate how easy it is to break the security guarantees of DHE-Diffie-Hellman and how to prevent it in practice. We also provide recommendations for selecting secure parameters for DHE-Diffie-Hellman and other key exchange protocols.

The current best practice in improving the security of DHE-Diffie-Hellman is to ensure that all parties are using a strong prime and a large field size. In this work, we describe how to carry out this attack and how to prevent it in practice. We also provide recommendations for selecting secure parameters for DHE-Diffie-Hellman and other key exchange protocols.

The main purpose of SSL is to provide confidentiality and integrity against an active man-in-the-middle attacker. Even when the network is completely trustworthy, DHE-Diffie-Hellman is vulnerable to such attacks. In practice, this is the case for a significant portion of the Internet. In this work, we investigate the security of DHE-Diffie-Hellman and provide recommendations for selecting secure parameters.

The current best practice in improving the security of DHE-Diffie-Hellman is to ensure that all parties are using a strong prime and a large field size. In this work, we describe how to carry out this attack and how to prevent it in practice. We also provide recommendations for selecting secure parameters for DHE-Diffie-Hellman and other key exchange protocols.

The main purpose of SSL is to provide confidentiality and integrity against an active man-in-the-middle attacker. Even when the network is completely trustworthy, DHE-Diffie-Hellman is vulnerable to such attacks. In practice, this is the case for a significant portion of the Internet. In this work, we investigate the security of DHE-Diffie-Hellman and provide recommendations for selecting secure parameters.

The current best practice in improving the security of DHE-Diffie-Hellman is to ensure that all parties are using a strong prime and a large field size. In this work, we describe how to carry out this attack and how to prevent it in practice. We also provide recommendations for selecting secure parameters for DHE-Diffie-Hellman and other key exchange protocols.

The main purpose of SSL is to provide confidentiality and integrity against an active man-in-the-middle attacker. Even when the network is completely trustworthy, DHE-Diffie-Hellman is vulnerable to such attacks. In practice, this is the case for a significant portion of the Internet. In this work, we investigate the security of DHE-Diffie-Hellman and provide recommendations for selecting secure parameters.

The current best practice in improving the security of DHE-Diffie-Hellman is to ensure that all parties are using a strong prime and a large field size. In this work, we describe how to carry out this attack and how to prevent it in practice. We also provide recommendations for selecting secure parameters for DHE-Diffie-Hellman and other key exchange protocols.

The main purpose of SSL is to provide confidentiality and integrity against an active man-in-the-middle attacker. Even when the network is completely trustworthy, DHE-Diffie-Hellman is vulnerable to such attacks. In practice, this is the case for a significant portion of the Internet. In this work, we investigate the security of DHE-Diffie-Hellman and provide recommendations for selecting secure parameters.

The current best practice in improving the security of DHE-Diffie-Hellman is to ensure that all parties are using a strong prime and a large field size. In this work, we describe how to carry out this attack and how to prevent it in practice. We also provide recommendations for selecting secure parameters for DHE-Diffie-Hellman and other key exchange protocols.

The main purpose of SSL is to provide confidentiality and integrity against an active man-in-the-middle attacker. Even when the network is completely trustworthy, DHE-Diffie-Hellman is vulnerable to such attacks. In practice, this is the case for a significant portion of the Internet. In this work, we investigate the security of DHE-Diffie-Hellman and provide recommendations for selecting secure parameters.

The current best practice in improving the security of DHE-Diffie-Hellman is to ensure that all parties are using a strong prime and a large field size. In this work, we describe how to carry out this attack and how to prevent it in practice. We also provide recommendations for selecting secure parameters for DHE-Diffie-Hellman and other key exchange protocols.

The main purpose of SSL is to provide confidentiality and integrity against an active man-in-the-middle attacker. Even when the network is completely trustworthy, DHE-Diffie-Hellman is vulnerable to such attacks. In practice, this is the case for a significant portion of the Internet. In this work, we investigate the security of DHE-Diffie-Hellman and provide recommendations for selecting secure parameters.

The current best practice in improving the security of DHE-Diffie-Hellman is to ensure that all parties are using a strong prime and a large field size. In this work, we describe how to carry out this attack and how to prevent it in practice. We also provide recommendations for selecting secure parameters for DHE-Diffie-Hellman and other key exchange protocols.

The main purpose of SSL is to provide confidentiality and integrity against an active man-in-the-middle attacker. Even when the network is completely trustworthy, DHE-Diffie-Hellman is vulnerable to such attacks. In practice, this is the case for a significant portion of the Internet. In this work, we investigate the security of DHE-Diffie-Hellman and provide recommendations for selecting secure parameters.

The current best practice in improving the security of DHE-Diffie-Hellman is to ensure that all parties are using a strong prime and a large field size. In this work, we describe how to carry out this attack and how to prevent it in practice. We also provide recommendations for selecting secure parameters for DHE-Diffie-Hellman and other key exchange protocols.

The main purpose of SSL is to provide confidentiality and integrity against an active man-in-the-middle attacker. Even when the network is completely trustworthy, DHE-Diffie-Hellman is vulnerable to such attacks. In practice, this is the case for a significant portion of the Internet. In this work, we investigate the security of DHE-Diffie-Hellman and provide recommendations for selecting secure parameters.
An Empirical Study of Cryptographic Misuse in Android Applications

Manuel Espele, David Brumley
Dartmouth College
{mrogole,cbrumley}@dmc.edu

Yarick Fantonico, Christopher Kruegel
University of California, Santa Barbara
{yarick.fontonico, yarick.kruegel}@ucsb.edu

ABSTRACT
Developers use cryptographic APIs in Android with the intent of securing data such as passwords and personal information on mobile devices. In this paper, we ask whether developers use the cryptographic APIs in a fashion that provides true cryptographic security, e.g., IND-CPA security. We develop program analysis techniques to automatically track programs on the Google Play marketplace, and find that 10,372 out of 11,728 applications that use cryptographic APIs are vulnerable to at least one common cryptographic misusage. Those numbers show that applications do not use cryptographic APIs in a fashion that maximizes overall security. We then suggest specific remediation based on our analysis to improve overall cryptographic security in Android applications.

Categories and Subject Descriptors
D.2.7 [Software Engineering]: Distribution, Maintenance, and Enhancement—Reengineering, reverse engineering, and redistribution

General Terms
Android program analysis, Misuse of cryptographic primitive

Keywords
Software Security, Program Analysis

1 Introduction
Developers use cryptographic primitives like those offered by the Android KitKat platform to provide strong security guarantees and the wrong way mistakenly leads to breaches.

In this paper, we ask whether developers know how to use cryptographic APIs in a cryptographically correct fashion. In particular, given code that uses an API and compiler, does the implementation code use cryptographic primitives correctly to achieve typical definitions of security? We assume that, if the primitive is used correctly, the code will also be used correctly. We then provide a tool that can perform this analysis and recommend specific remediation techniques that can be used to correct common misuses.

2 Related Work
There are many tools that analyze cryptographic implementations to detect common misuses. Some of these tools are designed to detect specific common misuses, while others are more general. The tools in this section are designed to detect common misuses.

3 Methodology
Our tool, called CryptoLint, is designed to detect common misuses of cryptographic APIs in Android applications. The tool takes an Android application and output a report that includes specific recommendations to improve the application's cryptographic security.

Rule 1: Do not use ECB mode for encryption. [6]
Rule 2: Do not use a non-random IV for CBC encryption. [6, 23]
Rule 3: Do not use constant encryption keys.
Rule 4: Do not use constant salts for PBE. [2, 5]
Rule 5: Do not use fewer than 1,000 iterations for PBE. [2, 5]
Rule 6: Do not use static seeds to seed SecureRandom().

CryptoLint tool to perform static analysis on Android apps to detect how they are using crypto libraries.
CRYPTO MISUSE IN ANDROID APPS

15,134 apps from Google play used crypto; Analyzed 11,748 of them
CRYPTO MISUSE IN ANDROID APPS

15,134 apps from Google play used crypto;
Analyzed 11,748 of them

<table>
<thead>
<tr>
<th># apps</th>
<th>violated rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>5,656</td>
<td>Uses ECB (BouncyCastle default) (R1)</td>
</tr>
<tr>
<td>3,644</td>
<td>Uses constant symmetric key (R3)</td>
</tr>
<tr>
<td>2,000</td>
<td>Uses ECB (Explicit use) (R1)</td>
</tr>
<tr>
<td>1,932</td>
<td>Uses constant IV (R2)</td>
</tr>
<tr>
<td>1,636</td>
<td>Used iteration count &lt; 1,000 for PBE(R5)</td>
</tr>
<tr>
<td>1,629</td>
<td>Seeds SecureRandom with static (R6)</td>
</tr>
<tr>
<td>1,574</td>
<td>Uses static salt for PBE (R4)</td>
</tr>
<tr>
<td>1,421</td>
<td>No violation</td>
</tr>
</tbody>
</table>
15,134 apps from Google play used crypto; Analyzed **11,748** of them

<table>
<thead>
<tr>
<th># apps</th>
<th>violated rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>5,656</td>
<td>Uses ECB <em>(BouncyCastle default)</em> (R1)</td>
</tr>
<tr>
<td>3,644</td>
<td>Uses constant symmetric key (R3)</td>
</tr>
<tr>
<td>2,000</td>
<td>Uses ECB <em>(Explicit use)</em> (R1)</td>
</tr>
<tr>
<td>1,932</td>
<td>Uses constant IV (R2)</td>
</tr>
<tr>
<td>1,636</td>
<td>Used iteration count &lt; 1,000 for PBE(R5)</td>
</tr>
<tr>
<td>1,629</td>
<td>Seeds SecureRandom with static (R6)</td>
</tr>
<tr>
<td>1,574</td>
<td>Uses static salt for PBE (R4)</td>
</tr>
<tr>
<td>1,421</td>
<td>No violation</td>
</tr>
</tbody>
</table>
NEVER use ECB
(but over 50% of Android apps do)
• BouncyCastle is a library that conforms to Java’s `Cipher` interface:

```java
Cipher c =
    Cipher.getInstance("AES/CBC/PKCS5Padding");

// Ultimately end up wrapping a ByteArrayOutputStream
// in a CipherOutputStream
```

• Java documentation specifies:

> If no mode or padding is specified, provider-specific default values for the mode and padding scheme are used. For example, the SunJCE provider uses ECB as the default mode, and PKCS5Padding as the default padding scheme for DES, DES-EDE and Blowfish ciphers.
<table>
<thead>
<tr>
<th>#Occurences</th>
<th>Symmetric encryption scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>5878</td>
<td>AES/CBC/PKCS5Padding</td>
</tr>
<tr>
<td>4803</td>
<td>AES *</td>
</tr>
<tr>
<td>1151</td>
<td>DES/ECB/NoPadding</td>
</tr>
<tr>
<td>741</td>
<td>DES *</td>
</tr>
<tr>
<td>501</td>
<td>DESede *</td>
</tr>
<tr>
<td>473</td>
<td>DESede/ECB/PKCS5Padding</td>
</tr>
<tr>
<td>468</td>
<td>AES/CBC/NoPadding</td>
</tr>
<tr>
<td>443</td>
<td>AES/ECB/PKCS5Padding</td>
</tr>
<tr>
<td>235</td>
<td>AES/CBC/PKCS7Padding</td>
</tr>
<tr>
<td>221</td>
<td>DES/ECB/PKCS5Padding</td>
</tr>
<tr>
<td>220</td>
<td>AES/ECB/NoPadding</td>
</tr>
<tr>
<td>205</td>
<td>DES/CBC/PKCS5Padding</td>
</tr>
<tr>
<td>155</td>
<td>AES/ECB/PKCS7Padding</td>
</tr>
<tr>
<td>104</td>
<td>AES/CFB8/NoPadding</td>
</tr>
</tbody>
</table>

Table 4: Distribution of frequently used symmetric encryption schemes. Schemes marked with * are used in ECB mode by default.
<table>
<thead>
<tr>
<th>#Occurences</th>
<th>Symmetric encryption scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>5878</td>
<td>AES/CBC/PKCS5Padding</td>
</tr>
<tr>
<td>4803</td>
<td>AES *</td>
</tr>
<tr>
<td>1151</td>
<td>DES/ECB/NoPadding</td>
</tr>
<tr>
<td>741</td>
<td>DES *</td>
</tr>
<tr>
<td>501</td>
<td>DESede *</td>
</tr>
<tr>
<td>473</td>
<td>DESede/ECB/PKCS5Padding</td>
</tr>
<tr>
<td>468</td>
<td>AES/CBC/NoPadding</td>
</tr>
<tr>
<td>443</td>
<td>AES/ECB/PKCS5Padding</td>
</tr>
<tr>
<td>235</td>
<td>AES/CBC/PKCS7Padding</td>
</tr>
<tr>
<td>221</td>
<td>DES/ECB/PKCS5Padding</td>
</tr>
<tr>
<td>220</td>
<td>AES/ECB/NoPadding</td>
</tr>
<tr>
<td>205</td>
<td>DES/CBC/PKCS5Padding</td>
</tr>
<tr>
<td>155</td>
<td>AES/ECB/PKCS7Padding</td>
</tr>
<tr>
<td>104</td>
<td>AES/CFB8/NoPadding</td>
</tr>
</tbody>
</table>

**Table 4:** Distribution of frequently used symmetric encryption schemes. Schemes marked with * are used in ECB mode by default.
15,134 apps from Google play used crypto; analyzed 11,748 of them.

<table>
<thead>
<tr>
<th># apps</th>
<th>violated rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>5,656</td>
<td>Uses ECB (BouncyCastle default) (R1)</td>
</tr>
<tr>
<td>3,644</td>
<td>Uses constant symmetric key (R3)</td>
</tr>
<tr>
<td>2,000</td>
<td>Uses ECB (Explicit use) (R1)</td>
</tr>
<tr>
<td>1,932</td>
<td>Uses constant IV (R2)</td>
</tr>
<tr>
<td>1,636</td>
<td>Used iteration count &lt; 1,000 for PBE (R5)</td>
</tr>
<tr>
<td>1,629</td>
<td>Seeds SecureRandom with static (R6)</td>
</tr>
<tr>
<td>1,574</td>
<td>Uses static salt for PBE (R4)</td>
</tr>
<tr>
<td>1,421</td>
<td>No violation</td>
</tr>
</tbody>
</table>
CRYPTO MISUSE IN ANDROID APPS

15,134 apps from Google play used crypto; Analyzed 11,748 of them

<table>
<thead>
<tr>
<th># apps</th>
<th>violated rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>48%</td>
<td>5,656 Uses ECB (BouncyCastle default) (R1)</td>
</tr>
<tr>
<td>31%</td>
<td>3,644 Uses constant symmetric key (R3)</td>
</tr>
<tr>
<td>17%</td>
<td>2,000 Uses ECB (Explicit use) (R1)</td>
</tr>
<tr>
<td>16%</td>
<td>1,932 Uses constant IV (R2)</td>
</tr>
<tr>
<td>14%</td>
<td>1,636 Used iteration count &lt; 1,000 for PBE(R5)</td>
</tr>
<tr>
<td>14%</td>
<td>1,629 Seeds SecureRandom with static (R6)</td>
</tr>
<tr>
<td>12%</td>
<td>1,574 Uses static salt for PBE (R4)</td>
</tr>
<tr>
<td></td>
<td>1,421 No violation</td>
</tr>
</tbody>
</table>

A failure of the programmers to **know the tools** they use
A failure of library writers to **provide safe defaults**
MISUSING CRYPTO

Avoid shooting yourself in the foot:

• Do not **roll your own** cryptographic mechanisms
  • Takes peer review
  • Apply Kerkhoff’s principle

• Do not **misuse** existing crypto

• Do not even **implement** the underlying crypto
WHY NOT IMPLEMENT AES/RSA YOURSELF?

• Not talking about creating a brand new crypto scheme, just implementing one that’s already widely accepted and used.

• Kerkhoff’s principle: these are all open standards; should be implementable.

• Potentially buggy/incorrect code, but so might be others’ implementations (viz. OpenSSL bugs, poor defaults in Bouncy castles, etc.)

• So why not implement it yourself?
Cryptography concerns the theoretical difficulty in breaking a cipher.
Cryptography concerns the **theoretical** difficulty in breaking a cipher

- Cryptographic processing (Encrypt/decrypt/sign/etc.)
- Secret keys

But what about the information that a particular **implementation** could leak?
- Attacks based on these are **“side-channel attacks”**
Cryptography concerns the *theoretical* difficulty in breaking a cipher.

But what about the information that a particular implementation could leak?

- Attacks based on these are "side-channel attacks"
SIMPLE POWER ANALYSIS (SPA)

- Interpret *power traces* taken during a cryptographic operation

- Simple power analysis can reveal the sequence of instructions executed
SPA ON DES

Figure 1: SPA trace showing an entire DES operation.

Overall operation clearly visible:
Can identify the **16 rounds of DES**
Overall operation clearly visible:
Can identify the 16 rounds of DES
Figure 3: SPA trace showing individual clock cycles.

Specific instructions are also discernible.
Figure 3: SPA trace showing individual clock cycles.

Specific instructions are also discernible.
HypotheticalEncrypt(msg, key) {
    for(int i=0; i < key.len(); i++) {
        if(key[i] == 0)
            // branch 0
        else
            // branch 1
    }
}
HypotheticalEncrypt(msg, key) {
    for (int i=0; i < key.len(); i++) {
        if (key[i] == 0) {
            // branch 0
        } else {
            // branch 1
        }
    }
}
HypotheticalEncrypt(msg, key) {
    for(int i=0; i < key.len(); i++) {
        if(key[i] == 0) { // branch 0
            What if branch 0 had, e.g., a jmp that branch 1 didn’t?
        } else { // branch 1
        }
    }
}

Implementation issue: If the execution path depends on the inputs (key/data), then SPA can reveal keys
HypotheticalEncrypt(msg, key) {
    for(int i=0; i < key.len(); i++) {
        if(key[i] == 0) {
            // branch 0
        } else {
            // branch 1
        }
    }
}

Implementation issue: If the execution path depends on the inputs (key/data), then SPA can reveal keys.
HypotheticalEncrypt(msg, key) {
    for (int i = 0; i < key.len(); i++) {
        if (key[i] == 0) {
            // branch 0
        } else {
            // branch 1
        }
    }
}

Implementation issue: If the execution path depends on the inputs (key/data), then SPA can reveal keys.

What if branch 0 had, e.g., a jmp that branch 1 didn’t?

What if branch 0 - took longer? (timing attacks)
HypotheticalEncrypt(msg, key) {
    for(int i=0; i < key.len(); i++) {
        if(key[i] == 0) // branch 0
            // branch 0
        else // branch 1
            // branch 1
    }
}

What if branch 0 had, e.g., a jmp that branch 1 didn’t?
What if branch 0
- took longer? (timing attacks)
- gave off more heat?

Implementation issue: If the execution path depends on the inputs (key/data), then SPA can reveal keys
HypotheticalEncrypt(msg, key) {
  for(int i=0; i < key.len(); i++) {
    if(key[i] == 0)
      // branch 0
    else
      // branch 1
  }
}

What if branch 0 had, e.g., a jmp that branch 1 didn’t?

What if branch 0
- took longer? (timing attacks)
- gave off more heat?
- made more noise?
- ...

Implementation issue: If the execution path depends on the inputs (key/data), then SPA can reveal keys
DIFFERENTIAL POWER ANALYSIS (DPA)

• SPA just visually inspects a single run

• DPA runs iteratively and reactively
  • Get multiple samples
  • Based on these, construct new plaintext messages as inputs, and repeat
MITIGATING SUCH ATTACKS

• Hide information by making the execution paths depend on the inputs as little as possible
  • Have to *give up some optimizations* that depend on particular bit values in keys
    - Some Chinese Remainder Theorem (CRT) optimizations permitted remote timing attacks on SSL servers

• The crypto community should seek to design cryptosystems under the assumption that some information is going to leak
POOR POLICIES FROM GOVERNMENTS

Exploits export-grade encryption

Figure 4: NSA’s VPN decryption infrastructure. This classified illustration published by Der Spiegel [67] shows captured IKE handshake messages being passed to a high-performance computing system, which returns the symmetric keys for ESP session traffic. The details of this attack are consistent with an efficient break for 1024-bit Diffie-Hellman.

1024-bit and smaller feasibly broken

Logjam downgrades to export-grade (512)
Clipper chip
A lesson in poorly designed protocols

Goal: Confidentiality
Support encrypted communication between devices

Goal: Key escrow
Permit law enforcement to obtain “session keys” with a warrant
Clipper chip: Design

Tamper-proof hardware

Skipjack
encryption algorithm

Skipjack Keys
Unit key
Global family key

Diffie-Hellman
key exchange

LEAF generation & validation

Hardware that is difficult to introspect (e.g., extract keys), alter (change the algorithms), or impersonate.
Clipper chip: Design

Tamper-proof hardware

Skipjack
encryption algorithm

Skipjack Keys
Unit key
Global family key

Diffie-Hellman
key exchange

LEAF generation & validation

Block cipher designed by the NSA, originally classified SECRET.

(Violates Kirchhoff’s principle)

Broken within one day of declassification.

80-bit key; similar algorithm to DES (also broken)
Clipper chip: Design

Tamper-proof hardware

**Skipjack**
- encryption algorithm

**Skipjack Keys**
- Unit key
- Global family key

**Diffie-Hellman**
- key exchange

**LEAF**
- generation
- & validation

Assigned when the hardware is manufactured.

Unit key is unique to this unit in particular (each Clipper chip also has a *unit ID*).

Global family key is the same across many units.
Clipper chip: Design

Tamper-proof hardware

**Skipjack**
- encryption algorithm

**Skipjack Keys**
- Unit key
- Global family key

**Diffie-Hellman**
- key exchange

**LEAF**
- generation & validation

Used for establishing a (symmetric) *session key*

Session keys are ephemeral (e.g., last only for a given connection, transaction, etc.)

General properties about session keys:
- Compromising one session key does not compromise others
- Compromising a long-term key should not compromise past session keys (*forward secrecy*)
Clipper chip: Design

Tamper-proof hardware

**Skipjack**
- encryption algorithm

**Skipjack Keys**
- Unit key
- Global family key

**Diffie-Hellman**
- key exchange

**LEAF**
- generation & validation

LEAF (Law Enforcement Access Field)

To permit wiretapping, law enforcement needs to be able to extract session keys, but only has access to what is sent during communication.

**Idea**: send data that has enough info to allow law enforcement to extract keys (but not any other eavesdropper).
LEAF protocol design

1. DH key exchange
2. Each send LEAF packet
3. Send data encrypted with the session key

The Clipper chips will not decrypt until it has received a valid LEAF packet

Law enforcement sees all packets.
• Cannot infer key from DH key exchange
• Can infer it from the LEAF packet
LEAF message structure

- Session key: 80 bits
- Unit Key
- Hash algorithm: 16 bits
- Unit ID
- Encrypted session key
- Hash
- Global family key
- Other variables

Flow:
- Unit Key → Skipjack
- Encrypted session key → Hash
- Global family key → Skipjack

Output: LEAF
LEAF message structure

The other Clipper chip also has the Global Family key

=> Can decrypt the LEAF to obtain this triple

Unit ID | Encrypted session key | Hash

Global family key \(\rightarrow\) Skipjack

LEAF
LEAF message structure

- Session key: 80 bits
- Hash algorithm
- Other variables

The other Clipper chip "verifies" the LEAF by making sure that the hash is correct.

Global family key → Skipjack

Unit Key → Skipjack

Hash → 16 bits
LEAF message structure

Law enforcement also has the Global Family Key

=> Can decrypt the LEAF to obtain this triple

Global family key \rightarrow \text{Skipjack} \rightarrow \text{LEAF}
Law enforcement *does not* have direct access to all unit keys; needs a **warrant** to get them.

Unit keys are split across two locations (one location gets a OTP, the other gets the XOR).
LEAF: failure

To verify the LEAF, the other Clipper chip only checks the hash.

Clipper chips also allow you to test a LEAF locally.
LEAF: failure

- Session key: 80 bits
- Hash algorithm: 16 bits
- Other variables

Generate a random LEAF => 1/2^{16} chance of a valid hash

But law enforcement will just see random ID & key

Validates at the other Clipper chip (so it will decrypt messages)
USEFUL TOOL: ZMAP

Goal: port-scan the entire Internet in less than an hour

Approaches:

- Non-blocking, stateless ➞ Highly parallelizable
- Randomize addresses ➞ Avoid takedown notices

Datasets: Rapid7, censys.io
**Unsafe Optimizations**

**TLS session ticket resumption**

Session ticket: session keys and other data to resume the session

Server sends an “opaque” ticket (encrypted with the Session Ticket Encryption Key, STEK)

Client sends the encrypted session ticket during handshake; server uses the STEK to recover it and pick up in one round-trip of communication.
UNSAFE OPTIMIZATIONS

Incentive to hold onto STEKs (lower RTTs)

But they’re holding onto them long enough for nation-states to recover them

Figure 3: STEK Lifetime—TLS connections cannot achieve forward secrecy until the STEK (the key used by the server to encrypt the session ticket) is discarded.

Figure 4: STEK Lifetime by Alexa Rank—We found 12 Alexa Top 100 sites that persisted STEKs for at least 30 days.
UNSAFE OPTIMIZATIONS

Incentive to hold onto STEKs (lower RTTs)

But they’re holding onto them long enough for nation-states to recover them

Figure 3: STek Lifetime—TLS connections cannot achieve forward secrecy until the STEK (the key used by the server to encrypt the session ticket) is discarded.

Figure 4: STek Lifetime by Alexa Rank—We found 12 Alexa Top 100 sites that persisted STEKs for at least 30 days.

Figure 5: Ephemeral Exchange Value Reuse—We measured how long Alexa Top Million websites served identical DHE and ECDHE values (note vertical scale is cropped).
SSL Handshake (RSA) Without Keyless SSL

Handshake:

1. Visitor sends hello, client random, and cipher suites supported

2. Server sends server random and public key certificate (also sent is a session ID for session resumption)

3. Visitor encrypts premaster secret with public key

4. CloudFlare decrypts the premaster secret with the private key

Both the visitor and CloudFlare create session keys from the client random, server random, and premaster secret. Now the visitor can request content from CloudFlare. (also sent is a session ticket for session resumption)
SSL Handshake (Diffie-Hellman) Without Keyless SSL

Handshake

Visitor

1. Visitor sends hello, client random, and cipher suites supported

CloudFlare

2a. Server sends server random and public key certificate (also sent is a session ID for session resumption)

2b. The key signs for client random, server random, and public key certificate

Server random

Public key certificate

Server DH parameter

Signature from key server

Client DH parameter

Premaster secret

Session key

3. Server sends the server DH parameter and a signature

4. Visitor sends the client DH parameter

Both the visitor and Cloudflare derive identical premaster secrets from the server DH parameter and client DH parameter.

Both the visitor and Cloudflare derive identical session keys from the client random, server random, and premaster secret. The visitor can request content from Cloudflare, and the request will be encrypted. (also sent is a session ticket for session resumption)
CloudFlare Keyless SSL (Diffie-Hellman)

Handshake

Visitor
1. Visitor sends hello, client random, and cipher suites supported.

CloudFlare
2. CloudFlare sends server random and public key certificate (may be sent as a session ID for session resumption).

2b. CloudFlare sends hash of client random, server random, and DH parameter.

Origin server
3. Origin server sends cached content or uncached content.

Key server
4. Key server sends private key.

Visitor
5. Visitor sends the client DH parameter.

6. Both the visitor and CloudFlare derive identical session keys from the client DH parameter and DH parameter.

7. Derive the Session key.

Premaster secret
8. Both the visitor and CloudFlare derive identical session keys from the server DH parameter and client DH parameter.

9. Derive the Session key.

Client DH parameter
10. Client DH parameter.

Server DH parameter
11. Server DH parameter.

Signature from key server
12. Signature from key server.

Public key certificate
13. Public key certificate.
Analysis of SSL Certificate Reissues and Revocations
In the Wake of Heartbleed

Liang Zhang, David Champagne, Dave Levin, Tudor Dumitru
Northwestern University
lzh0113@northwestern.edu, david.champagne@northwestern.edu, dave.levin@northwestern.edu, tudor.dumitru@northwestern.edu

POOR CERTIFICATE MANAGEMENT

Abstract

In the recent operation of a public key infrastructure (PKI), the ability to revoke certificates is crucial. While most of necessary mechanisms are already being taken place quickly in practice, revocation procedures typically require a human in the middle to release a new certificate and revoke the old one. This makes the process inherently dependent on how often systemwide certificate requests and resultant certificate issuance is under the command of an RSA. Unfortunately, SSL is typically slower to measure with this relative ease to determine when a certificate is revoked. It is difficult to determine whether and when administrators start to have revocation.

In this paper, we see a much faster and more secure revocation as a new experiment. We have measured and revoking certificate in the last 24 hours, and we have observed that most of certificates have been revoked relatively quickly, with some certificates being revoked within minutes.

SSL, TLS, HTTPS, X.509, Certificate, Issuance, Revocation, Extension, validation

Categories and Subject Descriptors

Keywords
SSL, TLS, Reissue, Certificates, Issuance, Revocation, Extensions, validation

1. INTRODUCTION

SSL (Secure Sockets Layer) and TCP (Transmission Control Protocol) are the de facto standards for securing Internet communications such as domain name and directory. Along with a public key infrastructure (PKI), SSL provides trusted identities via certificate chains and private communications via encryption. Central to this is the issuance of new certificates to replace those that have reached their end of life.

In practice, the PKI uses a default model where potentially thousands of certificates are issued daily and their expiration dates can differ widely. Moreover, reissuance is a significant issue in revocation since the revocation process is not immediate.

Unfortunately, the PKI has not been able to effectively communicate with the certificate issuers, which results in a slow revocation process.

2. INTRODUCTION

Online and off-line authentication is a fundamental requirement to secure communications. On the Web, Secure Socket Layer (SSL) and Transport Layer Security (TLS) are the de facto standards for securing Internet communications such as domain name and directory. Along with a public key infrastructure (PKI), SSL provides trusted identities via certificate chains and private communications via encryption. To the sheers and consumers of SSL/TLS, users have developed a mature opinion to believe that if a site has a secure connection, it is trustworthy.

In this paper, we address the question using a new method for understanding the trust in SSL/TLS. In mid-2014, as SSL is still in an early stage of its life cycle, we have observed that there are still thousands of SSL/TLS users that only use a single certificate for the entire life of the certificate. This is not only a security risk, but also a privacy risk. In this paper, we present a new method for understanding the trust in SSL/TLS.
Heartbleed

OpenSSL
Heartbleed

"hi" 2

OpenSSL
Heartbleed

"hi" 2

"hi"

OpenSSL
Heartbleed

OpenSSL
Heartbleed

"hi" 22

OpenSSL
Heartbleed

"hi" 22

OpenSSL

"hi"

+20B from memory

< 2^16
Heartbleed exploits were undetectable

Potentially reveals user data and private keys
Why study Heartbleed?

- Discovered: 03/21
- Akamai patched: 04/02
- Publicly announced: 04/07
Why study Heartbleed?

Every vulnerable website should have:

1. Patched
2. Revoked
3. Reissued
Why study Heartbleed?

Every vulnerable website should have:
1. Patched
2. Revoked
3. Reissued

Heartbleed is a natural experiment:
How quickly and thoroughly do administrators act?
Dataset

Rapid7 data

22M certs
(~1/wk for 6mos)
Dataset

Rapid7 data
22M certs (~1/wk for 6mos)

2.8M certs
Alexa Top-1M

CAs
9k certs

filter
Dataset

- Rapid7 data: 22M certificates (~1/week for 6 months)
- Alexa Top-1M: 2.8M certificates
- CAs: 9k certificates
- Leaf Set: 628k certificates, 165k domains

Filters and Validates the dataset.
Dataset

- Rapid7 data
  - 22M certs (~1/wk for 6mos)

- CAs
  - 9k certs

- Alexa Top-1M
  - 2.8M certs

- Leaf Set
  - 628k certs
  - 165k domains

- Filter
- Validate

- Download CRLs
- Detect vulnerability
- Identify Heartbleed-induced reissues & revocations
Dataset

- **Rapid7 data**: 22M certs (~1/week for 6mos)
- **Alexa Top-1M**: 2.8M certs
- **CAs**: 9k certs
- **Leaf Set**: 628k certs, 165k domains

- **Filter**
- **Validate**
- **Download CRLs**
- **Detect vulnerability**
- **Identify Heartbleed-induced reissues & revocations**
Prevalence and patch rates

Fraction of Domains Vulnerable to Heartbleed

Alexa Site Rank (bins of 1000)

Was ever vulnerable
Still vulnerable after 3 weeks
Prevalence and patch rates

Fraction of Domains Vulnerable to Heartbleed

- Was ever vulnerable
- Still vulnerable after 3 weeks

Alexa Site Rank (bins of 1000)
Prevalence and patch rates

Patching rates are mostly positive
Only ~7% had not patched within 3 weeks
How quickly were certs revoked?
How quickly were certs revoked?

Reaction ramps up quickly
How quickly were certs revoked?

Reaction ramps up quickly
How quickly were certs revoked?

Reaction ramps up quickly

Security takes the weekends off
Certificate update rates

Frac. of Vulnerable Certs not Revoked/Reissued

Date

04/07 04/21 05/05 05/19 06/02 06/16 06/30 07/14 07/28

0.6 0.65 0.7 0.75 0.8 0.85 0.9 0.95 1.0

3 wks
Certificate update rates

Frac. of Vulnerable Certs not Revoked/Reissued

Date

04/07 04/21 05/05 05/19 06/02 06/16 06/30 07/14 07/28

Not revoked

3 wks
Certificate update rates

- Date:
  - 04/07
  - 04/21
  - 05/05
  - 05/19
  - 06/02
  - 06/16
  - 06/30
  - 07/14
  - 07/28

- Frac. of Vulnerable Certs not Revoked/Reissued:
  - 3 wks
Certificate update rates

Frac. of Vulnerable Certs not Revoked/Reissued

Date

04/07 04/21 05/05 05/19 06/02 06/16 06/30 07/14 07/28

0.6 0.65 0.7 0.75 0.8 0.85 0.9 0.95 1

Not revoked

Not reissued

Similar pattern to patches:
Exponential drop-off, then levels out

After 3 weeks: 13% Revoked 27% Reissued
Reissue ⇒ New key?

Fraction of New Certificates Reissued with the Same Key

- All reissues
- Heartbleed-induced reissues

Date of Birth

11/2013 12/2013 01/2014 02/2014 03/2014 04/2014 05/2014
Reissue $\Rightarrow$ New key?

Fraction of New Certificates Reissued with the Same Key

Date of Birth

11/2013 12/2013 01/2014 02/2014 03/2014 04/2014 05/2014

All reissues

Heartbleed-induced reissues
Reissuing the same key is common practice

4.1% Heartbleed-induced
Can we wait for expiration?
Can we wait for expiration?

Vulnerable but not revoked
Can we wait for expiration?

Vulnerable but not revoked

\[ \sim 40\% \text{ did not expire after one year} \]
Can we wait for expiration?

Vulnerable but not revoked

~8% of vulnerable certs still unexpired

~40% did not expire after one year
Can we wait for expiration?

We may be dealing with Heartbleed for years
Security is an economic concern
Security is an economic concern

- Browser
- Website
- Certificate Authority

Revoke?
Security is an economic concern

**Browsers** face tension between security and **page load times**

**CAs** face tension between security and **bandwidth costs**
But OCSP Stapling rarely activated by admins: Our scan: 3% of normal certs; 2% of EV certs
Testing browser behavior

- **Revocation protocols**
  - Browsers *should* support all major protocols
    - CRLs, OCSP, OCSP stapling

- **Availability of revocation info**
  - Browsers *should* reject certs they cannot check
    - E.g., because the OCSP server is down

- **Chain lengths**
  - Browsers *should* reject a cert if *any* on the chain fail
    - Leaf, intermediate(s), root
Testing browser behavior

- Browsers should support all major protocols
  - CRLs, OCSP, OCSP stapling

- Browsers should reject certs they cannot check
  - E.g., because the OCSP server is down

- Browsers should reject a cert if any on the chain fail
  - Leaf, intermediate(s), root
Testing browser behavior

Revocation protocols

- Browsers should support all major protocols
  - CRLs, OCSP, OCSP stapling

Availability of revocation info

- Browsers should reject certs they cannot check
  - E.g., because the OCSP server is down

Chain lengths

- Browsers should reject a cert if any on the chain fail
  - Leaf, intermediate(s), root
Testing browser behavior

- **Revocation protocols**
  - Browsers *should* support all major protocols
    - CRLs, OCSP, OCSP stapling

- **Availability of revocation info**
  - Browsers *should* reject certs they cannot check
    - E.g., because the OCSP server is down

- **Chain lengths**
  - Browsers *should* reject a cert if *any* on the chain fail
    - Leaf, intermediate(s), root

![Diagram showing the signing process of a certificate chain with root, intermediate, and leaf certificates.](diagram.png)
Testing browser behavior

- **Revocation protocols**
  - Browsers *should* support all major protocols
    - CRLs, OCSP, OCSP stapling

- **Availability of revocation info**
  - Browsers *should* reject certs they cannot check
    - E.g., because the OCSP server is down

- **Chain lengths**
  - Browsers *should* reject a cert if *any* on the chain fail
    - Leaf, intermediate(s), root

Diagram:
- Root (signs)
- Intermediate
- ... Intermediate
- Leaf
Test harness

Implemented 192 tests using fake root certificate + Javascript
• Unique DNS name, cert chain, CRL/OCSP responder, …
## Results across all browsers

<table>
<thead>
<tr>
<th>CRL</th>
<th>Firefox</th>
<th>Opera</th>
<th>Safari</th>
<th>IE</th>
<th>iOS</th>
<th>Andr. 4.1-5.1</th>
<th>IE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revoked Unavailable</td>
<td>35-37</td>
<td>12.17</td>
<td>28.0</td>
<td>6-8</td>
<td>7-9</td>
<td>10-11</td>
<td>6-8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OCSP</th>
<th>Chrome 42</th>
<th>Desktop Browsers</th>
<th>Mobile Browsers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revoked Unavailable</td>
<td>OS X</td>
<td>Win.</td>
<td>Linux</td>
</tr>
<tr>
<td>Leaf</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OCSP Stapling</td>
<td>Request OCSP Staple</td>
<td>Respect Revoked Staple</td>
<td></td>
</tr>
</tbody>
</table>

- ✔️ Passes test
- ✗ Fails test
- EV Passes for EV certs
- I Ignores OCSP Staple
- A Pops up alert to user
- L/W Passes on Linux/Win.
Results across all browsers

Chrome

Generally, only checks for EV certs
~3% of all certs

Allows if revocation info unavailable

Supports OCSP stapling

<table>
<thead>
<tr>
<th>CRL</th>
<th>OS X</th>
<th>Win.</th>
<th>Linux</th>
</tr>
</thead>
<tbody>
<tr>
<td>Int. 1</td>
<td>EV</td>
<td>✓</td>
<td>EV</td>
</tr>
<tr>
<td>Revoked</td>
<td>EV</td>
<td>✓</td>
<td>–</td>
</tr>
<tr>
<td>Unavailable</td>
<td>×</td>
<td>×</td>
<td>–</td>
</tr>
<tr>
<td>Leaf</td>
<td>EV</td>
<td>EV</td>
<td>EV</td>
</tr>
<tr>
<td>Revoked</td>
<td>×</td>
<td>×</td>
<td>–</td>
</tr>
<tr>
<td>Unavailable</td>
<td>×</td>
<td>×</td>
<td>–</td>
</tr>
<tr>
<td>OCSP</td>
<td>EV</td>
<td>EV</td>
<td>EV</td>
</tr>
<tr>
<td>Revoked</td>
<td>×</td>
<td>×</td>
<td>–</td>
</tr>
<tr>
<td>Unavailable</td>
<td>×</td>
<td>×</td>
<td>–</td>
</tr>
<tr>
<td>Int. 2+</td>
<td>EV</td>
<td>EV</td>
<td>EV</td>
</tr>
<tr>
<td>Revoked</td>
<td>×</td>
<td>×</td>
<td>–</td>
</tr>
<tr>
<td>Unavailable</td>
<td>×</td>
<td>×</td>
<td>–</td>
</tr>
<tr>
<td>Leaf</td>
<td>EV</td>
<td>EV</td>
<td>EV</td>
</tr>
<tr>
<td>Revoked</td>
<td>×</td>
<td>×</td>
<td>–</td>
</tr>
<tr>
<td>Unavailable</td>
<td>×</td>
<td>×</td>
<td>–</td>
</tr>
<tr>
<td>OCSP Stapling</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Request OCSP Staple</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Respect Revoked Staple</td>
<td>×</td>
<td>✓</td>
<td>–</td>
</tr>
</tbody>
</table>

✔ Passes test
✗ Fails test
I Ignores OCSP Staple
A Pops up alert to user
L/W Passes on Linux/Win.
### Firefox

**Results across all browsers**

<table>
<thead>
<tr>
<th></th>
<th>Desktop Firefox</th>
<th>35–37</th>
</tr>
</thead>
<tbody>
<tr>
<td>Int. 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRL</td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td>Revoked</td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td>Unavailable</td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td>Int. 2+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revoked</td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td>Unavailable</td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td>Leaf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revoked</td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td>Unavailable</td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td>Int. 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OCSP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revoked</td>
<td>EV</td>
<td></td>
</tr>
<tr>
<td>Unavailable</td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td>Int. 2+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revoked</td>
<td>EV</td>
<td></td>
</tr>
<tr>
<td>Unavailable</td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td>Lcafe</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revoked</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Unavailable</td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td>OCSP Stapling</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Request OCSP Staple</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Respect Revoked Staple</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

- ✔ Passes test
- ✗ Fails test
- EV Passes for EV certs
- I Ignores OCSP Staple
- A Pops up alert to user
- L/W Passes on Linux/Win.
Results across all browsers

<table>
<thead>
<tr>
<th>Safari 6-8</th>
<th>CRL</th>
<th>OCSP</th>
<th>OCSP Stapling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Int. 1</td>
<td>Revoked</td>
<td>Revoked</td>
<td>Request OCSP Staple</td>
</tr>
<tr>
<td></td>
<td>Unavailable</td>
<td>Unavailable</td>
<td>Respect Revoked Staple</td>
</tr>
<tr>
<td>Leaf</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Int. 2+</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Leaf</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Safari

- Checks CRLs and OCSP
- Allows if revocation info unavailable
- Except for first intermediate, for CRLs

- Does not support OCSP stapling

✔ Passes test
✗ Fails test
I Ignores OCSP Staple
A Pops up alert to user
L/W Passes on Linux/Win.

Passes for EV certs
### Results across all browsers

<table>
<thead>
<tr>
<th></th>
<th>CRL</th>
<th>OCSP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Internet Explorer</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Int. 1</strong></td>
<td>Revoked</td>
<td>Revoked</td>
</tr>
<tr>
<td></td>
<td>Unavailable</td>
<td>Unavailable</td>
</tr>
<tr>
<td><strong>Int. 2+</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Leaf</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>OCSP</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Int. 1</strong></td>
<td>Revoked</td>
<td>Revoked</td>
</tr>
<tr>
<td></td>
<td>Unavailable</td>
<td>Unavailable</td>
</tr>
<tr>
<td><strong>Int. 2+</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Leaf</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>OCSP Stapling</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Passes test</strong></td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td><strong>EV</strong></td>
<td>Passes for EV certs</td>
<td></td>
</tr>
<tr>
<td><strong>Pops up alert to user</strong></td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td><strong>I</strong></td>
<td>Ignores OCSP Staple</td>
<td></td>
</tr>
<tr>
<td><strong>L/W</strong></td>
<td>Passes on Linux/Win.</td>
<td></td>
</tr>
</tbody>
</table>

Checks CRLs and OCSP

Often rejects if revocation info unavailable

Pops up alert for leaf in IE 10+

Supports OCSP stapling
Results across all browsers

<table>
<thead>
<tr>
<th></th>
<th>Mobile Browsers</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>iOS 6–8</td>
<td>Andr. 4.1–5.1</td>
<td>Stock</td>
<td>Chrome</td>
<td>IE 8.0</td>
</tr>
<tr>
<td><strong>CRL</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Int. 1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Revoked</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unavailable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Int. 2+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revoked</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unavailable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Leaf</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revoked</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unavailable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>OCSP</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Int. 1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Revoked</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unavailable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Int. 2+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revoked</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unavailable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Leaf</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revoked</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unavailable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>OCSP Stapling</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Request OCSP Staple</td>
<td>x</td>
<td>1</td>
<td>1</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Respect Revoked Staple</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Uniformly never check**
- **Android browsers request Staple**
  ...and promptly ignore it

- ✔️ Passes test
- ✗ Fails test
- ❑ Passes for EV certs
- I Ignores OCSP Staple
- A Pops up alert to user
- L/W Passes on Linux/Win.
## Results across all browsers

<table>
<thead>
<tr>
<th></th>
<th>Chrome 42</th>
<th>Desktop Browsers</th>
<th>Safari</th>
<th>IE 7–9</th>
<th>IE 10–11</th>
<th>iOS 6–8</th>
<th>Mobile Browsers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OS X</td>
<td>Win.</td>
<td>Linux</td>
<td>Firefox</td>
<td>Opera 12</td>
<td>28.0</td>
<td></td>
</tr>
<tr>
<td><strong>CRL</strong></td>
<td>EV</td>
<td>✔</td>
<td>EV</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td><strong>Revoked</strong></td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td><strong>Unavailable</strong></td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td><strong>Leaf</strong></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td><strong>OCSP</strong></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td><strong>Revoked</strong></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td><strong>Unavailable</strong></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td><strong>Lcaf</strong></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td><strong>OCSP Stapling</strong></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Request OCSP Staple</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Respect Revoked Staple</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

- **✔** Passes test
- **✗** Fails test
- **EV** Passes for EV certs
- **I** Ignores OCSP Staple
- **A** Pops up alert to user
- **L/W** Passes on Linux/Win.
## Results across all browsers

<table>
<thead>
<tr>
<th></th>
<th>Chrome 42</th>
<th>Desktop Browsers</th>
<th>Safari</th>
<th>IE</th>
<th>Mobile Browsers</th>
<th>iO S</th>
<th>Andr. 4.1–5.1</th>
<th>Chrome 8.0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OS X</td>
<td>Win.</td>
<td>Linux</td>
<td>Firefox</td>
<td>Opera</td>
<td>6–8</td>
<td>7–9</td>
<td>10–11</td>
</tr>
<tr>
<td><strong>CRL</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Int. 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revoked</td>
<td>EV</td>
<td>✓</td>
<td>EV</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Unavailable</td>
<td>EV</td>
<td>✓</td>
<td>–</td>
<td>x</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Leaf</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revoked</td>
<td>EV</td>
<td>✓</td>
<td>EV</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Unavailable</td>
<td>EV</td>
<td>✓</td>
<td>–</td>
<td>x</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>OCSP</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Int. 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revoked</td>
<td>EV</td>
<td>EV</td>
<td>EV</td>
<td>EV</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Unavailable</td>
<td>EV</td>
<td>✓</td>
<td>–</td>
<td>x</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Int. 2+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revoked</td>
<td>EV</td>
<td>EV</td>
<td>EV</td>
<td>EV</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Unavailable</td>
<td>EV</td>
<td>✓</td>
<td>–</td>
<td>x</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Leaf</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revoked</td>
<td>EV</td>
<td>✓</td>
<td>EV</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Unavailable</td>
<td>EV</td>
<td>✓</td>
<td>–</td>
<td>x</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>OCSP Stapling</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Request OCSP Staple</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Respect Revoked Staple</td>
<td>x</td>
<td>✓</td>
<td>–</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Browser developers are not doing what the PKI needs them to do.
Subject Alternate Name (SAN) Lists

Spirit:

Multiple names for the same organization
Subject Alternate Name (SAN) Lists

**Spirit:**
Multiple names for the same organization

**Practice:**
Different organizations lumped together
Subject Alternate Name (SAN) Lists

Spirit: Multiple names for the same organization

Practice: Different organizations lumped together
Subject Alternate Name (SAN) Lists

**Spirit:** Multiple names for the same organization

**Practice:** Different organizations lumped together
Subject Alternate Name (SAN) Lists

Spirit: Multiple names for the same organization

Practice: Different organizations lumped together
Subject Alternate Name (SAN) Lists

Spirit: Multiple names for the same organization

Practice: Different organizations lumped together

Who gets the private key?
Who manages it?

Cruise-liner Certificate
How prevalent is key sharing?

CDF

Number of Third-Party Hosting Providers Used

Organizations

0

1

10

100

1000

10000

100000
How prevalent is key sharing?

CDF

Number of Third-Party Hosting Providers Used

Organizations

0
1
10
100
1000
10000
100000

0.0
0.2
0.4
0.6
0.8
1.0
How prevalent is key sharing?

CDF

Number of Third-Party Hosting Providers Used

Organizations

23.5% Self-hosted
How prevalent is key sharing?

- 23.5% Self-hosted
- 76.5% share at least 1 key
How prevalent is key sharing?

Who?

76.5% share at least 1 key

23.5% Self-hosted

CDF

Number of Third-Party Hosting Providers Used

Organizations
Who shares?

Fraction of Domains Hosted on Third-party Providers vs. Alexa Site Rank (bins of 10,000)
Who shares?

Fraction of Domains Hosted on Third-party Providers

Alexa Site Rank (bins of 10,000)

At least one key shared
All keys shared
Who shares?

43.2% (of Top 10k) share at least one

At least one key shared
All keys shared
Who shares?

43.2% (of Top 10k) share at least one

22.4% share all
Who shares?

Key sharing is common across the Internet.

- **43.2% (of Top 10k)** share at least one key.
- **22.4%** share all keys.

Key sharing is common across the Internet.
Does key sharing make enticing attack targets?
Does key sharing make enticing attack targets?

Cumulative Fraction of Domains’ Keys Acquired

Number of Hosting Providers Compromised

Alexa Top 1k
Alexa Top 1m
All Domains
Does key sharing make enticing attack targets?

Cumulative Fraction of Domains' Keys Acquired

Number of Hosting Providers Compromised

60% of Top 1K, same provider
Does key sharing make enticing attack targets?

Cumulative Fraction of Domains’ Keys Acquired

Number of Hosting Providers Compromised

- Alexa Top 1k
- Alexa Top 1m
- All Domains

60% of Top 1K, same provider
Does key sharing make enticing attack targets?

Cumulative Fraction of Domains’ Keys Acquired

Number of Hosting Providers Compromised

 ActiveRecord

60% of Top 1K, same provider

>40% of all sites, 10 providers

Alexa Top 1k
Alexa Top 1m
All Domains
Does key sharing make enticing attack targets?

Popular hosting services are prime targets for attack.
POOR CERTIFICATE MANAGEMENT

Websites aren’t properly revoking their certificates

Browsers aren’t properly checking for revocations

Websites aren’t keeping their secret keys secret
POOR CERTIFICATE MANAGEMENT

Websites aren’t properly revoking their certificates

Browsers aren’t properly checking for revocations

Websites aren’t keeping their secret keys secret

Why?

CAs have incentive to introduce disincentives (bandwidth costs)

Websites have disincentive to do the right thing (CAs charge; key management hard)

Browsers have a disincentive to do the right thing (page load times)