SSL / TLS

How broken is it?

Interesting times

From: [redacted]
To: "hanno@hboeck.de" <hanno@hboeck.de>
Subject: Re: Re: Einverstanden
Date: Thu, 12 Sep 2013 15:54:42 +0200
X-Mailer: iPad Mail (10B329)

```
--- Start of PGP/Inline encrypted data ---
```

golem.de
IT-NEWS FÜR PROFIS

HOME TICKER Suchen

TOP-THEMEN: NSA Ifa 2013 Xbox One Playstation 4 Haswell Windows 8 mehr

VERSCHLÜSSELUNG

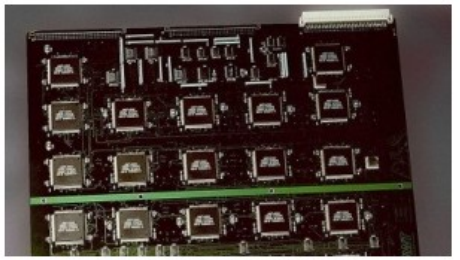
Was noch sicher ist

Die NSA soll auch in der Lage sein, verschlüsselte Verbindungen zu knacken. Ein Überblick über kryptographische Algorithmen und deren mögliche Probleme.

612 211 305

Recommer Twittern +1

Google-Anzeigen



Spezialchip zum Angriff auf das Verschlüsselungsverfahren DES (Bild: Electronic

Schneier on Security

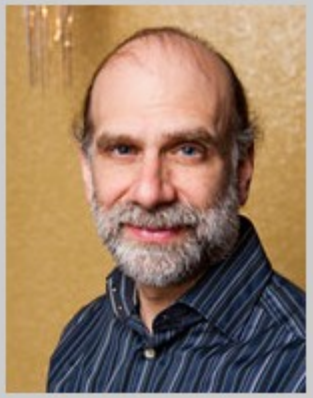
A blog covering security and security technology.

« The Effect of Money on Trust | Main | The NSA's Cryptographic Capabilities »

September 5, 2013

The NSA Is Breaking Most Encryption on the Internet
The new Snowden revelations are explosive. Basically, the NSA is able to decrypt most of the Internet. They're doing it primarily by cheating, not by mathematics.

It's joint reporting between the Guardian, the New York Times, and ProPublica.



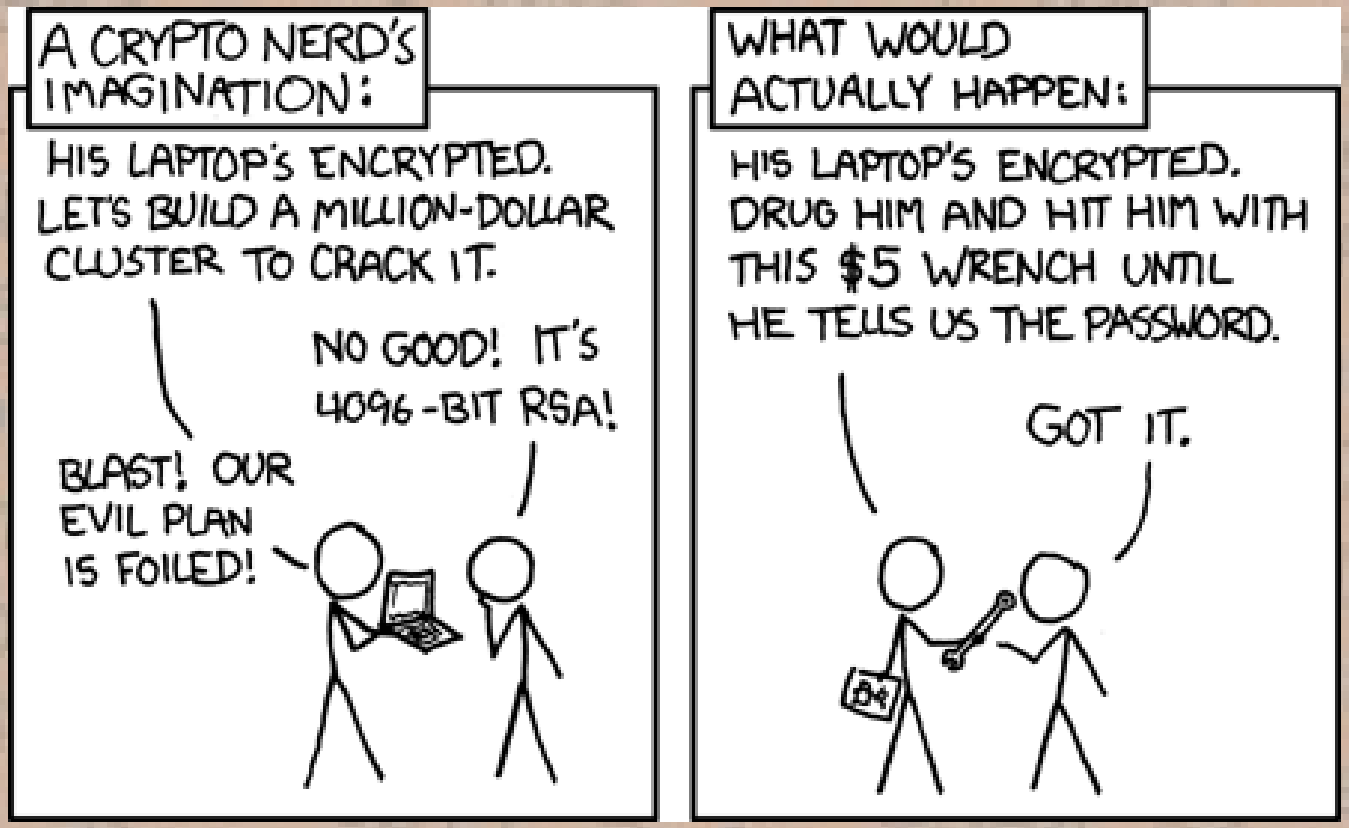
Introduction

- I'm Hanno Böck, freelance Journalist, administrator at schokokeks.org (webhosting)
- Long-time interest in crypto, diploma thesis on RSA-PSS
- This talk was planned before the NSA started the biggest crypto campaign of all time

Interactive test

- If you have a device with a browser in front of you, point it to:
  https://fancyssl.hboeck.de
- If you have an HTTPS server, check it:
  https://www.ssllabs.com/ssltest/

What's wrong here?



- The “wrench“ attack is active, detectable and doesn't work for secret mass surveillance.

Okay, let's talk about SSL / TLS

- Invented by Netscape in 1995 as SSLv2
- Completely broken
- SSLv3 in 1996
- Became renamed to TLS and IETF standard
- TLS 1.0 1999 (the one you're likely using today)
- TLS 1.1 2006
- TLS 1.2 2008 (support: near zero – it's just too new)

X.509 certificates

- If you use TLS, you usually also use X.509
- Certificate authorities are a mess
- But nobody has a good idea how to fix it
- EFF tried (sovereign keys – watch their talk)
- But CA fails are not our topic today
- CA attacks are active, detectable and not suitable for mass surveillance

TLS connection – what happens

- Client asks Server for Connection
- Server shows cert with public key (usually RSA), client checks cert
- Client/server use the public key in the cert to create a secret key (two ways to do it)
- They exchange encrypted and authenticated data with secret key (usually AES and some broken way of authenticating it, sometimes also RC4)

Attacks

- BEAST attack (2011)
- CRIME attack (2012)
- Lucky Thirteen attack (2013)
- RC4 attack without a cool name (2013)
- BREACH attack (2013)
- Many issues were documented before
- All protocol bugs, not software bugs

But let's look at certs first

- Cert is signed by certificate authorities cert (sometimes with intermediate cert)
- Every cert contains a public key, usually RSA
- You can also use DSA or ECDSA , but nobody does
- Key size can be almost anything
- But CAs today usually don't issue keys below 2048 bits (I'm sure there are ones that still do)

RSA key size

- How difficult is factoring?
- 512 bit can be broken by everyone with a current computer, a day of time and ability to compile a weird tool
- EFF found 512 bit “Extended Validation“ key in 2010
- 768 bit can be broken (2009)
- 1024 bit can be broken by someone rich (Eran Tromer estimates 1 million $)
- All major sites moved away from 1024 bit

DSA / ElGamal / DiffieHellman

- Discrete Logarithm Problem (for a^x=b mod p find x)
- Key size / security roughly the same as RSA
- DSA was fixed to 1024 bit for quite some time
- DSA is messy anyway, just don't use it. It completely breaks with bad random numbers.
- Not a big issue in SSL, but many GPG keys
- DiffieHellman with 1024 bits is used a lot and almost as insecure as RSA with 1024 bits (we'll get back to that)

Elliptic curves

- Discrete logarithm also works in a structure called elliptic curves
- Shorter key sizes should be secure
- ECDSA (public key signatures) and ECDH supported by SSL
- ECDSA rarely used, ECDH a lot
- Uses usually NIST curves with somewhat unclear origin

These NIST curves

- Created by Jerry Solinas (NSA)
- They use output of a SHA-1 function. Idea: SHA-1 is not invertible, so we can't create curve with backdoor
- Uses input values like 3045ae6f c8422f64 ed579528 d38120ea e12196d5. Why?
- I've asked them, they promised me to get back on me. If they answer I'll tell you.

So we have a public key

- Two ways to generate session key
- Variant a: generate key, send it encrypted (bad)
- Variant b: use server key only for signing, do key exchange (Perfect Forward Secrecy)
- Diffie Hellman or Elliptic Curve Diffie Hellman
- DH: Apache uses 1024 bit and this can't be changed without patching
- Big mess: No proper way to negotiate size

Now we have a shared key

- We need confidentiality and authenticity
- AES encryption (or 3DES, Cammelia, all probably secure) in CBC mode with Padding and HMAC
- Or stream cipher RC4

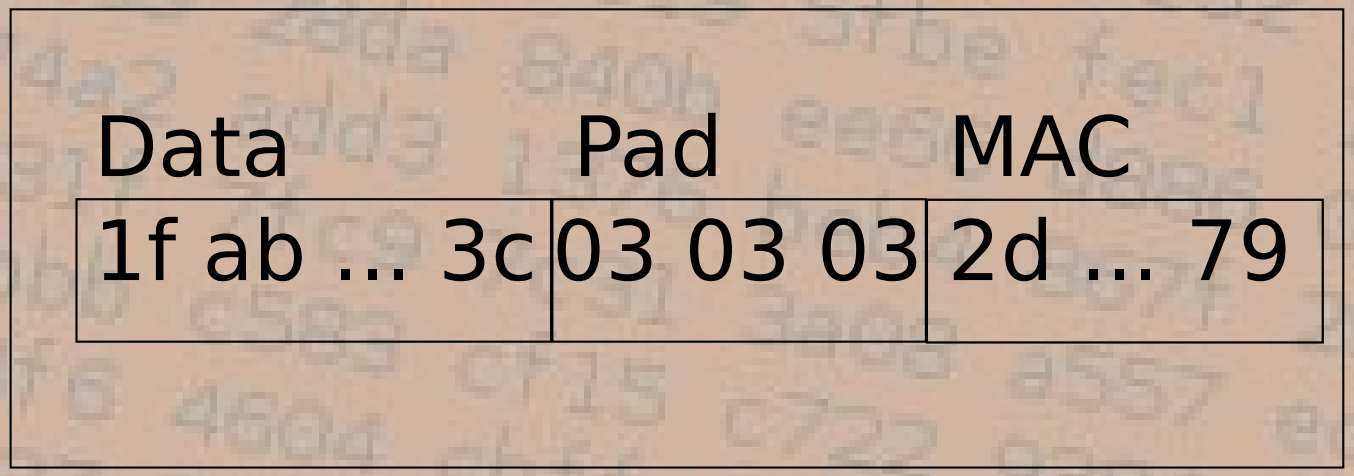
Combining AES-CBC and HMAC

- Three ways
- MAC-then-Encrypt (SSL, bad)
- MAC-and-Encrypt (SSH, very bad)
- Encrypt-then-MAC (good)

CBC/HMAC in TLS

- Looks like this:

AES-CBC



- Padding oracle – by bit-flipping and separating MAC and Padding errors, you can decrypt
- Fix: Always the same error. Good?

CBC/HMAC in TLS

- Padding oracle comes back as timing attack (Lucky Thirteen)
- Also other attacks due to bad IV (BEAST), which got fixed in TLS 1.1 (that almost nobody uses, because its so new - 2006)
- So a lot of people (e. g. Google) said “with all that mess with CBC/HMAC, let's use something different”
- Like RC4, because there's nothing else

RC4

- RC4 is great – it's fast, it's simple
- Developed 1987 by Ron Rivest, not public
- Leaked in 1994 in a newsgroup
- Since then used in successful standards like WEP and TLS
- It's only a few lines of code – you can just copy the code from Wikipedia

Unfortunately...

- RC4 is not very secure
- It has biases at certain bits in the key stream
- Fix: Throw away 256 stream bits
- But people found more biases in later stream bits
- So throw away more stream bits
- In 2013 Dan Bernstein shows attack on RC4/TLS
- Needs Gigabytes of traffic, not very practical
- But many think RC4 is a likely target for NSA breakthroughs

So we have

- Lots of problems in CBC/HMAC (that could be fixed if you would use Encrypt-then-MAC)
- Some hacky workarounds
- People switch to RC4, because it's the only thing left that's available in widely used TLS version 1.0
- Dan Bernstein shows RC4 is also vulnerable
- So we can choose between AES-CBC (bad) and RC4 (bad)

Hash

- MD5 is very broken, SHA-1 is broken
- SHA-2 looks good, it's created by the very skilled people from the NSA
- If this makes you nervous, SHA-3 will come – but really, most people think SHA-2 is fine
- If collision-broken hash functions (MD5, SHA-1) matter depends on the use case
- For sigs they matter, for HMAC they don't
- In doubt: use unbroken hash functions

What about TLS 1.1/1.2

- TLS 1.1 fixes BEAST/IV problem, but not padding oracle / Lucky Thirteen
- TLS 1.2 has AES with authenticated encryption: Galois/Counter Mode (GCM)
- That fixes it – really. Also gets rid of SHA-1 (probably no issue with SHA-1 here)
- But TLS 1.2 is so new (2008)
- Browsers slowly adopt TLS 1.2 and AES-GCM
- Webservers also slowly (but some people use Debian stable or Redhat Enterprise)

CRIME / BREACH

- CRIME attack on TLS compression
- Not a big problem: Rarely anyone uses TLS compression, just disable it
- BREACH attack focuses on HTTP compression
- That's a problem
- No simple solution and not clear where solution should be
- Involves web application security like CSRF-Tokens (web application security is a nightmare)

Downgrade attacks

- If connection through TLS 1.0/1.1/1.2 doesn't work, major browser will retry with SSLv3
- This happens quite often, e. g. when your Internet connection is bad (UMTS / WiFi)
- All improvements brought to you by TLS 1.1/1.2 are void – also non-security ones (like Server Name Indication)
- Reason: Broken old hardware, e. g. proxies
- Compatibility comes before security

Steps to a secure HTTPS Server

- Get a cert, 2048/4096 bit RSA, SHA256-signed
- Support TLS 1.2, prefer AES-GCM with ECDH (needs apache 2.4) or DH (patch apache) if you're worried about NIST curves
- Drop SSLv3 if you can (please!)
- Disable TLS compression
- Check Qualys SSL Labs test

Browser

- Install latest version, get TLS 1.2 support (Chrome already on some archs, Firefox soon)
- Install HTTPS Everywhere (maybe vulnerable HTTPS is better than no HTTPS)
- Disable SSLv3 if you can

Further interesting SSL features

- SNI – Several certs on one IP
- OCSP stapling – trying to fix the revocation mess (certificate revocation is seriously broken – Chrome “fixed” it recently by disabling)
- HSTS – enforce SSL, avoid stripping

Future

- Browsers will adopt TLS 1.2 with AES-GCM soon
- ChaCha20 stream cipher and Poly1305 mode for TLS
- What to do about elliptic Curves? There are alternatives, many people like Curve25519
- Or stick with classic algos. RSA could need some revamp by using PKCS #1 2.1 (but it's so new – from 2002) [read my thesis if you're interested]

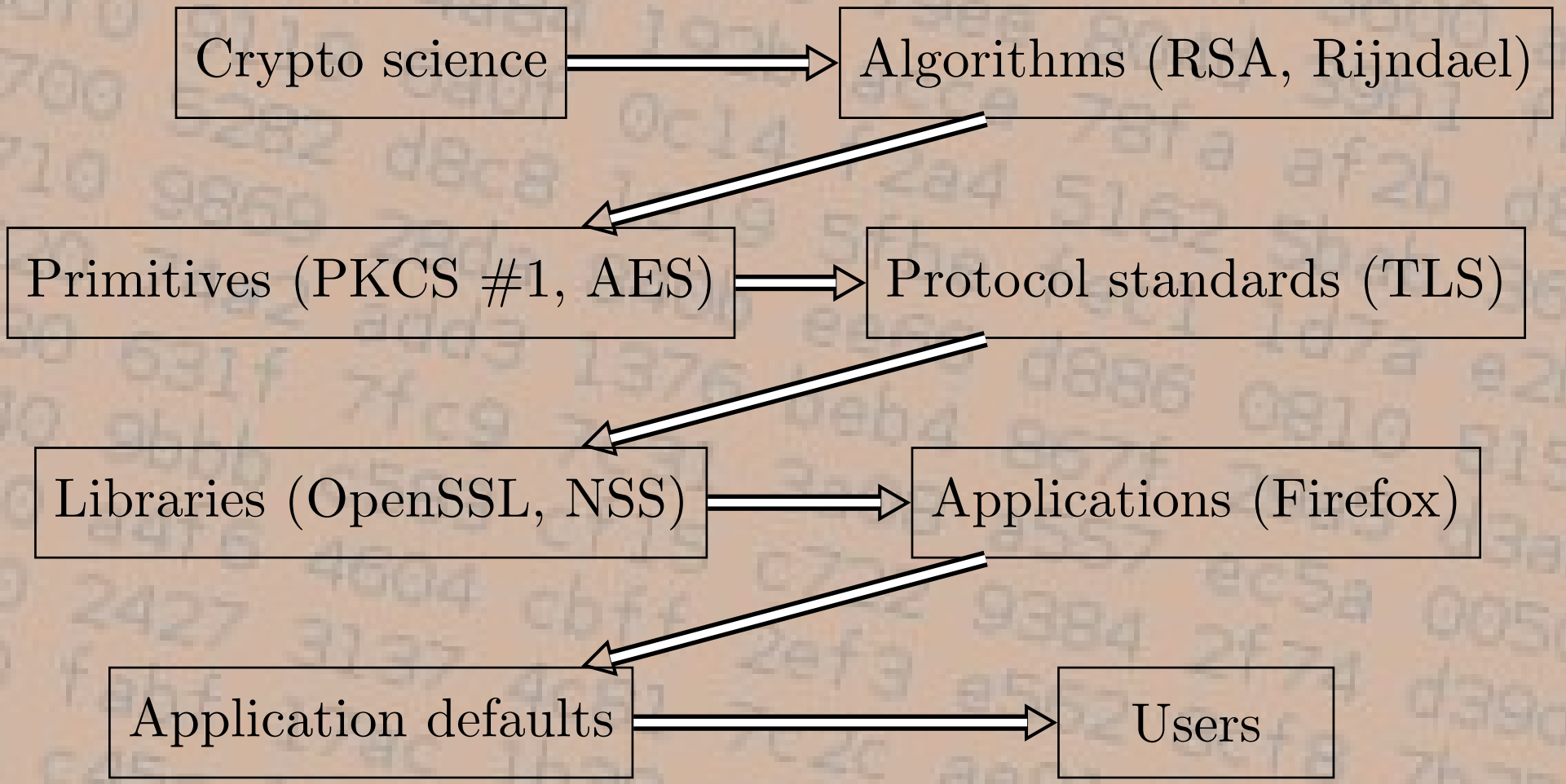
NIST

- Many people trusted NIST a lot
- AES and SHA-3 competition received lots of praise
- AES: Some people still think the wrong algorithm (Rijndael) won and Twofish / Serpent are the real winners
- But almost nobody doubts security of AES
- Except if you have much more capabilities than the NSA (4 Terawatt)

Quantum computer

- Shor's algorithm Can break RSA, DSA, ElGamal, DiffieHellman, elliptic Curves
- Can even break elliptic Curves easier
- IBM built 7 bit quantum computer – not a problem if your key size is > 7 bit
- D-Wave quantum computer: no threat
- Many people think: large QC can't be built
- Some people discuss Post-Quantum-Crypto. It's possible, but nothing ready yet.

The long way of crypto



- 8 steps - And then you still have backwards compatibility issues – almost forever

More Info

- I write regularly for Golem.de and others (usually German) http://hboeck.de/
- Read blogs of Matthew Green http://blog.cryptographyengineering.com/
- Bruce Schneier https://www.schneier.com/
- Adam Langley https://www.imperialviolet.org/
- Do online university course by Dan Boneh http://blog.cryptographyengineering.com/

Watch some CCC congress talks

- 29C3 FactHacks (Bernstein, Lange, Heninger)
- 28C3 Sovereign Keys (EFF tries to fix CAs)
- 27C3 SSLiverse (EFF SSL Observatory)
- 25C3 MD5 considered harmful today
- 20C3 1024-bit RSA ist unsicher
- http://media.ccc.de

Or some other talks

- Matthew Green on TLS https://www.youtube.com/watch?v=uP6np_oKVCk
- Adam Langley on HTTPS and web security https://www.youtube.com/watch?v=LBbCec4Bp10

Encrypt!

- Questions? My PGP key is BBB51E42