New Crypto 101

Internet2 Technology Exchange
Indianapolis, IN, Oct 30th, 2014

White River Ballroom D, 8:30AM

Joe St Sauver, Ph.D. (joe@stsauver.com)
https://www.stsauver.com/joe/new-crypto-101/

Disclaimer: all opinions strictly my own.

Introduction

- Recent revelations (such as the Snowden NSA-related disclosures as well as multiple high profile SSL/TLS vulnerabilities) have increased the need for all network engineers and system admins to understand basic emerging cryptographic best practices.
- This talk is meant to provide a brief introduction to the critical bits you need to understand, even if you're not a "crypto person" and even if you have a million other things to worry about, too.
- We also will NOT assume that you're a crypto math geek.

If You Do Only ONE Thing

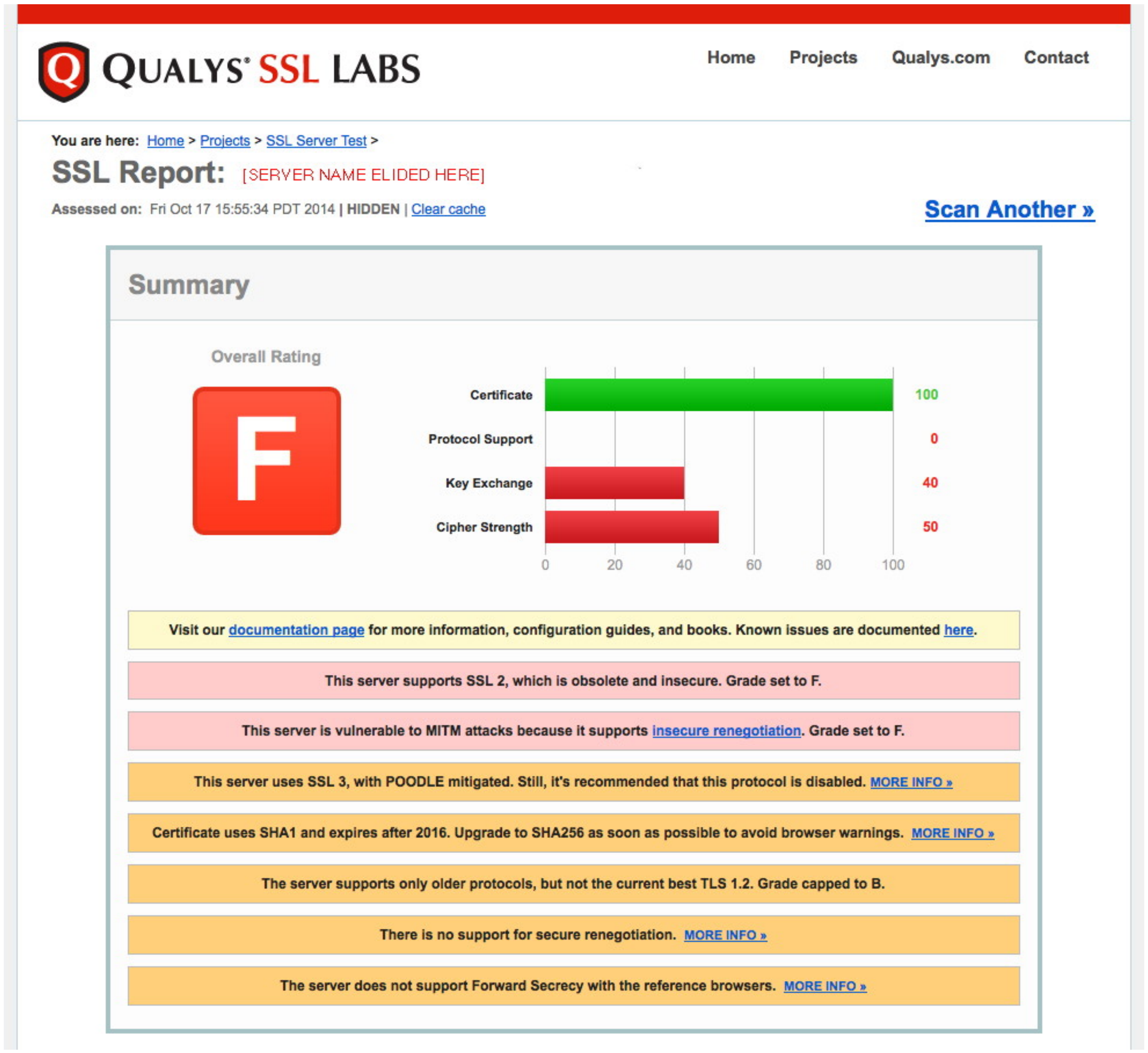
PLEASE check your "secure" web servers using the free Qualys SSL Tester:

https://www.ssllabs.com/ssltest/

➔ Note: you probably have MULTIPLE "secure" servers on your campus, not just https://www.whatever.edu/

Be sure you check ALL of them. 🡐

DON'T LET *THIS* BE *YOUR* SITE'S REPORT!



You are here: Home > Projects > SSL Server Test >

SSL Report: [SERVER NAME ELIDED HERE]

Assessed on: Fri Oct 17 15:55:34 PDT 2014 | HIDDEN | Clear cache

Scan Another »

Summary

Overall Rating

F

Certificate	100
Protocol Support	0
Key Exchange	40
Cipher Strength	50

Visit our documentation page for more information, configuration guides, and books. Known issues are documented here.

This server supports SSL 2, which is obsolete and insecure. Grade set to F.

This server is vulnerable to MITM attacks because it supports insecure renegotiation. Grade set to F.

This server uses SSL 3, with POODLE mitigated. Still, it's recommended that this protocol is disabled. MORE INFO »

Certificate uses SHA1 and expires after 2016. Upgrade to SHA256 as soon as possible to avoid browser warnings. MORE INFO »

The server supports only older protocols, but not the current best TLS 1.2. Grade capped to B.

There is no support for secure renegotiation. MORE INFO »

The server does not support Forward Secrecy with the reference browsers. MORE INFO »

Some Things To Note About That "F" Report

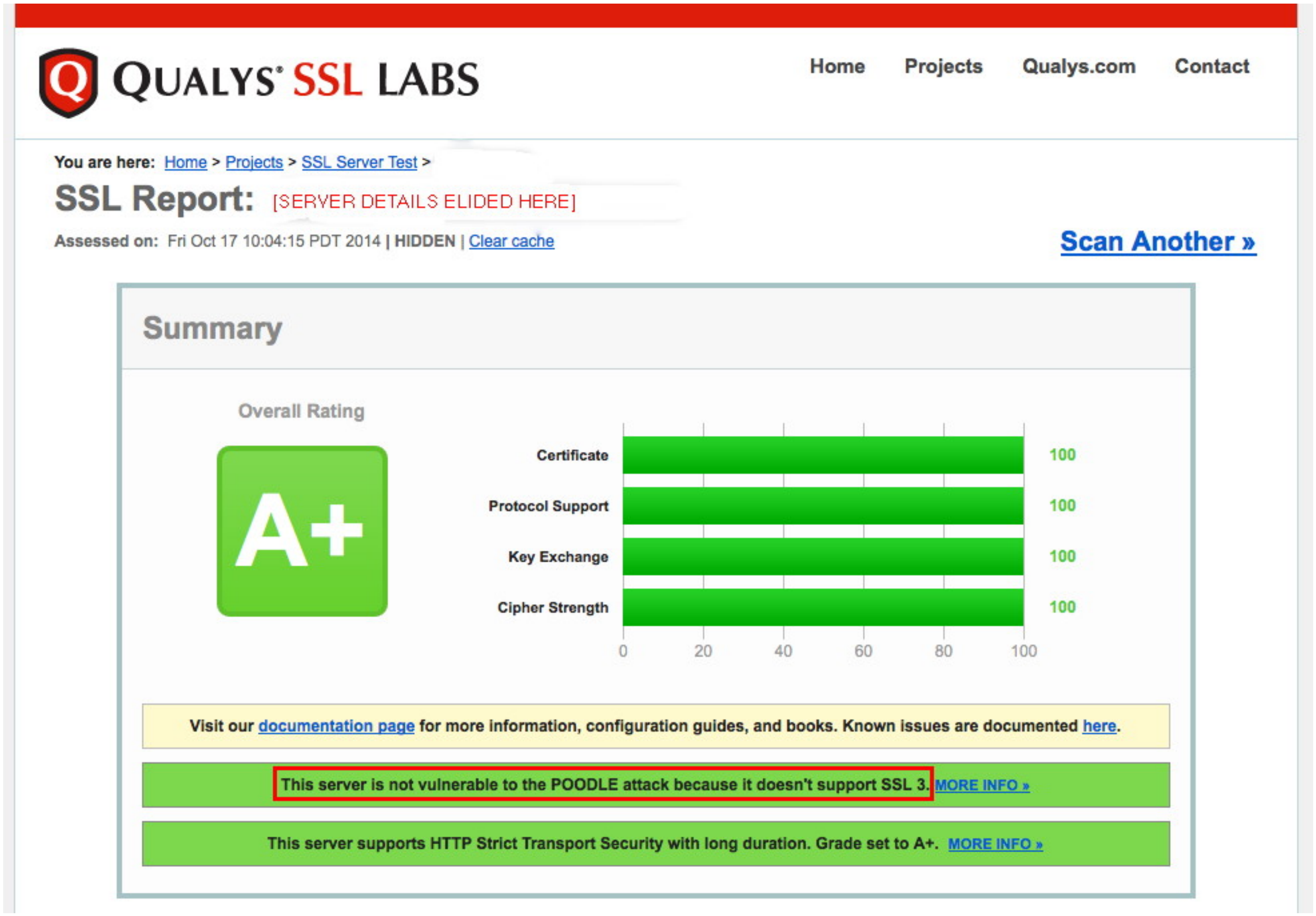
- That server's certificate was fine (THAT received a perfect score). The operator of this server also DID take steps to mitigate the recent high profile POODLE vulnerability (that's also great).

- HOWEVER, because they didn't fix all the **ADDITIONAL** vulnerabilities associated with their SSL/TLS installation, their **overall cryptographic security was AWFUL** – and that's why they ended up with a big fat **"F"**

- Bottom line, that "secure" server is actually highly **INSECURE** (and yes, I've reported it to that organization so it can be fixed)

- You do NOT have to just shrug and accept horrible SSL/TLS insecurities. You can get secure, and even get a perfect score.

The Sort of Report You *DO* Want to See



QUALYS SSL LABS

Home Projects Qualys.com Contact

You are here: Home > Projects > SSL Server Test >

SSL Report: [SERVER DETAILS ELIDED HERE]

Assessed on: Fri Oct 17 10:04:15 PDT 2014 | HIDDEN | Clear cache

Scan Another »

Summary

Overall Rating

A+

Certificate	100
Protocol Support	100
Key Exchange	100
Cipher Strength	100

Visit our documentation page for more information, configuration guides, and books. Known issues are documented here.

This server is not vulnerable to the POODLE attack because it doesn't support SSL 3. MORE INFO »

This server supports HTTP Strict Transport Security with long duration. Grade set to A+. MORE INFO »

Getting From An "F" to An "A+"

- Let me emphasize that while we talk about "F" "grades" and "A+" "grades," the "grades" themselves are NOT the actual objective. Those "grades" are just a proxy for the actual objective -- having a cryptographically secure system. The grades just provide an easy way for us to "keep score" or "measure how we're doing."

- The process of fixing a cryptographically insecure server isn't magic. There are a relatively small and straightforward set of steps that mere mortals can and should take. YOU should take these steps for all of your secure servers.

- We're going to go over what's involved, organized around the areas shown in the Qualys tester. We're also going to cover some additional cryptographic topics that are new/timely.

Part 1. Your Server's Certificate

Certificates Tie an "Identity" to a Key Pair

- Whenever we talk about web crypto, certificates are the first thing that always seem to come up.

- When you set up a secure web server, one of the tasks you'll perform is creating a **public/private key pair**. You'll do this as part of creating a "CSR" or certificate signing request.

- That public/private key pair handles the technical "heavy lifting" associated with your web site's crypto.

- So what role does the certificate serve? Well, your certificate ties an **identity** to the corresponding public/private key pair. An "identity" might be just a domain name, or it might be an actual organizational name.

Please Do NOT Use a Self-Signed Cert

- **To be trusted by common web browsers, your certificate needs come from a recognized certificate authority (CA).**
- 310 universities (and university systems) currently get their certificates from the InCommon Certificate Service (which resells Comodo certificates), see http://www.incommon.org/certificates/
- Others may prefer to just buy certificates "ala carte." For example, http://www.ssls.com/ currently sells globally-trusted Comodo "PositiveSSL" certificates for just **$4.99/year** (assuming you buy a five year cert).
- **The "A+"-rated server review shown on slide 6 is proof that you do NOT need to spend a lot of money when buying a cert... That "A+" report was for a server that was using an inexpensive $4.99/year cert.**
- Cost is no longer a reasonable justification for using a self-signed (and globally **UNtrusted**) certificate on your server!

So Why Do Some People Pay More for Certs?

- If a $4.99/year cert will do the job, why do some people pay a *whole lot more* for the same sort of certificate?

- Well, in some cases, purchasers seem to believe that a more expensive certificate is somehow "better" than a less expensive certificate (just like more expensive vodka is somehow "better" than less expensive vodka :-;)

- In reality, as long as your certificate chains to a root certificate trusted by your users' browsers, you're going to be okay.

- There ***<u>are</u>*** some different types of certs, however, so just to eliminate any confusion, let's talk about those differences briefly.

Domain Validated Certs

- *Domain validation (DV) is based around the process of an applicant demonstrating* ***technical*** *control over a domain.*
- For example, to do DV, you might need to *reply to email sent to an administrative contact address*, or demonstrate the ability to create a *specified web page*, or show the ability to create a *specified DNS entry* – if you can one of those things, CAs believe you've shown "technical control" over that domain, and thus should be allowed to get a cert for that domain. For most certs, that's **all** that's done. There's no attempt made to verify the identity of the company that owns the domain. (That's why certs that only do DV don't show an org's "real name" in the browser)
- DV, while not perfect, is still important -- it ensures that an applicant that doesn't technically control a particular domain can't get a cert that they shouldn't be getting (Random people shouldn't be able to get a cert for google.com, whitehouse.gov, etc.!)

Organizational Validation

- All InCommon Certificate Program certs are protected with domain validation, but that's only PART of a more complex process of doing *organizational validation.*

- Organizational validation requires *additional steps*, including verification of the domain's whois registrant information, and confirmation that a requester is authorized to request certs for his or her organization.

- This additional processing typically requires **manual intervention and review**, and that's why you will be made to (at least temporarily) remove any privacy or proxy registration protections you may have in place while organizational ownership of the domain is confirmed.

Extended Validation Certs

- Going beyond organizational validation certs, there's an even cooler option: "extended validation" ("green bar") certificates.
- Extended validation certificates are **normally quite expensive** (some certificate authorities charge as much as ~$1,499 for just one of them), but they're bundled at no additional charge for InCommon Certificate Service subscribers – all you "pay" is the time associated with completing the required extended validation paperwork.
- Extended validation certs aren't ***cryptographically*** "stronger" than a regular domain validated cert, but because they involve much tighter identity proofing, they're rarely seen associated with abuse. Knowledgeable users may thus be more inclined to trust your site if it has gone through the trouble to get an EV ("green bar") cert.
- Downsides to EV certs (besides the paperwork)? EV certs have a max lifetime of two years, and you can't get wildcard EV certs.

[Speaking of Wildcard Certs: Please Avoid Them!]

- If you hate screwing around ordering certs, the thought of getting one "wildcard" cert that you could then use for ALL your domain's systems (e.g., *.example.edu) might be superficially attractive (and relatively cheap). Resist the temptation!
- Wildcard certs have a tendency to be used on both high security systems (like your school's ERP system) as well as less critical systems (like the local student bridge club's web server), ugh.
- If you have a wildcard cert installed on hundreds of campus servers and any ONE of those systems gets breached, you're looking at a fire drill: you'll need to generate a new public/private key pair and then get your new certificate installed on ALL of those hundreds of systems, rather than just fixing one system.
- If you just need a single cert to work for a defined set of hosts, a multi domain cert (for up to 100 names) is a better alternative.

Part 2. "Advanced" Cert Topics

(a) Support for Raw IPv6 Addresses in Certs?

- It is currently possible to get SSL/TLS certificates for raw IPv4 addresses from most Certificate Authorities. However:

  -- You often cannot request certs for **BLOCKS** of IP addresses (e.g., you can't specify an arbitrary CIDR block, or even just a starting and ending IP address). This should be fixed.

  -- You also CAN'T currently get SSL/TLS certificates for raw **IPv6 addresses** (even for single IPv6 addresses). Given that we're rapidly exhausting IPv4 address space, shouldn't we be working on this?

  And ESPECIALLY in IPv6 land, I REALLY want to be able to specify network blocks (e.g., IPv6 /64's) rather than just one potentially LONG raw IPv6 address at a time!

(b) Internal Domain Names/Private Address Space

- Some sites make up their own domain names (often things like "dot local" or "dot intranet") for use with internal sensitive systems. These systems also often live in non-routable RFC1918 address space. These internal systems are intentionally NOT meant to be globally accessible.
- Nonetheless, users often are interested in globally trusted certs that include those names (or those non-routable IP addresses)
- Problem: you and I may BOTH have servers with the same made-up internal names (or the same non-routable IP addresses) even though your system isn't the same as mine.
- If a certificate authority issues a cert to either of us, that cert could potentially be used beyond what's intended. The community is uncomfortable with that, and as a result, globally trusted certs for internal domain names are being phased out as a policy matter.

CAB Forum Policy on Internal Names

- **Globally trusted certs for internal domain names and/or non-routable address space may NOT be issued with expiration dates later than November 1st, 2015.**
  See https://cabforum.org/internal-names/

- Implicitly, if you're getting a one year cert for internal domains today, you're basically bumping into that expiration (or nearly enough)

- Either plan to rename/renumber those internal systems into name space or address space you explicitly control,

  OR replace globally trusted certs with locally issued/locally trusted certs.

Replace .local with .local.<yourdomain>.edu

- If you replace your use of a locally self-assigned TLD with a subdomain under your top level domain, you can continue to get and use globally trusted certs

- The trick then becomes ensuring that remote sites can't resolve those names, nor access the corresponding internal-only network segments. (This should not be much of a challenge for most DNS teams, assuming local recursive resolvers and authoritative name servers are properly architectected)

- Do not attempt to use a registered but totally offline domain – you still need to be able to DCV the domain, either via email, web or DNS

Replace RFC 1918 Address Space with Real IPs

- Similarly, if you currently use RFC 1918 address space for globally trusted certs, you should be planning to replace those RFC 1918 addresses with real IP addresses your school explicitly controls.

- Obviously you or your network engineers should take appropriate steps to ensure that those IP addresses aren't accessible from off campus, nor to other unauthorized users.

PAY ATTENTION: Potential Side Effect of Deploying Locally Issued Certs for Intranets

- If you decide to create your own local certificate authority, and force local users to accept your local self-signed root cert, you've just created an environment with a very high potential for misuse.
- Specifically, if you force your users your local root, you could then issue a locally-trusted cert for * (e.g., match all sites) and then potentially Man In The Middle ***ALL*** local traffic via a web gateway – yowza, Batman!
- You should be highly sensitive to the possibility of this occuring. At least some security officers totally freak out at the thought that they currently have zero visibility into user SSL/TLS traffic ("Malware! Phishing! Other bad content! Data exfiltration! And we can't see it or manage it on the network if it's encrypted!").
- The institutional risk implications associated with MITM'ing all site SSL/TLS traffic are HUGE. Do NOT do this.

(c) CRITICAL: SHA-1 Certificate Signatures

- When a certificate authority issues a certificate, they "digitally sign it" by computing a hash. Digitally signing a certificate protects the certificate against tampering (if anything gets altered, the digital signature becomes invalid).

- Even as late as February 2014, virtually all (literally 98+% of all certificates) were still signed with a SHA-1 signature.* The problem? In January 2011, NIST 800-131A** stated that "SHA-1 shall not be used for digital signatures after December 31, 2013."

-----

* *NIST continues using SHA-1 algorithm after banning it,* http://news.netcraft.com/archives/2014/02/04/nist-continues-using-sha-1-algorithm-after-banning-it.html

** *Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths,* January 2011, http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf

Microsoft and the SHA-1 $\rightarrow$ SHA-2 Migration

- What was the "holdup" moving to SHA-2? Well, a primary gating factor was that some versions of Windows XP didn't support SHA-2. This problem (more or less*) went away when Microsoft declared Windows XP end-of-support on April 8th, 2014.**
- Shortly thereafter, on November 12th, 2013, Microsoft announced its "SHA1 Deprecation Policy." **That policy required CAs participating in the Microsoft Root Certificate Program** (e.g., effectively all major globally-trusted Certificate Authorities) **to stop issuing SHA1-signed certificates by January 1st, 2016.*****
That date was generally believed to be pretty reasonable/doable.

-----

* *Support for Windows XP is over, but China still has 200 million PCs using it,*
http://www.techinasia.com/windows-xp-now-dead-but-200-million-machines-in-china-still-using-it/

** *Support for Windows XP has ended,*
http://www.microsoft.com/en-us/windows/enterprise/end-of-support.aspx

*** *SHA1 Deprecation Policy,*
http://blogs.technet.com/b/pki/archive/2013/11/12/sha1-deprecation-policy.aspx

Google Chrome, On The Other Hand, Announced A Much More Aggressive SHA-1 Time Schedule

- The Google Chrome people, makers of the world's most popular browser, announced that **as of Chrome version 39 (scheduled for release in NOVEMBER 2014), any https sites which have certs using SHA-1 AND which expire after January 1st, 2017 will no longer be shown as fully trustworthy** in the Chrome browser's user interface.* Obviously this would include most recently-issued three-year SHA-1 certificates.
- **In general, this means that most sites will want to replace their SHA-1 certs with new SHA-2 certs as soon as possible. Most trusted certificate sources, including the InCommon Certificate Program, now have SHA-2 certs available.**
- **RECOMMENDATION: START USING SHA-2 CERTS!**

-----

* http://googleonlinesecurity.blogspot.com/2014/09/gradually-sunsetting-sha-1.html

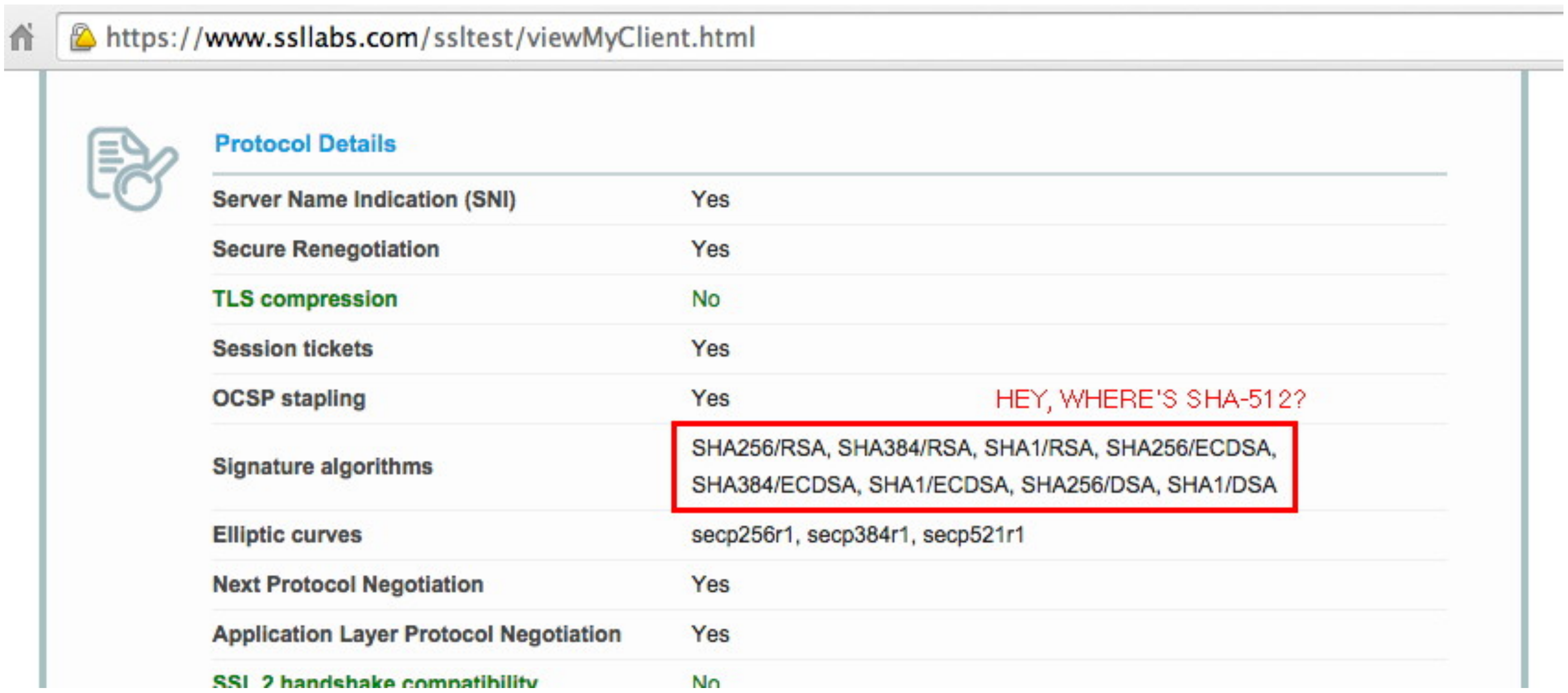
Caveats Associated With Moving to SHA-2

- The decision to move to SHA-2 is not without potential "gotchas." In particular, **some older systems or software may not be fully SHA-2 compatible.** My favorite page for SHA-2 compatibility information is probably GlobalSign's "SHA-256 Compatibility," https://support.globalsign.com/customer/portal/articles/1499561-sha-256-compatibility
- **If you are an InCommon Certificate Subscriber and have a system that isn't SHA-2 compatible:** the InCommon Certificate Service **is** currently still allowing its subscribers to get a **one year SHA-1** certificate in addition to the new now-industry-standard SHA-2 1, 2 or 3 year certificates. If you need to use one of those one year SHA-1 cert because of SHA-2 compatibility issues, please use your year's "grace period" to figure out how you're going to handle your SHA-2 incompatible system. **ALL OTHER SERVERS SHOULD NOW BE USING SHA-2 CERTS.**

[Aside: SHA-256, SHA-384, and SHA-512]

- SHA-2 is not just one hash function, it's actually a family of hashes, including SHA-256, SHA-384, and SHA-512.
  While you might expect implementations to support *all* members of the SHA-2 hash family if they support *any* of them, support for SHA-512 (at least when used with TLS 1.2) is incomplete.
- InCommon discovered this with Comodo based on user reports after we deployed a SHA-512 intermediate cert (e.g., the "glue" between the trusted root cert and the end-entity cert). That SHA-512 intermediate worked fine for most modern systems and browsers, but had problems with some others, even with patches applied (e.g., see "SHA512 is disabled in Windows when you use TLS 1.2," http://support.microsoft.com/kb/2973337)
- As a result, InCommon worked with Comodo to drop our SHA-2 intermediate cert back to SHA-384. That SHA-384 intermediate is now working as expected on all SHA-2 capable systems.

Sample Qualys SSL Client Check



https://www.ssllabs.com/ssltest/viewMyClient.html

Protocol Details

Server Name Indication (SNI)	Yes
Secure Renegotiation	Yes
TLS compression	No
Session tickets	Yes
OCSP stapling	Yes
Signature algorithms	SHA256/RSA, SHA384/RSA, SHA1/RSA, SHA256/ECDSA, SHA384/ECDSA, SHA1/ECDSA, SHA256/DSA, SHA1/DSA
Elliptic curves	secp256r1, secp384r1, secp521r1
Next Protocol Negotiation	Yes
Application Layer Protocol Negotiation	Yes
SSL 2 handshake compatibility	No

HEY, WHERE'S SHA-512?

Discussion of the ambiguity around the meaning of the signature algorithm field can be seen in this discussion thread:

http://www.ietf.org/mail-archive/web/tls/current/msg13599.html

(d) Strength of Your Public Key Crypto

- Remember how we mentioned that you create a public/private key pair when you create a certificate signing request?

- Those keys are virtually always **RSA-2048 bit** keys at this point.

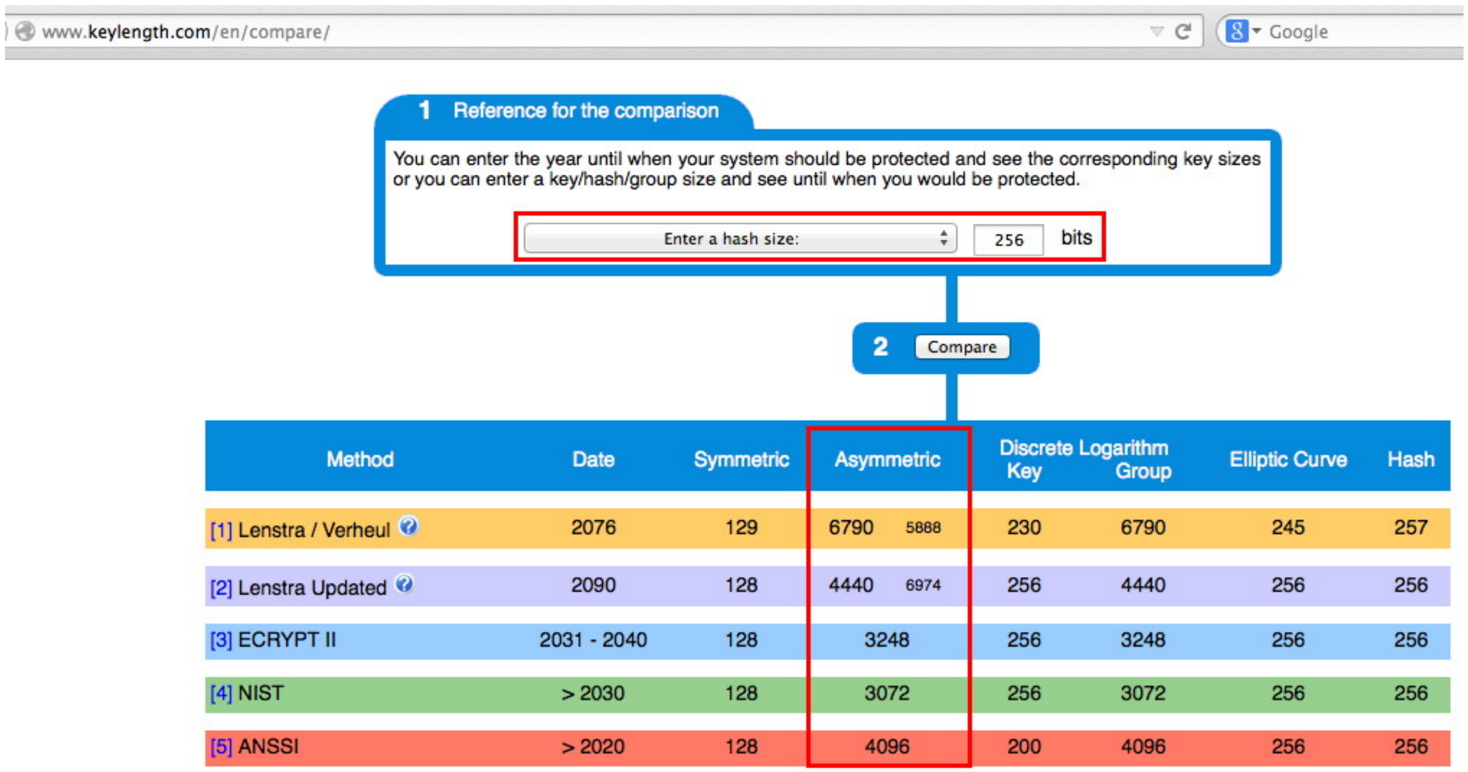
- The strength of those keys plays a major role in determining the strength of your SSL/TLS crypto overall.

- Show how did you pick RSA-2048? In most cases, you're using RSA-2048 because... well, because everyone else is, too.

- That doesn't mean it's the right choice. Let's assume that we want all parts of our crypto to be of roughly the same strength, instead.

If We Need SHA-2 For Our Hashes...

- When you replace your SHA-1 cert with a stronger SHA-2 cert, this may also be a good time to think about whether your want to stay with RSA-2048, or if you want to start using RSA-4096 to keep your asymmetric crypto at an "equivalent level of strength."


www.keylength.com/en/compare/

1 Reference for the comparison

You can enter the year until when your system should be protected and see the corresponding key sizes or you can enter a key/hash/group size and see until when you would be protected.

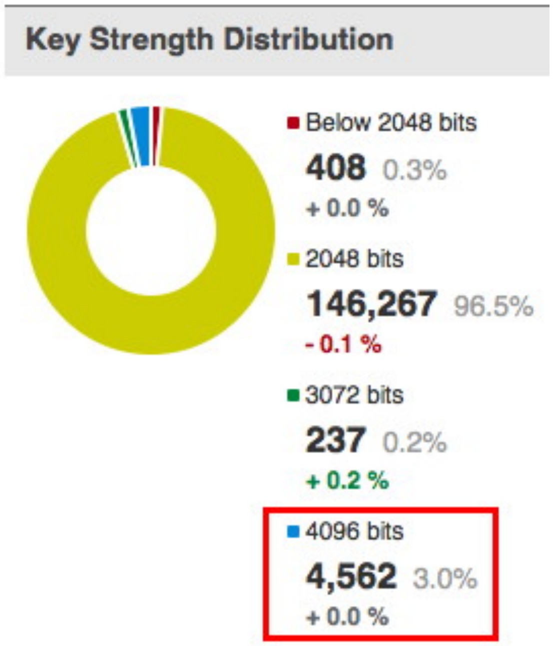
Enter a hash size: 256 bits

2 Compare

Method	Date	Symmetric	Asymmetric		Discrete Logarithm Key	Discrete Logarithm Group	Elliptic Curve	Hash
[1] Lenstra / Verheul	2076	129	6790	5888	230	6790	245	257
[2] Lenstra Updated	2090	128	4440	6974	256	4440	256	256
[3] ECRYPT II	2031 - 2040	128	3248		256	3248	256	256
[4] NIST	> 2030	128	3072		256	3072	256	256
[5] ANSSI	> 2020	128	4096		200	4096	256	256

Roughly 3% of Site Are ALREADY Using RSA-4096 Bit Keys...

- Most SSL/TLS server certs currently use RSA-2048 (see the "SSLPulse" results to the right). RSA-2048 appears to be okay for now, however some sites ARE beginning to migrate to RSA-4096. Many CAs, including Comodo, will allow you to submit a CSR for either RSA-2048 or RSA-4096 (just use the appropriate command line flag when generating your CSR with OpenSSL).
- Note: if you do elect to just stay with RSA-2048, you will NOT be able to get a "perfect score" from the Qualys SSL Tester.
- Please also note: using RSA-4096 <u>will</u> likely reduce the capacity of your server; be careful if you're running on the "ragged edge."



RSA-2048 vs RSA-4096 Relative Performance

- For one sample system:

```
$ openssl speed rsa2048 rsa4096
[snip]
                  sign        verify        sign/s  verify/s
rsa 2048 bits 0.002786s 0.000089s    358.9  11218.8
rsa 4096 bits 0.020470s 0.000329s     48.9   3041.2
```

- Doing the math:

  -- RSA-4096 **signing** takes roughly **7.3X** as long as RSA-2048
  -- RSA-4096 **verifying** takes roughly **3.7X** as long as RSA-2048

- Your mileage WILL vary. **Test your system to avoid surprises.**

(e) Trust Chains: Mind Your Intermediates

- The old days (when trusted root certs directly signed end-entity certs) are long gone. These days, trusted root certs are used to sign an *intermediate cert*, and that intermediate cert is then used to sign the end-entity cert used by your servers. (Sometimes there may even be a 2nd or 3rd intermediate cert in that trust chain)
- One sometimes overlooked fact is that YOUR SERVER needs to provide any required intermediate certs as well as its own server cert. (The root cert's already present in the browser trust anchors, so you don't need to provide that) While you might expect the browser to be able to "automagically" find and download any required (but missing) intermediates, it just doesn't work that way.
- **Therefore, as part of installing your certs, be SURE you also install the required intermediate(s),** and in the **right order** (if you're using more than one intermediate cert). [Consult your server's documentation for the correct intermediate ordering]

[Can There Be Multiple Trust Paths? Yes]

- At least in the case of InCommon's SHA-2 certs, there are actually two paths to a trusted root that are provided.
- The **short path** is all that's needed for browsers that already trust the USERTrust RSA Certification Authority [SHA-2] trust anchor. The **extended path** (with cross-signing to Comodo's AddTrust External CA Root) is a temporary solution for other browsers.
- Short path:

  USERTrust RSA Certification Authority [SHA-2]

  InCommon RSA Server CA [SHA-2]

  End-Entity Certificate (your server's cert) [SHA-2]
- Extended path:

  ***AddTrust External CA Root* [SHA-1]**

  USERTrust RSA Certification Authority [SHA-2]

  InCommon RSA Server CA [SHA-2]

  End-Entity Certificate (your server's cert) [SHA-2]

"But Joe! There's A *SHA-1 Root* In That Chain!"

- Some users carefully check the certificate chain used by their certificates. If so, they may noticed that while their **end entity cert** uses SHA-2, and their **intermediate cert(s)** use SHA-2, those certs actually chain back to a SHA-1 trusted root in some cases. For example, that's the case for the AddTrust External CA Root in the case just discussed.

- **Because of how root certs come to be trusted, this is not an issue.**

- As browser vendors update their trust anchors, the need to rely on legacy SHA-1 signed root certs will go away (this is expected to be completed in Comodo's case sometime after the first of the year).

Finally, It SHOULD Go Without Saying, But...

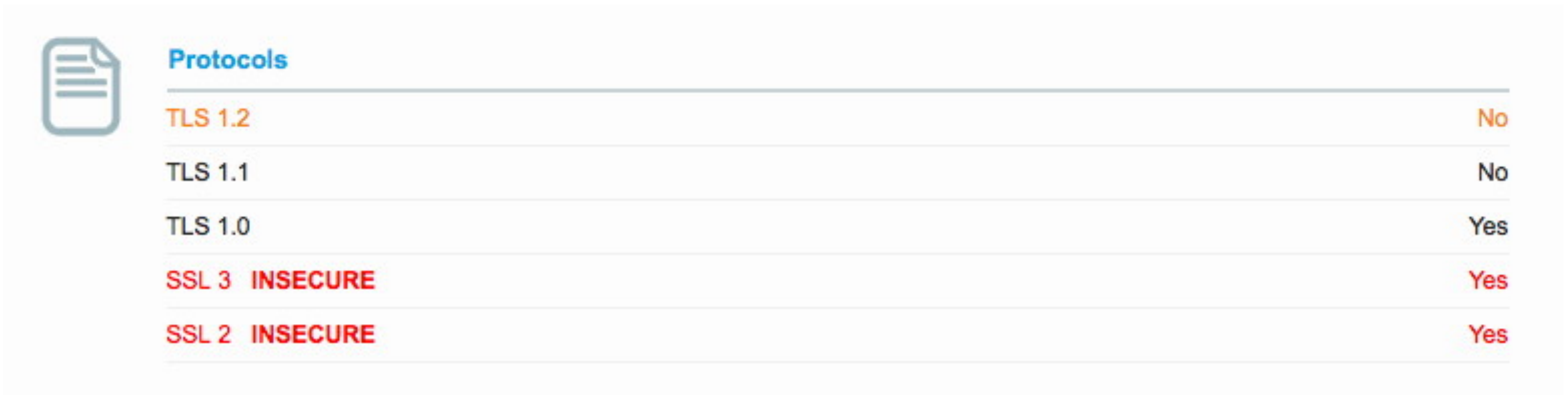
- The name on the cert MUST MATCH the name of the server it's used on
- The cert must NOT be expired (and shouldn't be used BEFORE it's valid, either!)
- The cert must NOT be revoked, either.
- If any of the preceding conditions aren't met, your cert won't be trusted.
- All that said, enough about certs! Let's move on to talk about the SSL/TLS protocols themselves.

Part 3. SSL/TLS Protocols

(a) SSL/TLS and Compatibility

- SSL/TLS has evolved over the years, typically in response to vulnerabilities discovered in whatever was the then-current version of those protocols.
- Some web servers support "all" versions of SSL/TLS for "compatibility reasons." **PLEASE DON'T.**
- **SSLv2** is ancient and grossly insecure and **MUST NOT** be used any more.
- **SSLv3**, the focus of the recent **POODLE vulnerability**, is now roughly 16 years old and **ALSO MUST NOT** be used any more.
- Strive to ALSO avoid offering **TLSv1.1** (or **TLSv1.0**) if at all possible.
- If you're running a current version of your server's crypto libraries, and if your users employ a modern web browser, **USE TLS 1.2**

An Example Of A Disappointing Qualys SSL Server TLS Protocol Report



Protocols	
TLS 1.2	No
TLS 1.1	No
TLS 1.0	Yes
SSL 3 **INSECURE**	Yes
SSL 2 **INSECURE**	Yes

Remember:

If you support SSLv2 (ancient and grossly insecure!), you'll get an "F" (and you deserve it!)

If you support SSLv3, for now, you won't get better than a "C" (and you WILL be vulnerable, so don't do SSLv3)

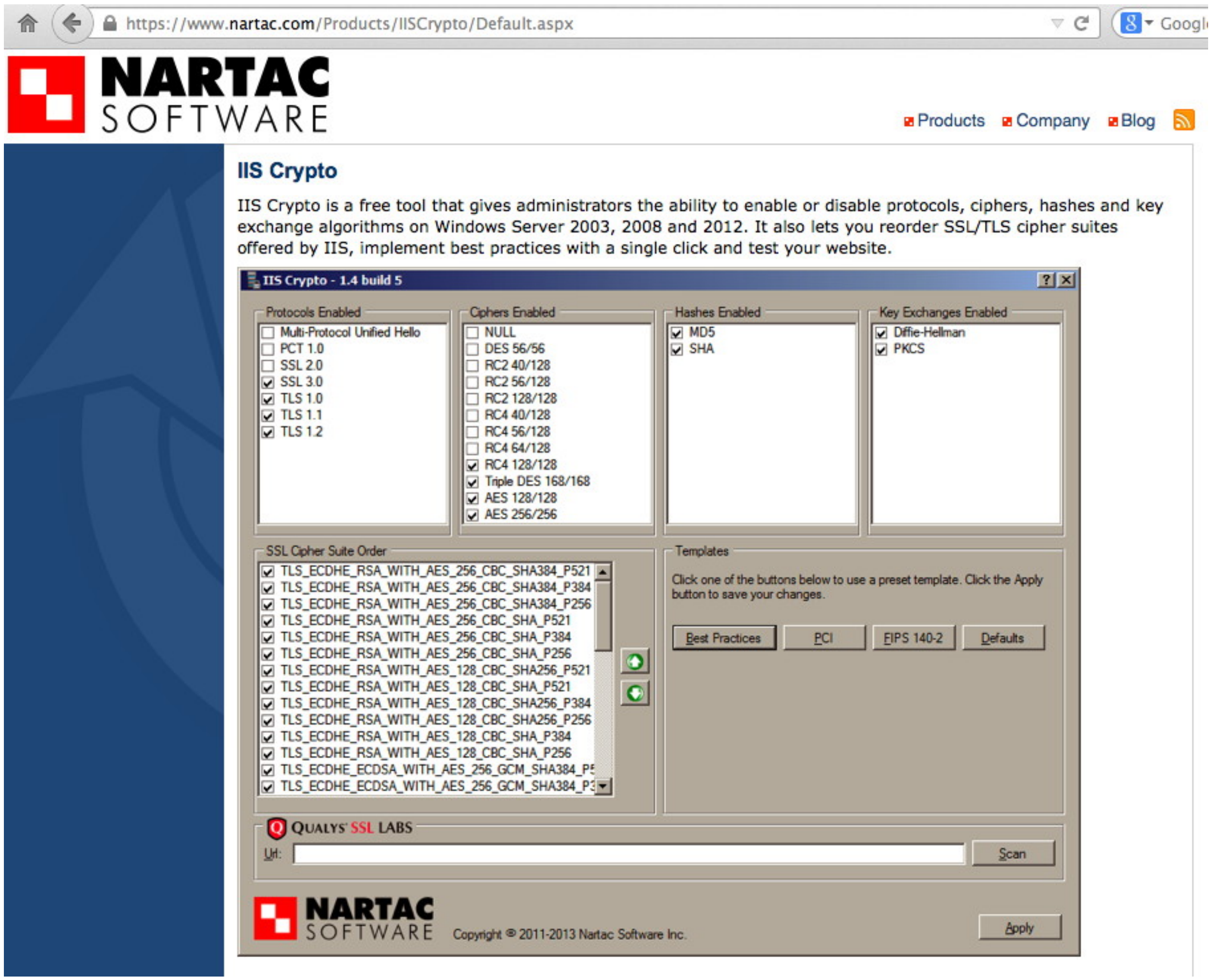
If you <u>don't</u> support TLSv1.2, you won't get better than a "B" on the Qualys Tester.

Tweaking Your Configuration to Use TLSv1.2

- This process is not magic. Ensure you're running the **latest version of your web server's software**. Again, as of the time this was written, nginx was on mainline release 1.7.6, see http://nginx.org/en/download.html (Yes, at this point I am now recommending nginx over Apache and (other server options) due to nginx's improved support for strong crypto/advanced protocols)
- Ensure you're running the **latest version of your crypto libraries**. At the time this was written, OpenSSL had released 1.0.1j on October 15th, 2014, see https://www.openssl.org/source/
- Consult your web server software's configuration guide for information on how to set the version(s) of SSL and TLS you want to use. Typically this is just one line in the config file. For example in nginx.conf, you simply use the command:

```
ssl_protocols TLSv1.2;
```

"Joe! What if I'm using MS IIS?"



https://www.nartac.com/Products/IISCrypto/Default.aspx

NARTAC SOFTWARE

Products Company Blog

IIS Crypto

IIS Crypto is a free tool that gives administrators the ability to enable or disable protocols, ciphers, hashes and key exchange algorithms on Windows Server 2003, 2008 and 2012. It also lets you reorder SSL/TLS cipher suites offered by IIS, implement best practices with a single click and test your website.

IIS Crypto - 1.4 build 5

Protocols Enabled
- [ ] Multi-Protocol Unified Hello
- [ ] PCT 1.0
- [ ] SSL 2.0
- [x] SSL 3.0
- [x] TLS 1.0
- [x] TLS 1.1
- [x] TLS 1.2

Ciphers Enabled
- [ ] NULL
- [ ] DES 56/56
- [ ] RC2 40/128
- [ ] RC2 56/128
- [ ] RC2 128/128
- [ ] RC4 40/128
- [ ] RC4 56/128
- [ ] RC4 64/128
- [x] RC4 128/128
- [x] Triple DES 168/168
- [x] AES 128/128
- [x] AES 256/256

Hashes Enabled
- [x] MD5
- [x] SHA

Key Exchanges Enabled
- [x] Diffie-Hellman
- [x] PKCS

SSL Cipher Suite Order
- [x] TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384_P521
- [x] TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384_P384
- [x] TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384_P256
- [x] TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA_P521
- [x] TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA_P384
- [x] TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA_P256
- [x] TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256_P521
- [x] TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA_P521
- [x] TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256_P384
- [x] TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256_P256
- [x] TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA_P384
- [x] TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA_P256
- [x] TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384_P5
- [x] TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384_P3

Templates

Click one of the buttons below to use a preset template. Click the Apply button to save your changes.

NARTAC SOFTWARE Copyright © 2011-2013 Nartac Software Inc.

Do I ***Really*** Need To Run The Latest Versions?

- **YES! YOU REALLY, *REALLY* DO.**

- While things like POODLE get a lot of "air play," there are many OTHER web security and crypto security issues that are quietly handled in updates. If you're NOT running the latest versions, you're likely vulnerable to one or more of those security issues.

- **The easiest way to avoid that is by running the latest version of your cryptographic libraries and the latest version of your web server's software.**

- That said, I know this discussion triggers the **"distro problem."** The "distro problem" is probably the leading reason why people don't run "safe and sane" SSL/TLS configurations.

What Is The "Distro Problem?"

- Many sites have elected to run a standardized "enterprise grade" distribution emphasizing **stability** above everything else.
- These enterprise grade distributions do NOT chase the latest version of every package, they typically only back port and merge major bug fixes and security-critical patches into their updates, with everything else waiting for the next major release.
- Mind bogglingly, crypto updates (or generic web server software updates) may not always be considered "security critical."
- Site may not even run the latest version, or even the latest *update* for whatever version they *are* using. Local policy may forbid sys admins from manually upgrading packages outside the distribution framework.
- **Recommendation:** run the **latest version** of the **latest distribution**, & encourage your vendor to at least keep the crypto and web server software in those builds completely current!

Not Convinced That Problems Exist In Earlier Versions of Crypto and Web Server Software?

- Check out the list of vulnerabilities for OpenSSL at https://www.openssl.org/news/vulnerabilities.html
- Check out the nginx security advisories at http://nginx.org/en/security_advisories.html
- Check out the Apache 2.4 httpd security advisories at http://httpd.apache.org/security/vulnerabilities_24.html
- Check out the Apache 2.2 httpd security advisories at http://httpd.apache.org/security/vulnerabilities_22.html
- **Etc., etc., etc! ALWAYS RUN THE LATEST VERSIONS!**

(b) HTTP Strict Transport Security

- Historically, only "sensitive" information was protected with SSL/TLS security, everything else was transmitted in plain text via regular HTTP.
- These days, now that we have a much better basis for assessing the extent to which our traffic is subject to pervasive monitoring, we know that we should really be encrypting ALL traffic, whether sensitive or seemingly routine.
- The HTTP Strict Transport Security policy mechanism essentially says, "For this domain, ALL traffic should be exchanged over an encrypted channel."
- As part of this process, webmasters will normally rewrite requests made over insecure (regular) http, forcing them to go over https
- This whole process is relatively straightforward in modern web servers (such as nginx)

Configuring Strict Transport Security in nginx

```
server {
   listen  80;
   rewrite ^(.*) https://$host$request_uri permanent;
        }
server {
   server_name  www.example.com;
   listen  443 ssl;
   add_header
      Strict-Transport-Security "max-age=31536000;
         includeSubdomains";
      }
```

CAUTION: After a browser sees that Strict-Transport-Security header, that browser will ONLY connect to that site via HTTPS for the specified # of seconds (31,536,000 seconds=roughly 1 year)

WITHOUT HTTP Strict Transport Security...

- If you don't use HTTP Strict Transport Security, it's extremely easy to end up with HTTPS "Mixed Content" warnings, e.g., a page that delivers a mixture of content that is – and ISN'T – protected by SSL/TLS. This is a VERY common problem.

- Ivan Ristic of Qualys states that "We tend to talk a lot about other aspects of SSL/TLS, but **mixed content is arguably the easiest way to completely mess up your web site encryption**." (see https://community.qualys.com/blogs/securitylabs/2014/03/19/ https-mixed-content-still-the-easiest-way-to-break-ssl)

- [And if you DON'T do HTTP Strict Transport Security, you'll also never get that A+ you want from Qualys.]

Joe! HTTP Strict Transport Sounds "Scary"...

- It really shouldn't. For example, if you follow the IETF work around HTTP/2, the next major version of HTTP, you will see that major browser vendors are indicating that they will ONLY support HTTP/2 over TLS secured connections.

- You might as well get used to working in what amounts to an HTTP strict transport security-equivalent world now, because the next version of HTTP will be operating in that environment by default.

- [In the interest of fairness, if you'd like to read the argument FOR allowing a NON-encrypted HTTP/2 mode for use with proxies, see https://github.com/http2/http2-spec/wiki/Proxy-User-Stories]

(c) Optional (But Worth Considering): SPDY

- Speaking of HTTP/2, the next version of HTTP, if you believe that ever-trustworthy oracle, Wikipedia, "is based on SPDY." (see http://en.wikipedia.org/wiki/HTTP/2 and http://en.wikipedia.org/wiki/SPDY)
- SPDY is a Google-originated effort to reduce web page load times while also improving web security. It is currently still somewhat experimental, but the latest version of most browsers support it.
- If you want to try it on your web site, you'll need to be running a SPDY-capable version of your crypto libraries, such as OpenSSL 1.01j, and a SPDY-capable web server, such as nginx 1.7.6.
- If building nginx from source, be sure to configure with **`--with-http_spdy_module`**
- In your nginx config file, specify spdy as an option to the listen command: `listen 443 ssl spdy;`

How Much SPDY'er is SPDY?

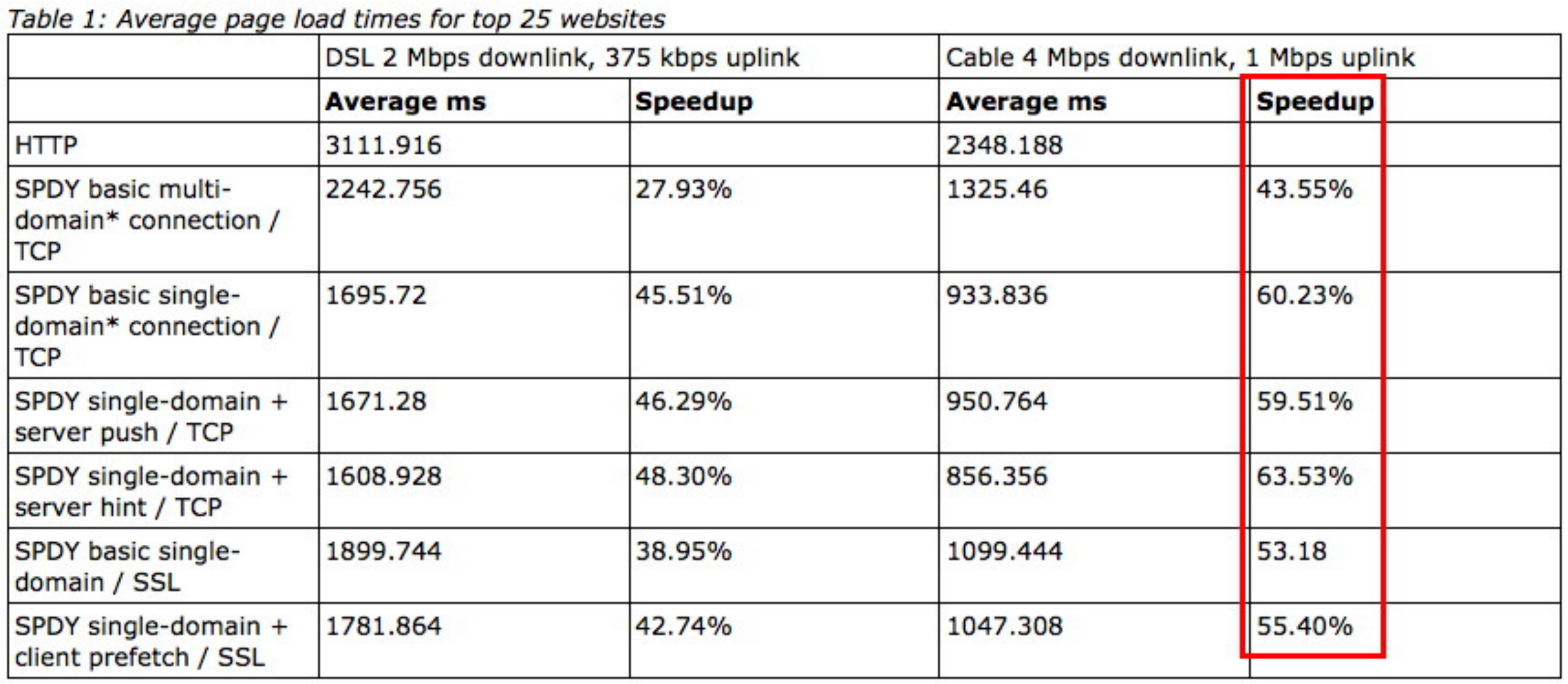


Table 1: Average page load times for top 25 websites

	DSL 2 Mbps downlink, 375 kbps uplink		Cable 4 Mbps downlink, 1 Mbps uplink	
	Average ms	**Speedup**	**Average ms**	**Speedup**
HTTP	3111.916		2348.188	
SPDY basic multi-domain* connection / TCP	2242.756	27.93%	1325.46	43.55%
SPDY basic single-domain* connection / TCP	1695.72	45.51%	933.836	60.23%
SPDY single-domain + server push / TCP	1671.28	46.29%	950.764	59.51%
SPDY single-domain + server hint / TCP	1608.928	48.30%	856.356	63.53%
SPDY basic single-domain / SSL	1899.744	38.95%	1099.444	53.18
SPDY single-domain + client prefetch / SSL	1781.864	42.74%	1047.308	55.40%

Source: http://www.chromium.org/spdy/spdy-whitepaper

(d) FIPS Ready

- One of the things you'll see in the SSL Labs report is a statement "FIPS Ready." Sometimes folks wonder what that involves. If so, check out:

  https://github.com/ssllabs/research/wiki/FIPS-Requirements

- Short form version:

  -- Trusted cert
  -- NO SSL v2 and NO SSL v3
  -- Strong private key (4096 bit RSA will be fine for now, if you followed that earlier recommendation)
  -- Strong cipher suite selection (the recommendations in this talk should leave you in great shape here, too)

(e) PCI Compliant

- Similarly, there's also a "PCI Compliant" line. To understand what's involved on that one, check out:

  https://github.com/ssllabs/research/wiki/PCI-SSL-Requirements

- If you are in good shape for the "FIPS Ready" item, you'll also have most of this one covered, but I'd also particularly draw your attention to the following additional item (among others):

  -- DH parameters 1024+ bits

  Again, if you're following the recommendations from later in this talk, you should end up with a PCI compliant server.

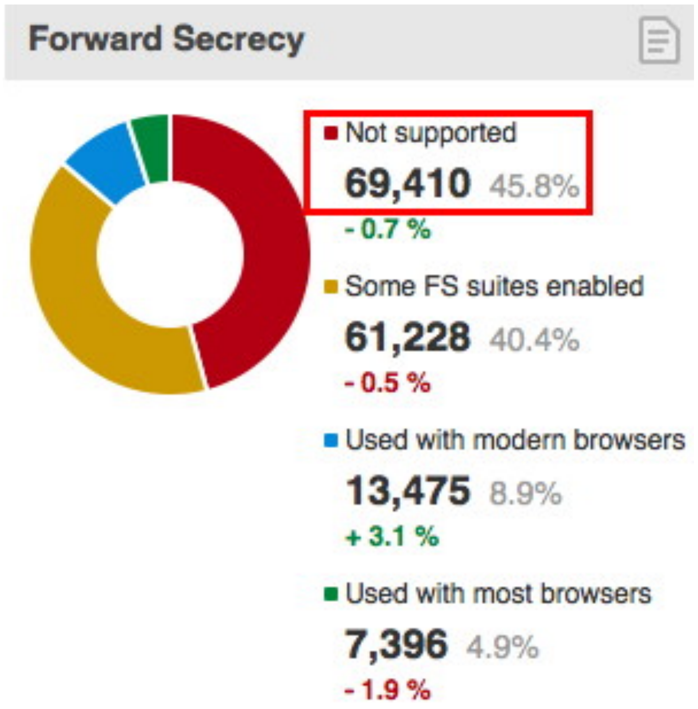
Part 4. Key Exchange

(a) The Two Jobs of the Key Exchange Process

- Job one: am I talking to whom I THINK I'm talking? (e.g., am I protected **against MITM?**) Remember that public-private key pair you created when you requested your certificate? This is where that public-private key pair comes into play.

- Job two: creating a **safe channel over which symmetric keys can be agreed** for use in encrypting the bulk of the session's traffic. The RSA public-private key pair CAN be used for this, too, but if you DO use it for this function, the worrying scenario then becomes:
  -- Adversary hoovers up all your encrypted traffic, and saves it
  -- Adversary somehow obtains access to your private key (e.g., via Heartbleed or maybe an untrustworthy privileged user)
  -- Adversary then decrypts ALL your captured traffic (much sad)

Ephemeral Key Exchange to the Rescue!



- To avoid the hoover-and-eventually-decrypt scenario, you should use an ephemeral key exchange protocol:

  Diffie Hellman Ephemeral, or Elliptic Curve Diffie Hellman Ephemeral

  Those key exchange protocols CANNOT be retrospectively exploited. We'll show how to configure nginx for those in the next section of this talk.

- For now, let's talk about how to ensure we use strong Diffie Hellman parameters for the key exchange.

(b) Generating Stronger DH Parameters

- DHE and ECDHE default parameters are often small (1024 bit).

- Fortunately, you can readily generate an configure stronger parameters using OpenSSL:

  ```
  cd /etc/ssl/certs
  openssl dhparam -out dhparam.pem 4096
  ```

  NOTE: This will take a LONG time (*hours* in some cases), but just set it up to generate and then go work on something else.

- Once that file's been created, add the following line to nginx.conf:

  ```
  ssl_dhparam /etc/ssl/certs/dhparam.pem;
  ```

(c) Key Exchange Badness That Must Also Be Avoided: ANONYMOUS Diffie Hellman

- Don't use anonymous cipher suites. **ANONYMOUS** Diffie Hellman is NOT the same as **EPHEMERAL** Diffie Hellman!
- Anonymous Diffie Hellman doesn't make any attempt to verify the identity of the site to which you connect, allowing for easy Man In the Middle Attacks. ADH cipher suites include:

```
$ openssl ciphers ADH
ADH-AES256-GCM-SHA384:ADH-AES256-SHA256:
ADH-AES256-SHA:ADH-CAMELLIA256-SHA:
ADH-DES-CBC3-SHA:ADH-AES128-GCM-SHA256:
ADH-AES128-SHA256:ADH-AES128-SHA:
ADH-SEED-SHA:ADHCAMELLIA128-SHA:
ADH-RC4-MD5:ADH-DES-CBCSHA:
EXP-ADH-DES-CBC-SHA:EXP-ADH-RC4-MD5
```

Part 4. Cipher Suite Selection

(a) Choice of Ciphers

- We're closing on the last bits you need to know in order to secure your sever and get a good evaluation from the Qualys tester.

- The last thing we need to talk about is our choice of symmetric ciphers, and how strong they need to be.

- If you ONLY need to support modern browsers, and you've configured your server as previously described, the recommendation is simple...

Simple Recommendation: Use AES-256

- AES-256 is generally agreed to be a very strong symmetric cipher.
- If the clients accessing your site support it (and all modern browsers will), and your server has the horsepower to handle the load (most normally will), it is your best option.
- If your server CANNOT handle the load, you can safely drop back to AES-128 (in fact, some authorities will out-and-out recommend AES-128 instead of AES-256, but I'd prefer to see you have additional cryptographic margin "just in case")

Suggested nginx.conf Cipher Suite Configuration

```
ssl_ciphers  AES256+EECDH:AES256+EDH:!aNULL:!eNULL:
!LOW:!3DES:!MD5:!EXP:!PSK:!DSS:!RC4:!SEED;

ssl_prefer_server_ciphers   on;

ssl_ecdh_curve secp384r1;
```

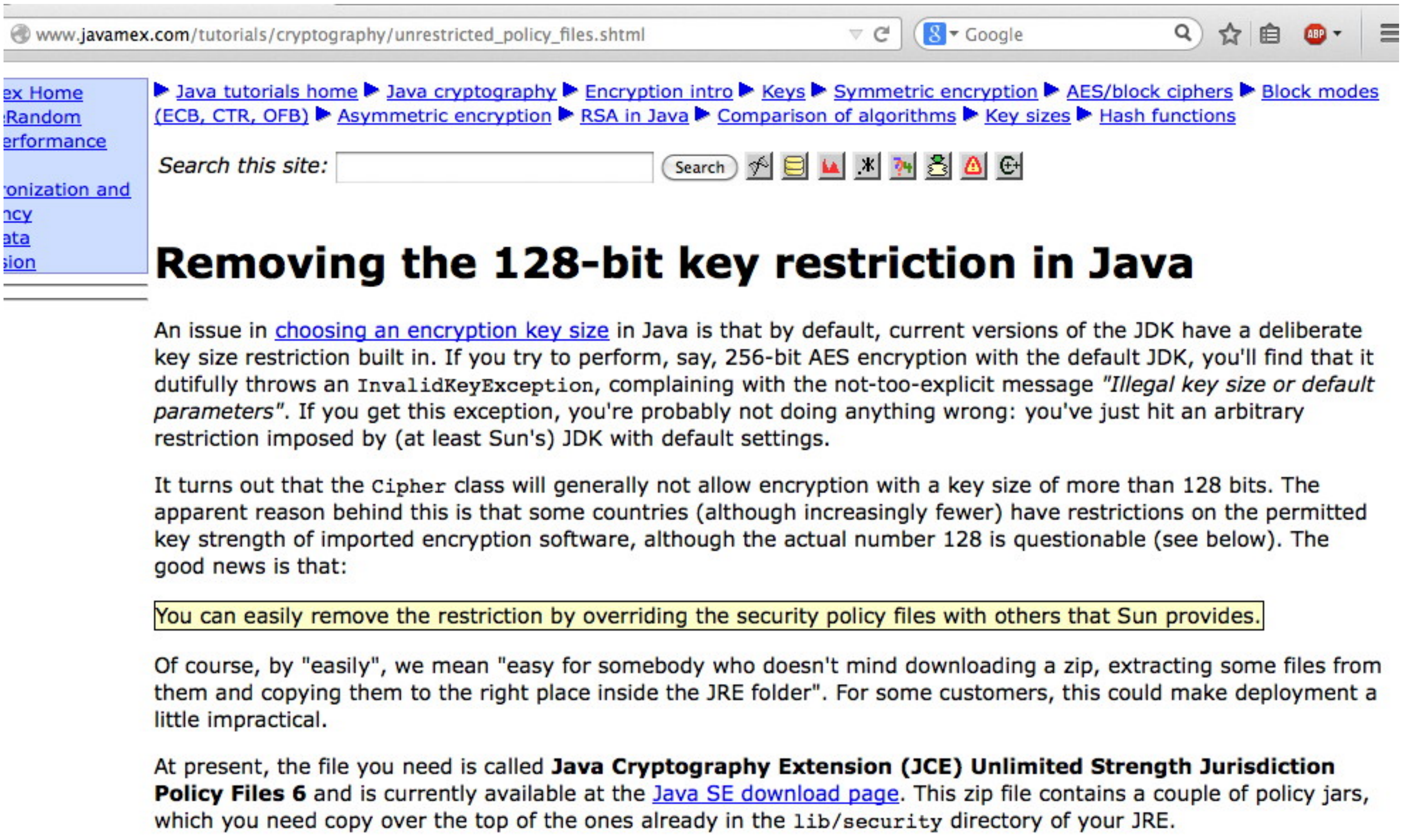
How That "Looks" For Various Selected Browsers

If you configure that string and then test your site with the Qualys SSL Server tester, it shows the resulting cipher suite that gets negotiated. Popular browsers look like:

Chrome 37 / OS X R	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)	FS	256
Firefox 32 / OS X R	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)	FS	256
IE 11 / Win 8.1 R	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xc028)	FS	256
Safari 7 / OS X 10.9 R	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xc028)	FS	256

NOTE: If you care about non-current browsers or out-of-date operating systems, the cipher suite I suggested will likely be **TOO RESTRICTIVE**, and it will be impossible for the user's browser and your server to negotiate a mutually agreeable cipher.

If You're Using Java, You May Have To Say "Captain May I?" To Get Access to AES-256



www.javamex.com/tutorials/cryptography/unrestricted_policy_files.shtml

ex Home
Random
erformance

onization and
ncy
ata
sion

▶ Java tutorials home ▶ Java cryptography ▶ Encryption intro ▶ Keys ▶ Symmetric encryption ▶ AES/block ciphers ▶ Block modes (ECB, CTR, OFB) ▶ Asymmetric encryption ▶ RSA in Java ▶ Comparison of algorithms ▶ Key sizes ▶ Hash functions

Search this site: Search

Removing the 128-bit key restriction in Java

An issue in choosing an encryption key size in Java is that by default, current versions of the JDK have a deliberate key size restriction built in. If you try to perform, say, 256-bit AES encryption with the default JDK, you'll find that it dutifully throws an `InvalidKeyException`, complaining with the not-too-explicit message *"Illegal key size or default parameters"*. If you get this exception, you're probably not doing anything wrong: you've just hit an arbitrary restriction imposed by (at least Sun's) JDK with default settings.

It turns out that the `Cipher` class will generally not allow encryption with a key size of more than 128 bits. The apparent reason behind this is that some countries (although increasingly fewer) have restrictions on the permitted key strength of imported encryption software, although the actual number 128 is questionable (see below). The good news is that:

You can easily remove the restriction by overriding the security policy files with others that Sun provides.

Of course, by "easily", we mean "easy for somebody who doesn't mind downloading a zip, extracting some files from them and copying them to the right place inside the JRE folder". For some customers, this could make deployment a little impractical.

At present, the file you need is called **Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 6** and is currently available at the Java SE download page. This zip file contains a couple of policy jars, which you need copy over the top of the ones already in the `lib/security` directory of your JRE.

(b) Alternative Cipher Suite Recommendations

- **If my preferred cipher suite specification proves to be just too intolerably stringent** (it really SHOULDN'T BE), you may want to use one of the alternative cipher suite specifications from:

  -- https://bettercrypto.org/static/applied-crypto-hardening.pdf
  -- https://www.ssllabs.com/projects/best-practices/
  -- https://wiki.mozilla.org/Security/Server_Side_TLS

- **In a nutshell, the less-rigorous option(s) will typically involve enabling AES-128 (if some countries are okay with AES-128 but not AES-256, that should tell you something).**

  It will ALSO typically involve permitting TLS 1.0 (which is DEFINITELY NOT GOOD).

(c) Also, Do NOT Use RC4 Ciphers

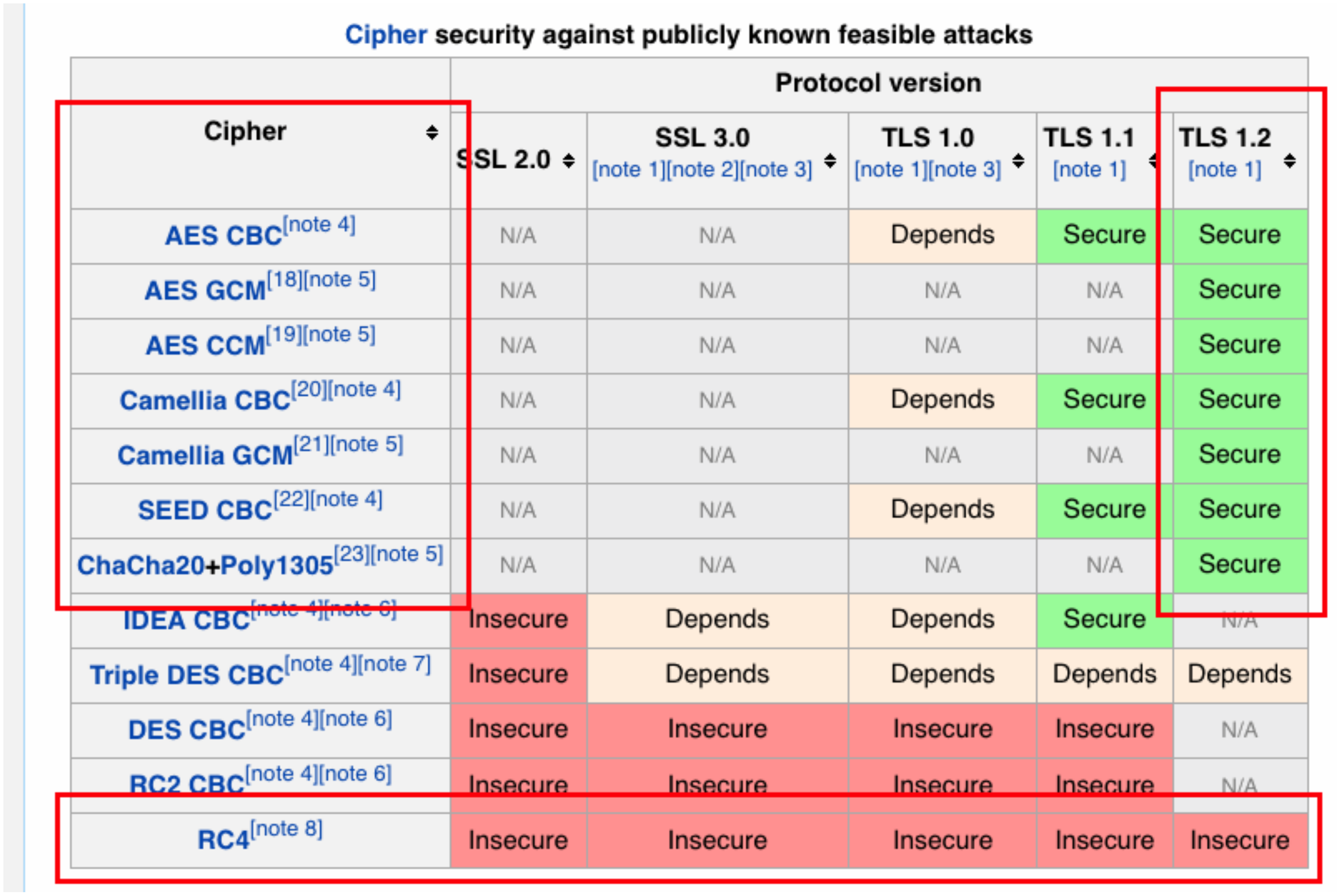
- At the time of an earlier SSL/TLS attack (the BEAST vulnerability), RC4 (the only stream cipher that most SSL/TLS servers supported) became popular as a quick fix for that bug.

- We now know that RC4 is NOT a safe option (see for example http://www.isg.rhul.ac.uk/tls/).

- We recommend that you **DISABLE RC4**, as does Microsoft ("Security Advisory 2868725: Recommendation to disable RC4," http://blogs.technet.com/b/srd/archive/2013/11/12/security-advisory-2868725-recommendation-to-disable-rc4.aspx)

- See also the IETF Draft "Prohibiting RC4 Cipher Suites," https://tools.ietf.org/html/draft-ietf-tls-prohibiting-rc4-01

Nice Summary of RC4 (vs. TLS 1.2) Security



Cipher security against publicly known feasible attacks

Cipher	Protocol version				
	SSL 2.0	SSL 3.0 [note 1][note 2][note 3]	TLS 1.0 [note 1][note 3]	TLS 1.1 [note 1]	TLS 1.2 [note 1]
AES CBC[note 4]	N/A	N/A	Depends	Secure	Secure
AES GCM[18][note 5]	N/A	N/A	N/A	N/A	Secure
AES CCM[19][note 5]	N/A	N/A	N/A	N/A	Secure
Camellia CBC[20][note 4]	N/A	N/A	Depends	Secure	Secure
Camellia GCM[21][note 5]	N/A	N/A	N/A	N/A	Secure
SEED CBC[22][note 4]	N/A	N/A	Depends	Secure	Secure
ChaCha20+Poly1305[23][note 5]	N/A	N/A	N/A	N/A	Secure
IDEA CBC[note 4][note 6]	Insecure	Depends	Depends	Secure	N/A
Triple DES CBC[note 4][note 7]	Insecure	Depends	Depends	Depends	Depends
DES CBC[note 4][note 6]	Insecure	Insecure	Insecure	Insecure	N/A
RC2 CBC[note 4][note 6]	Insecure	Insecure	Insecure	Insecure	N/A
RC4[note 8]	Insecure	Insecure	Insecure	Insecure	Insecure

Source: http://en.wikipedia.org/wiki/Transport_Layer_Security

(d) DO NOT USE the NULL "Cipher" Suites

- You should also not use the NULL cipher suites because that doesn't involve doing ANY encryption! Doh!

  ```
  $ openssl ciphers NULL
  ECDHE-RSA-NULL-SHA:ECDHE-ECDSA-NULL-SHA:
  AECDH-NULL-SHA:ECDH-RSA-NULL-SHA:
  ECDHE-CDSA-NULL-SHA:NULL-SHA256:
  NULL-SHA:NULL-MD5
  ```

  Let me emphasive:

  Do NOT make a mistake and accidental end up using the NULL cipher suites

Other Cipher Suites You Should Also NOT Use

- **DO NOT USE "Export" or "Low" grade (weak) ciphers...**
  $ openssl ciphers EXPORT,LOW

- **DO NOT USE "DES" ciphers**
  $ openssl ciphers DES

- **DO NOT USE "MD5" (this includes ALL SSLv2 ciphers)...**
  $ openssl ciphers MD5

- The above are ALL weak!

- Speaking of wanting strong ciphers, is there anything stronger than RSA-4096?

Part 5. Elliptic Curve Crypto (ECC)

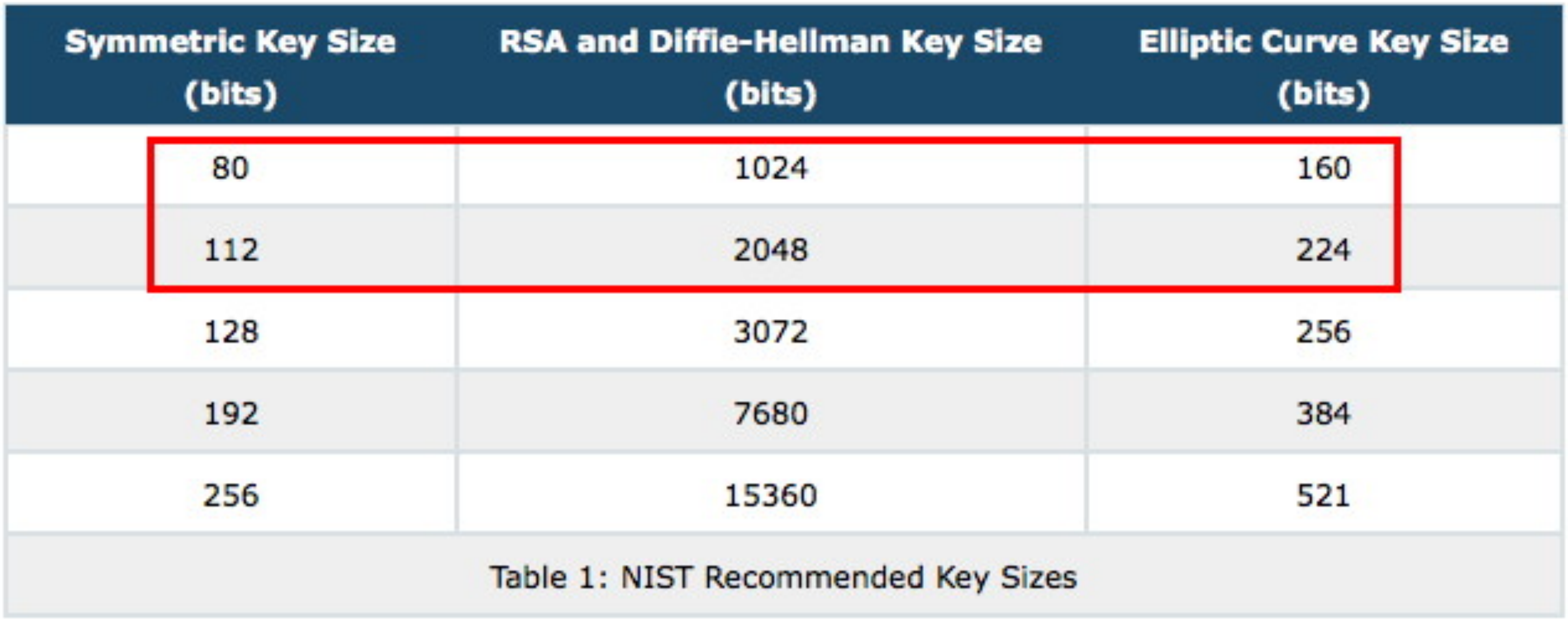
RSA Public Key Crypto vs. ECC

- Traditionally, **RSA public key crypto has been based around our (limited) ability to quickly factor large integers.** For those who may have forgotten factoring from high school or grade school, factoring is the ability to find numbers that divide into an integer evenly. For example, 3 and 5 are factors of 15. While that's easy, factoring a 2048 or 4096 bit-long integer is just a *bit* more difficult.
- About ten years ago, the community began to work on moving to **elliptic curve** cryptography, which relies on the difficulty of solving the discrete logarithm problem. You likely ***didn't study*** the discrete logarithm problem in grade school or high school. ☺
- The best (relatively) easy-to-understand introduction to elliptic curves that I've seen is probably this one: http://arstechnica.com/security/2013/10/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography/

Do We "Need" Elliptic Curve Crypto *Today*?

- Right now, we might not. RSA appears to be working okay.
- That said, ECC is also generally able to deliver stronger crypto for a given size key, while being less taxing on CPUs than RSA, so if you want keys stronger than RSA-4096, ECC's basically your only realistic/practical option today.
- You might also want to be ready with alternative crypto system *IF* there's a breakthrough in factoring long integers (e.g., perhaps through use of quantum computing and Shor's Algorithm see http://en.wikipedia.org/wiki/Shor%27s_algorithm). The hypothetical fall of RSA, etc., is what Tom Ritter and his co-authors refer to as the "cryptopocalypse", see: https://isecpartners.com/media/105564/ritter_samuel_stamos_bh_2013_cryptopocalypse.pdf)
- If you're interested in post-quantum crypto, see also http://en.wikipedia.org/wiki/Post-quantum_cryptography

RSA vs. Elliptic Curve Strength Equivalence



Symmetric Key Size (bits)	RSA and Diffie-Hellman Key Size (bits)	Elliptic Curve Key Size (bits)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

Table 1: NIST Recommended Key Sizes

http://www.nsa.gov/business/programs/elliptic_curve.shtml

National Security Crypto and ECC

- Because the government routinely relies on "commercial off the shelf" (COTS) technology for its own applications, the government needs to provide guidance about how to safely use publicly-available crypto to secure sensitive information, for classified information up to and including for TOP SECRET information.

- Some of those recommendations involve use of public standards (rather than classified government cryptographic methods).

- See the excerpt from CNSSP No. 15 Annex B on the next slide, describing use of ECC for up to TOP SECRET information.

ANNEX B

Suite B – NIST cryptographic algorithms approved by NSA to protect National Security Systems and the information that resides therein.

Algorithm	Function	Specification	Parameters
Advanced Encryption Standard (AES)	Symmetric block cipher used for information protection	FIPS PUB 197 (reference g.)	Use 128 bit keys to protect up to SECRET. Use 256 bit keys to protect up to TOP SECRET*
Elliptic Curve Diffie-Hellman (ECDH) Key Exchange	Asymmetric algorithm used for key establishment	NIST SP 800-56A (reference h.)	Use Curve P-256[1] to protect up to SECRET. Use Curve P-384 to protect up to TOP SECRET.*
Elliptic Curve Digital Signature Algorithm (ECDSA)	Asymmetric algorithm used for digital signatures	FIPS PUB 186-3 (reference f.)	Use Curve P-256 to protect up to SECRET. Use Curve P-384 to protect up to TOP SECRET.*
Secure Hash Algorithm (SHA)	Algorithm used for computing a	FIPS PUB 180-4 (reference e.)	Use SHA-256 to protect up to

ECC Trusted Roots?

- In order to be able to do Suite B-recommended ECC crypto, you need a cert that chains to an ECC root.

- Currently there are just seven (7) ECC trusted roots in the Firefox trust anchor (see http://www.mozilla.org/en-US/about/governance/policies/security-group/certs/included/):

  Comodo ECC Certification Authority
  DigiCert Assured ID Root G3
  DigiCert Global Root G3
  GeoTrust Primary Certification Authority – G2
  Thawte Primary Root CA - G2
  Trend Micro Affirm Trust Premium ECC
  VeriSign Class 3 Public PCA - G4

Choice of Curves

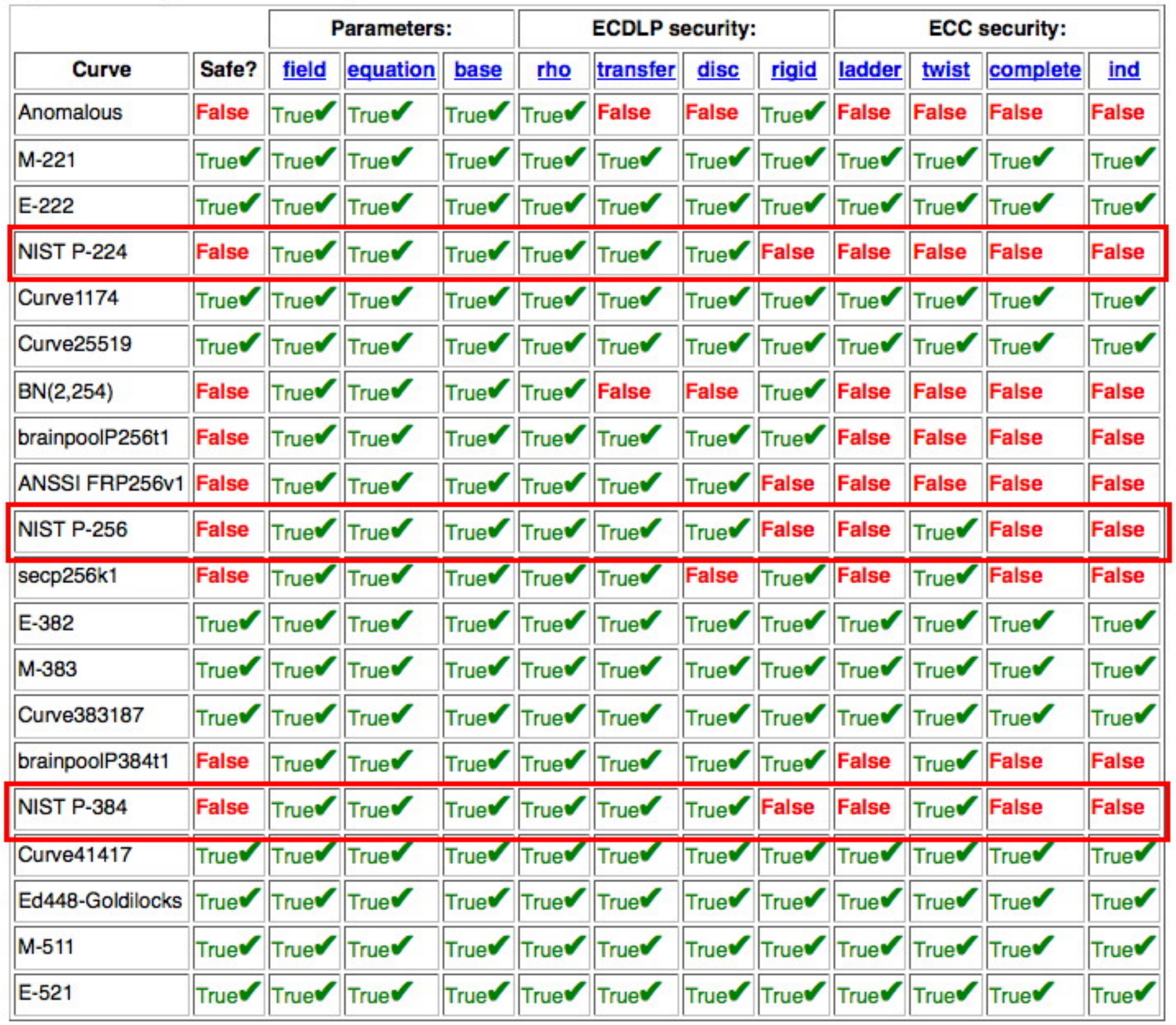
- When it comes to working with elliptic curve cryptography, note that ECC is actually a ***family*** of methods that rely on **different** elliptic curves.

- You (or, more accurately, the coders of the crypto libraries you use) need to decide on the specific elliptic curves you're going to use.

- **NIST has recommended a set of curves. SO HAVE OTHERS. If you're considering doing ECC (and you should be), I'd urge you to review http://safecurves.cr.yp.to/ <u>Not all ECC curves are equally good</u>.**

- **IMPORTANT NOTE: if you DON'T allow the NIST curves, you may not have a solution that will work for Windows users.**

http://safecurves.cr.yp.to/



		Parameters:			ECDLP security:				ECC security:			
Curve	Safe?	field	equation	base	rho	transfer	disc	rigid	ladder	twist	complete	ind
Anomalous	False	True✓	True✓	True✓	True✓	False	False	True✓	False	False	False	False
M-221	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓
E-222	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓
NIST P-224	False	True✓	True✓	True✓	True✓	True✓	True✓	False	False	False	False	False
Curve1174	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓
Curve25519	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓
BN(2,254)	False	True✓	True✓	True✓	True✓	False	False	True✓	False	False	False	False
brainpoolP256t1	False	True✓	True✓	True✓	True✓	True✓	True✓	True✓	False	False	False	False
ANSSI FRP256v1	False	True✓	True✓	True✓	True✓	True✓	True✓	False	False	False	False	False
NIST P-256	False	True✓	True✓	True✓	True✓	True✓	True✓	False	False	True✓	False	False
secp256k1	False	True✓	True✓	True✓	True✓	True✓	False	True✓	False	True✓	False	False
E-382	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓
M-383	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓
Curve383187	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓
brainpoolP384t1	False	True✓	True✓	True✓	True✓	True✓	True✓	True✓	False	True✓	False	False
NIST P-384	False	True✓	True✓	True✓	True✓	True✓	True✓	False	False	True✓	False	False
Curve41417	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓
Ed448-Goldilocks	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓
M-511	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓
E-521	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓

"What Does All The Stuff In That Table Mean?"

- Read this paper:

  "Security Dangers of the NIST Curves"
  http://cr.yp.to/talks/2013.05.31/slides-dan+tanja-20130531-4x3.pdf

Some Pretty Smart Folks Use Alternative Curves

- For example the team at Silent Circle have announced that they are going to non-NIST curves from Daniel Bernstein & Tanja Lange: https://blog.silentcircle.com/nncs/ https://blog.silentcircle.com/this-one-goes-to-414/
- Google Chrome is also moving to crypto from DJ Bernstein, see http://googleonlinesecurity.blogspot.com/2014/04/speeding-up-and-strengthening-https.html plus http://tools.ietf.org/html/draft-nir-cfrg-chacha20-poly1305-02 and http://tools.ietf.org/html/draft-mavrogiannopoulos-chacha-tls-02
- Will more cryptographic products and libraries move to support non-NIST elliptic curves from Bernstein and Lange? I think so.

Part 6. Certificates OTHER THAN Traditional SSL/TLS Certs For The Web

InCommon Client Certs

- InCommon Client Certs now are available in SHA-2 format, too.
- If you're an InCommon Certificate Service subscriber and you'd like to begin using SHA-2 client certs at your site, please send in the client cert request form that's at: https://spaces.internet2.edu/display/InCCollaborate/Client+Certificate+Request+Form
- Another item "for the record:" InCommon is no longer offering PKI hard tokens or PKI smart cards for use with client certificates (but you can still obtain PKI hard tokens or PKI smart cards from the usual major online computer and networking gear sources if you need/want them)

Code Signing Certs: Now Using SHA-2 As Well

- Code signing certs are used by programmers to cryptographically sign apps. [Code signing certs are bundled at no extra charge with the InCommon Certificate Service]

- Historically, code signing certs used SHA-1 for signatures. Now, just like other types of certs, code signing certs are migrating to SHA-2.

- There are some excellent campus pages available around code signing; see for example:

  "Digital Certificate - Use a Code Signing Certificate"
  https://answers.uchicago.edu/page.php?id=19495

IGTF Server Certs

- These are another type of specialized certificates that meet the unique requirements of the Interoperable Global Trust Federation "Grid" community. Note: if you're NOT trying to set up a server to use with the IGTF, this is NOT the sort of SSL/TLS certificate you want.

- For more information, see:

  -- http://www.igtf.net/

  -- http://www.incommon.org/certificates/igtf/index.html

  -- https://www.xsede.org/security/certificates

Intel AMT Device Management Certificates

- This is a new type of certificate available at no additional charge from Comodo/the InCommon Certificate Service. Quoting from: en.wikipedia.org/wiki/Intel_Active_Management_Technology

  "Intel AMT is hardware and firmware technology that builds certain functionality into business PCs in order to monitor, maintain, update, upgrade, and repair PCs. Intel AMT is part of the Intel Management Engine, which is built into PCs with Intel vPro technology. Intel AMT is designed into a secondary (service) processor located on the motherboard."

- See also https://software.intel.com/sites/manageability/AMT_Implementation_and_Reference_Guide/default.htm?turl=WordDocuments%2Fintelamtandsecurityconsiderations1.htm

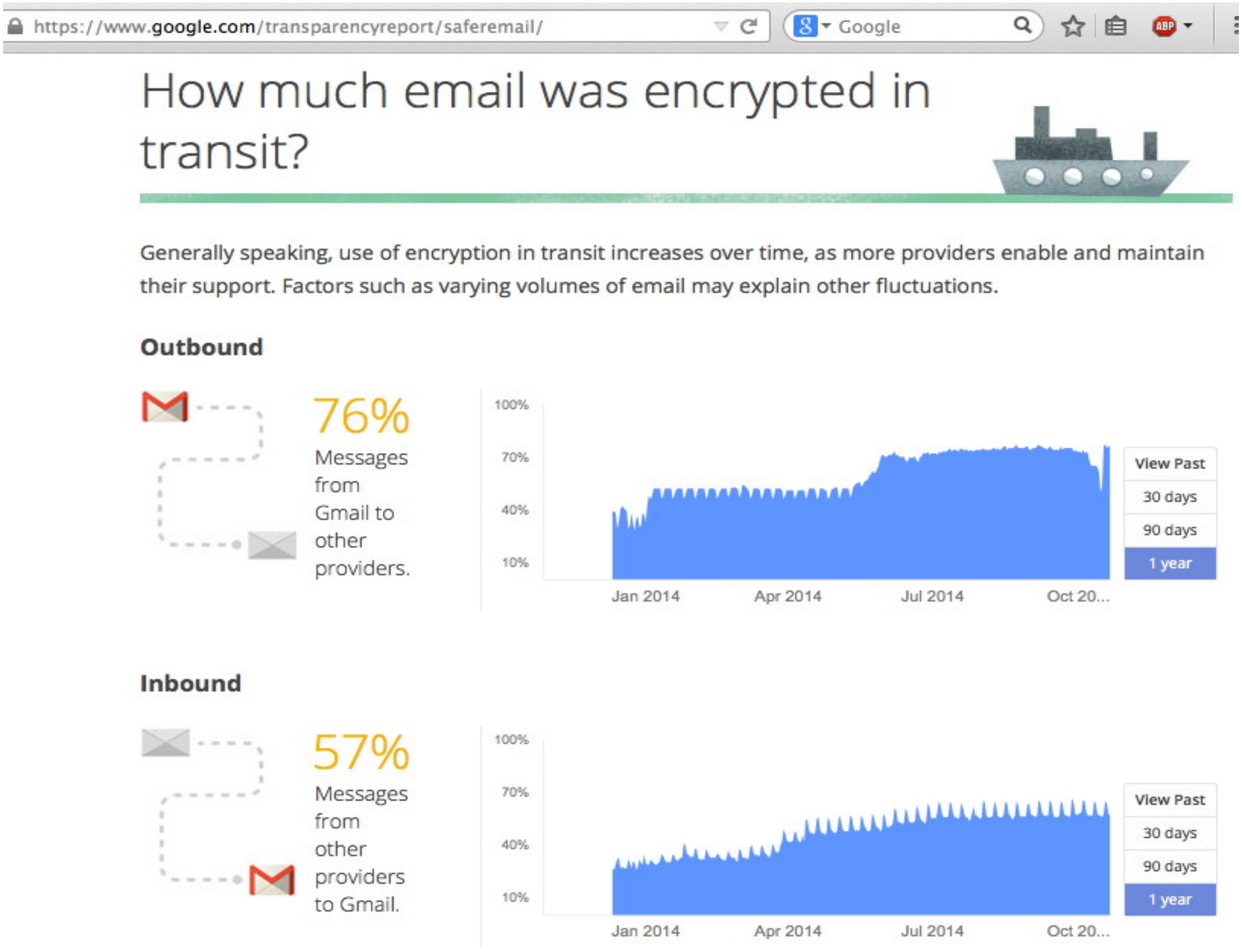
  "Remote platform management applications can access Intel AMT securely, even when the platform is turned off, as long as the platform is connected to line power and to a network."

Opportunistic Encryption of SMTP Traffic

- When thinking about uses for SSL/TLS certificates, don't overlook how they can help to secure non-web services. For example, SSL/TLS can be used to secure POP and IMAP access to user email via clients such as Thunderbird or Outlook.

- SSL/TLS can also be used to secure mail flows between mail transfer agents (MTAs), such as Postfix, Exim, etc.

- Recently major mail providers have made a major effort to get a majority of their SMTP traffic protected with opportunistic encryption. See the Gmail email transparency report on the next page...



How much email was encrypted in transit?

Generally speaking, use of encryption in transit increases over time, as more providers enable and maintain their support. Factors such as varying volumes of email may explain other fluctuations.

Outbound

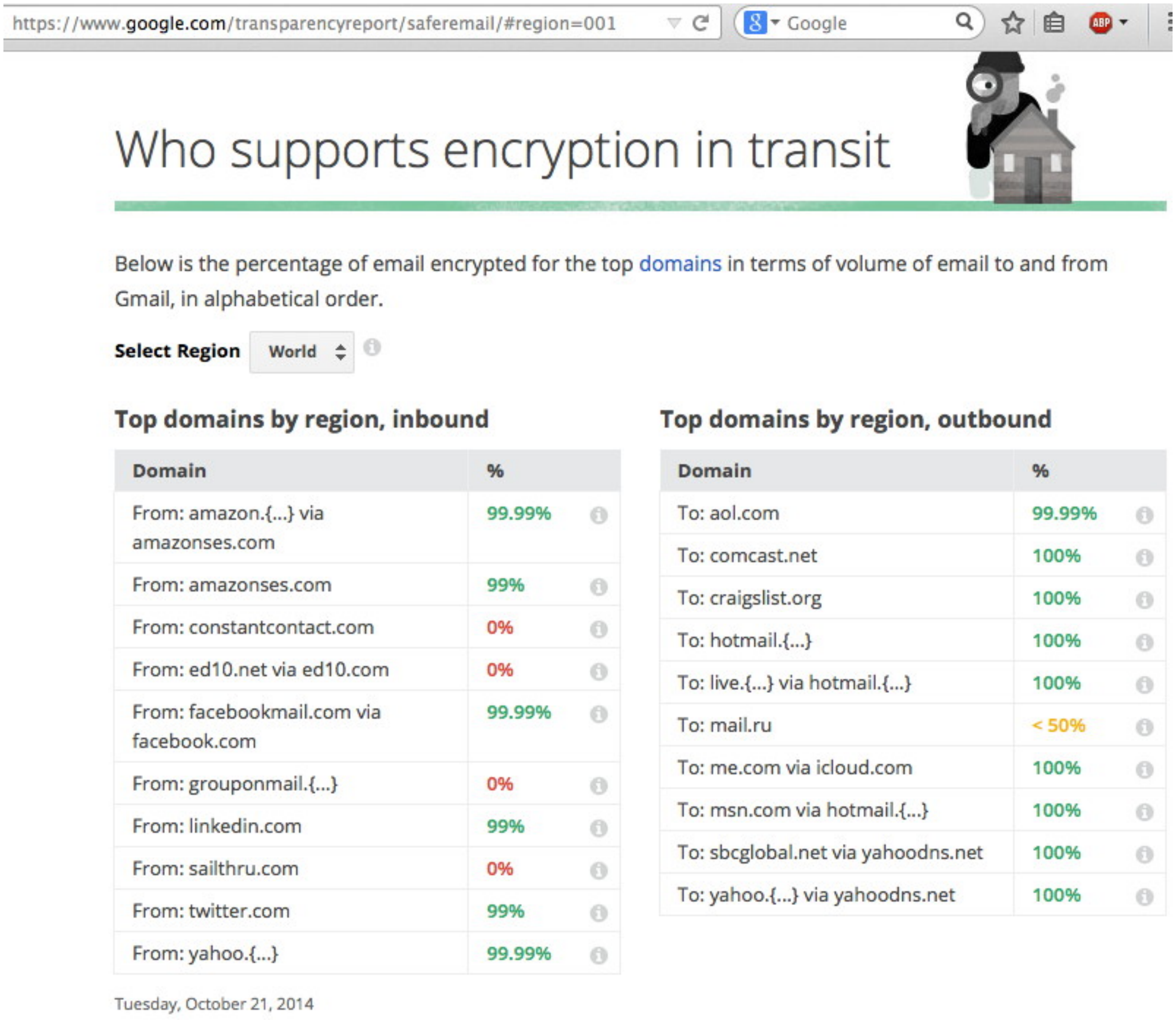
76%

Messages from Gmail to other providers.

Inbound

57%

Messages from other providers to Gmail.


https://www.google.com/transparencyreport/saferemail/#region=001

Google

Who supports encryption in transit

Below is the percentage of email encrypted for the top domains in terms of volume of email to and from Gmail, in alphabetical order.

Select Region World

Top domains by region, inbound

Domain	%
From: amazon.{...} via amazonses.com	99.99%
From: amazonses.com	99%
From: constantcontact.com	0%
From: ed10.net via ed10.com	0%
From: facebookmail.com via facebook.com	99.99%
From: grouponmail.{...}	0%
From: linkedin.com	99%
From: sailthru.com	0%
From: twitter.com	99%
From: yahoo.{...}	99.99%

Top domains by region, outbound

Domain	%
To: aol.com	99.99%
To: comcast.net	100%
To: craigslist.org	100%
To: hotmail.{...}	100%
To: live.{...} via hotmail.{...}	100%
To: mail.ru	< 50%
To: me.com via icloud.com	100%
To: msn.com via hotmail.{...}	100%
To: sbcglobal.net via yahoodns.net	100%
To: yahoo.{...} via yahoodns.net	100%

Tuesday, October 21, 2014

Certs on Wireless Access Point

- Another place where certs sometimes turn up is on wireless access points. This is an important use (you don't want spoofed wireless auth, for example), but can pose some challenges, chicken-and-egg style.
- Specifically, before we can trust a cert presented by an server, we need to make sure it hasn't been revoked. This is normally done by using the Online Certificate Status Protocol (OCSP), or by downloading and checking a CRL (Certificate Revocation List).
- However, recall that on a wireless access point, until you accept the cert you're offered, you won't be online, so you can't check OCSP or CRLs, so... Beware this circularity.
- Ideally, configure your wireless access point to allow OCSP or CRL traffic even for UNAUTHENTICATED wireless users so proper certificate revocation checking can happen.

Finishing Up My Work With the Cert Service

- Finally, I also wanted to make sure that folks were aware that my last day managing the InCommon Certificate Service (under contract through UO) will be the end of this week, given Internet2's decision not to renew that contract.

- On November 1st, 2014, I'm accepting a new position with Paul Vixie's data-driven security company, Farsight Security (see http://www.farsightsecurity.com/)

- It has been a real pleasure and privilege working with you all, and I wish everyone all the best in the future when it comes to their cryptographic work.

Thanks for The Chance to Talk

- Are there any questions?

- If you come up with any questions later:

  joe@stsauver.com

- These slides are available from

  https://www.stsauver.com/joe/new-crypto-101/