

Cryptographic Best Practices in the Post-Snowden Era

Joe St Sauver, Ph.D. (joe@uoregon.edu)

Security Professionals 2014

St Louis, Missouri

May 8th, 2014

<http://pages.uoregon.edu/joe/crypto-bcp/>

Disclaimer: all opinions strictly my own.

Good Morning!

- I'd like to thank the program committee for the opportunity to speak with you today.
- As one of the closing sessions for this year's meeting, I'd also like to thank all of you for sticking around for this session.
- Yes, this is another one of those odd "Joe talks" but remember, you don't need to try to read everything on these slides. Think of my slides as a way to keep me on track, or a resource for someone who couldn't be here in person or online, or "closed captioning" for any in the audience who may be deaf or hard of hearing.
- Finally, let me emphasize that what I'll discuss and say today represents my own opinion, not the opinion of any other entity.

A Few Process Notes

- Please do NOT make any changes based solely on what we discuss here! **Do your OWN due diligence**, carefully and thoroughly evaluating any crypto changes you may consider making.
- Let me also emphasize that **this deck is meant to surface issues, not dictate solutions**. Different answers may be appropriate for different sites, given unique aspects of those sites' environments. I may offer some suggestions, but they're just that.
- Finally, note that this talk was prepared in April/May of 2014. Cryptographic practices are moving fast, have already evolved substantially in the year since Snowden's revelations began, and will likely continue to do so. **As this presentation ages, be sure to consider any subsequently-available relevant information.**

This Is Largely A Web Crypto-Focused Session.

Do I Really Need It?

- Let's do a quick check of one very public aspect of your site's cryptographic practices. Check your main campus web server at:

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

(note that you can ask for a "checkup" w/o having your results broadcast to the world by ticking the box under the URL entry box)

- The great part about that site, if you're from higher education, is that you get a "grade," just like introduction to physics. What grade do you get? An "A" (or at least an "A-")? No major issues? If so, yes, you can "cut class" – at least if you're confident that there's nothing else cryptographic worth worrying about.

"This Session Will Be Over My Head!"

- We're going to do our best to ensure that this session is NOT over your head, or anyone else's head (including mine), either.
- We're not going to do anything mathematically complex today, and we're going to work to keep the technical level as approachable for everyone as we can.
- The emphasis will be on background to bring you up to speed, and on some PRACTICAL/DEPLOYABLE considerations for you to take away with you.
- We will also try to provide **lots of links** to other resources, so you can get any backfill you may require.

Related Prior Presentations

- For background, I've previously done a variety of other presentations related to crypto, including (but not limited to):
 - "Giving Your Use of Server Certificates a Hard Look,"
<http://pages.uoregon.edu/joe/hardlook/hard-look.pdf>
 - "Leveraging Certificates for Improved Security,"
<http://pages.uoregon.edu/joe/leveraging-certificates/>
 - "Client Certificates: A Security Professionals 2012 Preconference Seminar," <http://pages.uoregon.edu/joe/secprof2012/>
- **This talk** has more of an emphasis on **crypto algorithms and protocols**, and is informed by Snowden-related disclosures over the past year. Let's dive in.

I. Introduction

"One of the most complex areas of the security industry is cryptography. [...] [T]he survey of the IT industry experts identified that many of the cryptographic solutions that they audit and test are poorly deployed and insecure. [...]"

Study on the use of cryptographic techniques in Europe,
European Network and Information Security Agency,
https://www.enisa.europa.eu/activities/identity-and-trust/library/the-use-of-cryptographic-techniques-in-europe/at_download/fullReport

Crypto Is Complex

- If you want to **design cryptographic algorithms**, doctoral-level training in mathematics would help, as would a number of years spent apprenticing with experts from the crypto community. The rest of us need to professionally rely on others.
- **Cleanly implementing cryptographic algorithms**, particularly in the cross-platform case, can be challenging unless you're an expert coder, and even the best of coders can make mistakes. **We need the community to carefully scrutinize and test open source crypto**, helping to find bugs and improve implementations.
- Even just **configuring & deploying cryptography** is fraught with a million potential wrong turns and bad neighborhoods. There are many seemingly worthy options that can turn out to be subtly problematic, and few good guides are available for practitioners.
- And yet, we all continue to rely on crypto for much of our security.

Crypto Use in Cyber Security: It's Everywhere

- Cyber security technologies depend heavily on cryptography. Examples include:
 - Hashed passwords (as well as federated auth, e.g., SAML)
 - Multifactor auth solutions (TOTP, HOTP, PKI, etc.)
 - Digital signatures on documents
 - Use of hashes to uniquely identify malware
 - Code signing (e.g., for mobile apps)
 - Whole disk encryption solutions
 - Wireless access point security with WPA2 and 802.1X
 - SSL/TLS with digital certificates for web servers, IMAPS, etc.
 - ssh/sftp as an encrypted replacement for telnet and ftp
 - Email signed/encrypted with PGP/GnuPrivacyGuard or S/MIME
 - IPsec (e.g., as used by some VPN solutions)
 - DNSSEC
 - RPKI

Before Snowden...

- If you don't remember the original "crypto wars" of 80's/ 90's, until Edward Snowden's recent revelations about pervasive monitoring of the Internet, crypto was widely neglected. Few people thought much about it, if they thought about it at all.
- When crypto actually did get discussed pre-Snowden, **the primary focus was often on protecting user passwords (and credit card numbers) from being intercepted by criminal hacker/crackers.**
- A tiny number of people went further, doing things like:
 - deploying opportunistic encryption for services such as SMTP
 - encrypting their email end-to-end using PGP or S/MIME
 - using (or running) a cipherspunks anonymizing remailer, or
 - using Tor (or running a node in the Tor network) for protection against traffic analytic attacks.
- But, candidly, many folks probably thought that doing so was sort of whacky – at least until Snowden's revelations of June 2013.

The Conceptual Idyllic (Pre-Snowden) World



Source: <http://www.geograph.org.uk/photo/1329439>

II. Snowdonia

"Nobody does the right thing."

Marie Kreutz, *The Bourne Identity*, 2002

Obumbrata et velata, michi quoque niteris
[In the dark and secretly, you work against me]

"O Fortuna," *Carmina Burana*, 13th century poem

June 2013: The Crypto World Turns Upside Down

- On June 6th, 2013, Glenn Greenwald published an article in *The Guardian* revealing that **the NSA had been vacuuming up phone records for millions of American customers who use Verizon.** The Internet suddenly tilted. See <http://www.theguardian.com/world/2013/jun/06/nsa-phone-records-verizon-court-order>
- The next day, the online world turned completely upside down when the *Washington Post* subsequently reported on PRISM, see **"U.S., British intelligence mining data from nine U.S. Internet companies in broad secret program,"** www.washingtonpost.com/investigations/us-intelligence-mining-data-from-nine-us-internet-companies-in-broad-secret-program/2013/06/06/3a0c0da8-cebf-11e2-8845-d970ccb04497_story.html

Drilling Holes In the Bottom of Our Own Cryptographic "Boats"

- Another profoundly disturbing revelation was that the **NSA** may **have intentionally weakened or compromised the strength and technical integrity of some cryptographic protocols as part of the standards development process.** We all need to be able to rely on these protocols to secure confidential information online, but we can't if they've been intentionally weakened or compromised.

A specific example of this: **RSA** has now publicly told its customers to stop using the NSA-influenced **Dual_EC_DRBG random number generator.** Dual_EC_DRBG had been used for key parts of some RSA products (see <http://arstechnica.com/security/2013/09/stop-using-nsa-influence-code-in-our-product-rsa-tells-customers/>)

[An Aside: Why Random Numbers Are a Big Deal]

- If I were to suggest that a **slot machine's** output wasn't truly random, you'd immediately get why that's a big deal: that slot machine wouldn't be "fair," and someone's going to get rich.
- Well, just like gaming devices, modern cryptography depends on an ability to generate good quality pseudorandom numbers for things like generated session passwords. An excellent discussion of why random numbers are very important can be seen at <http://blog.cloudflare.com/why-randomness-matters>
- Unfortunately, most systems don't have good built-in hardware sources of entropy, and inexpensive third party hardware random number generators can't be produced fast enough to meet market demand. For example, <http://www.entropykey.co.uk/shop/> states *"Please note that there is a very long waiting period for Entropy Keys at the moment. We currently have no stock and do not have a date for when we will have more."*

Judicially Compelled Disclosure of Private Keys

- A fourth example of a disturbing revelation related to the Snowden incident was judicially-compelled disclosure of private keys. This happened to Ladar Levison, owner of the ISP "Lavabit." There was an excellent Q&A session on this at NANOG, Oct 9th, 2013: http://www.youtube.com/watch?v=uo9-0So2A_g
- Under threat of contempt, Ladar was compelled to provide a copy of his SSL/TLS private keys, keys that protected the data of over 400,000 customers. With a copy of those private keys, the security of all Lavabit users could have been completely undercut. After being forced to surrender his keys under seal, Ladar took the only action he felt was morally left to him: he shuttered his business.
- When Godaddy learned that their customer's private keys may have been compromised, they revoked his certificate: www.forbes.com/sites/kashmirhill/2013/10/09/godaddy-pulls-lavabits-security-creds-because-the-government-got-a-hold-of-its-encryption-keys/

More Recently: The NSA and the Heartbleed Bug

www.bloomberg.com/news/print/2014-04-11/nsa-said-to-have-used-heartbleed-bug-exposing-consun Google

Bloomberg

[Print](#) [Back to story](#)

NSA Said to Exploit Heartbleed Bug for Intelligence for Years

By Michael Riley - Apr 12, 2014

The U.S. National Security Agency knew for at least two years about a flaw in the way that many websites send sensitive information, now dubbed the Heartbleed bug, and regularly used it to gather critical intelligence, two people familiar with the matter said.

The agency's reported decision to keep the bug secret in pursuit of national security interests threatens to renew the rancorous debate over the role of the government's top computer experts. The NSA, after declining to comment on the report, subsequently denied that it was aware of Heartbleed until the vulnerability was made public by a private security report earlier this month.

"Reports that NSA or any other part of the government were aware of the so-called Heartbleed vulnerability before 2014 are wrong," according to an e-mailed statement from the Office of the Director of National Intelligence.

Did the NSA Know About Heartbleed, or Not?

- One of the following things is true:
 - The NSA did know about Heartbleed and said nothing, exploiting the bug for years. Because they wanted to be able to exploit the bug, they knowingly left hundreds of thousands of American sites at risk. If this is true, it would represent a profound failure with respect to the NSA's information assurance responsibilities.
 - Alternatively, maybe the NSA didn't know about (as they've claimed). If that's true, why didn't the NSA identify this vulnerability? Shouldn't they have proactively found, reported and helped to fix this flaw?
- Poor judgment? Negligence/incompetence? Neither is very encouraging.

\$52.6 Billion Spent on National Intelligence

- The Washington Post Reports that we currently spend \$52.6 BILLION dollars a year on the national intelligence budget (see www.washingtonpost.com/wp-srv/special/national/black-budget/)
- That's a very big number, and one that's hard to wrap your head around except by thinking about what else we could buy with it. For example, we could buy and just *give away* over a quarter million \$200,000 houses a year, *outright*, with that kind of money.
- **HOW is it possible that we spend THAT MUCH and STILL not have secure open source crypto libraries?** Well, that's not the NSA's job. They exist to collect foreign intelligence, and to protect the federal government, **the NSA is NOT responsible for protecting YOUR site's cybersecurity or YOUR privacy.**

"Surely You're Wrong, Joe!"

- I'm not. You can read what the NSA's mission actual includes at:

<http://www.nsa.gov/about/mission/index.shtml>

It may not be what you think it should be.

- No where there does it say, "Keep Joe's computers safe" or "Protect Wagon Wheel State University's networks from cyber attack." Go check for yourself. Those hypothetical roles aren't there. The NSA is focused on protecting the **federal government**, and **collecting intel**, plus some other stuff. We're just free riders.
- So if you don't like the service you're getting from the government, change the NSA's mission (or ask for your \$52.6 billion back) ☺

Bottom Line, We're On Our Own. We Need To Be Paying Attention to Crypto Issues Ourselves.

- We CANNOT afford to ignore the challenging cryptographic environment we now know exists out there...
- We MUST be realistic about the steps we need to take to protect our systems and networks...
- In saying that, I know that some people won't (completely) agree.

Objection I... "We Shouldn't Be Talking About Leaked Classified Information!"

- Some may focus on the fact that **the materials leaked by Snowden and his journalist collaborators are classified**, arguing that as a result **we shouldn't access them or discuss them**.
- From my perspective, once a program has been broadly disclosed in the national or international media, "**you can't un-ring that bell.**" Snowden's secrets ARE publicly known / "in the wild."
- You might *want* to pretend those disclosures didn't happen, but you ignore those disclosures at your peril. **Many others** -- including your cyber adversaries -- ***will* be looking at all those leaks**.
- That said, I am not advising anyone with a clearance out there **to take any actions that might jeopardize their cleared status** (e.g., if you are cleared, note the DOD security policy dated June 7th 2013, at <http://www.fas.org/sgp/othergov/dod/notice.pdf>)

Objection II... "Snowden Is a Traitor! We Shouldn't Legitimize Or Give Attention to What He Did!"

- We also really don't want to get sidetracked around a discussion of whether or not Snowden are traitors (or heroes).
- Different people have radically different perspectives on this particular point, and we've all probably heard persuasive arguments in both directions.
- Ultimately, history will judge Snowden's actions, as well as the righteousness of the various clandestine programs he exposed.
- Rather than trying to judge the morality of what Snowden did, we'd like to focus on the practical technical impact of those disclosures, instead.

We Now Know: The Patient Is Very Sick

- **Broad and deep attention to cryptographic implementations mean that latent vulnerabilities are being looked for – and are being *found*.**
- Each vulnerability that gets uncovered and corrected makes those products more secure, but the process can be brutal, much like finding and eliminating malignant tumors in a sick patient.

Can We Even Trust The Algorithms We're Using?

- Attempts to subvert at least some cryptographic standards call into question the credibility of ALL cryptographic algorithms coming from the same agencies.
- After all, while problems were found with one set of algorithms, the same people who alleged subverted that standard also had input into other algorithms.
- But if not those algorithms, which ones should we be using instead?

Commercial Service Providers Are Reeling

- **Knowledge of U.S. pervasive monitoring programs may raise question about the privacy and security of commercial service providers, reducing customer trust and deterring adoption.**
- Victimized commercial providers are still struggling to harden their infrastructures so as to overcome damage to their reputations, and to re-earn their customers' trust.
- Hampering the competitiveness of American businesses doesn't enhance our economy, or our national security.

The Cyber Criminal Underground Is Watching

- **Criminal miscreants are paying close attention to what's being disclosed.**
- They don't care if they compromise your hosts via malware, vulnerable plugins you've added to a content management system, weak cryptographic implementations, or something else. They'll hit you any way they can.
- Unfortunately, **flawed crypto may dramatically expand their options.**
- And flawed crypto can translate into data breaches involving PII, ugh!

Foreign Governments Will "Do Unto Us"

- Snowden's revelations largely focused on US intelligence community programs, or collaborative international intelligence collection activities lead by the United States.
- However, we also need to assume (as a purely pragmatic reality) that if **America** runs surveillance programs targeting foreign nationals, **foreign countries will reciprocate in kind with similar programs that target Americans.** [See, e.g., "Uproar over French plan to extend online spying," <http://www.thelocal.fr/20131126/france-surveillance-privacy-internet-online-snowden-nsa>]
- Thus, even if you don't worry about what the United States does, this isn't just about what the United States does!

Yes, We Really ARE Being Watched...



http://commons.wikimedia.org/wiki/File:Grabill_-_General_Miles_and_staff.jpg

[A Brief Historical Digression About That Photo]

- The image on the preceding slide is of **General Nelson Miles**.
- Quoting Wikipedia "In the winter of 1877, [General Miles] drove his troops on a forced march across Montana and intercepted the Nez Percé band led by Chief Joseph. For the rest of Miles' career, he would quarrel with General Oliver O. Howard over credit for Joseph's capture. While on the Yellowstone, he developed expertise with the **heliograph for sending communications signals, establishing a 140-mile-long (230 km) line of heliographs connecting Fort Keogh and Fort Custer, Montana in 1878.**"
[Note, our first state-scale Northern Tier optical network!]
- Many know "**Chief Joseph**" as the leader of the Wallowa band of the Nez Perce, a Native American tribe indigenous to the Wallowa Valley in **northeastern Oregon**. He is the native leader famous for having said, "**I am tired; my heart is sick and sad. From where the sun now stands I will fight no more forever.**"

"Take Away" Ideas From This Section

- The threat is no longer "just" criminal hackers seeking passwords and credit card numbers. As a result of Snowden's revelations, we now know that another threat to our privacy and security really is pervasive monitoring by government agencies.
- Even if that pervasive monitoring is being done with the best of intentions (such as keeping us safe from national security threats), what we're learning is STILL disquieting and WILL likely have unanticipated negative side effects.
- EVERYTHING needs encryption with strong crypto.
- **But whose cryptographic standards should we use?**

III. The Cryptographic Standards Process

"I've upped my standards. Now, up yours."

Presidential campaign slogan of Pat Paulsen,
American comedian and satirist

"The IETF community's technical assessment is that PM [Pervasive Monitoring] is an attack on the privacy of Internet users and organizations. The IETF community has expressed strong agreement that PM is an attack that needs to be mitigated where possible, via the design of protocols that make PM significantly more expensive or infeasible."

<http://tools.ietf.org/html/draft-farrell-perpass-attack-06>

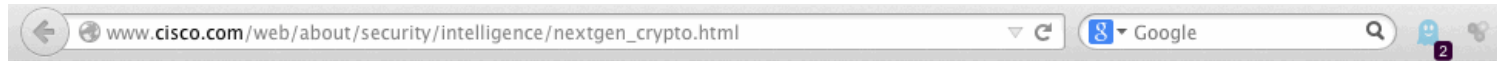
NIST, The Government, and Non-NNS Crypto

- The National Institute of Standards and Technology (NIST), part of the U.S. Department of Commerce, has traditionally lead the development and standardization of cryptographic protocols for use in *non-national security federal information systems*.
[emphasis added]
- NIST recently released an updated guide to the standards process they now follow: "NIST Cryptographic Standards and Guidelines Development Process (draft)," February 2014, http://csrc.nist.gov/publications/drafts/nistir-7977/nistir_7977_draft.pdf
- NIST aggressively discharges its cryptographic responsibilities, a reality that can be seen in the numerous cryptographic projects described at <http://csrc.nist.gov/groups/ST/index.html>

That Said, NIST Crypto Standards Tend To Be Adopted By Everybody – More or Less By Default

- As a practical matter, (1) in the absence of alternative credible cryptographic advice, and (2) to ensure that products will interoperate and meet minimum federal crypto requirements, MANY non-federal-government users use the cryptographic standards that NIST promulgates, even though they are not officially required to do so.
- In fact, pretty much *everybody* (at home or abroad) still tends to use NIST standardized cryptographic algorithms (notwithstanding worries about intentional flaws in things like Dual_EC_DRBG , see http://en.wikipedia.org/wiki/Dual_EC_DRBG)

Example: Vendor Recommending NIST Crypto



Recommendations for Cryptographic Algorithms

The following table can help customers migrate from legacy ciphers to current or more secure ciphers. The table explains each available cryptographic algorithm, the operations it supports, and whether it is Cisco's best recommendation. Customers should pay particular attention to algorithms designated as *Avoid* or *Legacy*. The status labels are explained following the table.

Table 1. Recommendations for Cryptographic Algorithms

Algorithm	Operation	Status	Alternative	Mitigation
DES	Encryption	Avoid	AES	—
3DES	Encryption	Legacy	AES	Short key lifetime
RC4	Encryption	Avoid	AES	—
AES-CBC mode	Encryption	Acceptable	AES-GCM	—
AES-GCM mode	Authenticated encryption	NGE ¹	—	—
DH-768, -1024	Key exchange	Avoid	DH-2048 (Group 14)	—
RSA-768, -1024	Encryption		RSA-2048	—
DSA-768, -1024	Authentication		DSA-2048	—
DH-2048	Key exchange	Acceptable	ECDH-256	—
RSA-2048	Encryption		—	—
DSA-2048	Authentication		ECDSA-256	—
MD5	Integrity	Avoid	SHA-256	—
SHA-1	Integrity	Legacy	SHA-256	—
SHA-256	Integrity	NGE	SHA-384	—
SHA-384			—	—
SHA-512			—	—

What About The National Security Side of Things?

- "The Committee on National Security Systems (CNSS) sets national-level Information Assurance policies, directives, instructions, operational procedures, guidance and advisories for *United States Government (USG) departments and agencies for the security of National Security Systems (NSS)*. It provides a comprehensive forum for strategic planning and operational decision-making to protect NSS and approves the release of INFOSEC products and information to Foreign Governments."

See <https://www.cnss.gov/CNSS/index.cfm> [emphasis added]

- This includes specific guidance around approved cryptographic algorithms; see for example CNSSP No. 15, "Use of Public Standards for the Secure Sharing of Information Among NSS," 10/01/2012, <https://www.cnss.gov/CNSS/issuances/Policies.cfm>

NSS Crypto and COTS Technology

- Note the title of the standard mentioned on the preceding slide:

"Use of *Public Standards* for the Secure Sharing of Information Among *NSS* [*National Security Systems*]"
(emphasis added).

- 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.

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

Implications of Those CNSS Recommendations

- That document presents us with another one of those interesting either/or logic questions...
- EITHER the CNSS is knowingly recommending and permitting the use of insecure crypto for the protection of U.S. classified information (and frankly, that's a "hard sell," at least for me)
- OR the CNSS is recommending crypto that actually IS highly resistant to compromise, including being resistant to all known/expected attacks from foreign nation state adversaries, **and even resistant to technical access by their own cryptanalysts?**
- And if Suite B IS perfectly adequate, why is there ALSO a classified Suite A option? Why isn't everyone just using Suite B?

What About International Non-NIST Alternatives?

- Some terrific crypto work does get done internationally. For example, Lange and Bernstein are both associated with tue.nl, and Adi Shamir (the "S" in RSA) is an Israeli cryptographer.
- There have been noteworthy international ciphers, too, including:
 - http://en.wikipedia.org/wiki/Camellia_%28cipher%29
 - http://en.wikipedia.org/wiki/GOST_%28block_cipher%29
 - <http://en.wikipedia.org/wiki/Rijndael>
 - http://en.wikipedia.org/wiki/Serpent_%28cipher%29
 - <http://en.wikipedia.org/wiki/SMS4>
- In a few cases, an international cipher gets tweaked/standardized by US standards bodies, e.g., Rijndael evolved into what's now AES, see <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>

Is There Anything Comparable to NIST Abroad?

- The main **European Union** cryptographic subject matter expert (SME) group is at <http://www.ecrypt.eu.org/>

That group's work includes

"Improv[ing] our understanding of existing algorithms and protocols, [e]xpand[ing] the theoretical foundations of cryptology, and [d]evelop[ing] better cryptographic algorithms, protocols and implementations ..."

In fact, though, many of the recommendations from that organization seem to rely on standards from NIST. See for example "ECRYPT II Yearly Report on Algorithms and Keysizes (2011-2012)", <http://www.ecrypt.eu.org/documents/D.SPA.20.pdf> (and they don't seem as prolific at generating standards as NIST)

What About Asia?

- **Japan's** CRYPTREC e-Government Recommended Ciphers List is at <http://www.cryptrec.go.jp/english/list.html> but most of the ciphers recommended by CRYPTREC (both in the 2003 English-language document as well as in the 2012 Japanese-language-only revision) tie back to NIST standardized ciphers. Thus, if you were considering Japan as a potential source of novel alternative ciphers, that strategy will largely fail, except for Camellia.
- **China?** Cryptographic products must be reviewed by the "State Commercial Cryptography Administration" (www.oscca.gov.cn) and the "National Secrecy Science Technology Evaluation Center" (www.isstec.org.cn). Reportedly, products must also use Chinese-developed cryptographic algorithms, but many Chinese crypto standards aren't widely available, although there are some exceptions (see <http://tools.ietf.org/html/draft-shen-sm2-ecdsa-02> and see <http://eprint.iacr.org/2008/329.pdf> , for example)

Russian Federation?

- Like many countries abroad, Russia carefully controls diffusion of cryptographic technologies, see <http://www.cryptolaw.org/cls2.htm#ru>
- The best known publicly disclosed Russian cipher is probably GOST, which was approved for use and widely used in Russia, even for use with classified materials. It was even considered for ISO standardization, but it ended up rejected due to flaws, see Nicholas Courtois' "Security Evaluation of Russian GOST Cipher," <http://events.ccc.de/congress/2012/Fahrplan/events/5225.en.html>
- GOST was implemented and remains available as an option in some cryptographic libraries (such as OpenSSL, however)

UK? Canada? Australia? New Zealand?

- The UK, Canada, Australia and New Zealand are all part of the so-called "Five Eyes" intelligence sharing community (see http://en.wikipedia.org/wiki/Five_Eyes), lead by the United States.
- As such the crypto policies of those member states tend to be harmonized with (and responsive to) recommendations of the US government agencies that are most concerned with cryptographic topics.
- Your expectations for novel and cryptographic standards from the other members of the Five Eyes should thus be low, although work from academic sources in the Five Eyes (such as the Serpent cipher) can still be worthy of note.

One Caution... Beware Amateur Cryptographers

- Both Daniel J Bernstein and Tanja Lange are literally world class cryptographers. I'm very comfortable about the quality of their work based on community review of their recommendations.
- That said, you should be extremely careful of any new or novel crypto schemes dreamt up by little-known cryptographers. Bruce Schneier does an excellent job of explaining why in:

"Memo to the amateur cipher designer,"

<https://www.schneier.com/crypto-gram-9810.html#cipherdesign>

Or as <http://www.lauradhamilton.com/10-cryptography-mistakes-amateurs-make> mentions:

"The #1 rule of cryptography is "Don't invent your own." ☺

"What About the IETF?"

- The IETF is the traditional Internet standards development body, and they're influential in the cryptographic realm, too.
- I've chosen to discuss IETF crypto activity near the end of this talk (see the community engagement section).

III. Choice of Cipher Suites

"People tend to hold overly favorable views of their abilities in many social and intellectual domains. The authors suggest that this overestimation occurs, in part, because people who are unskilled in these domains suffer a dual burden: Not only do these people reach erroneous conclusions and make unfortunate choices, but their incompetence robs them of the metacognitive ability to realize it."

"Unskilled and Unaware of It: How Difficulties in Recognizing One's Own Incompetence Lead to Inflated Self-Assessments,"
Kruger and Dunning, 1999

A Cipher Suite == Four Components

- In the case of SSL/TLS, a cipher suite consists of four components:
 - key transport: RSA, DH, DHE, ECDH, ECDHE, etc.
 - public keys: RSA, DSA, ECDSA, etc.
 - data encryption: RC4_128, 3DES, AES128, AES256, CHACHA20_POLY1305, etc.
 - hash function: SHA (e.g., SHA1), SHA256, SHA384, etc.
- For example:
TLS_RSA_WITH_RC4_128_SHA
- Another example:
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256

So Which Specific Cipher Suites Should We Use?

- **Choosing the cipher suite you're going to use is a fundamental (and potentially difficult) decision.**
- **If you choose badly, you may end up with a cipher suite that fails to adequately protect the information you're encrypting.**
- **A bad choice can also mean that you end up with a cipher selection that's highly secure, but perhaps isn't broadly supported: if that happens, web servers and browser may not be able to practically negotiate a mutually agreeable cryptographic configuration. But we're getting ahead of ourselves.**
- **Let's begin with some ciphers that you should NOT use...**

(a) Null, Anonymous, Export Ciphers, etc.

- **DON'T USE "NULL" "ciphers"...**

\$ openssl ciphers NULL

ECDHE-RSA-NULL-SHA:ECDHE-ECDSA-NULL-SHA:AECDH-NULL-SHA:ECDH-RSA-NULL-SHA:ECDH-ECDSA-NULL-SHA:NULL-SHA256:NULL-SHA:NULL-MD5

- **DON'T USE "Anonymous" ciphers...**

\$ 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:ADH-CAMELLIA128-SHA:ADH-RC4-MD5:ADH-DES-CBC-SHA:EXP-ADH-DES-CBC-SHA:EXP-ADH-RC4-MD5

Null, Anonymous and Export Ciphers, Etc (cont.)

- **DON'T USE "EXPORT" or "LOW" grade (weak) ciphers...**

\$ openssl ciphers EXPORT,LOW

EXP-EDH-RSA-DES-CBC-SHA:EXP-EDH-DSS-DES-CBC-SHA:EXP-ADH-DES-CBC-SHA:EXP-DES-CBC-SHA:EXP-RC2-CBC-MD5:EXP-RC2-CBC-MD5:EXP-ADH-RC4-MD5:EXP-RC4-MD5:EXP-RC4-MD5:EDH-RSA-DES-CBC-SHA:EDH-DSS-DES-CBC-SHA:ADH-DES-CBC-SHA:DES-CBC-SHA:DES-CBC-MD5

- **DON'T USE DES ciphers**

\$ openssl ciphers DES

EDH-RSA-DES-CBC-SHA:EDH-DSS-DES-CBC-SHA:ADH-DES-CBC-SHA:DES-CBC-SHA:DES-CBC-MD5:EXP-EDH-RSA-DES-CBC-SHA:EXP-EDH-DSS-DES-CBC-SHA:EXP-ADH-DES-CBC-SHA:EXP-DES-CBC-SHA

Null, Anonymous and Export Ciphers, Etc (cont.)

- **DON'T USE "MD5" (includes all SSLv2 ciphers)...**

\$ openssl ciphers MD5

DES-CBC3-MD5:IDEA-CBC-MD5:RC2-CBC-MD5:ADH-RC4-MD5:RC4-MD5:RC4-MD5:DES-CBC-MD5:EXP-RC2-CBC-MD5:EXP-RC2-CBC-MD5:EXP-ADH-RC4-MD5:EXP-RC4-MD5:EXP-RC4-MD5:NULL-MD5

(b) RSA for Key Transport: Don't Use It, Either

- RSA has traditionally been used for TLS on "most" web servers.
- It can be used for public key authentication (and that's fine),
or as the basis for symmetric key transport, which is NOT.
If RSA is used for key transport, the client picks a random value, and encrypts it with the server's RSA public key.
- However, note that if your RSA-encrypted network traffic gets vacuumed up over time (and it is), AND that attacker manages to get your RSA private key by hook or by crook (or by Heartbleed!), then that attacker with a big cache of traffic can retrospectively decrypt ALL that intercepted-and-saved traffic.
- This is true for ALL ciphers that don't offer "forward secrecy."

DO Strive To Deploy Forward Secrecy

- Alternatives to RSA that offer forward secrecy include:
 - Diffie-Hellman **Ephemeral** (DHE)
 - Elliptic Curve Diffie-Hellman **Ephemeral** (ECDHE)
- <https://www.trustworthyinternet.org/ssl-pulse/> reports on support for Forward Secrecy:
 - **51.7% Forward secrecy is NOT supported**
 - 42.0% SOME FS suites enabled
 - 2.6% FS **USED WITH MODERN** browsers
 - 3.7% FS **USED WITH MOST** browsers

RSA Key Transport Removed in TLS 1.3

- From: "Joseph Salowey (jsalowey)" <jsalowey@cisco.com>
Date: Sat, 26 Apr 2014 15:24:41 +0000
Cc: "<tls@ietf.org>" <tls@ietf.org>
Subject: Re: [TLS] Confirming Consensus on removing RSA key
Transport from TLS 1.3

The discussion on this list and others supports the consensus in IETF 89 to remove RSA key transport cipher suites from TLS 1.3. The Editor is requested to make the appropriate changes to the draft on github.

More discussion is needed on both DH and ECDH are used going forward and on if standard DHE parameters will be specified.

Joe

[For the chairs]

[remainder snipped here]

What About RSA For Certificate Public Keys?

- RSA (at least RSA-2048 or stronger) can continue to be used for cert public keys. That's good, since **over 99% of publicly trusted certificates were recently found to rely on RSA public keys.**

See "Analysis of the HTTPS Certificate Ecosystem," October 2013, <https://jhalderm.com/pub/papers/https-imc13.pdf> at 6.1

- Those authors did note "Over the course of the past year, we found 47 certificates that contain ECDSA public keys; none were present in our March 22 scan and none were browser trusted." Table 9 in that paper also mentions 17 DSA-signed keys out of millions of certificates inspected. :-;
- **It's an RSA world** when it comes to cert public keys right now.

(c) SHA Hashes (Message Authentication Codes)

- SHA-1 (or just SHA) is a fingerprint (or signature value) associated with the contents of a file, referred to as a "hash." Example:

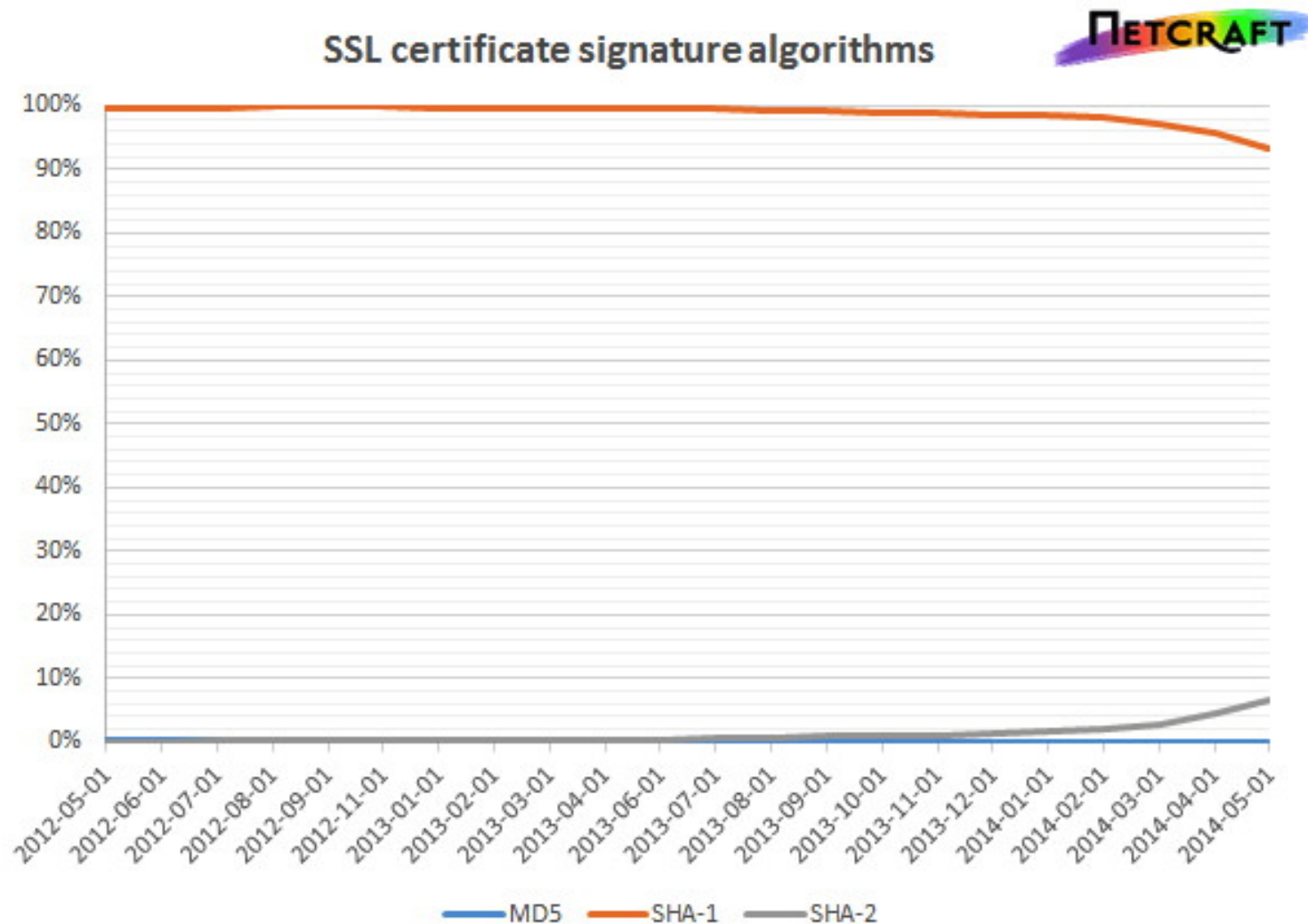
```
% sha1sum index.html
```

```
318f9256603aa09aec7552a027f09d35ff2e4eab index.html
```

Changing even one bit of a hashed file will result in a different hash. Like fingerprints or snowflakes, two non-identical files should never have the same hash (this is referred to as a "hash collision").

- Hash functions are routinely used on SSL/TLS certificates. "Analysis of the HTTPS Certificate Ecosystem," mentioned earlier, **reported 98.7% of all trusted certs used SHA-1**. In a later study, Netcraft actually reported some progress moving certs toward SHA-2 (e.g., SHA-256) signatures. See the following slide.

SHA-1 vs. SHA-256 For SSL Certs Per Netcraft



Source: <http://news.netcraft.com/wp-content/uploads/2014/05/cert-sig-alg2.png>

The Problem With SHA-1?

Table 9: Hash Function Transitions

Hash Function	Use	
SHA-1	Digital signature generation	Acceptable through 2010 Deprecated from 2011 through 2013 Disallowed after 2013
	Digital signature verification	Acceptable through 2010 Legacy-use after 2010
	Non-digital signature generation applications	Acceptable
SHA-224		

Source: "Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths," <http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf> at PDF page 17.

What Are SHA-256 or SHA-384 Hashes Like For Comparison?

- Conceptually, they look a lot like a SHA-1 value, they're just longer/cryptographically stronger:

```
% sha1sum index.html
```

```
318f9256603aa09aec7552a027f09d35ff2e4eab index.html
```

```
% sha256sum index.html
```

```
4b0c64ae31a08a6852f4f66eff6205c5ee99ee257edc1b91053d6cb31  
210d77b index.html
```

```
% sha384sum index.html
```

```
4c9cc986fd33b5dabd8bdbb5f387ef6faf5fd589b24b34e7bde8920  
38fa596e26a9e703a956352ec110d2cfbc295f6 index.html
```

CA Advice on Transitioning Away From SHA-1

- While most Certificate authorities are careful to note that some older systems may not be able to support SHA-2, most CAs are pretty clear with their advice for transitioning from SHA-1 to SHA-2 – basically "do it." See for example:
- <http://www.comodo.com/e-commerce/SHA-2-transition.php>
- <http://www.digicert.com/sha-2-ssl-certificates.htm>
- <https://www.globalsign.com/ssl-information-center/transitioning-to-sha-256.html>
- [And yes, the InCommon Cert Service will be modified to support SHA-2 as an option for InCommon Cert Service users shortly]

(d) RC4: Stop Using It, Too

- RC4 (aka ARCFOUR, see <http://en.wikipedia.org/wiki/RC4>) is reportedly the most widely used (and often the only supported) SSL/TLS stream cipher.
- **Many sites do in fact routinely support it.**
<https://www.trustworthyinternet.org/ssl-pulse/> says that of the world's top 200,000 or so SSL-enabled web sites, **only 8.7% DO NOT support RC4.**
- Looking at a broader sample of five million sites, Microsoft found that uptake isn't quite as strong, with at least 58% of servers in their sample NOT using it, and only 3.9% REQUIRING its use.
See <http://blogs.technet.com/b/srd/archive/2013/11/12/security-advisory-2868725-recommendation-to-disable-rc4.aspx>

Thank Goodness Required Use of RC4 Is Low...

- Low required use is good, because RC4 has serious issues.
- Many are recommending that people stop using it altogether, see for example:

-- <https://tools.ietf.org/html/draft-popov-tls-prohibiting-rc4-02>

-- <http://blog.cloudflare.com/killing-rc4>

-- <https://community.qualys.com/blogs/securitylabs/2013/03/19/rc4-in-tls-is-broken-now-what>

-- <http://blogs.technet.com/b/srd/archive/2013/11/12/security-advisory-2868725-recommendation-to-disable-rc4.aspx>

So How Did RC4 End Up So Broadly Deployed???

- RC4 was (a) broadly supported in popular crypto libraries, and (b) relatively efficient. That efficiency made it popular with major web sites. Amateur cipher connoisseurs would look to see what the "big guys" used, and then they'd often emulate them without understanding the implications of their choice.
- RC4 was also one of the only options originally available for mitigating the BEAST attack back in the Fall of 2011 (see en.wikipedia.org/wiki/Transport_Layer_Security#BEAST_attack). Later versions of TLS (e.g., TLS 1.1 and 1.2) eliminated the BEAST issue. **Bottom line, RC4 is all over the place out there, but pretty much everyone agrees that it *needs* to go.**

Sample Problematic RC4 Configuration



Cipher Suites (SSL 3+ suites in server-preferred order, then SSL 2 suites where used)

TLS_RSA_WITH_RC4_128_SHA (0x5)	128
TLS_RSA_WITH_AES_128_CBC_SHA (0x2f)	128
TLS_RSA_WITH_AES_256_CBC_SHA (0x35)	256
TLS_RSA_WITH_3DES_EDE_CBC_SHA (0xa)	112
TLS_RSA_WITH_AES_128_CBC_SHA256 (0x3c)	128
TLS_RSA_WITH_AES_256_CBC_SHA256 (0x3d)	256



Handshake Simulation

Android 2.3.7 No SNI ²	TLS 1.0	TLS_RSA_WITH_RC4_128_SHA (0x5)	No FS	RC4	128
Android 4.0.4	TLS 1.0	TLS_RSA_WITH_RC4_128_SHA (0x5)	No FS	RC4	128
Android 4.1.1	TLS 1.0	TLS_RSA_WITH_RC4_128_SHA (0x5)	No FS	RC4	128
Android 4.2.2	TLS 1.0	TLS_RSA_WITH_RC4_128_SHA (0x5)	No FS	RC4	128
Android 4.3	TLS 1.0	TLS_RSA_WITH_RC4_128_SHA (0x5)	No FS	RC4	128
Android 4.4.2	TLS 1.2	TLS_RSA_WITH_RC4_128_SHA (0x5)	No FS	RC4	128
BingBot Dec 2013 No SNI ²	TLS 1.0	TLS_RSA_WITH_RC4_128_SHA (0x5)	No FS	RC4	128
BingPreview Dec 2013	TLS 1.0	TLS_RSA_WITH_RC4_128_SHA (0x5)	No FS	RC4	128
Chrome 33 / Win 7 R	TLS 1.2	TLS_RSA_WITH_RC4_128_SHA (0x5)	No FS	RC4	128
Firefox 24.2.0 ESR / Win 7	TLS 1.0	TLS_RSA_WITH_RC4_128_SHA (0x5)	No FS	RC4	128
Firefox 27 / Win 8 R	TLS 1.2	TLS_RSA_WITH_RC4_128_SHA (0x5)	No FS	RC4	128

Q. What Symmetric Cipher Should I Use Instead of RC4?

- **A. Use AES-128 or AES-256, probably in GCM mode.**
- If you don't like AES for some reason, use CAMELIA-128 or CAMELIA-256, or try CHACHA20.
- Note: if you are using AES-GCM, you **also** need to be using TLS 1.2. (I wouldn't recommend continuing to use TLS 1.1 or TLS 1.0, nor SSL 3.0 or SSL 2.0)
- You may want to review "Recommendations for Secure Use of TLS and DTLS", <http://tools.ietf.org/html/draft-ietf-uta-tls-bcp-00> March 27, 2014 at section 3.3
- See also "Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations," www.nist.gov/manuscript-publication-search.cfm?pub_id=915295
- **<https://bettercrypto.org/static/applied-crypto-hardening.pdf>**

If You Are 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

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.

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

(e) Elliptic Curve Crypto with NIST Curves

- 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 bit value is just a **bit** more difficult.
- About ten years ago, the community began to move 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/>

ECC Trusted Roots?

- In order to be able to do Suite B-recommended crypto, you need a cert that chains to an ECC root.
- Currently there are just **four (4)** 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 ECC (EV)

Trend Micro Affirm Trust Premium ECC (EV)

Verisign/Symantec GeoTrust Primary Certification Authority ECC (NOT EV)

Verisign/Symantec Thawte Primary Root CA - G2 ECC (NOT EV)

Why Do We "Need" Elliptic Curve Crypto?

- Short answer: arguably you want to be ready if there's a breakthrough in factoring long integers and RSA crypto ceases to be an option (this 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)

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

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 elliptic curve you're try.
- **NIST has recommended curves. SO DO OTHERS.** If you're considering doing ECC (and you should be), I'd urge you to review <http://safecurves.cr.yp.to/> Not all ECC curves are as good as others.
- **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/>

Curve	Safe?	Parameters:			ECDLP security:				ECC security:			
		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.

IV. Cryptographic Implementation Flaws (Including Problems With OpenSSL)

Anything worth doing, is worth doing right.

Hunter S. Thompson, 70's "Gonzo" Journalist

One Specific OpenSSL Bug: Heartbleed



The Heartbleed Bug

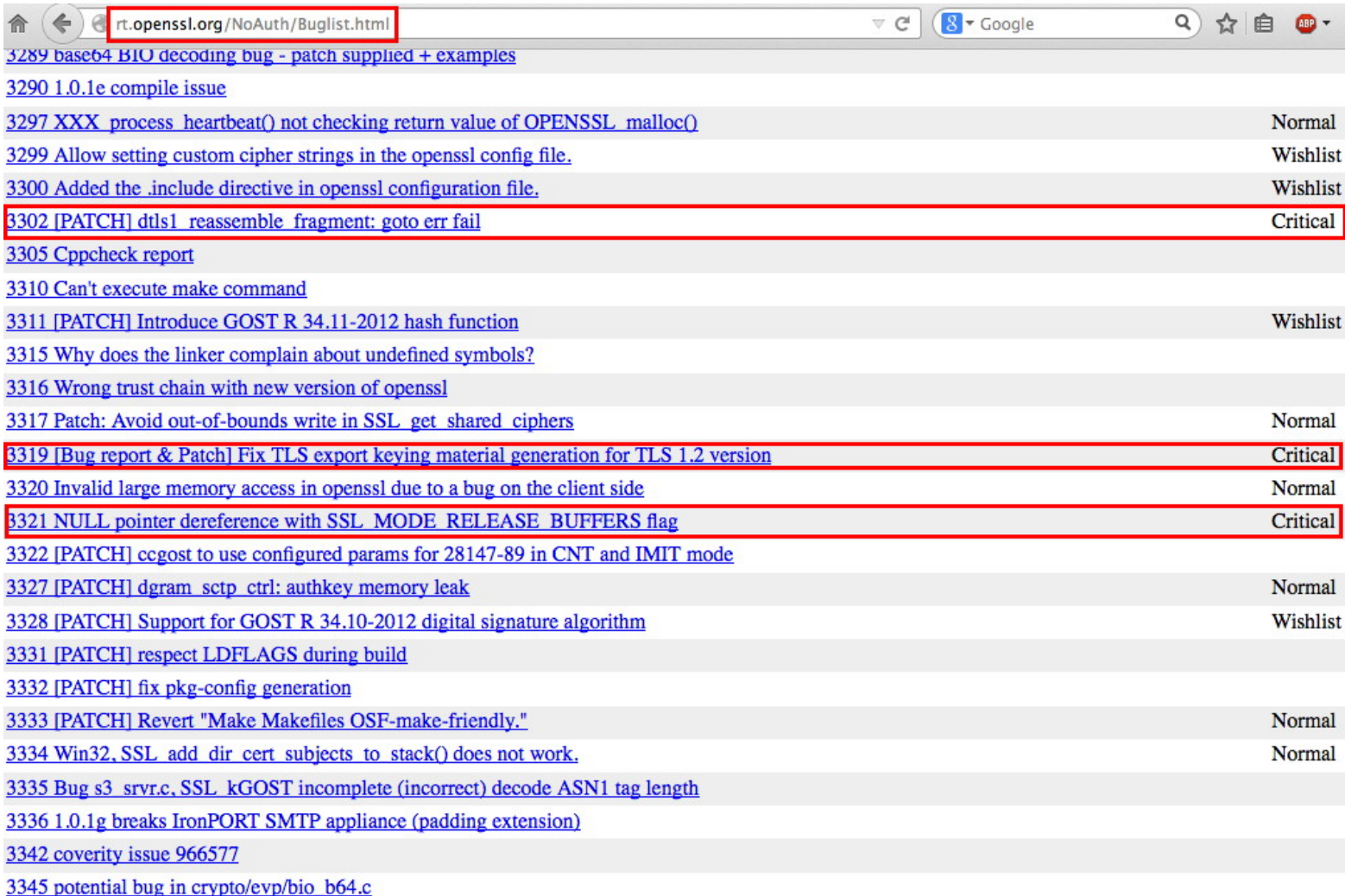
The Heartbleed Bug is a serious vulnerability in the popular OpenSSL cryptographic software library. This weakness allows stealing the information protected, under normal conditions, by the SSL/TLS encryption used to secure the Internet. SSL/TLS provides communication security and privacy over the Internet for applications such as web, email, instant messaging (IM) and some virtual private networks (VPNs).

The Heartbleed bug allows anyone on the Internet to read the memory of the systems protected by the vulnerable versions of the OpenSSL software. This compromises the secret keys used to identify the service providers and to encrypt the traffic, the names and passwords of the users and the actual content. This allows attackers to eavesdrop on communications, steal data directly from the services and users and to impersonate services and users.



Other Open OpenSSL Bugs:






<http://rt.openssl.org/NoAuth/Buglist.html>



3289 base64 BIO decoding bug - patch supplied + examples	
3290 1.0.1e compile issue	
3297 XXX process heartbeat() not checking return value of OPENSSL_malloc()	Normal
3299 Allow setting custom cipher strings in the openssl config file.	Wishlist
3300 Added the .include directive in openssl configuration file.	Wishlist
3302 [PATCH] dtls1_reassemble_fragment: goto err fail	Critical
3305 Cppcheck report	
3310 Can't execute make command	
3311 [PATCH] Introduce GOST R 34.11-2012 hash function	Wishlist
3315 Why does the linker complain about undefined symbols?	
3316 Wrong trust chain with new version of openssl	
3317 Patch: Avoid out-of-bounds write in SSL_get_shared_ciphers	Normal
3319 [Bug report & Patch] Fix TLS export keying material generation for TLS 1.2 version	Critical
3320 Invalid large memory access in openssl due to a bug on the client side	Normal
3321 NULL pointer dereference with SSL_MODE_RELEASE_BUFFERS flag	Critical
3322 [PATCH] ccgost to use configured params for 28147-89 in CNT and IMIT mode	
3327 [PATCH] dgram_sctp_ctrl: authkey memory leak	Normal
3328 [PATCH] Support for GOST R 34.10-2012 digital signature algorithm	Wishlist
3331 [PATCH] respect LDFLAGS during build	
3332 [PATCH] fix pkg-config generation	
3333 [PATCH] Revert "Make Makefiles OSF-make-friendly."	Normal
3334 Win32, SSL_add_dir_cert_subjects_to_stack() does not work.	Normal
3335 Bug s3_srvr.c, SSL_kGOST incomplete (incorrect) decode ASN1 tag length	
3336 1.0.1g breaks IronPORT SMTP appliance (padding extension)	
3342 coverity issue 966577	
3345 potential bug in crypto/evp/bio_b64.c	

An *OpenBSD* Fork of *OpenSSL*

-- One Representative Change Notice --

 freshbsd.org/search?project=openbsd&q=file.name%3Alibssl&page=3   Google

OpenBSD — lib/libssl/src/crypto/bio b_sock.c bss_dgram.c
miod @ master - 728fcd80 - 2014-04-23 20:59:36

The usual idiom to cope with systems not defining `socklen_t` is to add a `#define socklen_t int` somewhere (or a typedef, whatever gives you an integer type of the size your system expects as the 3rd argument of `accept(2)`, really).

OpenSSL here is a bit more creative by using an union of an `int` and a `size_t`, and extra code if `sizeof(int) != sizeof(size_t)` in order to recover the proper size. With a comment mentioning that this has no chance to work on a platform with a stack growing up and `accept()` returning an `int`, fortunately this seems to work on HP-UX.

Switch to the light side of the force and declare and use `socklen_t` variables, period. If your system does not define `socklen_t`, consider bringing it back to your vendor for a refund.

ok matthew@ tedeu

+5	-35	lib/libssl/src/crypto/bio/b_sock.c
+3	-14	lib/libssl/src/crypto/bio/bss_dgram.c
+8	-49	2 files

Another Attempt At Fixing OpenSSL: The [Corporate] Core Infrastructure Initiative



Core Infrastructure Initiative

The Core Infrastructure Initiative is a multi-million dollar project housed at The Linux Foundation to fund open source projects that are in the critical path for core computing functions. Inspired by the Heartbleed OpenSSL crisis, The Initiative's funds will be administered by the Linux Foundation and a steering group comprised of backers of the project as well as key open source developers and other industry stakeholders.



The steering group will work with an advisory board of esteemed open source developers to identify and fund open source projects in need. Support from the initiative can include funding for fellowships for key developers to work full time on the open source project, security audits, computing and test infrastructure, travel, face-to-face meeting coordination and other support. Early supporters include:



We expect more to follow suit in the coming weeks and months. Members of CII will evaluate open source projects that are essential to global computing infrastructure and are experiencing under-investment. These companies recognize the need for directed funds for highly critical open source software projects they all consume and that run much of modern day society. They also value and invest in developers and collaborative software development and want to support this important work.

» [View FAQ For More Details](#)

HeartBleed Followup Check: Revoking Old Certs

- After Heartbleed, due to the potential for private keys to have been compromised, many people obtained new certificates, or rekeyed existing certificates.
- However, we know from inspecting publicly available certificate revocation lists (CRLs) that at least some sites have NOT revoked their old (now replaced) certificates.
- Nice graph available at <http://isc.sans.edu/crls.html>
- If your site had vulnerable systems that may have been affected by Heartbleed, and you've installed new certificates (or rekeyed existing certificates) **be SURE that your old certificate has been revoked.**
- **If you don't revoke your old certificate and someone manages to get that cert and your private key, then your site can be undetectably impersonated via a MITM attack. That's bad.**

BUT... Revocation Doesn't Always Work

- If you go through the trouble of revoking a certificate, you probably hope that people will notice it's been revoked (if they somehow bump into it) either checking the Online Certificate Status Protocol (OCSP) or Certificate Revocation Lists (CRLs).
- Unfortunately, increasingly many browsers are NOT checking the revocation status of certs via OCSP or CRL mechanisms. See for example the discussion that's at:

<https://www.imperialviolet.org/2014/04/19/revchecking.html>

<https://wiki.mozilla.org/CA:ImprovingRevocation>

- Curious about your favorite browser? Check it:

<https://www.grc.com/revocation.htm>

Chrome Results for The GRC Revocation Check



Security Certificate Revocation Awareness Test

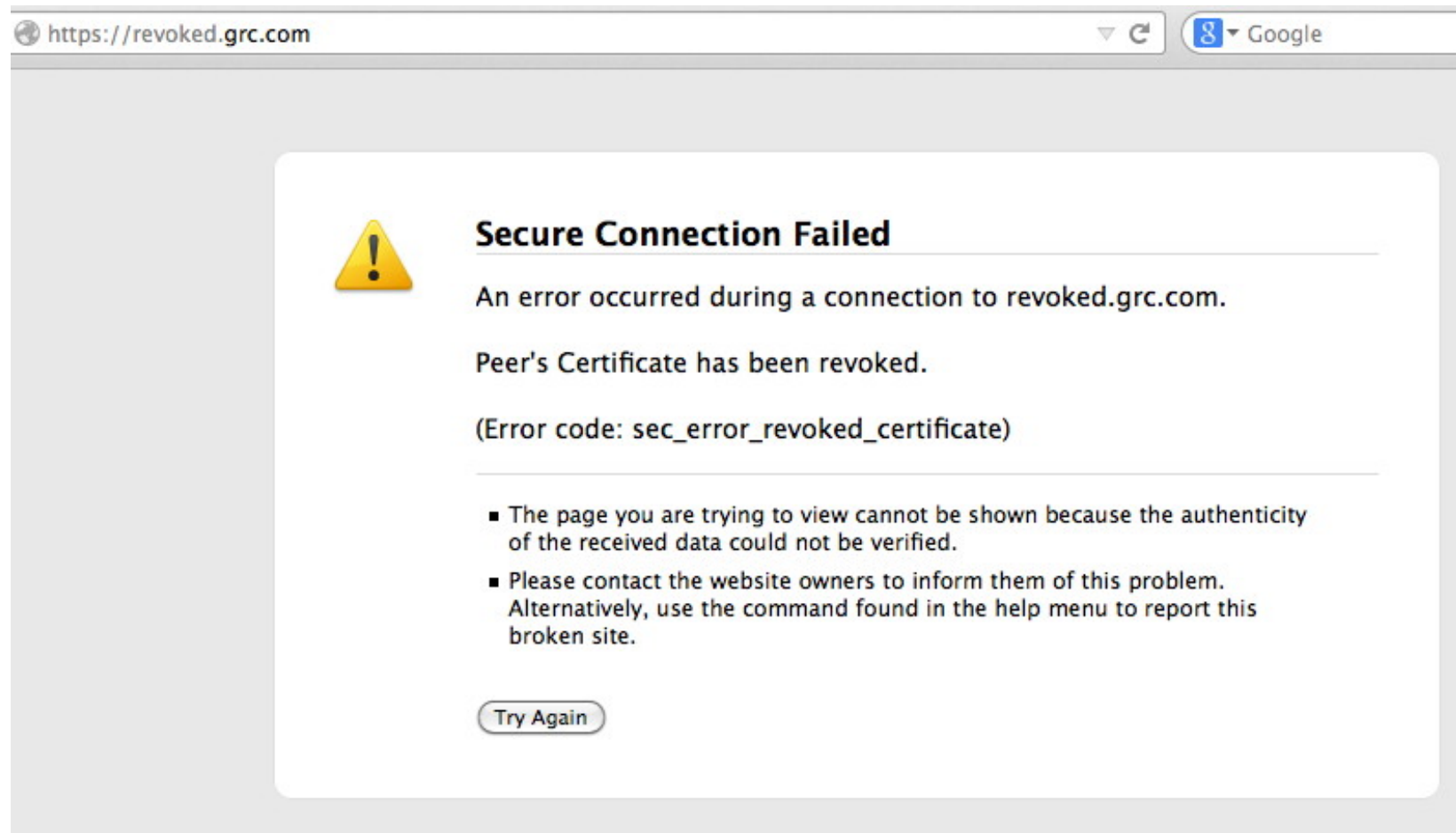
If you can see this (and apparently you can), you
are using a **revocation UNaware** web browser!

The SSL/TLS security certificate for this special website has been deliberately revoked. Since you **are** seeing this page, we know that this web browser is allowing a site with a **known invalid certificate** to display its pages. This is likely not the behavior you would choose.

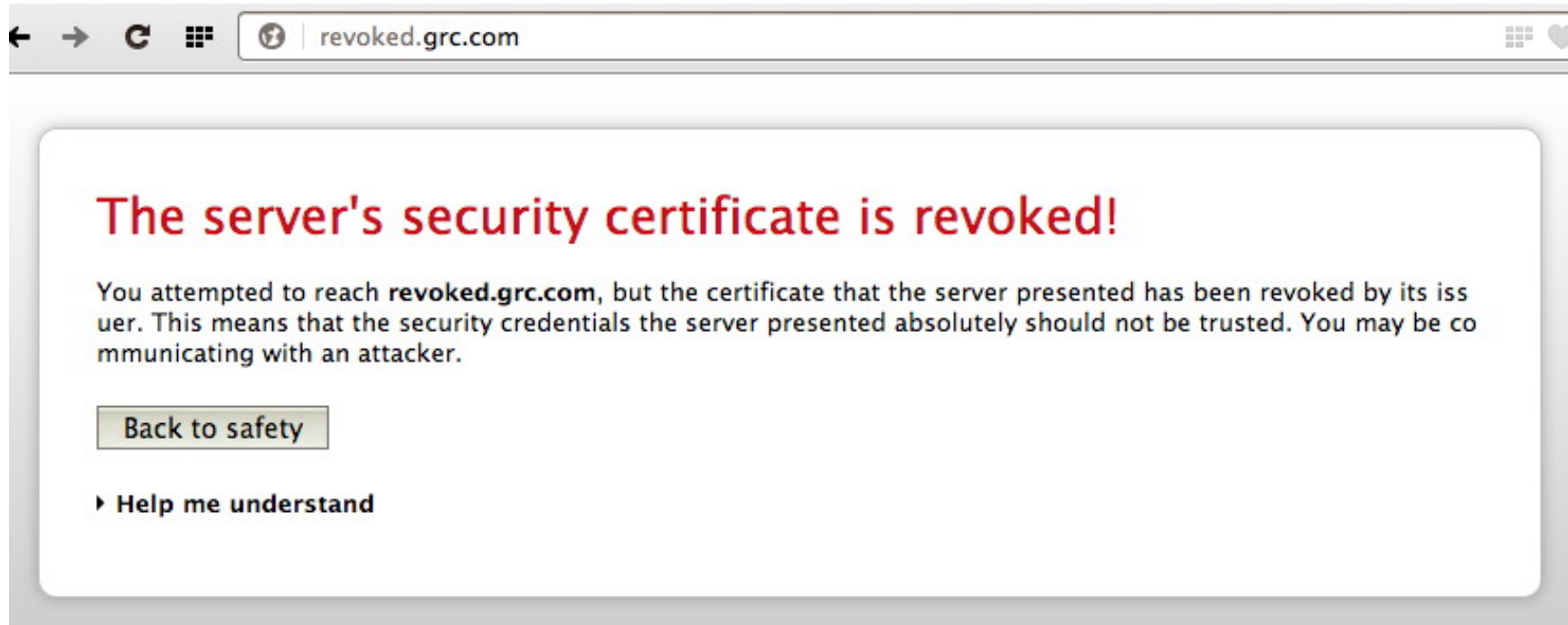
Here's what we know:

- **The Firefox browser** currently leads the industry in certificate revocation checking security. It incorporates its own mature internal technology and Firefox checks for revocation by default (thus protecting all users). And it does this on every operating system platform.
- **Google's Chrome browser** is the least certificate-secure browser on the Internet. It puts speed before security, so *it is the only browser on the Internet* to disable certificate checking by default.
- **Internet Explorer** uses Windows' built-in revocation checking which is also enabled by default.
- **The mobile Android platform** currently offers **no** certificate revocation checking of its own, so Android apps (including all users of Google's Chrome browser) are vulnerable to malicious certificate abuse. The only way to use Android securely today is with Firefox, which brings along its own certificate security.
- **iOS is only a bit better than the Android disaster.** It only checks for revocation of extended validation (EV) certificates and trusts everything else. On iOS, Safari and the LastPass Tab browser get this benefit. But incredibly, Chrome on iOS doesn't pay attention to any certificate revocation.

Firefox Results for The GRC Revocation Check



Opera Results for The GRC Revocation Check



OpenSSL Wasn't the Only Crypto Library Bug: Consider, For Example The Apple "goto fail" Bug

https://www.imperialviolet.org/2014/02/22/applebug.html

So here's the Apple bug:

```
static OSStatus
SSLVerifySignedServerKeyExchange(SSLContext *ctx, bool isRsa, SSLBuffer signedParams,
                                uint8_t *signature, UInt16 signatureLen)
{
    OSStatus      err;
    ...

    if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
        goto fail;
        goto fail;
    if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
        goto fail;
    ...

fail:
    SSLFreeBuffer(&signedHashes);
    SSLFreeBuffer(&hashCtx);
    return err;
}
```

(Quoted from [Apple's published source code](#).)

Note the two `goto fail` lines in a row. The first one is correctly bound to the `if` statement but the second, despite the indentation, isn't conditional at all. The code will always jump to the end from that second `goto`, `err` will contain a successful value because the SHA1 update operation was successful and so the signature verification will never fail.

Or The GNUTLS Cert Verification Bug...

arstechnica.com/security/2014/03/critical-crypto-bug-leaves-linux-hundreds-of-apps-open-to-eavesdropping/

Critical crypto bug leaves Linux, hundreds of apps open to eavesdropping

This GnuTLS bug is worse than the big Apple "goto fail" bug patched last week.

by Dan Goodin - Mar 4, 2014 6:56 pm UTC

HACKING PRIVACY 406



A. Strakey

Hundreds of open source packages, including the Red Hat, Ubuntu, and Debian distributions of Linux, are susceptible to attacks that circumvent the most widely used technology to prevent eavesdropping on the Internet, thanks to an extremely critical vulnerability in a widely used cryptographic code library.

I Think There's LOTS More Still To Come

We design, implement, and apply the first methodology for large-scale testing of certificate validation logic in SSL/TLS implementations. Our first ingredient is “frankencerts,” synthetic certificates that are randomly mutated from parts of real certificates and thus include unusual combinations of extensions and constraints. Our second ingredient is differential testing: if one SSL/TLS implementation accepts a certificate while another rejects the same certificate, we use the discrepancy as an oracle for finding flaws in individual implementations.

Differential testing with frankencerts uncovered 208 discrepancies between popular SSL/TLS implementations such as OpenSSL, NSS, CyaSSL, GnuTLS, PolarSSL, MatrixSSL, etc. Many of them are caused by serious security vulnerabilities. For example, any server with a valid X.509 version 1 certificate can act as a rogue certificate authority and issue fake certificates for any domain, enabling man-in-the-middle attacks against MatrixSSL and GnuTLS. Several implementations also accept certificate authorities created by unauthorized issuers, as well as certificates not intended for server authentication.

We also found serious vulnerabilities in how users are warned about certificate validation errors. When presented with an expired, self-signed certificate, NSS, Safari, and Chrome (on Linux) report that the certificate has expired—a low-risk, often ignored error—but not that the connection is insecure against a man-in-the-middle attack.

Source: https://www.cs.utexas.edu/~shmat/shmat_oak14.pdf

V. Crypto in The Browser

"... the browser is the dominant layer, the one nexus for software, the one switchboard where all power lies. It needs from the operating system a rectangle to draw the Web page, a bit of storage space, and a TCP/IP feed. It does everything else in a cross-platform way that is, when all is considered, relatively free of bugs and other issues."

<http://www.infoworld.com/d/applications/10-reasons-the-browser-becoming-the-universal-os-230812>

The Browser As "Arbiter of the Possible"

- In order for an encrypted connection to be successfully negotiated, both the web server and the web client (e.g., the web browser) need to be in agreement about a mutually supported cipher suite.
- CURRENT browsers rarely run into crypto issues, but problems can arise if you are running an antiquated browser. (Of course, if you're running an antiquated browser, it likely has a range of broader and deeper security issues, anyhow!). **Explicit recommendation: don't let old versions of XP (now end of life!) or ancient browsers force you toward supporting weak crypto.**
- From a cipher selection point of view, different browsers do *prefer* different cipher selections... you can test your favorite browser(s) with <https://www.ssllabs.com/ssltest/viewMyClient.html>

Firefox Cipher Preference

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

Google



Cipher Suites (in order of preference)

TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)	Forward Secrecy	128
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)	Forward Secrecy	128
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)	Forward Secrecy	256
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009)	Forward Secrecy	128
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)	Forward Secrecy	128
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)	Forward Secrecy	256
TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA (0xc012)	Forward Secrecy	112
TLS_ECDHE_ECDSA_WITH_RC4_128_SHA (0xc007)	Forward Secrecy	128
TLS_ECDHE_RSA_WITH_RC4_128_SHA (0xc011)	Forward Secrecy	128
TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x33)	Forward Secrecy	128
TLS_DHE_DSS_WITH_AES_128_CBC_SHA (0x32)	Forward Secrecy*	128
TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA (0x45)	Forward Secrecy	128
TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x39)	Forward Secrecy	256
TLS_DHE_DSS_WITH_AES_256_CBC_SHA (0x38)	Forward Secrecy*	256
TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA (0x88)	Forward Secrecy	256

WHY Did Firefox Pick That Prioritization?

https://wiki.mozilla.org/Security/Server_Side_TLS#Recommended_Cipher ▾ ↻  Google



Prioritization logic

1. ECDHE+AESGCM ciphers are selected first. These are TLS 1.2 ciphers and not widely supported at the moment. No known attack currently target these ciphers.
2. [PFS](#) ciphersuites are preferred, with ECDHE first, then DHE.
3. AES 128 is preferred to AES 256. There has been [discussions](#) on whether AES256 extra security was worth the cost, and the result is far from obvious. At the moment, AES128 is preferred, because it provides good security, is really fast, and seems to be more resistant to timing attacks.
4. AES is preferred to RC4. [BEAST](#) attacks on AES are mitigated in TLS 1.1 and above, and difficult to achieve in TLS 1.0. In comparison, attacks on RC4 are not mitigated and likely to become more and more dangerous.
5. RC4 is on the path to removal, but still present for backward compatibility, see the discussion in [#RC4_weaknesses](#)

Chrome Cipher Preference

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



Cipher Suites (in order of preference)

TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcc14)	Forward Secrecy	256
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcc13)	Forward Secrecy	256
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)	Forward Secrecy	128
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)	Forward Secrecy	128
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 (0x9e)	Forward Secrecy	128
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)	Forward Secrecy	256
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009)	Forward Secrecy	128
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)	Forward Secrecy	128
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)	Forward Secrecy	256
TLS_ECDHE_ECDSA_WITH_RC4_128_SHA (0xc007)	Forward Secrecy	128
TLS_ECDHE_RSA_WITH_RC4_128_SHA (0xc011)	Forward Secrecy	128
TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x33)	Forward Secrecy	128

Opera Cipher Preference



Cipher Suites (in order of preference)

TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcc14)	Forward Secrecy	256
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcc13)	Forward Secrecy	256
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)	Forward Secrecy	128
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)	Forward Secrecy	128
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 (0x9e)	Forward Secrecy	128
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)	Forward Secrecy	256
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009)	Forward Secrecy	128
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)	Forward Secrecy	128
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)	Forward Secrecy	256
TLS_ECDHE_ECDSA_WITH_RC4_128_SHA (0xc007)	Forward Secrecy	128
TLS_ECDHE_RSA_WITH_RC4_128_SHA (0xc011)	Forward Secrecy	128
TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x33)	Forward Secrecy	128

Maxthon Cipher Preference

Protocol Features



Protocols*

TLS 1.2	No
TLS 1.1	Yes
TLS 1.0	Yes
SSL 3	Yes
SSL 2	No


(*) This test reliably detects only the highest supported protocol.



Cipher Suites (in order of preference)

TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA(0xc00a) Forward Secrecy	256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA(0xc014) Forward Secrecy	256
TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA(0x88) Forward Secrecy	256
TLS_DHE_DSS_WITH_CAMELLIA_256_CBC_SHA(0x87) Forward Secrecy*	256
TLS_DHE_RSA_WITH_AES_256_CBC_SHA(0x39) Forward Secrecy	256
TLS_DHE_DSS_WITH_AES_256_CBC_SHA(0x38) Forward Secrecy*	256
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA(0xc00f)	256
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA(0xc005)	256
TLS_RSA_WITH_CAMELLIA_256_CBC_SHA(0x84)	256
TLS_RSA_WITH_AES_256_CBC_SHA(0x35)	256
TLS_ECDHE_ECDSA_WITH_RC4_128_SHA(0xc007) Forward Secrecy	128
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA(0xc009) Forward Secrecy	128

howssmyssl.com on Maxthon

→ ↺ ↻ ⌂ ↶ ☆ <https://www.howssmyssl.com> 

How's My SSL? Home About API

Version

Improvable Your client is using TLS 1.1. It would be better to be TLS 1.2, but at least it isn't susceptible to the BEAST attack. But, it also doesn't have the AES-GCM cipher suite available.

[Learn More](#)

Ephemeral Key Support

Good Ephemeral keys are used in some of the cipher suites your client supports. This means your client may be used to provide [forward secrecy](#) if the server supports it. This greatly increases your protection against snoopers, including global passive adversaries who scoop up large amounts of encrypted traffic and store them until their attacks (or their computers) improve.

[Learn More](#)

Session Ticket Support

Good Session tickets are supported in your client. Services you use will be able to scale out their TLS connections more easily with this feature.

[Learn More](#)

TLS Compression

Good Your TLS client does not attempt to compress the settings that encrypt your connection, avoiding information leaks from the [CRIME attack](#).

[Learn More](#)

BEAST Vulnerability

Good Your client is not vulnerable to the [BEAST attack](#) because it's using a TLS protocol newer than TLS 1.0. The BEAST attack is only possible against clients using TLS 1.0 or earlier using [Cipher-Block Chaining](#) cipher suites that do not implement the 1/n-1 record splitting mitigation.

[Learn More](#)

Insecure Cipher Suites

Bad Your client supports cipher suites that are known to be insecure:

- [SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA](#): This cipher was meant to die with SSL 3.0 and is of unknown safety.
- [TLS_DHE_DSS_WITH_RC4_128_SHA](#): This cipher uses keys smaller than 128 bits in its encryption.

[Learn More](#)

81,665 Lines of Code Down to 4,167 Lines of Code

<https://blog.mozilla.org/security/2014/04/24/exciting-updates-to-certificate-verification-in-gecko/>

APR 24 2014

Exciting Updates to Certificate Verification in Gecko

cviecco 9

Today we're excited to announce a new certificate verification library for Mozilla Products – mozilla::pkix! While most users will not notice a difference, the new library is more robust and maintainable. The new code is more robust because certificate path building attempts all potential trust chains for a certificate before giving up (acknowledging the fact that the certificate space is a cyclic directed graph and not a [forest](#)). The new implementation is also more maintainable, with only 4,167 lines of C++ code compared to the previous 81,865 lines of code which had been auto-translated from Java to C. The new library benefits from C++ functionality such as memory cleanup tools (e.g., [RAII](#)).

To provide some more background, Gecko has historically used the certificate verification processing in [NSS](#) to ensure that the certificates presented during a TLS/SSL handshake is valid. NSS currently has two code paths for doing certificate verification: "classic" used by Gecko for [Domain Validated \(DV\)](#) certificate verification, and libPKIX used by Gecko for [Extended Validation \(EV\)](#) certificate verification. The NSS team has wanted to replace the "classic" verification with libPKIX for some time because libPKIX [handles cross-signed certificates better](#) and properly handles certificate policies required for [Enhanced Validation \(EV\)](#) certificates. However, libPKIX has proven to be very difficult to work with.

More That Can Be Done to Harden the Browser?

- At the request of the Educause Technologies, Operations and Practices (TOP) Working Group (formerly the Educause Security Effective Practices Working Group), I prepared a draft whitepaper that explains how to browse the web more securely and privately with Firefox on a Mac.

If you're interested, feel free to check out the 0.1 draft version of that document (about 85 pages. "Joe style," sorry) at:

<http://pages.uoregon.edu/joe/browsing-securely-mac-firefox/browsing-securely-mac-firefox.pdf>

Take a look if you're interested. Feedback is always appreciated. It talks about a lot of stuff we don't have time to go over this morning.

VI. Engaging With The Community

"[...] if a beachhead of cooperation may push back the jungle of suspicion, let both sides join in creating a new endeavor, not a new balance of power, but a new world of law, where the strong are just and the weak secure and the peace preserved. All this will not be finished in the first 100 days. Nor will it be finished in the first 1,000 days, nor in the life of this Administration, nor even perhaps in our lifetime on this planet. But let us begin."

John F. Kennedy's Inaugural Address, January 20, 1961

How Will YOU Make a Difference?

- One of the most positive aspects of the Snowden revelations and things like the Heartbleed incident has been the way the community has become mobilized to work on the security and privacy challenges we collectively face. It's been a wake up call, quite frankly.
- What will your contribution to that effort be? Someday, when your children or grandchildren ask, "*Mommy (or Daddy), what did you do during the second crypto wars?*" what will you be able to say?
- I'd encourage you to consider becoming involved, lending your perspective and your abilities to helping to harden the Internet.
- Many new opportunities exist, particularly in the IETF community.

IETF SAAG

- "The SAAG List is the IETF mailing list for the Security Area Advisory Group which meets once at each IETF meeting as part of the Security Area."

<https://www.ietf.org/mailman/listinfo/saag>

- Sample topics from a recent day off the mailing list can be seen on the next slide.

A Recent Day From The SAAG List

- **May 02 2014**

- [Re: \[saag\] Algorithm agility](#), *Daniel Kahn Gillmor*
- [Re: \[saag\] Algorithm agility \(was: Please review draft-iab-crypto-alg-agility-00.txt\)](#), *Mouse*
- [Re: \[saag\] Algorithm agility \(was: Please review draft-iab-crypto-alg-agility-00.txt\)](#), *Matthew Chalmers*
- [Re: \[saag\] Algorithm agility \(was: Please review draft-iab-crypto-alg-agility-00.txt\)](#), *Salz, Rich*
- [Re: \[saag\] Algorithm agility \(was: Please review draft-iab-crypto-alg-agility-00.txt\)](#), *Viktor Dukhovni*
- [Re: \[saag\] Algorithm agility \(was: Please review draft-iab-crypto-alg-agility-00.txt\)](#), *S Moonesamy*
- [Re: \[saag\] Please review draft-iab-crypto-alg-agility-00.txt](#), *Viktor Dukhovni*
- [Re: \[saag\] Algorithm agility](#), *Salz, Rich*
- [Re: \[saag\] Please review draft-iab-crypto-alg-agility-00.txt](#), *ianG*
- [Re: \[saag\] Algorithm agility](#), *ianG*
- [Re: \[saag\] Please review draft-iab-crypto-alg-agility-00.txt](#), *Mouse*
- [Re: \[saag\] Algorithm agility \(was: Please review draft-iab-crypto-alg-agility-00.txt\)](#), *Salz, Rich*
- [Re: \[saag\] Algorithm agility \(was: Please review draft-iab-crypto-alg-agility-00.txt\)](#), *Mouse*
- [Re: \[saag\] Algorithm agility](#), *Salz, Rich*
- [Re: \[saag\] Please review draft-iab-crypto-alg-agility-00.txt](#), *ianG*
- [Re: \[saag\] Algorithm agility](#), *ianG*
- [Re: \[saag\] Algorithm agility \(was: Please review draft-iab-crypto-alg-agility-00.txt\)](#), *Salz, Rich*
- [Re: \[saag\] Please review draft-iab-crypto-alg-agility-00.txt](#), *Eliot Lear*
- [\[saag\] Algorithm agility \(was: Please review draft-iab-crypto-alg-agility-00.txt\)](#), *S Moonesamy*

IETF PERPASS

- "The perpass list is for IETF discussion of pervasive monitoring.

"IETF specifications need to be designed to protect against pervasive monitoring where possible. This list is intended for technical discussions attempting to meet that goal.

"Discussion is limited to specific technical proposals for improvements in IETF protocols, their implementation or deployment and to IETF process changes aiming to increase the likelihood that development, implementation and deployment of IETF protocols results in better mitigation for pervasive monitoring."

<https://www.ietf.org/mailman/listinfo/perpass>

A Few Recent Days Of The Perpass List

- **May 07 2014**
 - [Re: \[perpass\] Delivering TLS Best Practices](#), *Carl S. Gutekunst*
 - [Re: \[perpass\] Delivering TLS Best Practices](#), *Trevor Freeman*
 - [Re: \[perpass\] Delivering TLS Best Practices](#), *Paul Ferguson*
 - [\[perpass\] Delivering TLS Best Practices](#), *Trevor Freeman*
- **May 05 2014**
 - [\[perpass\] Summary of IETF Activities on Privacy](#), *Hannes Tschofenig*
- **May 04 2014**
 - [\[perpass\] Traffic peeking - draft-moonesamy-traffic-peeking-02](#), *S Moonesamy*
- **May 01 2014**
 - [Re: \[perpass\] FW: \[IP\] Details of how Turkey is intercepting Google Public DNS](#), *Phillip Hallam-Baker*
- **Apr 30 2014**
 - [Re: \[perpass\] Is DNSDEC a viable technology for perpass?](#), *Dan York*
 - [Re: \[perpass\] Is DNSDEC a viable technology for perpass?](#), *Nicholas Weaver*
 - [Re: \[perpass\] Is DNSDEC a viable technology for perpass?](#), *Dan York*
 - [Re: \[perpass\] Is DNSDEC a viable technology for perpass?](#), *manning bill*

IETF TLS

- The primary purpose of the working group is to develop (D)TLS v1.3. Some of the main design goals are [...]
 - o Update record payload protection cryptographic mechanisms and algorithms to address known weaknesses in the CBC block cipher modes and to replace RC4.
[...]
 - o The WG will consider the privacy implications of TLS1.3 and where possible (balancing with other requirements) will aim to make TLS1.3 more privacy-friendly, e.g. via more consistent application traffic padding, more considered use of long term identifying values, etc.
[...]

<http://datatracker.ietf.org/wg/tls/charter/>

A Recent Day From The TLS List

• May 06 2014

- [Re: \[TLS\] The risk of misconfiguration](#), *Viktor Dukhovni*
 - [Re: \[TLS\] The risk of misconfiguration](#), *Andrei Popov*
 - [Re: \[TLS\] The risk of misconfiguration](#), *Nico Williams*
 - [Re: \[TLS\] The risk of misconfiguration](#), *Viktor Dukhovni*
 - [Re: \[TLS\] The risk of misconfiguration](#), *James Cloos*
 - [Re: \[TLS\] The risk of misconfiguration](#), *Alyssa Rowan*
 - [\[TLS\] The risk of misconfiguration](#), *Watson Ladd*
 - [Re: \[TLS\] Confirming Consensus on supporting only AEAD ciphers](#), *Manuel Pégourié-Gonnard*
 - [Re: \[TLS\] \(offline note\) Re: Confirming Consensus on supporting only AEAD ciphers](#), *Michael StJohns*
 - [Re: \[TLS\] \(offline note\) Re: Confirming Consensus on supporting only AEAD ciphers](#), *Manuel Pégourié-Gonnard*
 - [Re: \[TLS\] Confirming Consensus on removing RSA key Transport from TLS 1.3](#), *Nikos Mavrogiannopoulos*
 - [Re: \[TLS\] \(offline note\) Re: Confirming Consensus on supporting only AEAD ciphers](#), *Michael StJohns*
 - [Re: \[TLS\] \(offline note\) Re: Confirming Consensus on supporting only AEAD ciphers](#), *Michael StJohns*
 - [Re: \[TLS\] \(offline note\) Re: Confirming Consensus on supporting only AEAD ciphers](#), *Martin Rex*
 - [Re: \[TLS\] Confirming Consensus on supporting only AEAD ciphers](#), *Michael StJohns*
 - [Re: \[TLS\] Confirming Consensus on removing RSA key Transport from TLS 1.3](#), *Watson Ladd*
 - [Re: \[TLS\] Confirming Consensus on removing RSA key Transport from TLS 1.3](#), *Viktor Dukhovni*
 - [Re: \[TLS\] \(offline note\) Re: Confirming Consensus on supporting only AEAD ciphers](#), *Joseph Salowey (jsalowey)*
 - [\[TLS\] \(offline note\) Re: Confirming Consensus on supporting only AEAD ciphers](#), *Rene Struik*
 - [Re: \[TLS\] Triple Handshake Fix.](#), *Bodo Moeller*
 - [Re: \[TLS\] Confirming Consensus on supporting only AEAD ciphers](#), *Fedor Brunner*
 - [Re: \[TLS\] Confirming Consensus on removing RSA key Transport from TLS 1.3](#), *Nikos Mavrogiannopoulos*
 - [Re: \[TLS\] Confirming Consensus on supporting only AEAD ciphers](#), *Joseph Salowey (jsalowey)*
-

IETF UTA (Using TLS in Applications)

- "This WG has the following tasks:
 - Update the definitions for using TLS over a set of representative application protocols. This includes communication with proxies, between servers, and between peers, where appropriate, in addition to client/server communication.
 - Specify a set of best practices for TLS clients and servers, including but not limited to recommended versions of TLS, using forward secrecy, and one or more ciphersuites and extensions that are mandatory to implement. [...]"

<http://datatracker.ietf.org/wg/uta/charter/>

Relatively light mailing list traffic recently

IRTF Crypto Forum Research Group (CFRG)

- "The Crypto Forum Research Group (CFRG) is a general forum for discussing and reviewing uses of cryptographic mechanisms, both for network security in general and for the IETF in particular.
- The CFRG serves as a bridge between theory and practice, bringing new cryptographic techniques to the Internet community and promoting an understanding of the use and applicability of these mechanisms via Informational RFCs (in the tradition of, e.g., RFC 1321 (MD5) and RFC 2104 (HMAC) [...])"

<https://irtf.org/cfrg>

A Couple of Recent Days On The CFRG List

- **May 07 2014**

- [Re: \[Cfrg\] DDH, GapDH and Static DH \[was RE: On Strong DH\]](#), *Watson Ladd*
- [\[Cfrg\] Reduction and pseudoreduction of static DHP to TLS-DH](#), *Dan Brown*
- [Re: \[Cfrg\] GapDH groups: a long-term research question](#), *Dan Brown*
- [Re: \[Cfrg\] Recording and chat transcript CFRG Spring 2014 Interim Meeting on ECC](#), *James Cloos*
- [\[Cfrg\] DDH, GapDH and Static DH \[was RE: On Strong DH\]](#), *Dan Brown*
- [Re: \[Cfrg\] Some observations after the ECC interim meeting](#), *Patrick Longa Pierola*

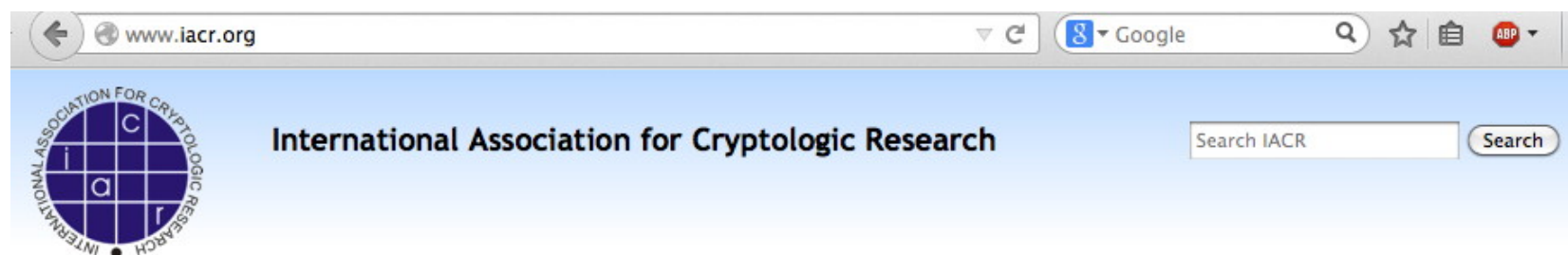
- **May 06 2014**

- [Re: \[Cfrg\] Task looming over the CFRG](#), *Watson Ladd*
- [\[Cfrg\] On Strong DH](#), *Watson Ladd*
- [Re: \[Cfrg\] Task looming over the CFRG](#), *Paul Lambert*
- [\[Cfrg\] Recording and chat transcript CFRG Spring 2014 Interim Meeting on ECC](#), *David McGrew*
- [Re: \[Cfrg\] Task looming over the CFRG](#), *Johannes Merkle*
- [Re: \[Cfrg\] Task looming over the CFRG](#), *Igoe, Kevin M.*
- [Re: \[Cfrg\] Task looming over the CFRG](#), *Johannes Merkle*

Academic Cryptology

- Beyond the IETF, we need a strong and vibrant academic cryptography community, doing innovative research and vigorously challenging what may be proposed by governments or their peers.
- If you are so inclined and have a suitable background, I'd encourage you to consider engaging with professional academic cryptographic efforts such as the IACR, see the following slide.

IACR



The International Association for Cryptologic Research (IACR) is a non-profit scientific organization whose purpose is to further research in cryptology and related fields. Cryptology is the science and practice of designing computation and communication systems which are secure in the presence of adversaries.

Meetings

Conferences — The IACR organizes three main international conferences in cryptology each year.

- [Eurocrypt 2014](#), May 11-May 15, 2014, Copenhagen, Denmark.
- [Crypto 2014](#), August 17-August 21, 2014, Santa Barbara, USA.
- [Asiacrypt 2014](#), December 7-December 11, 2014, Kaohsiung, Taiwan.

Workshops — The IACR organizes four annual specialist workshops in various areas of cryptology.

- [Theory of Cryptography Conference](#), February 24-February 26, 2014, La Jolla, CA, USA.
- [Fast Software Encryption 2014](#), March 3-March 5, 2014, London, United Kingdom.
- [17th International Conference on Practice and Theory of Public-Key](#) March 26-March 28 2014 Buenos Aires

News Updates from IACR



[Volunteers wanted for IACR online services](#)
(2014-05-07)

Jobs

[Ph. D student, CEA SAS \(Secure Architectures & Systems\) Lab, France](#) (2014-05-05)



[On the Complexity of Finding Low-Level Solutions, by Bjoern Grohmann](#) (2014-05-02)



[Weaknesses of Password Authentication Scheme Based on Geometric Hashing, by Martin Stanek](#) (2014-04-26)



[ASK 2014: The Fourth Asian Workshop on Symmetric Key Cryptography](#) (2014-04-25)

IACR Calendar of Cryptology Events

<https://www.iacr.org/events/>

Google

Calendar of Events in Cryptology

Events on this page are sorted **by date**. This includes workshops and conferences, but not special issues of journals. The latter are only shown in the view *by submission deadline*. Switch to view events [sorted by submission deadline](#) or [displayed on a map](#).

The IACR calendar lists events (conferences, workshops, ...) that may be of interest to IACR members or deal with research in cryptology. Information here is provided as a service to the IACR membership. The accuracy of the listing is the responsibility of the submitters. No endorsement by IACR of the conference nor any other information contained in the listing can be assumed.

If you want to have an event listed here, please [fill out this form](#). (The current condition for being listed is that the description of an event must contain the substring "crypt" anywhere.)

For incorporation into your personal calendar, the events sorted by date are also available in [iCal format](#).

See also the list of IACR [conferences](#), IACR [workshops](#), their [history](#), and [events in cooperation](#) with the IACR. Past events are [archived here](#).

2014

- [Information Security Practice & Experience Conference \(ISPEC '14\)](#), Fuzhou, China, *May 5-May 8*
 - [Workshop on Theory and Practice of Secure Multiparty Computation \(MPC14\)](#), Aarhus, Denmark, *May 5-May 9*
 - [Eurocrypt 2014 \(Eurocrypt\)](#), Copenhagen, Denmark, *May 11-May 15*
 - [IEEE Symposium on Security and Privacy \(SSP\)](#), San Jose, United States, *May 18-May 21*
 - [Central European Conference on Cryptology \(CECC14\)](#), Budapest, Hungary, *May 21-May 23*
 - [Africacrypt 2014](#), Marrakech, Morocco, *May 28-May 30*
-
- [Summer school on Design and security of crypto algorithms and devices](#), Sibenik, Croatia, *June 1-June 6*
 - [29th IFIP TC11 SEC 2014 Int Conf ICT Systems Security & Privacy Protection \(SEC 2014\)](#), Marrakech, Morocco, *June 2-June 4*
 - [2nd ACM ASIA Public-Key Cryptography Workshop \(ASIAPKC 2014\)](#), Kyoto, Japan, *June 3-June 3*
 - [The Second International Workshop on Security in Cloud Computing \(AsiaCCS-SCC\)](#), Kyoto, Japan, *June 3-June 3*
 - [9th ACM Symposium on Information, Computer and Communications Security \(ASIACCS 2014\)](#), Kyoto, Japan, *June 4-June 6*
 - [3rd Workshop on Current Trends in Cryptology \(CTCrypt 2014\)](#), Moscow, Russia, *June 5-June 6*
 - [Yet Another Conference on Cryptography \(YACC 2014\)](#), Ile de Porquerolles, France, *June 9-June 13*
 - [12th International Conference on Applied Cryptography and Network Security \(ACNS'14\)](#), Lausanne, Switzerland, *June 10-June 13*
 - [SECITC'14: The 7th International Conference on Security for IT&C \(SECITC'14\)](#), Bucharest, Romania, *June 12-June 12*

VII. Everything Else

"No snowflake in an avalanche ever feels responsible."

Voltaire

I Wanted To Leave Some Time For Discussion

- I can never guess what the community would like to talk about, or what I should have covered, so I wanted to leave some time for less structured discussion.
- What else would you like to talk about?
 - Traffic analysis?
 - Web authentication-related risks?
 - PGP?
 - IPsec (VPN or otherwise)?
 - Opportunistic encryption?

Thanks for the Chance to Talk Today

- Are there any questions?
- Copies of these slides can be obtained online at

<http://pages.uoregon.edu/joe/crypto-bcp/>