Hard and Not-necessarily-hard Problems in Cryptography

Peter Gutmann

University of Auckland

Hard Crypto Problems

Some crypto problems have no known general solution

- This may be the first talk ever to admit that there exist security problems for which adding more cryptography isn't the answer

Why are you telling us about them if there's no solution?

- To warn you about them so you can try alternatives
- To let you know that if you're finding it hard to deal with them then it's not your fault

Example: Secure Bootstrapping of Comms

How do you securely initiate communications with an entity that you've never communicated with before?

- The killer problem
- The elephant in the room
- The mixed metaphor

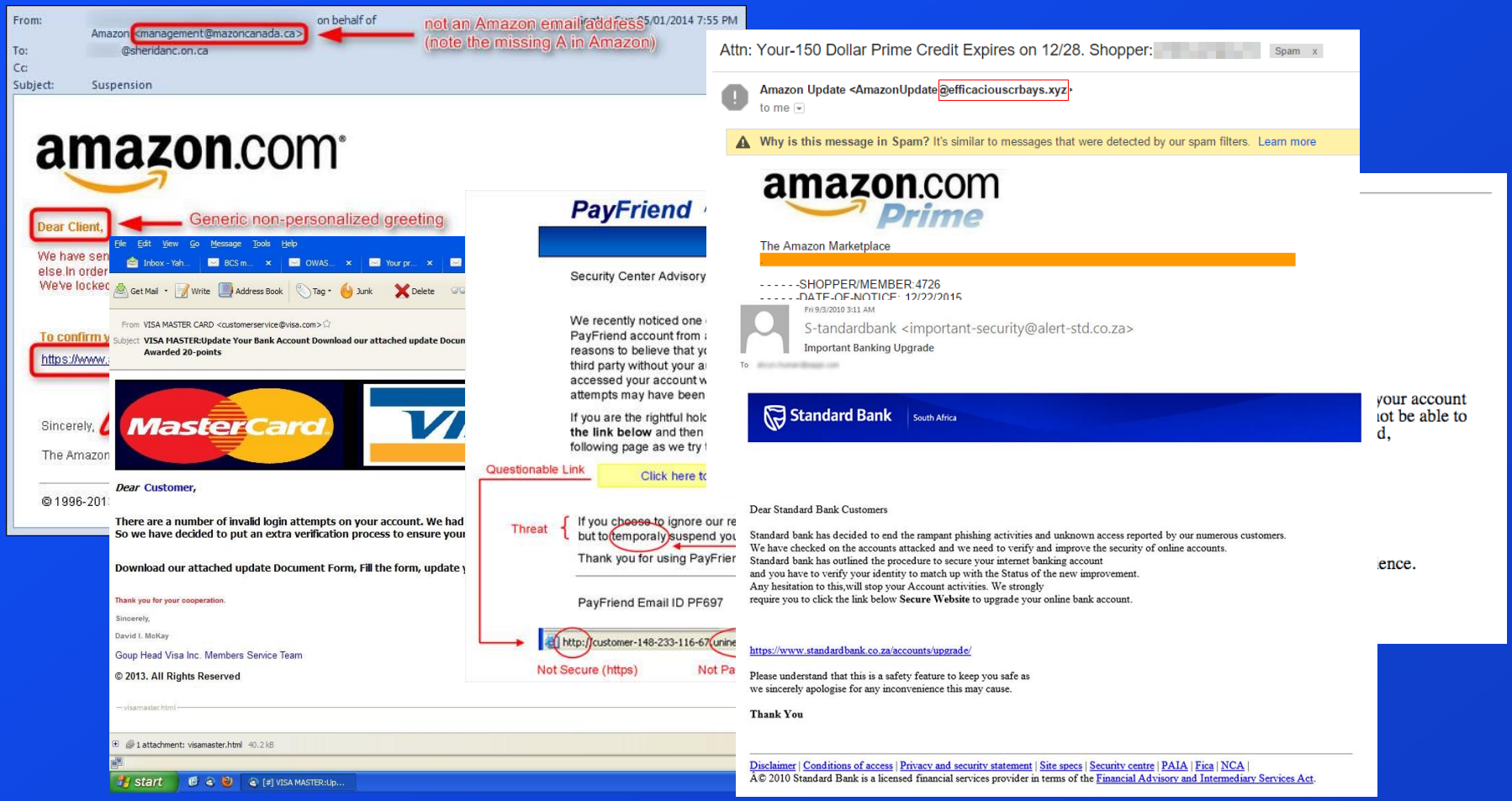
… of Internet security protocols

We simply have no way of doing this

Example: Secure Bootstrapping of Comms

This makes a lot of otherwise good crypto a lot less useful

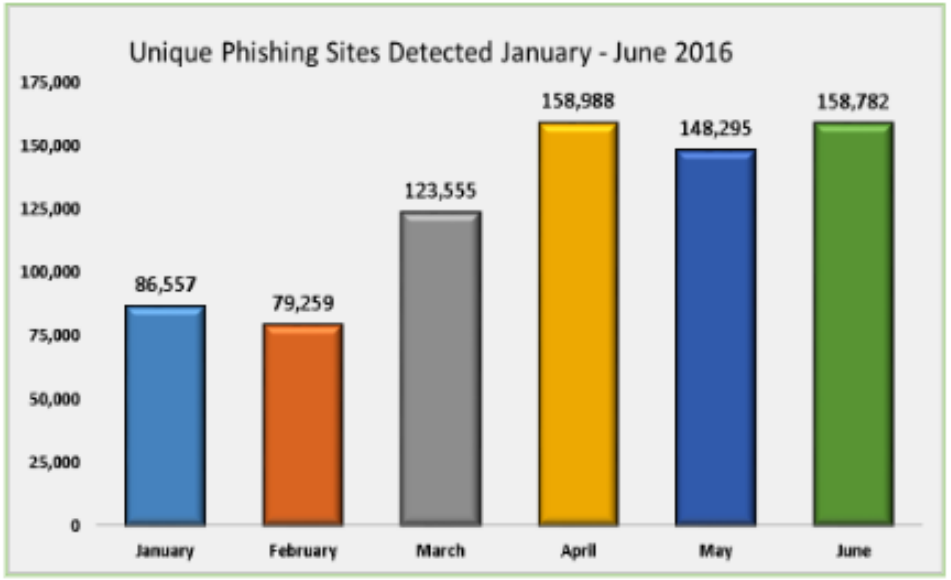
- Something that the bad guys are very good at exploiting



Example: Secure Bootstrapping of Comms

Until we solve this problem…

Phishing Attacks In Q2 2016 Shatter All Records to Reach All-Time High

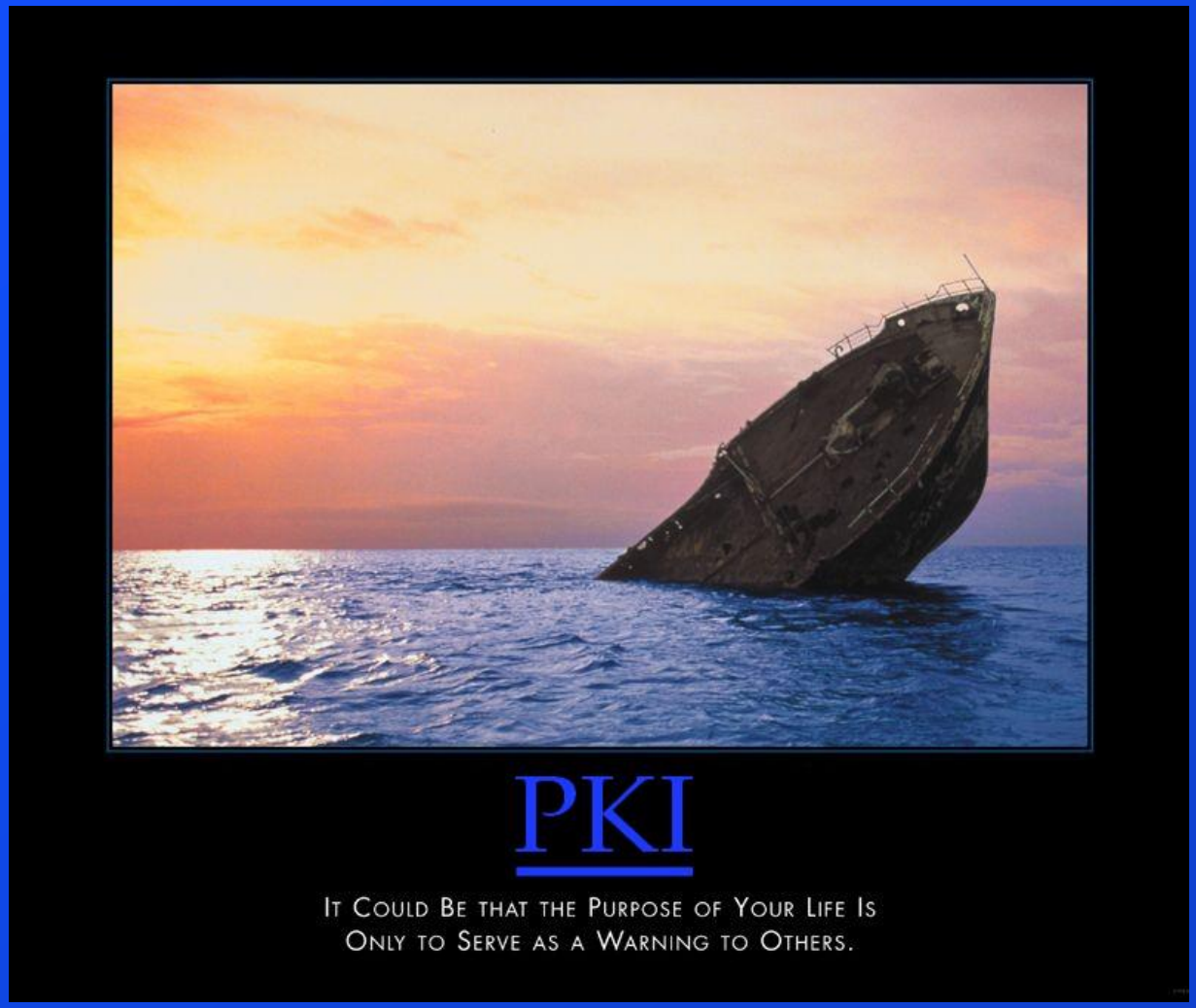


Source: APWG

… we may as well just be using RSA-512 with RC4/40

Example: Secure Bootstrapping of Comms

What about PKI?



Source: Despair

Example: Secure Bootstrapping of Comms

What about SSH?

Do SSH Fingerprints Increase Security?

Peter Gutmann
Department of Computer Science
University of Auckland

Abstract

No.

Example: Secure Bootstrapping of Comms

What about *insert-pet-mechanism-here*?

- No, that won't solve it either

Wicked Problems

So why is this so hard?

This (and many other issues) are examples of wicked problems

- Concept from the field of social planning

Source: Wkikpedia

- Proposed in the 1970s as a means of modelling the process for dealing with social, environmental, and political issues

Wicked Problems (ctd)

Amongst a wicked problem's weaponry are such diverse elements as…

Lack of any definitive formulation of the problem

Lack of a stopping rule

- One of the core requirements for dealing with a wicked problem becomes not deciding too early which solution you're going to apply

Solutions that are rateable only as "better" or "worse" and not true or false

- Particularly bad for security geeks
- There are only two options, absolutely secure or absolutely insecure

Wicked Problems (ctd)

No clear idea of a which steps or operations are necessary to get to the desired goal

A variety of ideological and political differences among stakeholders

> The difference between them is simple: [algorithm design] is ‘hard science’. [Security] is ‘people wanking around with their opinions’
>
> — Linus Torvalds, 2007

Wicked Problems (ctd)

A wicked problem presents...

- No clear idea of what the problem is
- No clear idea of how to get to a solution
- No easy way to tell whether you've reached your goal or not
- All of the participants are pulling in different directions

Example: High-performance Sports Cars

Fit a more powerful engine

- Adds extra weight
  - Slows it down again
- Adds size
  - If taken to extremes leaves little room for anything else, including a driver

Example: High-performance Sports Cars (ctd)

Reduce weight by fitting a lighter engine

- Have to make the car lighter to compensate for the less powerful engine

If taken to extremes leads to a car that's little more than an exoskeleton with a motorcycle engine

- Has limited appeal to the general market

Example: High-performance Sports Cars (ctd)

Use exotic materials like carbon fibre to decrease weight

- Raises the price and again discourages buyers

Example: High-performance Sports Cars (ctd)

Strip out as many weight-adding features as possible

- Trade-off between performance and comfort
- Some jurisdictions have safety regulations that affect what you can and can’t do
- Tradeoff between being able to sell the car in a particular market and making performance-reducing changes

Example: Audio Woo-Woo

High-end audio is like high-performance sports car design, only much sillier

Example: Audio Woo-Woo (ctd)

OK, that's not really true…

Only stopping rule is "how much money does ~~the sucker~~ the customer have?"

- Any solution *you* sell is better than what everyone else has (by definition)

Source: Higher-Fi

Limits are defined by how much woo-woo you can come up with

Example: Audio Woo-Woo (ctd)

WALKER AUDIO

Home | All Systems | Analog Systems | About Us | Shop

Home » Shop » System-Enhancing Products » Black Diamond Room Treatment Crystals

Black Diamond Room Treatment Crystals

$350.00 – $590.00

NEW PRICING!! We've been discovering the benefits of our Black Diamond crystal technology for several years now and learning more about their AMAZING properties. Based on our extensive and meticulous research, we now offer you two complementary, but different, products – Black Diamond Component Crystals and **Black Diamond Room Treatment Crystals**.

When properly applied, the Black Diamond Room Treatment Crystals actually manipulate the way sound behaves in your space, with phenomenal results. You can expect dramatically improved focus and a greater sense of realism and dimension in the soundstage as the walls and ceiling seem to open up. No longer will your sound be "electronic", but it will have a presence and excitement of a live performance.

Anything goes…

Example: Audio Woo-Woo (ctd)

Of course we'd never go for this in the security field…

Example: Audio Woo-Woo (ctd)

- $30,000 iPod dock demo'd at CES 2012
- Behringer iNuke Boom car-sized dock

Example: Audio Woo-Woo

Example: Wavac SH-833 (Amps slide) using 833 tubes

- 1938 vintage RCA radio transmitting tube

DESCRIPTION

GL-833-A is a three-electrode transmitting tube of the high-mu type for use as a radio-frequency amplifier, oscillator, and Class B modulator. Be-

As a result of its construction, the 833-A provides exceptional efficiency at high frequencies. It can be operated in Class C telegraph service with

Source: eBay

Design use: Class B or C RF modulator/power amp

- Wavac use: Class A audio amp

The audio equivalent of WEP

- Why pound a screw with a wrench when you can use a spanner?

Example: Crypto Woo-Woo

There are equivalents to this in crypto…

Wireless USB (WUSB)

- Short-range, low-power communications

In 2004:

- 4096-bit DH!
- 3072-bit RSA!
- SHA-256!
- AES-CCM!
- Did we miss out anything else we could throw in?

Source: FTDI

Implementing it on $0.15 chip is Someone Else's Problem

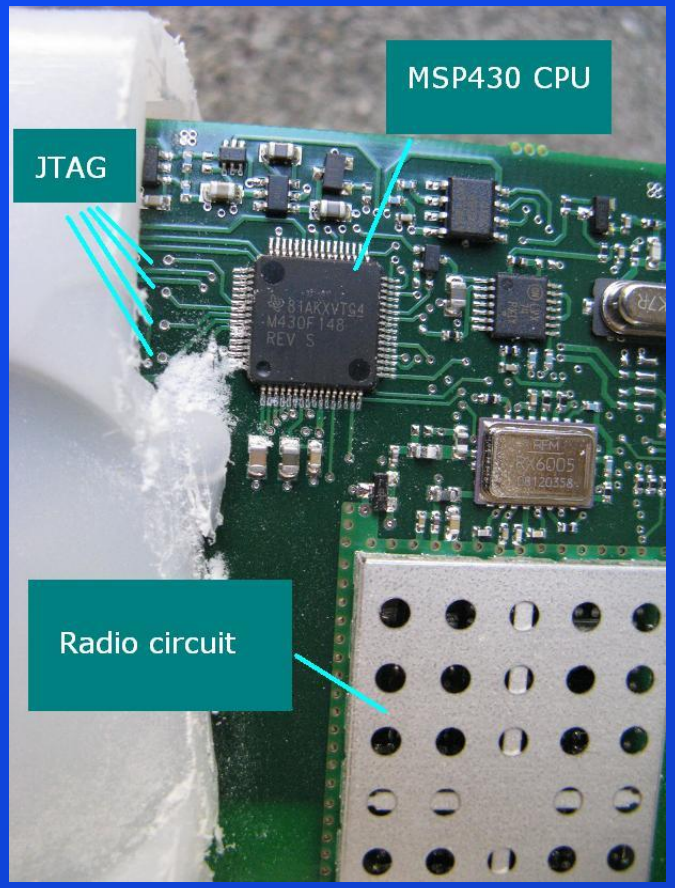
Example: Crypto Woo-Woo (ctd)

"Smart" meters

- Digital signatures!
- X.509 certificates!
- CRLs!
- The whole PKI shebang!

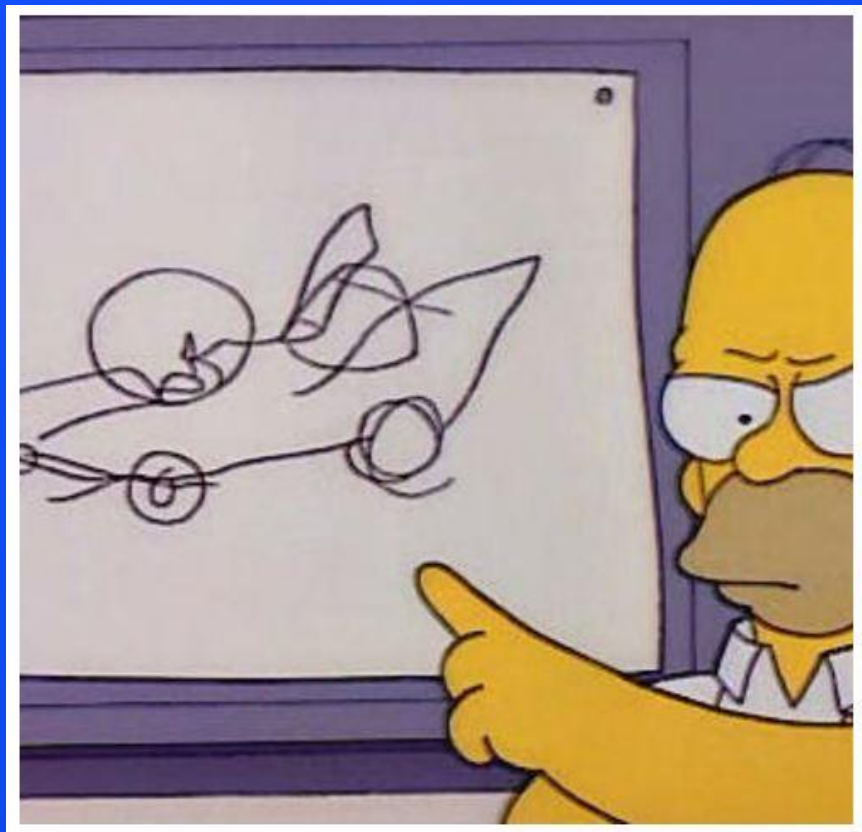
MSP430F148 CPU

- 8 MHz 16-bit CPU
- 16-bit multiplier as external functional unit (no divide)
- 2kB RAM, 48kB flash
- Additional analog/digital circuitry for a power meter



You can guess how much PKI this actually implements…

Getting Back to Sports Cars



Source: StarsAndCars

Wicked Problems

This perfectly illustrates the characteristics of a wicked problem…

No definitive formulation of what's required for a sports car

No stopping rule to tell you that you've definitely reached your goal

- Running out of money is one oft-encountered stopping rule

The various options can only be rated in terms of tradeoffs against each other

continues…

Wicked Problems (ctd)

…continued

It's not obvious which steps are the best ones to take in getting to your goal

All manner of externalities

- Participants' opinions of which option is best
- Bikeshedding comes as an automatic built-in
- Externally-applied materials and regulatory constraints on what you can and can't do

Source: Top Gear

Problem: Secure Ops on Insecure Systems

Trying to perform safe operations on an untrusted system has historically been mostly of academic interest

- "Programming Satan's Computer", Anderson and Needham, 1995
  - Satan's Computer in 1995 was positively benign compared to what's hiding inside many current PCs
- On the off chance that your machine gets compromised, reformat and reinstall from clean media

Today the sheer scale and scope of the problem has made this approach all but impossible

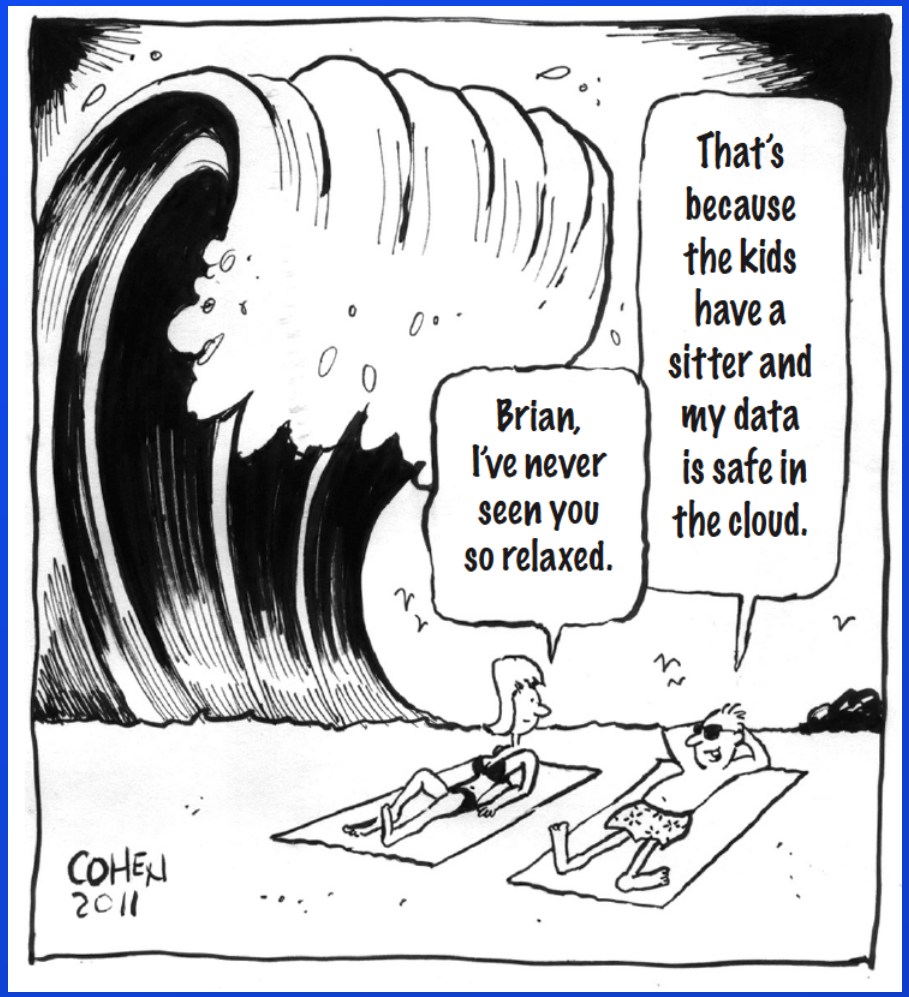
Problem: Secure Ops on Insec.Systems (ctd)

In any case this was before the cloud came along

- "The cloud" = marketing-speak for "someone else's computer"

"How do I secure my data when it's ~~in the cloud~~ on someone else's computer?"

- This is a trick question, right?



Source: Novelaspect

Problem: Secure Ops on Insec.Systems (ctd)

What about trusted computing?

- Yeah, any year now...

Even if it could be deployed, it can protect only a small part of the system, typically the OS core

- A fully-protected computer on which you can’t make any changes isn’t terribly useful

Problem: Secure Ops on Insec.Systems (ctd)

Even for the portions that it does protect, all it guarantees is that they're unchanged from the state they were in when the TPM initially examined them

- Just because it's TPM-verified doesn't mean that it's safe

TPMs and cloud-based VMs are even messier

Source: Supermicro

Problem: Secure Ops on Insec.Systems (ctd)

A typical industry figure for code defects is about twenty bugs in every thousand lines of code (KLOC)

- Feel free to substitute your own pet value at this point
- The important thing isn't the absolute value but the rough order of magnitude for estimation purposes

Widely-used operating systems like Linux and Windows weigh in at 50-100 million lines of code

- Again, depending on which version and what you count as being part of "Linux" and "Windows"

Problem: Secure Ops on Insec.Systems (ctd)

That's between one and two million bugs in the OS

- Ignores the additional code that'll be added in the form of user-installed device drivers and other kernel components
  - A majority of Windows OS crashes are due to these additional drivers

Ignores the perpetual churn of updates

- TPMs weren't really designed for a constantly-changing code base

So your TPM-verified boot guarantees that you're loading an OS core with only a million bugs

- As opposed to a tampered one with a million and one bugs

Problem: Secure Ops on Insec.Systems (ctd)

Accept the fact that you can never really trust anything that's done on a PC and treat it purely as a router?

- Forward encrypted/authenticated content from a remote server to a self-contained device via USB or Bluetooth or NFC

Problem: Secure Ops on Insec.Systems (ctd)

This “solution” gets re-invented every six to twelve months by academics and vendors

Problem: Secure Ops on Insec.Systems (ctd)

The process has been ongoing for at least *thirty years*

This card is from 1986!

It’s a remarkably persistent meme

- Latest iteration was only recently, with Google pushing U2F tokens as the solution to all your authentication problems

Problem: Secure Ops on Insec.Systems (ctd)

PC-as-a-router for secure tokens doesn't work too well as a general solution…

- Expensive
- Requires deployment of specialised hardware
- Requires custom protocols and mechanisms both on the client and the server in order to handle the constraints imposed by the attached device
- A royal pain to use

Usefulness/inconvenience ratio is just too big

Problem: Secure Ops on Insec.Systems (ctd)

What if we use a cellphone as the external device?

- That one's been reinvented for about ten to fifteen years too

Cellphone ≡ Windows 95 PC

- Bloated OS riddled with buggy, never-updated (Android) components running every random app the user can download
- Hack like it's 1997!

Source: ZDNet

Problem: Security vs. Availability

Availability concerns dictate that in the case of a problem the system allows things to continue

- Security concerns dictate that in the case of a problem the system doesn't allow things to continue

This is an umbrella problem that encompasses several other unsolvable sub-problems (covered later) as subclasses

- Unattended key storage
- Upgrade a product or device after a security breach

Source: Sparkfun

Problem: Security vs. Availability (ctd)

Availability concerns can be a powerful motivator

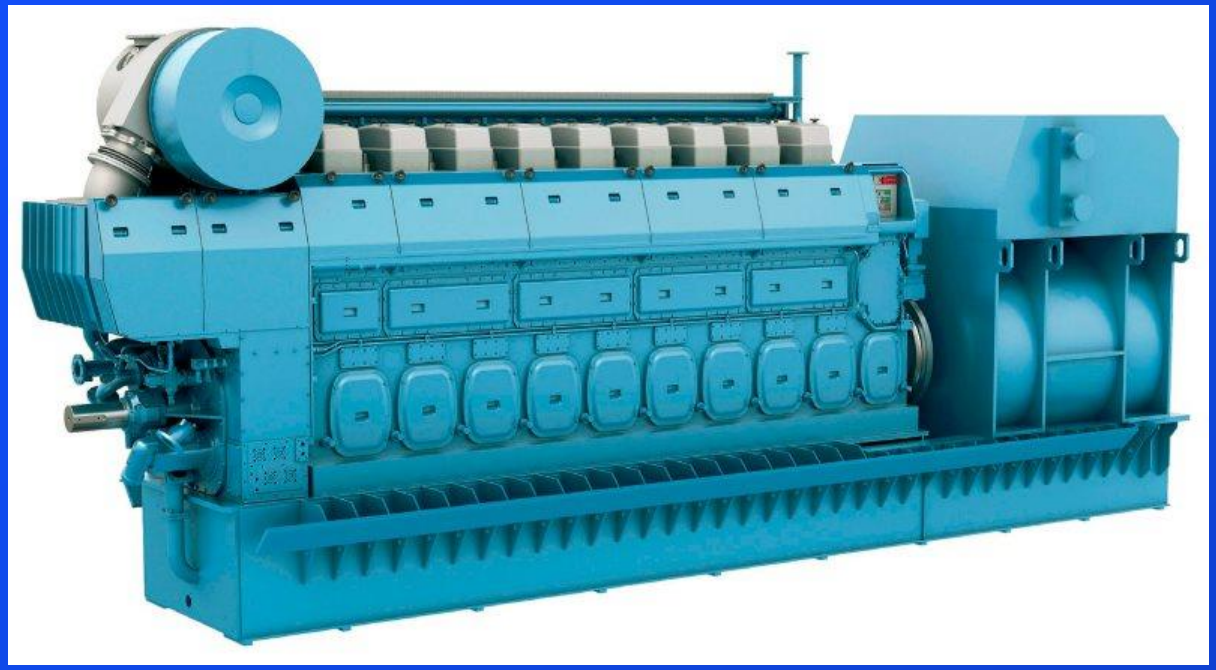
Data centre was built with marine diesel generators for backup power

- Marine diesels come with a built-in cooling system
- That would be “the ocean”

Source: Thinglink

Problem: Security vs. Availability (ctd)

Less concern about marine diesels overheating than conventional generators



Source: DirectIndustry

- Makes them more compact than standard generators
- Space was a concern for the data centre in question

Problem: Security vs. Availability (ctd)

The data centre wasn't anywhere near the ocean

Used a stand-in consisting of a large water cistern whose contents were flushed through the generators' cooling systems

Source: ClimateTech

- When the cistern had emptied, the generators' thermal cut-outs shut them down

Problem: Security vs. Availability (ctd)

Management's response to this was to have the safety interlocks on the generators disabled

- Might get an extra five or ten minutes out of them
- Could potentially ride out a power outage that they wouldn't otherwise have survived

Source: Axon

Problem: Security vs. Availability (ctd)

Preferable to run the generators to destruction than to risk having the data centre go down

- You won't find this in the MCSE or CCNA training material

Source: Artzat

Problem: Security vs. Availability (ctd)

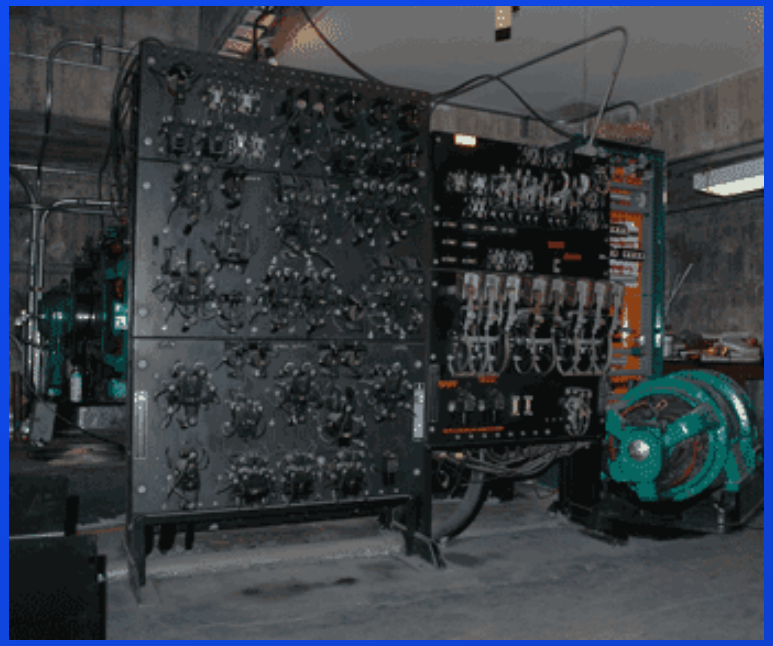
High-availability systems (e.g. SCADA) cannot go down

- Ever

MTBF requirements of ten years are not uncommon

- May be run for decades

Often can't be patched or updated



Source: Eletec Global

Problem: Security vs. Availability (ctd)

Problem: CA-issued certificates are valid for one year

- MTBF 12 months ≪ MTBF 10 years
- Ignore certificate expiry
  - In any case it's just a CA billing mechanism
  - Certificate that's perfectly fine on day *n* doesn't become completely insecure on day *n* + 1
- Issue your own certificates with infinite lifetimes

Problem: Certificates may suddenly stop working due to revocation

- Ignore CRLs and OCSP

Problem: Security vs. Availability (ctd)

Problem: Devices don't have DNS names, or even fixed IP addresses

- Identity = address often doesn't hold in any case
- Device can be identified by IEEE EUI64 ID (OUI + unique ID)
- ID the device by EUI64 or similar after you connect

The more checking you do, the greater the chance of something breaking

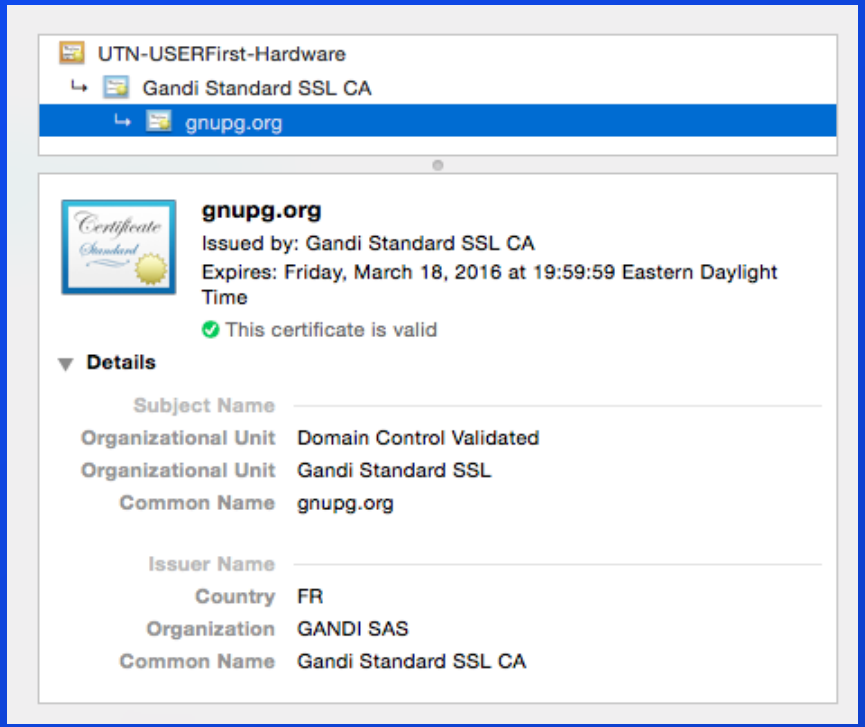
- Disable the ~~safety interlocks~~ nuisance checks to make sure things keep running

Problem: Security vs. Availability (ctd)

To ensure it works, ignore ID info, expiry dates, and revocations

- That's the entire certificate except for the key

And it now works fine on an MSP430!



Problem: Security vs. Availability (ctd)

Random number generation

- `/dev/random` blocks until enough entropy is available
- `/dev/urandom` doesn't

Tinfoil-hat response

- Only ever use `/dev/random`

Linux kernel provides a system call `getrandom()`

- In the kernel for years but wasn't supported in glibc
- Google "ulrich drepper"
- Some support finally added in late 2016

Problem: Security vs. Availability (ctd)

Blocks, but also uses `/dev/urandom`

- Worst of both worlds

`getrandom()` → `make_application_hang_at_random()`

- Quite literally so

If you disabled the blocking, what would happen?

- (Your application wouldn't appear to hang/crash at random any more)
- "Somewhere on the Internet there may be a system that may be running with reduced entropy"
- How is that exploitable by an attacker?

Problem: Security vs. Availability (ctd)

Sometimes you can reach a compromise…

Microsoft did this with the Windows XP SP2 firewall settings

- Finally, *finally* turned on by default in XP SP2

Found that home networks in which a computer acts as a file/print server were broken by having ports closed by default

- Open the ports required for print and file sharing…

  … but only for the local subnet

Problem: Security vs. Availability (ctd)

Home users are unlikely to be running computers on multiple subnets

- Anyone sophisticated enough do this will presumably know what a firewall is and what to do with it

Protects home users from Internet-based attacks without breaking their existing network setup

Security vs. Availability at the Design Level

Any RFC ever

… the server MUST NOT … the client MUST NOT …

More common is to leave it unspecified

Network Working Group
Request for Comments: 3093
Category: Informational

M. Gaynor
S. Bradner
Harvard University
1 April 2001

Firewall Enhancement Protocol (FEP)

Status of this Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

Abstract

Internet Transparency via the end-to-end architecture of the Internet has allowed vast innovation of new technologies and services [1]. However, recent developments in Firewall technology have altered this model and have been shown to inhibit innovation. We propose the Firewall Enhancement Protocol (FEP) to allow innovation, without violating the security model of a Firewall. With no cooperation from a firewall operator, the FEP allows ANY application to traverse a Firewall. Our methodology is to layer any application layer Transmission Control Protocol/User Datagram Protocol (TCP/UDP) packets over the HyperText Transfer Protocol (HTTP) protocol, since HTTP packets are typically able to transit Firewalls. This scheme does not violate the actual security usefulness of a Firewall, since Firewalls are designed to thwart attacks from the outside and to ignore threats from within. The use of FEP is compatible with the current Firewall security model because it requires cooperation from a host inside the Firewall. FEP allows the best of both worlds: the security of a firewall, and transparent tunneling thought the firewall.

Source: IETF

Security vs. Availability at Design Level (ctd)

Try finding a statement in a standard for a protocol (TLS, S/MIME, PGP, SSH, OCSP, SCEP, CMP, …) that tells you what to do if a crypto validation fails

- Not even a "if XYZ validation fails the client MUST terminate the connection"

There's just… nothing

- OK, one or two small notes specifically pointing out particular special-case oddball conditions

Security vs. Availability at Design Level (ctd)

You can write an implementation that ignores MAC failures, decryption errors, and invalid signatures, and be fully standards compliant

- As long as you deal with a small number of obscure corner cases specifically called out as MUST NOTs

Look at the spec for your favourite protocol after the talk

- Someone else's problem?
- It was never even considered?
- The experienced driver will usually know what's wrong?

Security vs. Availability at Design Level (ctd)

Any security RFC ever

- "Here is a security protocol. Whatever it happens to defend against is the threat model"

The Inside-Out Threat Model

Here is some crypto. If it happens to do what you want, go ahead and use it
— Matt Blaze?, Protocols Workshop

Security vs. Availability at Design Level (ctd)

Crypto designer assumptions: Our protocol is secure in the XYZ model

"Can a computationally unbounded attacker *who operates within the bounds of the protocol* compromise it?"

- An attacker that somehow has near-infinite computing power is however constrained to do what the defender wants

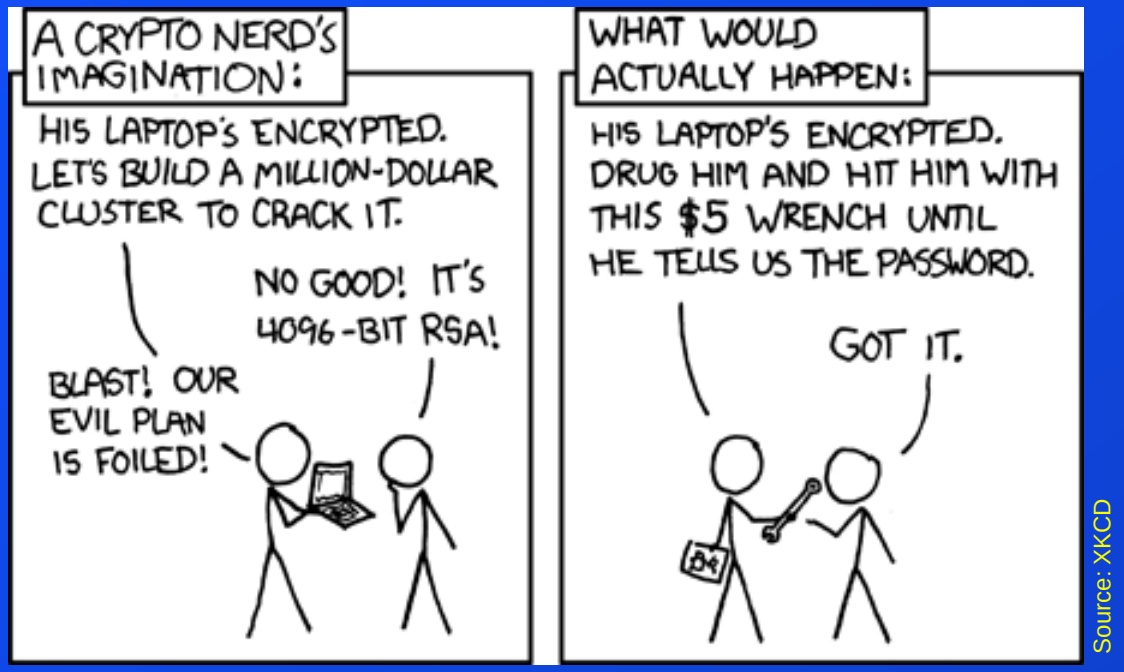
Prove that the protocol is secure within the XYZ model

- In other words, within the bounds that are defined by the defender

Security vs. Availability at Design Level (ctd)

Attacker assumptions: Ø

- Attackers don't even know that the XYZ model exists, let alone feel bound to abide by it



- If this cartoon was text-only, I'd have it permanently saved in a paste buffer for use in mailing list debates

Security vs. Availability at Design Level (ctd)

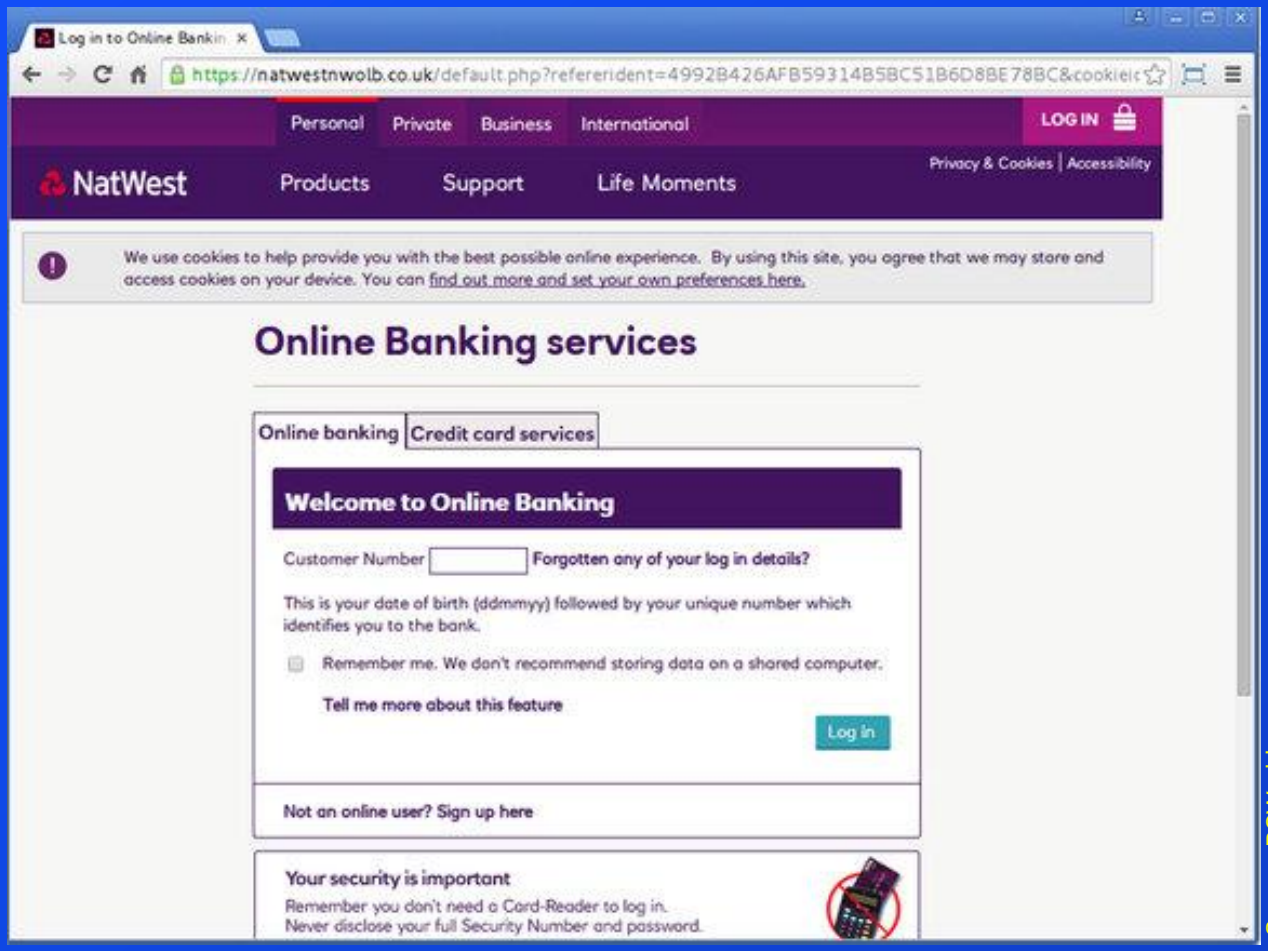
OK, we can't really create rubber-hose-proof crypto

- (No really, we can't. Stop trying to imagine that we can)

We should however be able to scope out the areas that the crypto doesn't defend against

Security vs. Availability at Design Level (ctd)

Question: Should TLS defend against phishing?



Source: PCWorld

Security vs. Availability at Design Level (ctd)

TLS designers: No, of course not. Everyone knows that

- Except 99.99% of all web users everywhere
- "If you can see the padlock/green bar, you're safe"

TLS has no threat model

- Nor does SSH, S/MIME, or PGP
- DNSSEC model was reverse-engineering from the spec a decade after it was published
- IPsec was similar

Security vs. Availability at Design Level (ctd)

IEEE standards require a rationale

- Explanation for why the standard does something

No major security RFC (TLS, SSH, PGP, S/MIME, IPsec, etc) contains one

- There are things in RFCs that cannot be rationally explained
- Seriously! When the authors were asked, they had no idea why their protocol design required XYZ

Rationale serves two purposes

- Guidance to implementers on how to apply a feature
- Sanity check on why the spec requires an action

Security vs. Availability in Practice

For browsers, mail clients, …

- Pop up a dialog and wait for the user to click “Continue anyway”

For unattended/non-interactive devices

- Continue anyway

Problem: Upgrading Insecure Crypto

How do you recover from the catastrophic compromise of a security system?

Extremely rare in properly-designed systems

- Actually, more like totally unknown

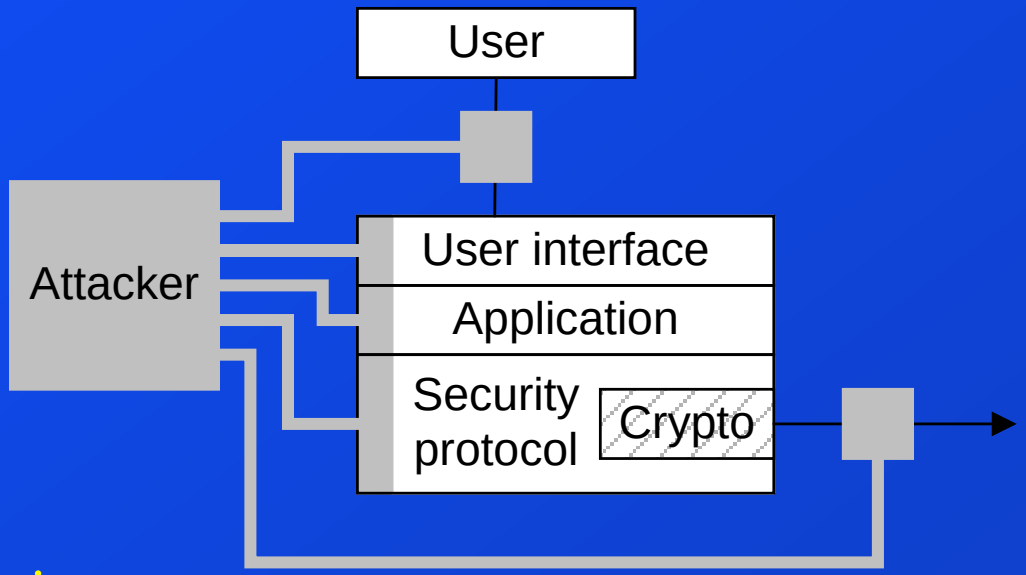
We've always had years and years of advance notice

- MD4, MD5, SHA-1, RC4, …

Problem: Upgrading Insecure Crypto

Attackers target the implementation, the way it's used, or some other aspect unrelated to the crypto

- Shamir's Law: "Crypto is bypassed, not attacked"
- Cryptographers are the people who are so busy patching the mouse holes in the floor that they don't notice that an entire wall of the barn is missing



Easiest approach is to ignore it and hope that it never occurs

Problem: Upgrading Insecure Crypto (ctd)

Consider a system that uses two authentication algorithms in case one fails

- Device receives a message authenticated with algorithm A saying “Algorithm B has been broken, don’t use it any more”
- Device also receives a message authenticated with algorithm B saying “Algorithm A has been broken, don’t use it any more”
- Device may also receive a third message saying “All Cretans are liars”

Could address this with fault-tolerant design concepts

- Voting protocols for algorithm replacement

Adds a huge amount of design complexity and new attack surface

Problem: Upgrading Insecure Crypto (ctd)

Capability will only be exercised in extremely rare circumstances

- Complex, error-prone code that's never really exercised
- Has to sit there unused (but resisting all attacks) for years until it's needed
- Has to work perfectly the first time

How do you safely load a replacement algorithm into a remote device when the existing algorithm that's required to secure the load has been broken?

Problem: Upgrading Insecure Crypto (ctd)

Security geeks want to replace half the security infrastructure that you're relying on as a side-effect of any algorithm upgrade

Source: Wawasmusings, 123RF

Problem: Upgrading Insecure Crypto (ctd)

Example: TLS 1.2

- Deployment lagged for years because the change from TLS 1.1 to 1.2 was far bigger than from SSL to TLS

Scan carried out by a browser vendor in mid-2010 found exactly two public web servers supporting TLS 1.2

- Both were specially set-up test servers

Even in 2016, the most widely-encountered TLS version was 1.0 from 1999

- In SCADA/embedded, TLS 1.0 will probably be around forever

Problem: Upgrading Insecure Crypto (ctd)

Example: TLS ~~1.3~~ ~~4.0~~ ~~2017~~ 1.3, a.k.a. TLS4Google

- Even worse than 1.2

Apart from the different algorithms, cipher suites, messages, message fields, message flow, handshaking, negotiation, extensions, and crypto, it's practically the same thing
— IETF-TLS list comment

- Complete redesign of the protocol to optimise performance for large content providers
- Zero input from embedded, SCADA, IoT, etc
- c.f. HTTP/2, a.k.a HTTP4Google, "anyone who doesn't like it can stay with HTTP 1.1"
- Sites using HTTP/2, in order: Google, Google, Google, Google, Facebook, Google, Google

Problem: Upgrading Insecure Crypto (ctd)

- If the TLS 1.2 experience (15+ year lag to general deployment) is anything to go by, we could see general adoption of 1.3 (outside of Google, Facebook, Akamai, etc) by 2030 or 2035
  - For SCADA/embedded, that date could be “never”
    - HTTP/2 was explicitly forked, HTTP/2 for large Silicon Valley Internet companies, HTTP 1.1 for everyone else
  - They’re still planning the move to 1.2 within the next 5-10 years

Problem: Upgrading Insecure Crypto (ctd)

Situation-specific solutions are possible…

Small number of high-cost units

- Courier out replacement devices that clone their state from the existing one
- Used by some hardware security modules (HSMs)

Source: Thales

Problem: Upgrading Insecure Crypto (ctd)

Remote boxes administered from a central server

- Boxes communicate their state to the central server
- Central site loads it into a new device that gets sent out
- Used by some VoIP boxes rented from a provider

Watch out for
supply-chain attacks!

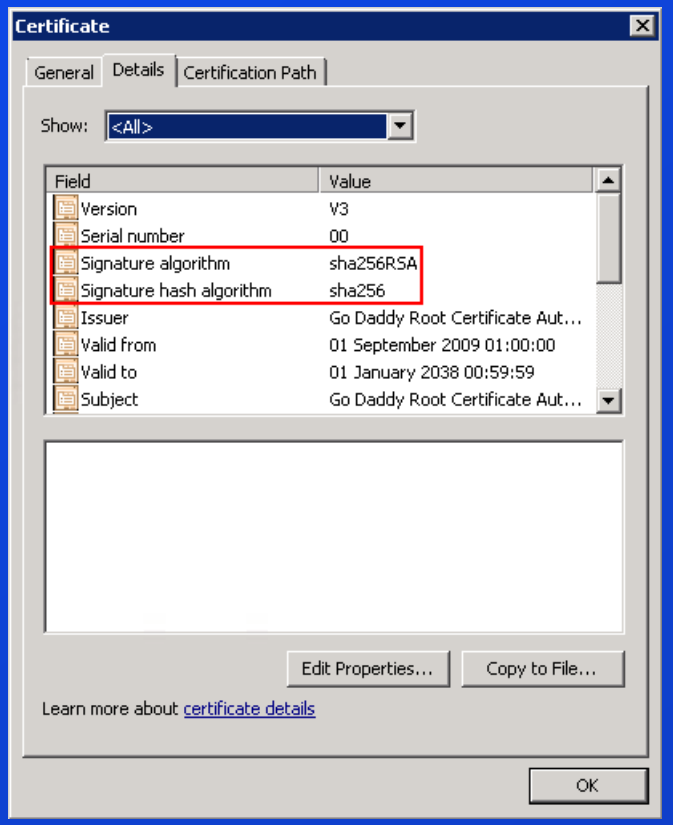
Source: Snowden

Problem: Upgrading Insecure Crypto (ctd)

Opportunistic upgrade of algorithms

- If the other side presents a certificate with algorithm $n + 1$ then switch all communication with the certificate owner to $n + 1$ as well
- Lots of fun for security geeks to play with

May be subject to rollback attacks



Source: MSDN

Problem: Key Storage for Unattended Use

Another variant of security vs. availability

Storing keys in plaintext form is a cardinal sin in cryptography

- A user is expected to enter a password or PIN to unlock or decrypt keys so that they can be used

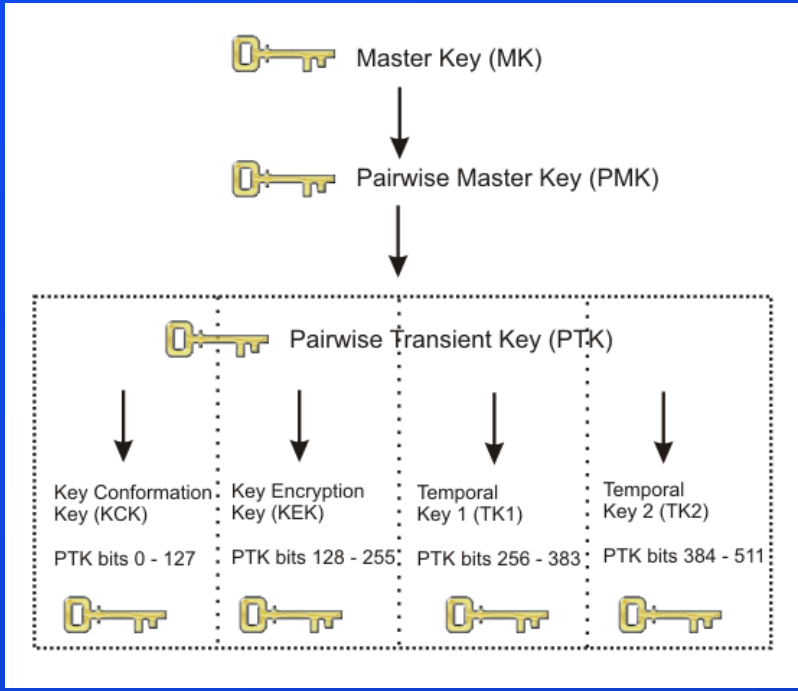
How do you do this for devices that have to be able to operate unattended?

How do you recover from a crash/power outage/OS upgrade/VM migration without explicit human intervention?

Problem: Key Storage for Unattended Use

Various cat-and-mouse games possible

- Poke hierarchies of keys into various locations
- Use them to decrypt other keys
- Hope that an attacker can't work their way back far enough to grab the real keys



Source: TDLP

For unattended operation at some point you need to fall back to a fixed key stored in plaintext-equivalent form that can survive a crash or reboot

Problem: Key Storage for Unattended Use

None of the "obvious" general-purpose solutions to this problem actually solve it

TPMs can only store the fixed storage-protection key that's required to decrypt the real key

- TPMs are just repurposed smart cards and don't have the horsepower to perform anything more than lightweight crypto themselves
- Can't offload the overall encryption processing to them

For unattended operation they have to release their secrets without a PIN

- Merely provide plaintext key storage with one level of indirection

Problem: Key Storage for Unattended Use

Add custom encryption hardware and perform all of the crypto in that

- Most manufacturers will be reluctant to add $500 of specialised encryption hardware to a $50 embedded device
- Scaled up to PC terms, a $20,000 hardware security module (HSM) added to a $2,000 server
- If the HSM vendor has particularly good salespeople they'll sell the client at least two $20,000 HSMs (each storing a single key) for disaster recovery purposes.

Problem: Key Storage for Unattended Use

Not very secure against an attack that compromises the host system

- All the HSM does is move the key from the compromised machine into an external box that does anything that the compromised host tells it to

Useful however for meeting auditing or regulatory requirements

- Adds an auditable physical artefact to the process
- “The box is still there, so we’ll assume that the key is also still there”

Problem: Key Storage for Unattended Use

If you're really concerned about security, move more of the security functionality into the HSM

- Instead of acting as a yes-box for crypto ops, implement whole portions of the underlying security protocol in the HSM
- Takes a large amount of programming effort

Problem: Key Storage for Unattended Use (c

IBM used to sell a fully programmable high-security crypto coprocessor

- Almost no-one took advantage of its programming capabilities

A good idea in theory but practical experience has shown that few users will make the effort

Source: IBM

Hard and Not-so-Hard Problems

So that was all the bad news

- Is there any good news?

Yes, if you're prepared to be flexible

- Some problems are most easily solved by moving the goalposts to where the ball is going, not the other way round

Source: Naturalgasnow

Hard and Not-so-Hard Problems (ctd)

Move the problem to an easier space and then solve that

Accept the fact that there are no perfect solutions

- Well, OK, there are perfect solutions
- Smart cards, PKI, biometrics, quantum anything, …
- Monorails, cold fusion, power too cheap to meter, …

Le mieux est l'ennemi du bien

— Voltaire (and others)

The good you can have right now, the perfect you'll have to wait forever for

Hard and Not-so-Hard Problems (ctd)

You don't need perfection

- Even a small change will stop at least some attackers

"The world's most ineffective CAPTCHA" (CodingHorror)

Please enter the word "orange" 

- Kept the comments section free of comment spam for many years
- (Now outsourced to Discourse, which requires creating an account, logging in, etc etc ☹)

Hard and Not-so-Hard Problems (ctd)

And that's not just effective on lesser-known blogs

CodingHorror is already a pretty popular blog, but it works just as well for major targets like this one

- "What was the colour of the Lone Ranger's white horse?"

Hard and Not-so-Hard Problems (ctd)

Boxes the attacker into smaller and smaller corners

- Standard defence-in-depth measure

Several relatively weak measures piled up can be phenomenally effective

- Or just cut down the overall noise level for an otherwise-unsolvable problem
- Allows you to focus on the real attackers, not the anklebiters

Source: Digventures

User Identification /Authentication

Allow users to sign up for online information (mailing lists, web sites)

- Fraudsters sign up in other people's names
  - Used for DoS, not just pure fraud
- Bots sign up large numbers of addresses to obtain accounts for spam purposes

Email-based Identification

Use the ability to receive mail as a form of (weak) authentication

- Sign up using an email address
- Server sends an authenticator to the given address
- Address owner responds with the authenticator to confirm the subscription
- Sometimes known as double opt-in

Widely used for password resets, mailing list subscriptions, blog registration

- Good enough unless the opponent is the ISP

Email-based Identification (ctd)

Self-auditing via email confirmation

- Attempting to use the account results in the legitimate owner being notified
- Changing the email address should result in a notification being sent to the original address

Enhanced version: Get users to set up a separate email-auth-only account

- Not used for anything else
- Not publicly visible
- Little chance of being phished

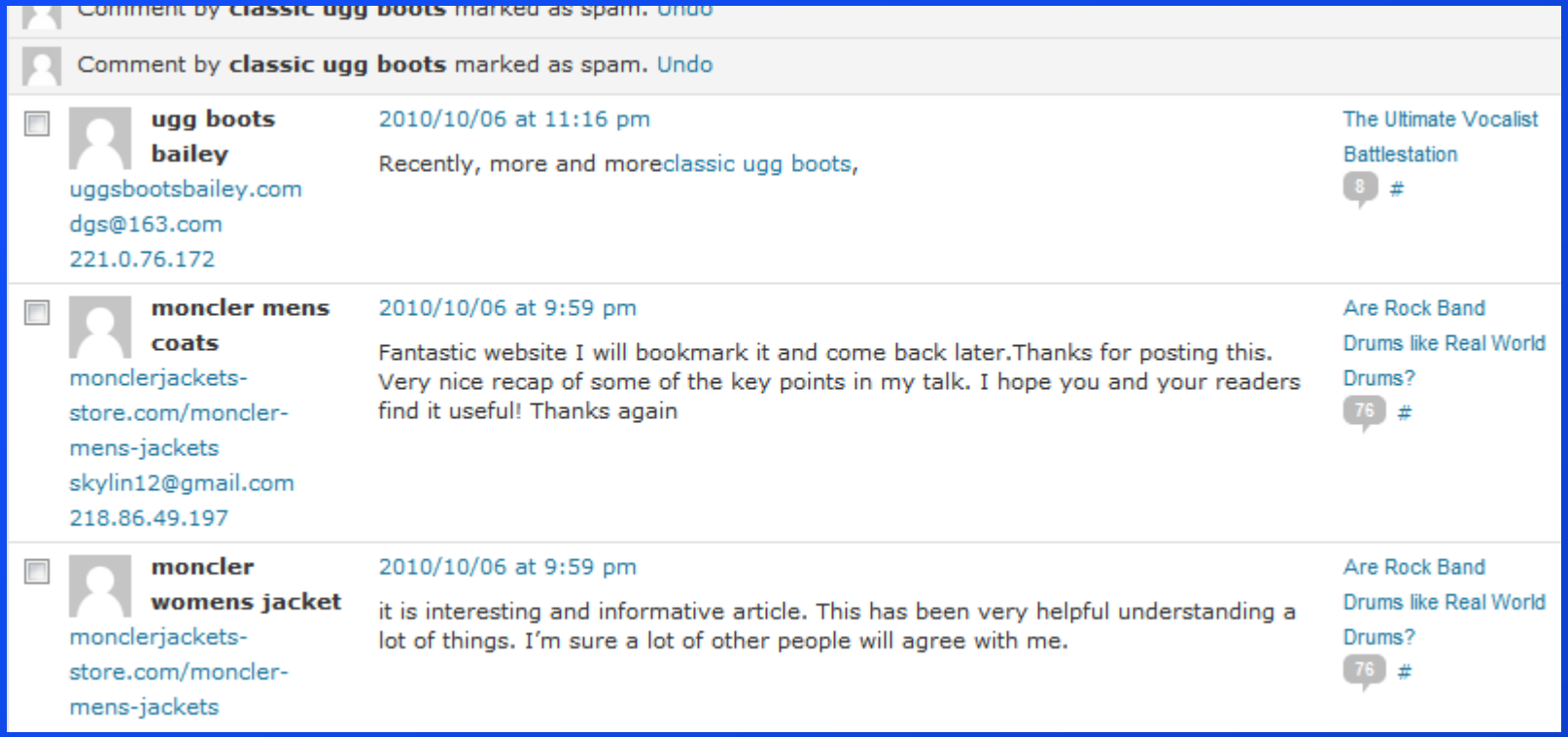
Email-based Identification (ctd)

Low-value authentication, but relatively difficult to defeat compared to what it's protecting

- An attacker who goes to the trouble of compromising your email account probably isn't interested in using it for mailing list access or blog spam

Comment/Link Spam

Use comments in blogs to post spam links



Comment by **classic ugg boots** marked as spam. Undo

Comment by **classic ugg boots** marked as spam. Undo

ugg boots bailey uggsbootsbailey.com dgs@163.com 221.0.76.172	2010/10/06 at 11:16 pm Recently, more and moreclassic ugg boots,	The Ultimate Vocalist Battlestation 8 #
moncler mens coats monclerjackets-store.com/moncler-mens-jackets skylin12@gmail.com 218.86.49.197	2010/10/06 at 9:59 pm Fantastic website I will bookmark it and come back later.Thanks for posting this. Very nice recap of some of the key points in my talk. I hope you and your readers find it useful! Thanks again	Are Rock Band Drums like Real World Drums? 76 #
moncler womens jacket monclerjackets-store.com/moncler-mens-jackets	2010/10/06 at 9:59 pm it is interesting and informative article. This has been very helpful understanding a lot of things. I'm sure a lot of other people will agree with me.	Are Rock Band Drums like Real World Drums? 76 #

Source: StackExchange

- Close enough to real posts to avoid triggering spam filters
- Can render the comments section of any blog unusable

Comment/Link Spam (ctd)

How to deal with this

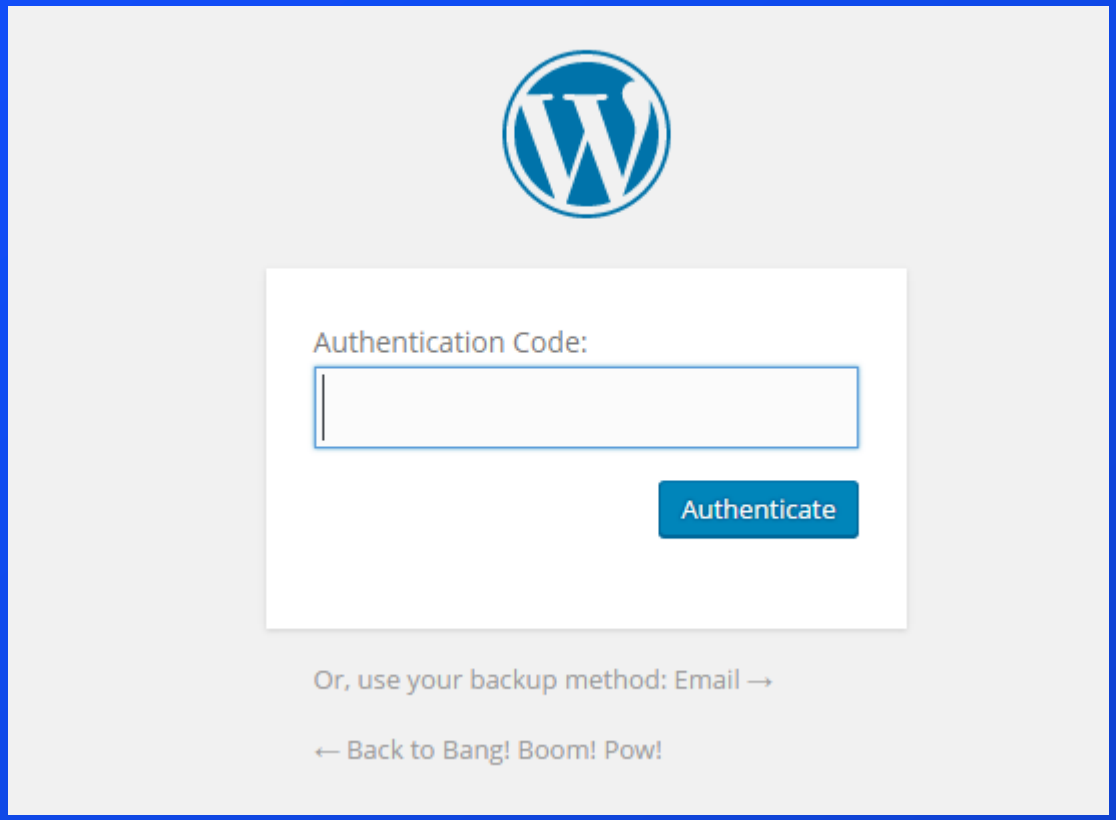
5 Tips to Prevent WordPress Spam Comments

1. Delete All Spam Comments
2. Hold Comments for Moderation
3. Modifying .htaccess to Prevent WordPress Spam Comments
   (Deny bots with no Referrer)
4. Ban the Spammer's IP Address
5. Install Anti-Spam Plugins

Source: UrbanAnomie

Comment/Link Spam (ctd)

OK, how else to deal with this



Source: Wordpress

- Go full crypto on them

Comment/Link Spam (ctd)

Gaahhh!! There's got to be a better way

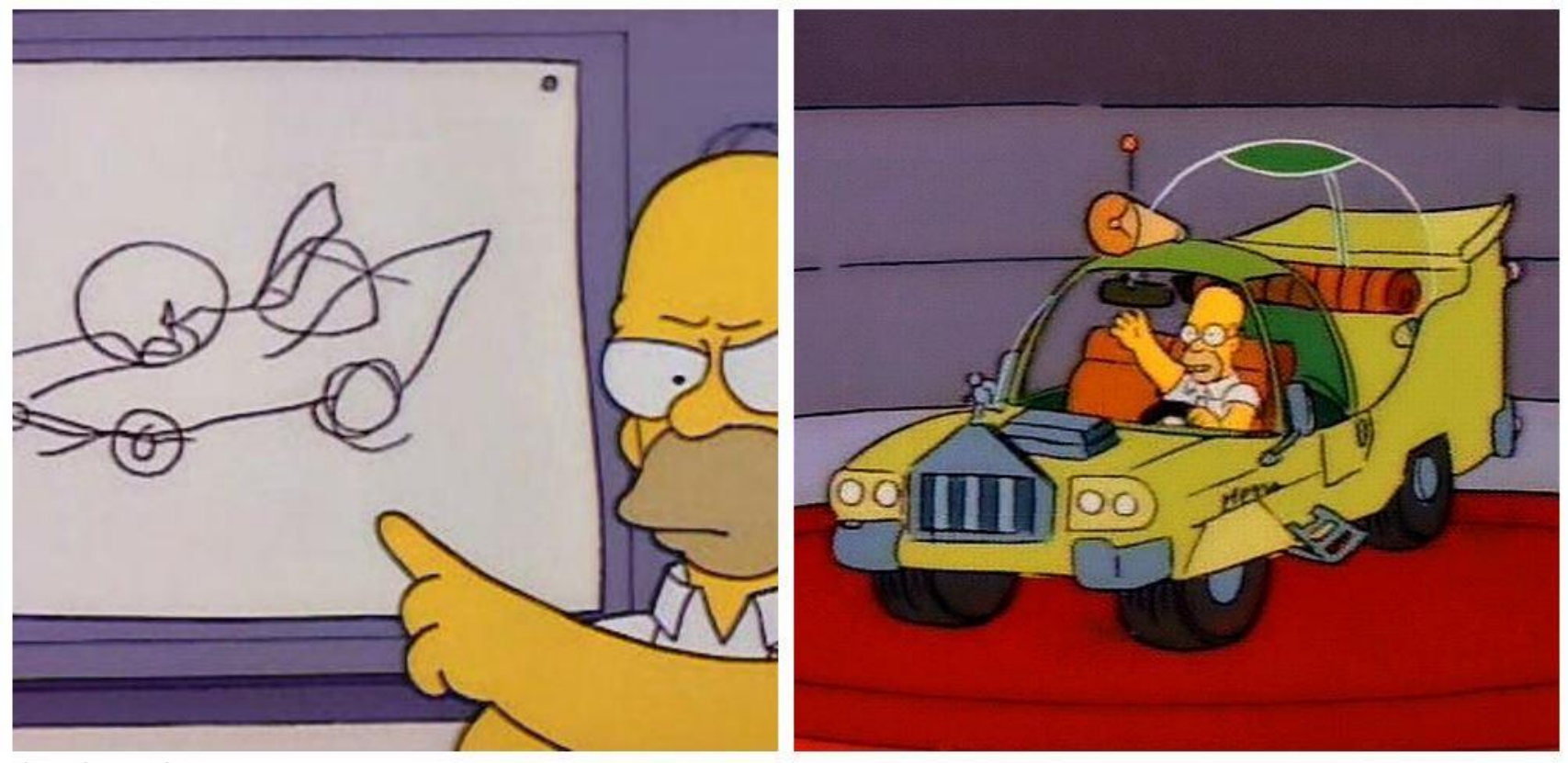
```
<span property="ann:trusted-content">
blog text
</span>
<span property="ann:untrusted-content">
user comments
</span>
```

Blog software knows which text came from the blogger and which came from random users

- Software can HT*Markup*L the untrusted content
- Using structures is also possible, but a bit more complex

Crypto non-Woo-Woo

Remember this?



- WUSB security specification

Crypto non-Woo-Woo (ctd)

The HomePlug folks had the same problem to address

- Needs to work with low-powered devices
- Can’t require a user input device or display (which WUSB does)

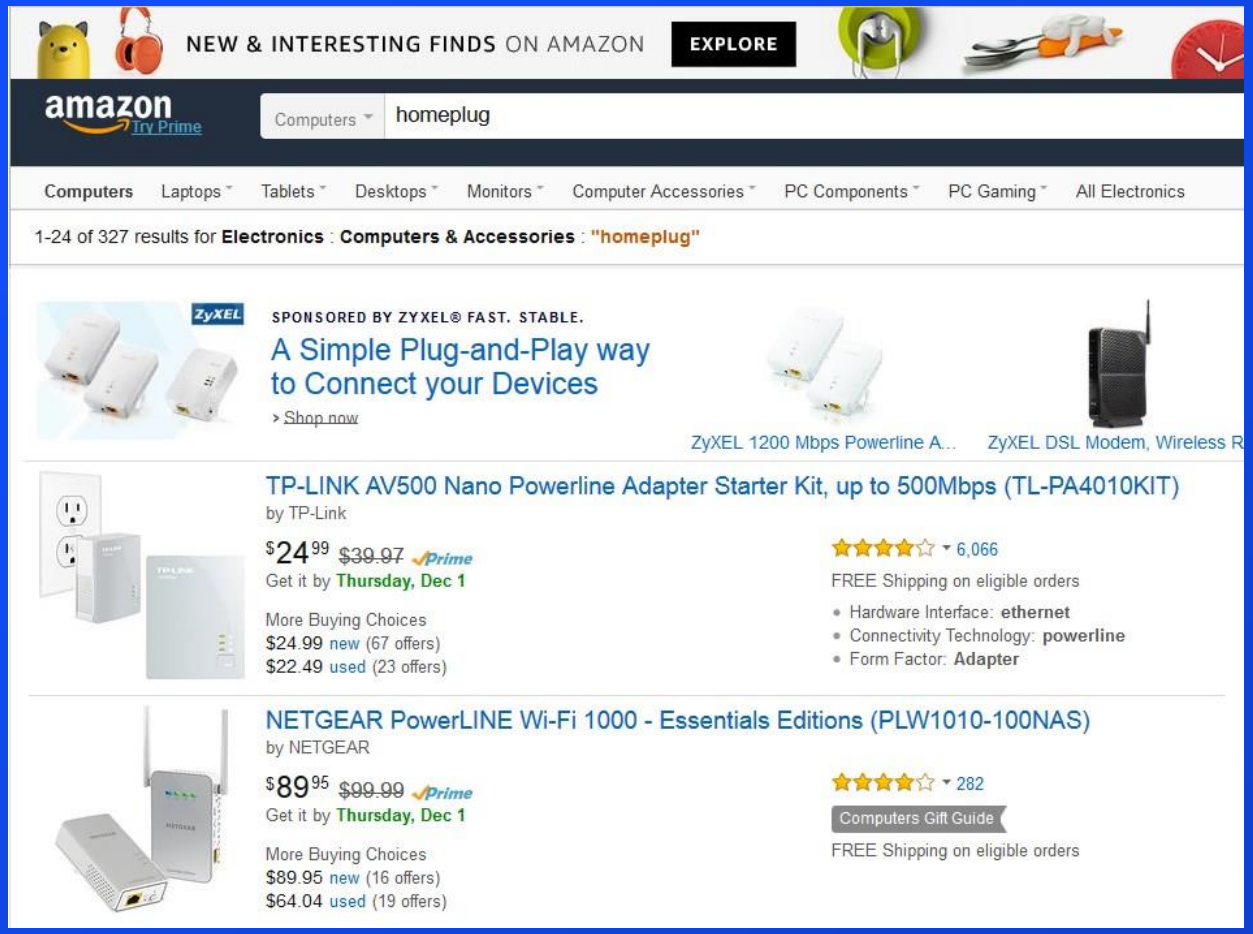
Move the goalposts

- Provide pre-paired adapters in sets of two
- Use location-limited channels
- Rely on attackers not being able to (easily) reverse-engineer OFDM tone maps

Source: Tenda

Crypto non-Woo-Woo (ctd)

Results: HomePlug

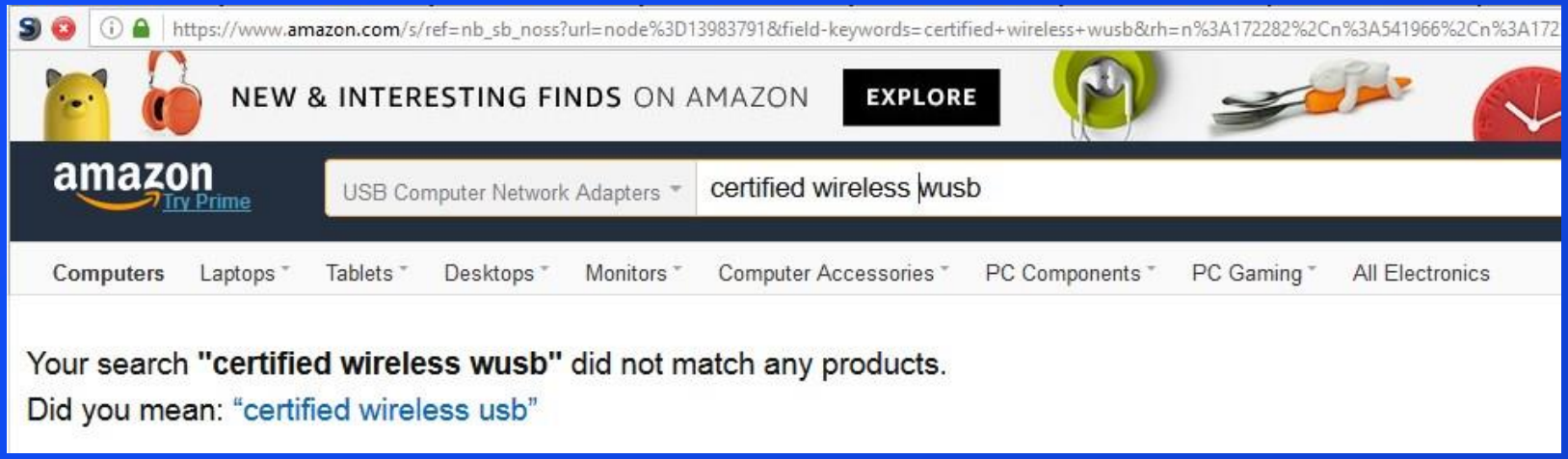


Source: Amazon

- Impressive considering it's long been superseded by WiFi

Crypto non-Woo-Woo (ctd)

Results: WUSB



Source: Amazon

- Need to be specific with search results since “wireless USB” returns all 802.11 results not WUSB, official name is “certified wireless USB” + WUSB
- “Wireless USB” means 802.11, not WUSB

Opportunistic Encryption

After twenty years of effort, S/MIME and PGP use is lost in the noise floor

- Most mail clients include S/MIME support
- Many (OSS) clients include PGP support

Usage is virtually nonexistent

- It's too much bother for most people

  The vast majority of users detest anything they must configure and tweak. Any really mass-appeal tool must allow an essentially transparent functionality as default behaviour; anything else will necessarily have limited adoption

  — Bo Leuf, "Peer to Peer: Collaboration and Sharing over the Internet"

STARTTLS/STLS/AUTH TLS

Opportunistic encryption for SMTP/POP/IMAP/FTP

```
220 mail.foo.com ESMTP server ready
EHLO server.bar.com
250-STARTTLS
STARTTLS
220 Ready to start TLS
<encrypted transfer>
```

- Totally transparent, (almost) idiot-proof, etc

Most commonly encountered in SMTP/POP/IMAP

- Protects mail in transit
- Authenticates sender/prevents unauthorised relaying/spamming

STARTTLS/STLS/AUTH TLS (ctd)

A year after first appearing, STARTTLS was protecting more email than all other email encryption protocols combined, despite their 10-15 year lead

- Just as SSH has displaced telnet, so STARTTLS has mostly displaced straight SMTP
- The fact that it helps authenticate/authorise users no doubt helped

Not perfect, but boxes attackers into narrower and narrower channels

Key Continuity Management

Where's the PKI?

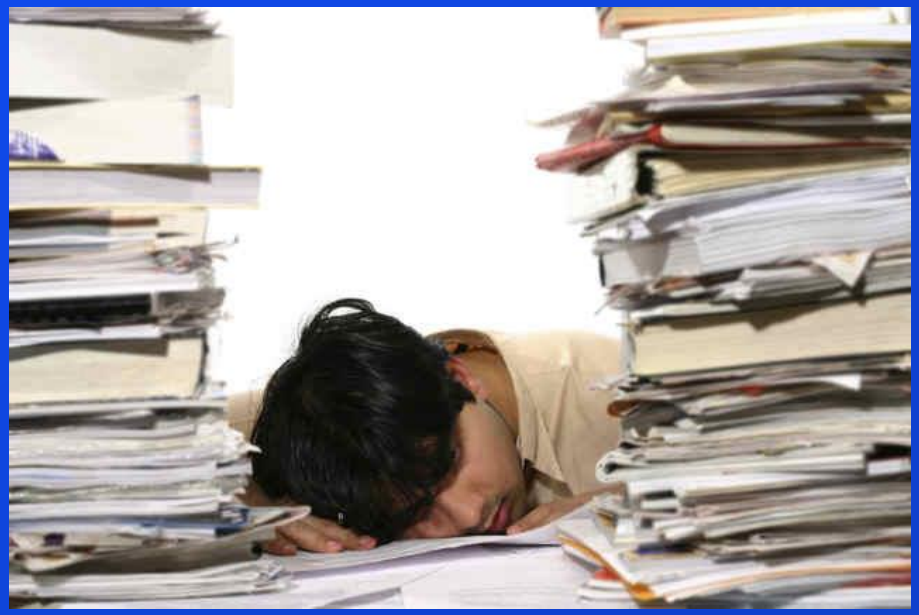
It's too…

- Expensive
- Complex
- Difficult to deploy
- Doesn't meet any real business need
- etc etc etc

Key Continuity Management (ctd)

The only visible use of PKI is SSL

- This is certificate manufacturing, not PKI
- Once a year, exchange a credit card number for a pile of bits
- See a near-infinite number of papers, blogs, and articles on the failure of web PKI to prevent any real attacks



Source: RandomThoughts

Assurance through Continuity

Continuity = knowing that what you're getting now is what you've had before/what you were expecting

- McDonalds primary product line is the same no matter which country you're in
- Coke is Coke no matter what shape bottle (or can) it's in, or what language the label is in

Image removed following copyright infringement claim from the Coca Cola Company

Assurance through Continuity (ctd)

Continuity is more important than third-party attestation

- Equivalent to brand loyalty in the real world
- Businesses place more trust in established, repeat customers

Use continuity for key management

- Verify that the current key is the same as the one you got previously

Key Continuity in SSH

First app to standardise its key management this way

On first connect, the client software asks the user to verify the key

- Done via the key fingerprint, a hash of the key components
- Standard feature for PGP, X.509, …

On subsequent connects, the client software verifies that the current server key matches the initial one

- Warn the user if it changes

Key Continuity in SSH (ctd)

Do SSH Fingerprints Increase Security?

Peter Gutmann
Department of Computer Science
University of Auckland

Abstract

No.

OK, so the fingerprint part doesn't work so well, but the continuity does

Key Continuity Abstract Model

Concept was formalised in the Resurrecting Duckling Security Model, Stajano and Anderson, 1999

- Device imprints on the first item that it sees
- Device trusts that item for future exchanges

Source: Uproxx

Key Continuity Abstract Model (ctd)

Already used by billions of devices worldwide

Key Continuity Abstract Model (ctd)

OK, so we still have a long way to go in some cases…

Update Mechanism	• Update sent without encryption • Updates not signed • Update location writable • Update verification • Update authentication • Malicious update • Missing update mechanism • No manual update mechanism

Source: OWASP

Key Continuity in SIP

Same general model as SSH

- First connect exchanges self-signed certificates
- Connection is authenticated via voice recognition

Same principle has been used in several secure IP-phone protocols

- Users read a hash of the session key over the link
- (This is 20-year-old tech)

Source: RandomThoughts

Key Continuity in SSL

The web guys had a go at this for SSL

RFC 6797: HTTP Strict Transport Security (HSTS)

- Server can specify a duration over which the client must connect using SSL
- No mention of tracking server key changes

Source: Wheelfanatyk

In any case it's not the host that should be controlling things but the client app

- On-by-default, not opt-in

Key Continuity in SSL (ctd)

Finally got it right at the second attempt

RFC 7469: Public Key Pinning Extension for HTTP

- Only accept one of the following set of certificates for the next time period *x*

Well, they tried…

- Tied to HTTP, so doesn't work for any other SSL use
- Google Chrome is the only major browser to support it
  - Guess who wrote the spec?

Key Continuity in S/MIME

S/MIME has a built-in mechanism to address the lack of a PKI

- Include all signing certificates in every message you send
- Lazy-update PKI distributes certificates on an on-demand basis

S/MIME gateways add two further stages

- Auto-generate certificates for new users
- Perform challenge-response for new certificates they encounter

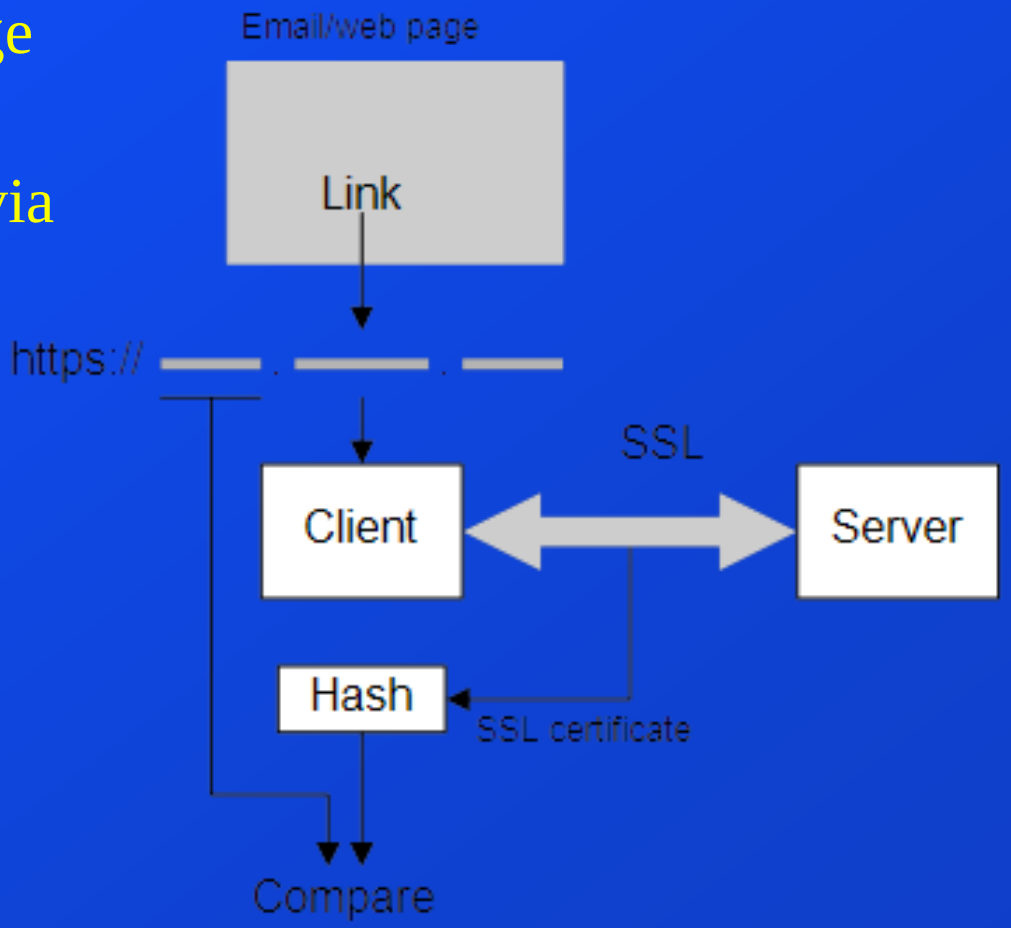
Self-Authenticating URLs

Uniquely tie a DNS name to a key

- URL posted on web page or sent in email
- Connect to SSL server via the URL

URL contains a hash of the key or certificate

- Only that URL can be accessed with that key



Self-Authenticating URLs (ctd)

Client compares the SSL key to the hash of the key in the URL

- If they match then it's the actual server in the URL, not a fake server or MITM from DNS spoofing

Fully compatible with existing applications, just with reduced security guarantees

But wait, this leads to ugly URLs!

```
https://a6ewc3n4p6ra27j2mexqd.downloadsite.com
```

- Have you looked at an Amazon/eBay/whatever URL recently?
- No worse than existing mangled URLs

Self-Authenticating URLs (ctd)

Used by PyPI (Python Package Index) to authenticate packages

Link to package is posted as `http://pypi.python.org/-packages/foo.tar.gz#sha1=23cb[…]e5fc`

- Link contains hash needed to check the package
- Packed into the HTTP fragment identifier

Python install tools automatically verify its integrity on download

Self-Authenticating URLs (ctd)

Deals with the global PPI (Per-Per-Install) industry

- Repackage existing distros to include malware

Transforms the problem of

- Signing every package
- Managing a PKI
- Providing client-side software capable of interpreting the code-signing data

To

- Providing a secure location to post URLs

That’s moving a very large goalpost!

Self-Authenticating URLs (ctd)

BitTorrent uses something a bit like this

- Actually just a fragment identifier to identify a piece of a large file
- Has the convenient side-effect that the torrent metadata also provides something like a self-authenticating URL

Other P2P protocols similarly use hashes to uniquely identify content online

More generally, DHTs use them to create self-certifying named objects

- Get me the object with this hash
- Does this object correspond to the hash I've got for it

Self-Authenticating URLs (ctd)

A general form was proposed as link fingerprints

Attempts to standardise it were torpedoed due to concerns that it sapped and impurified the precious bodily fluids of URLs

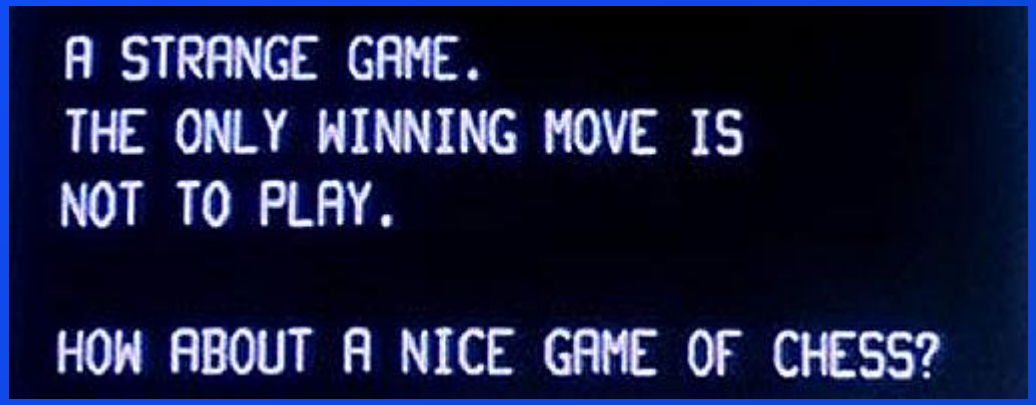
- Added to Firefox, but removed again due to concerns that people might actually use it
- Seriously!

Supported in various plugins and download managers

Conclusion

Yeah, OK, so playing with crypto is fun

- There are some problems that just aren't practically solvable with crypto
- That doesn't mean you can't publish fun papers on them, but still…



Source: Youtube

Nope, you can win if you change the rules of the game

- Redefine the problem to make it solvable