

Network Security

4

matt blaze 
@mattblaze Following

Cryptocurrency somehow combines everything we love about religious fanatics with everything we love about Ponzi schemes.

7:08 AM - 23 Oct 2017

68 Retweets 128 Likes

4 68 128

matt blaze 
@mattblaze · 18m

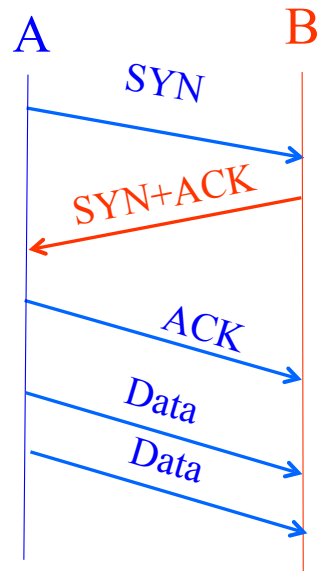
Replying to @mattblaze

There's a lot to dislike about the world we're in, but at least Ayn Rand didn't have bitcoin to write about.

Why Am I Not Hearing A Lecture?

- Because the start of the lecture is on the whiteboard and not recorded!
- It is, as announced, the "official" solution for Project 2.
- As it includes multiple discussions of architectures, attacks, etc, there is no recording
- Plus announcements: Project 3, wheee!!!!

Reminder: Establishing a TCP Connection



Each host tells its *Initial Sequence Number (ISN)* to the other host.

(Spec says to pick based on local clock)

Hmm, any way for the attacker to know *this*?

How Do We Fix This?

Use a (Pseudo)-Random ISN

Sure – make a non-spoofed connection *first*, and see what server used for ISN y then!

Summary of TCP Security Issues

- An attacker who can observe your TCP connection can manipulate it:
 - Forcefully terminate by forging a RST packet
 - Inject (spoof) data into either direction by forging data packets
 - Works because they can include in their spoofed traffic the correct sequence numbers (both directions) and TCP ports
 - Remains a major threat today

Summary of TCP Security Issues

- An attacker who can observe your TCP connection can manipulate it:
 - Forcefully terminate by forging a RST packet
 - Inject (spoon) data into either direction by forging data packets
 - Works because they can include in their spoofed traffic the correct sequence numbers (both directions) and TCP ports
 - Remains a major threat today
- If attacker could predict the ISN chosen by a server, could “blind spoof” a connection to the server
 - Makes it appear that host ABC has connected, and has sent data of the attacker’s choosing, when in fact it hasn’t
 - Undermines any security based on trusting ABC’s IP address
 - Allows attacker to “frame” ABC or otherwise avoid detection
 - Fixed (mostly) today by choosing random ISNs

But wasn't fixed completely...

- CVE-2016-5696
 - "Off-Path TCP Exploits: Global Rate Limit Considered Dangerous" Usenix Security 2016
 - <https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/cao>
- Key idea:
 - RFC 5961 added some global rate limits that acted as an **information leak**:
 - Could determine if two clients were communicating on a given port
 - Could determine if you could correctly guess the sequence #s for this communication
 - Required a third host to probe this and at the same time spoof packets
 - Once you get the sequence #s, you can then inject arbitrary content into the TCP stream (d'oh)

The SYN Flood DOS Attack...

- When a computer receives a TCP connection it decides to accept
 - It is going to allocate a significant amount of state
- So just send lots of SYNs to a server...
 - Each SYN that gets a SYN/ACK would allocate some state
 - So do a **lot of them**
 - And **spoof** the source IP
- Attack is a resource consumption DOS
 - Goal is to cause the server to consume memory and CPU
- Requires that the attacker be able to spoof packets
 - Otherwise would just rate-limit the attacker's IPs

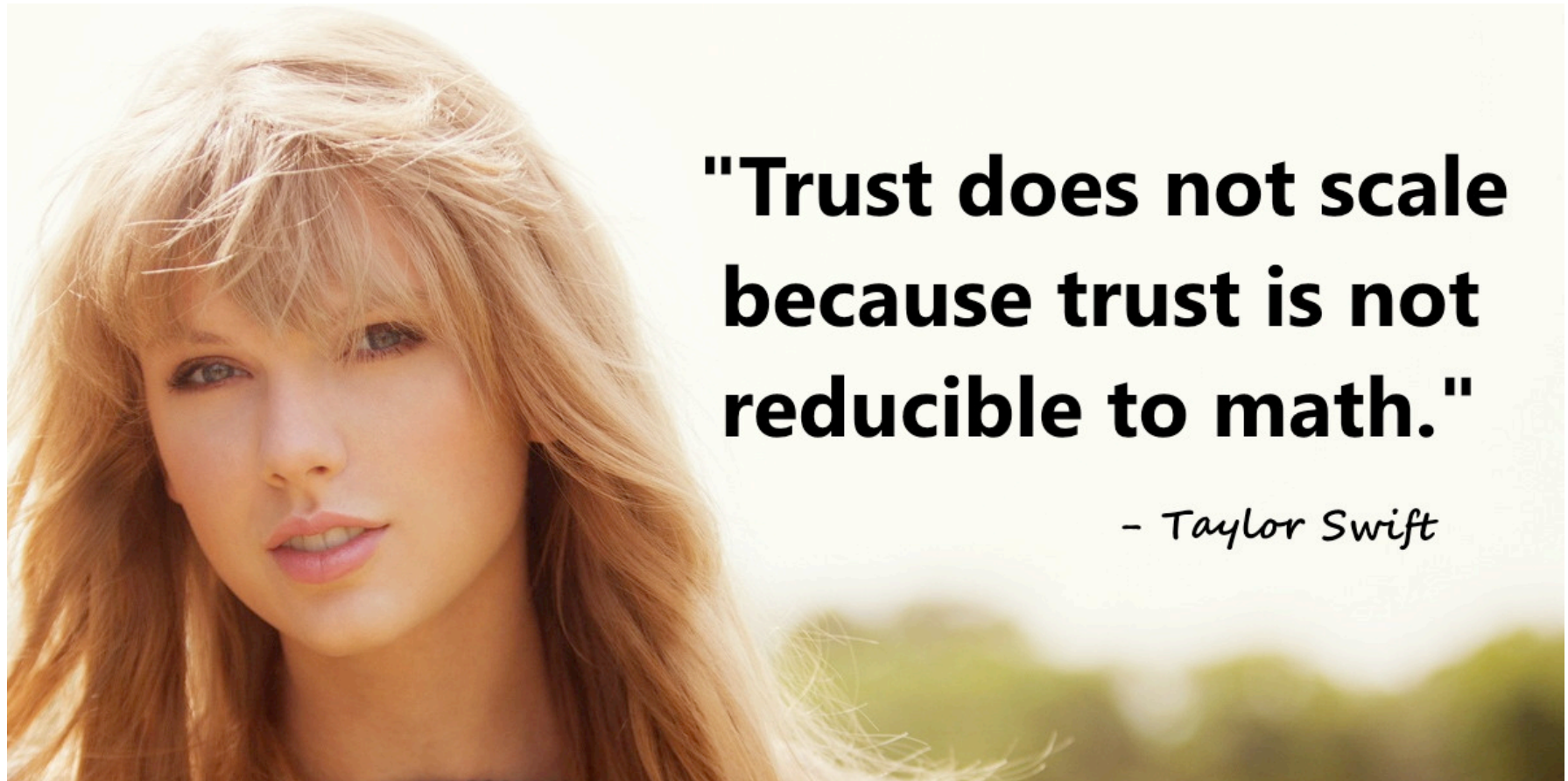
SYN-Flood Counter: SYN cookies

- Observation: Attacker needs to see or guess the server's response to complete the handshake
 - So don't allocate **anything** until you see the ACK...
But how?
- Idea: Have our initial sequence **not** be random...
 - But instead have it be **pseudo**-random
- So we create the SYN/ACK's ISN using the pseudo-random function
 - And then check that the ACK correctly used the sequence number

Easy SYN-cookies: HMAC

- On startup create a random key...
- For the server ISN:
 - $\text{HMAC}_k(\text{SIP}|\text{DIP}|\text{SPORT}|\text{DPORT}|\text{client_ISN})$
- Upon receipt of the ACK
 - Verify that ACK is based off $\text{HMAC}_k(\text{SIP}|\text{DIP}|\text{SPORT}|\text{DPORT}|\text{client_ISN})$
- Only ***then*** does the server allocate memory for the TCP connection
 - HMAC is very useful for these sorts of constructions:
Give a token to a client, verify that the client presents the token later

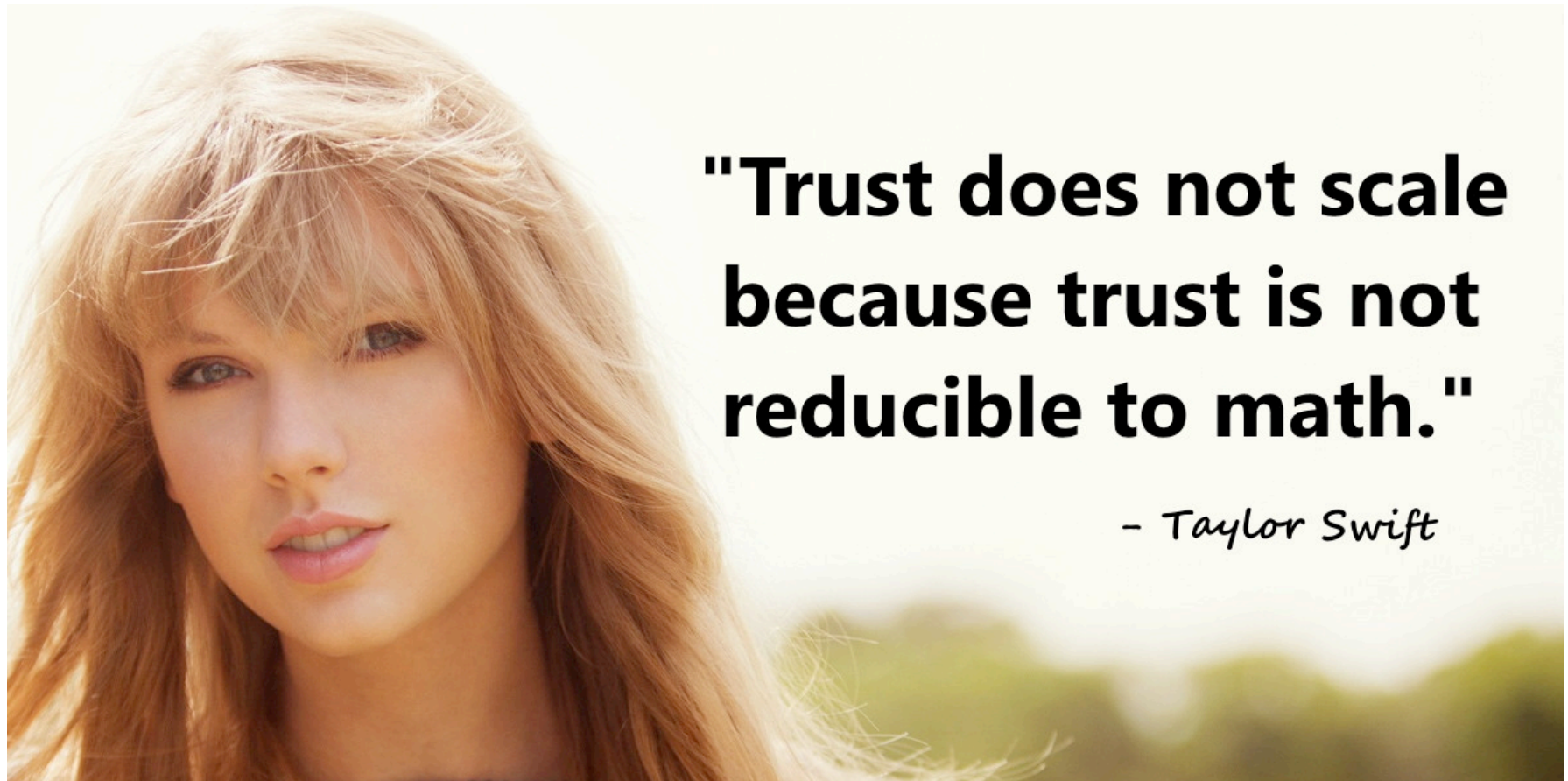
Theme of The Rest Of This Lecture...



**"Trust does not scale
because trust is not
reducible to math."**

- Taylor Swift

But Trust Can Be Delegated...



**"Trust does not scale
because trust is not
reducible to math."**

- Taylor Swift

The Rest of Today's Lecture:

- Applying crypto technology in practice
- Two simple abstractions cover 80% of the use cases for crypto:
 - “Sealed blob”: Data that is encrypted and authenticated under a particular key
 - Secure channel: Communication channel that can’t be eavesdropped on or tampered with
- Today: TLS – a secure channel
 - In network parlance, this is an “application layer” protocol but...
 - designed to have any application over it, so really “layer 6.5” is a better description

Building Secure End-to-End Channels

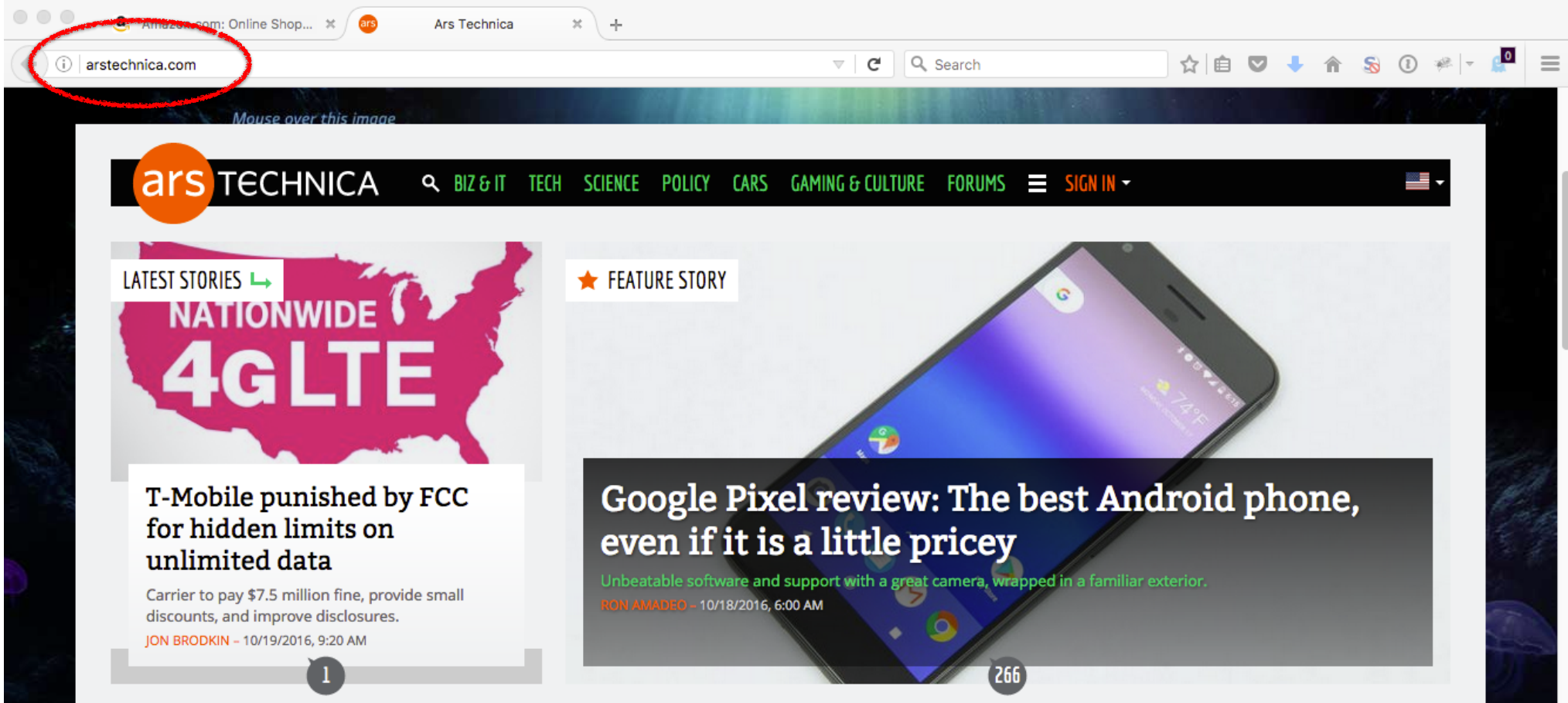
- End-to-end = communication protections achieved all the way from originating client to intended server
 - With no need to trust intermediaries
- Dealing with threats:
 - Eavesdropping?
 - Encryption (including session keys)
 - Manipulation (injection, MITM)?
 - Integrity (use of a MAC); replay protection
 - Impersonation?
 - Signatures

(What's missing?
Availability ...)

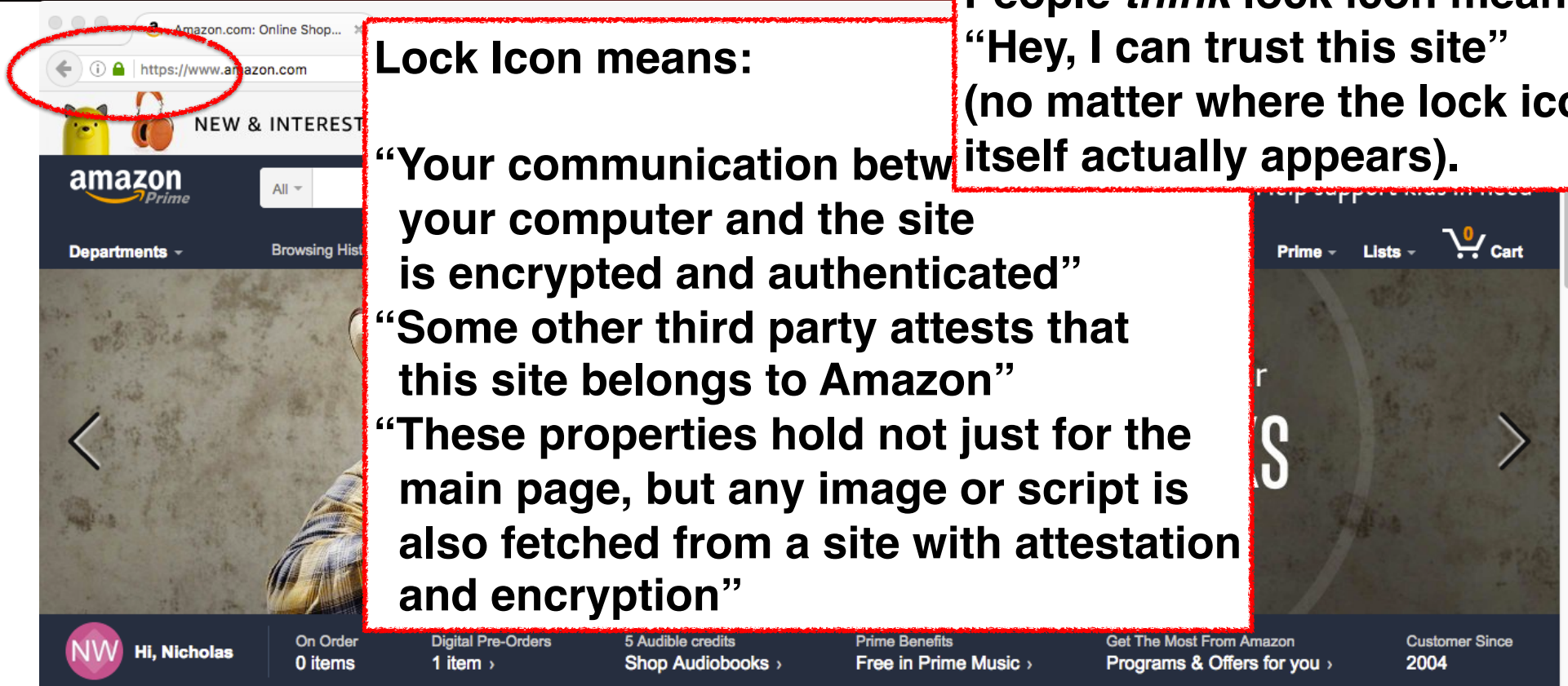
Building A Secure End-to-End Channel: SSL/TLS

- SSL = Secure Sockets Layer (predecessor)
- TLS = Transport Layer Security (standard)
 - Both terms used interchangeably
- Security for any application that uses TCP
 - Secure = encryption/confidentiality + integrity + authentication (of server, but not of client)
- Multiple uses
 - Puts the 's' in "https"
 - Secures mail sent between servers (STARTTLS)
 - Virtual Private Networks

An “Insecure” Web Page



A “Secure” Web Page

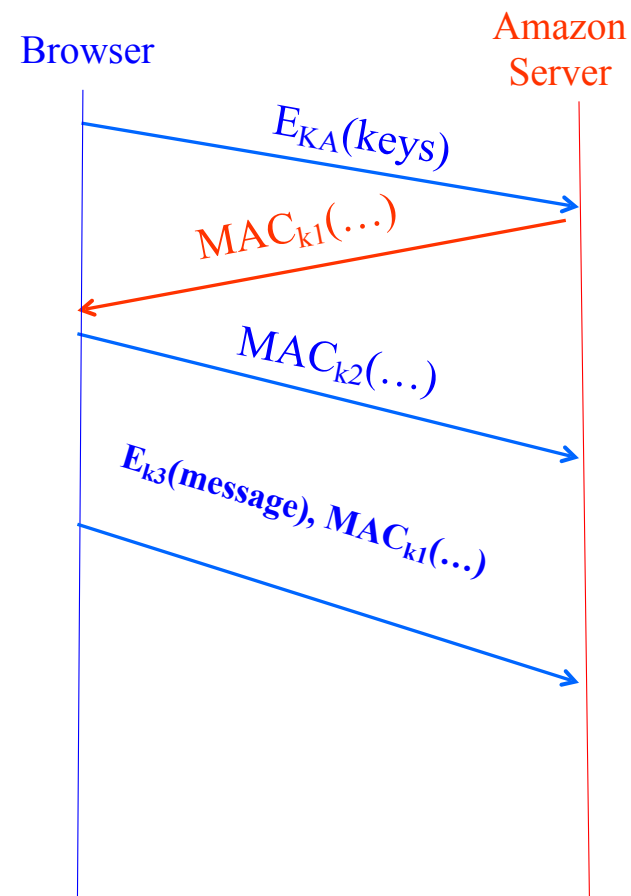


Explore AmazonFresh: Now just \$14.99/month [Learn more](#)

Amazon Gift Cards

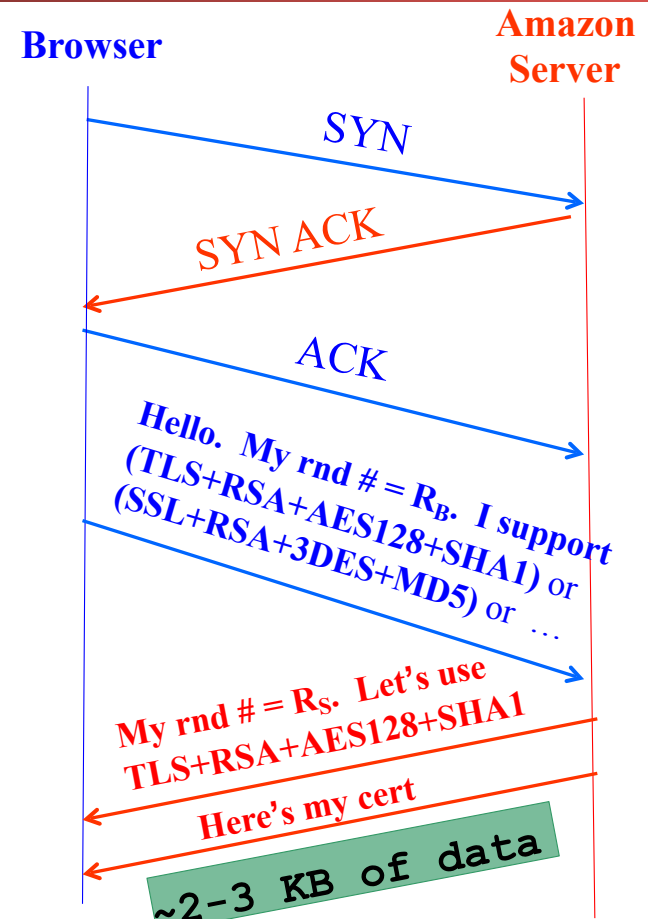
Basic idea

- Browser (client) picks some symmetric keys for encryption + authentication
- Client sends them to server, encrypted using RSA public-key encryption
- Both sides send MACs
- Now they use these keys to encrypt and authenticate all subsequent messages, using symmetric-key crypto



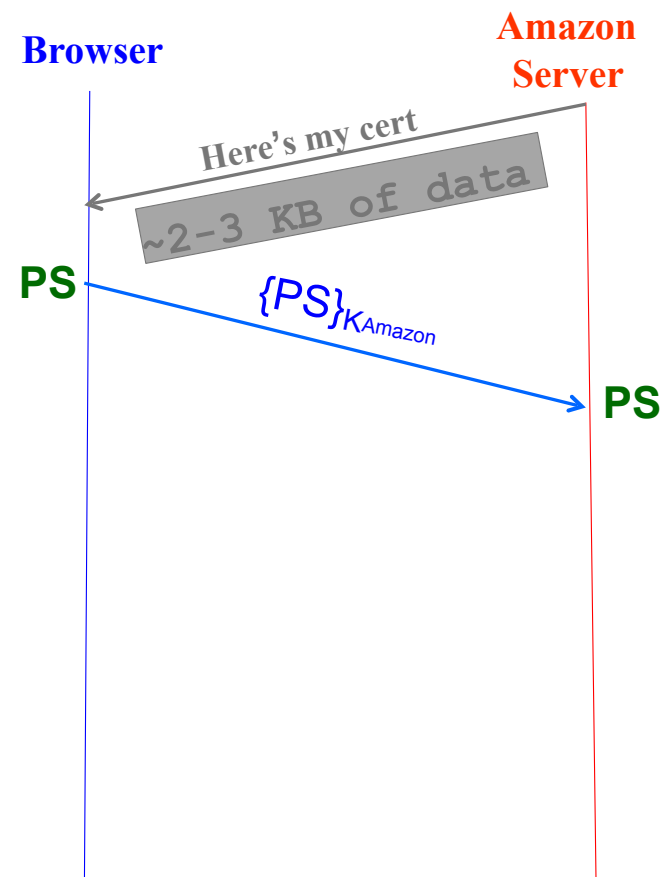
HTTPS Connection (SSL / TLS)

- Browser (client) connects via TCP to Amazon's HTTPS server
- Client picks 256-bit random number R_B , sends over list of crypto protocols it supports
- Server picks 256-bit random number R_S , selects protocols to use for this session
- Server sends over its certificate
 - (all of this is in the clear)
- Client now **validates** cert




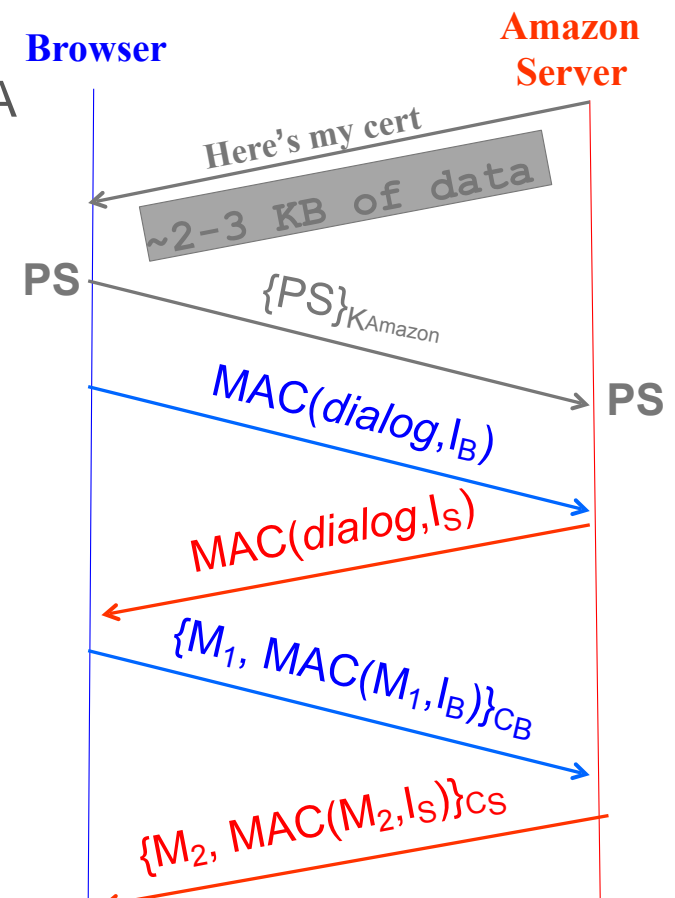
HTTPS Connection (SSL / TLS), cont.

- For RSA, browser constructs “Premaster Secret” PS
- Browser sends PS encrypted using Amazon’s public RSA key K_{Amazon}
- Using PS, R_B , and R_S , browser & server derive symmetric cipher keys (C_B , C_S) & MAC integrity keys (I_B , I_S)
 - One pair to use in each direction
 - Done by seeding a pRNG in common between the browser and the server:
Repeated calls to the pRNG then create the common keys



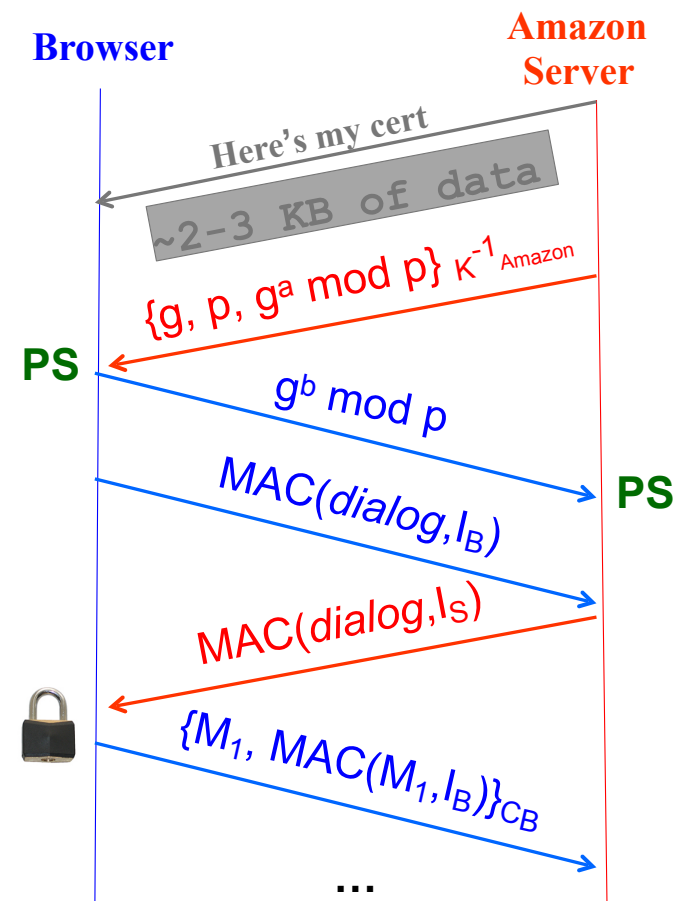
HTTPS Connection (SSL / TLS), cont.

- For RSA, browser constructs “Premaster Secret” PS
- Browser sends PS encrypted using Amazon’s public RSA key K_{Amazon}
- Using PS, R_B , and R_S , browser & server derive symm. cipher keys (C_B, C_S) & MAC integrity keys (I_B, I_S)
 - One pair to use in each direction
- Browser & server exchange MACs computed over entire dialog so far
- If good MAC, Browser displays 
- All subsequent communication encrypted w/ symmetric cipher (e.g., AES128) cipher keys, MACs
 - Sequence #'s thwart replay attacks



Alternative: Ephemeral Key Exchange via Diffie-Hellman

- For Diffie-Hellman, server generates random a , sends public parameters and $g^a \bmod p$
 - Signed with server's private key
- Browser verifies signature
- Browser generates random b , computes $PS = g^{ab} \bmod p$, sends $g^b \bmod p$ to server
- Server also computes $PS = g^{ab} \bmod p$
- Remainder is as before: from PS , R_B , and R_S , browser & server derive symm. cipher keys (C_B , C_S) and MAC integrity keys (I_B , I_S), etc...



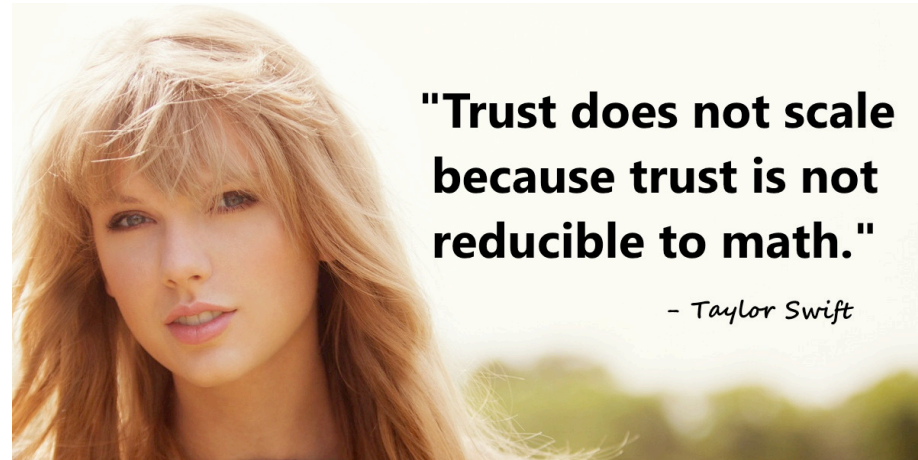
Big Changes for TLS 1.3

Diffie/Hellman and ECDHE only

- The RSA key exchange has a substantial vulnerability
 - If the attacker is ever able to compromise the server and obtain its RSA key... the attacker can decrypt any traffic captured
 - RSA lacks *forward secrecy*
- So TLS 1.3 uses DHE/ECDHE only
- TLS 1.3 also speeds things up:
 - In the client hello, the client includes $\{g^b \bmod p\}$ for preferred parameters
 - If the server finds it suitable, the server returns $\{g^a \bmod p\}$
 - Saves a round-trip time
- Also only supports AEAD mode encryptions and limited ciphersuites (e.g. GCM)

But What About that “Certificate Validation”

- Certificate validation is used to establish a chain of “trust”
- It actually is an *attempt* to build a scalable trust framework
- This is commonly known as a Public Key Infrastructure (PKI)
- Your browser is trusting the “Certificate Authority” to be responsible...



**"Trust does not scale
because trust is not
reducible to math."**

- Taylor Swift

Certificates

- Cert = signed statement about someone's public key
 - Note that a cert does not say anything about the identity of who gives you the cert
 - It simply states a given public key K_{Bob} belongs to Bob ...
 - ... and backs up this statement with a digital signature made using a different public/private key pair, say from Verisign (a "Certificate Authority")
- Bob then can prove his identity to you by you sending him something encrypted with K_{Bob} ...
 - ... which he then demonstrates he can read
 - ... or by signing something he demonstrably uses
- Works provided you trust that you have a valid copy of Verisign's public key ...
 - ... and you trust Verisign to use prudence when she signs other people's keys

Validating Amazon's Identity

- Browser compares domain name in cert w/ URL
 - Note: this provides an **end-to-end** property (as opposed to say a cert associated with an IP address)
- Browser accesses separate cert belonging to issuer
 - These are hardwired into the browser – **and trusted!**
 - There could be a chain of these ...
- Browser applies issuer's public key to verify signature **S**, obtaining the hash of what the issuer signed
 - Compares with its own SHA-1 hash of Amazon's cert
- Assuming hashes match, now have high confidence it's indeed Amazon's public key ...
 - assuming signatory is trustworthy, didn't lose private key, wasn't tricked into signing someone else's certificate, and that Amazon didn't lose their key either...

End-to-End \Rightarrow Powerful Protections

- Attacker runs a sniffer to capture our WiFi session?
 - But: encrypted communication is unreadable
 - No problem!
- DNS cache poisoning?
 - Client goes to wrong server
 - But: detects impersonation
 - No problem!
- Attacker hijacks our connection, injects new traffic
 - But: data receiver rejects it due to failed integrity check since all communication has a mac on it
 - No problem!
- Only thing a ***full man-in-the-middle*** attacker can do is inject RSTs, inject invalid packets, or drop packets: limited to a ***denial of service***

Validating Amazon's Identity, cont.

- Browser retrieves cert belonging to the issuer
 - These are hardwired into the browser – and trusted!
- But what if the browser can't find a cert for the issuer?



This Connection is Untrusted

You have asked Firefox to connect securely to **www.mikestoolbox.org**, but we can't confirm that your connection is secure.

Normally, when you try to connect securely, sites will present trusted identification to prove that you are going to the right place. However, this site's identity can't be verified.

What Should I Do?

If you usually connect to this site without problems, this error could mean that someone is trying to impersonate the site, and you shouldn't continue.

[Get me out of here!](#)

▼ Technical Details


www.mikestoolbox.org uses an

The certificate is not trusted by

(Error code: sec_error_untruste

► I Understand the Risks

Verify Certificate



Safari can't verify the identity of the website "www.mikestoolbox.org".

The certificate for this website was signed by an unknown certifying authority. You might be connecting to a website that is pretending to be "www.mikestoolbox.org", which could put your confidential information at risk. Would you like to connect to the website anyway?

[Show Certificate](#) [Cancel](#) [Continue](#)

Validating Amazon's Identity, cont.

- Browser retrieves cert belonging to the issuer
 - These are hardwired into the browser – and trusted!
- What if browser can't find a cert for the issuer?
- If it can't find the cert, then warns the user that site has not been verified
 - Can still proceed, just without authentication
- Q: Which end-to-end security properties do we lose if we incorrectly trust that the site is whom we think?
- A: All of them!
 - Goodbye confidentiality, integrity, authentication
 - Active attacker can read everything, modify, impersonate

SSL / TLS Limitations

- Properly used, SSL / TLS provides powerful end-to-end protections
- So why not use it for everything??
- Issues:
 - Cost of public-key crypto (fairly minor)
 - Takes non-trivial CPU processing (but today a minor issue)
 - Note: symmetric key crypto on modern hardware is effectively free
 - Hassle of buying/maintaining certs (fairly minor)
 - LetsEncrypt makes this almost automatic
 - Integrating with other sites that don't use HTTPS
 - Namely, you can't: Non-HTTPS content won't load!
 - Latency: extra round trips \Rightarrow 1st page slower to load

SSL / TLS Limitations, cont.

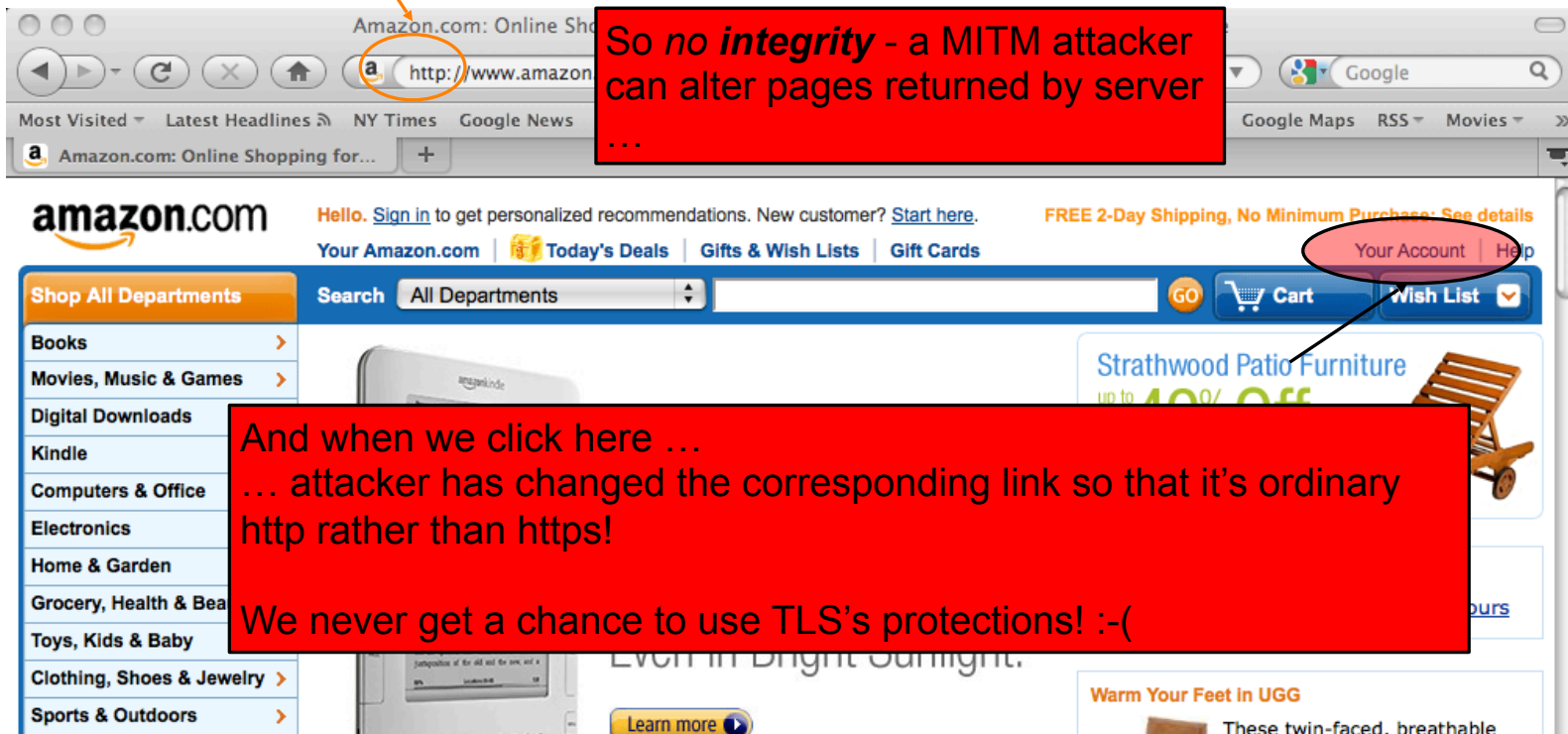
- Problems that SSL / TLS does not take care of ?
- Censorship:
 - The censor sees the certificate in the clear, so knows who the client is talking to
 - Optional Server Name Identification (SNI) is also sent in the clear
 - The censor can then inject RSTs or block the communication
- SQL injection/XSS/CSRF/server-side coding/logic flaws
- Vulnerabilities introduced by server inconsistencies

SSL/TLS Problem: Revocation

- A site screws up and an attacker steals the private key associated with a certificate, what now?
 - Certificates have a timestamp and are only good for a specified time
 - But this time is measured in years!?!?
- Two mitigations:
 - Certificate revocation lists
 - Your browser occasionally calls back to get a list of "no longer accepted" certificates
 - OSCP
 - Online Certificate Status Protocol:
https://en.wikipedia.org/wiki/Online_Certificate_Status_Protocol

“sslstrip” (Amazon FINALLY fixed this recently)

Regular web surfing: http: URL

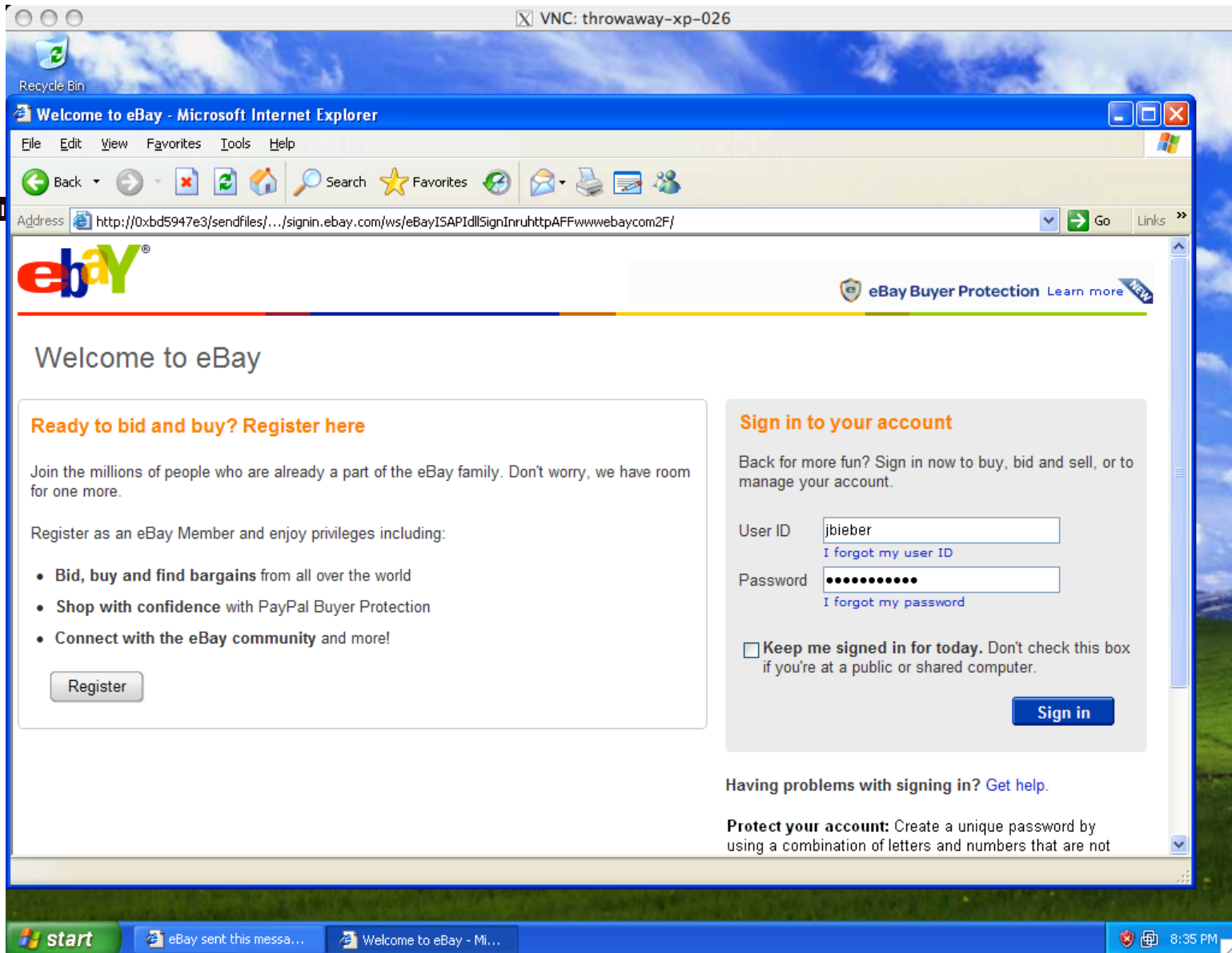


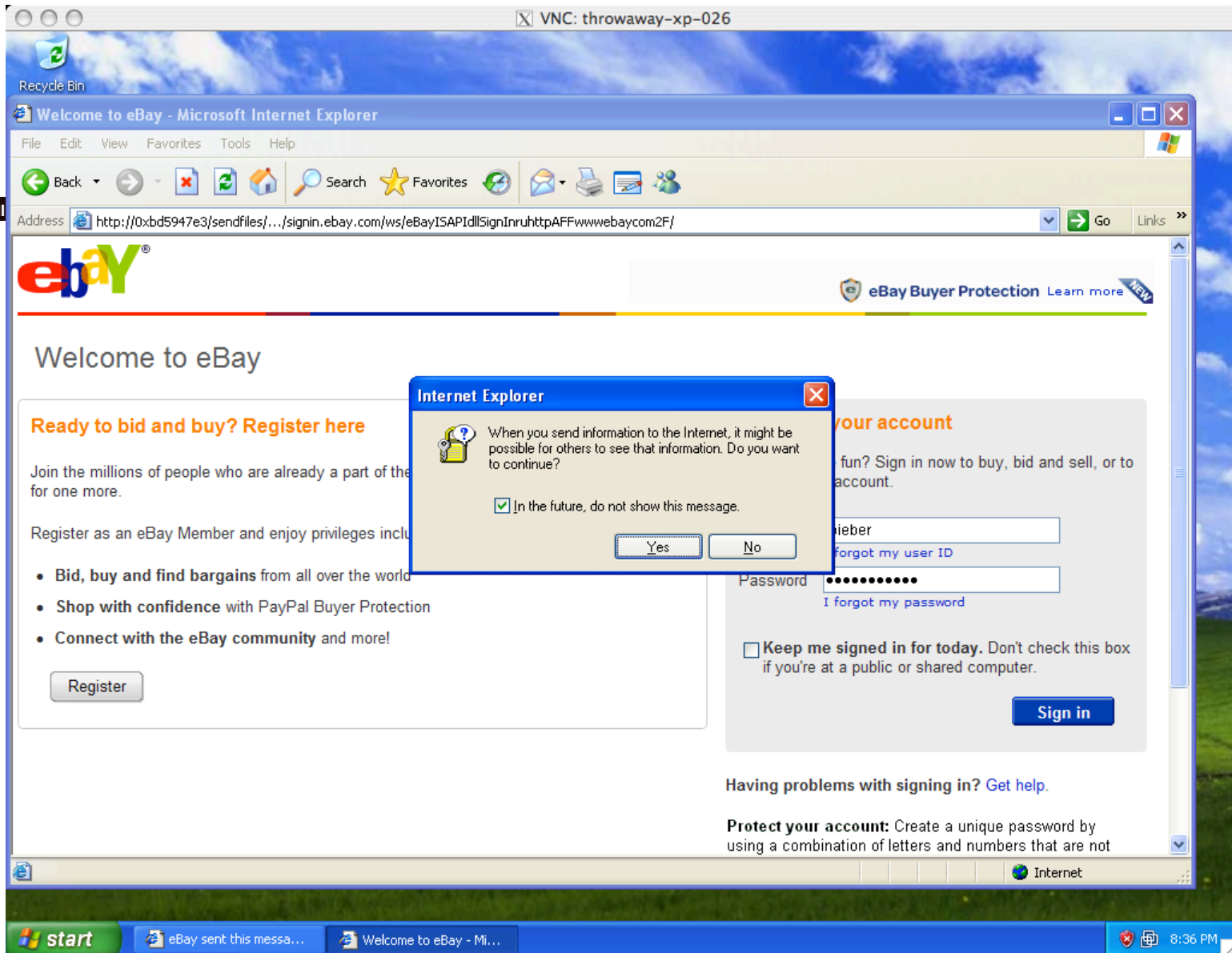
SSL / TLS Limitations, cont.

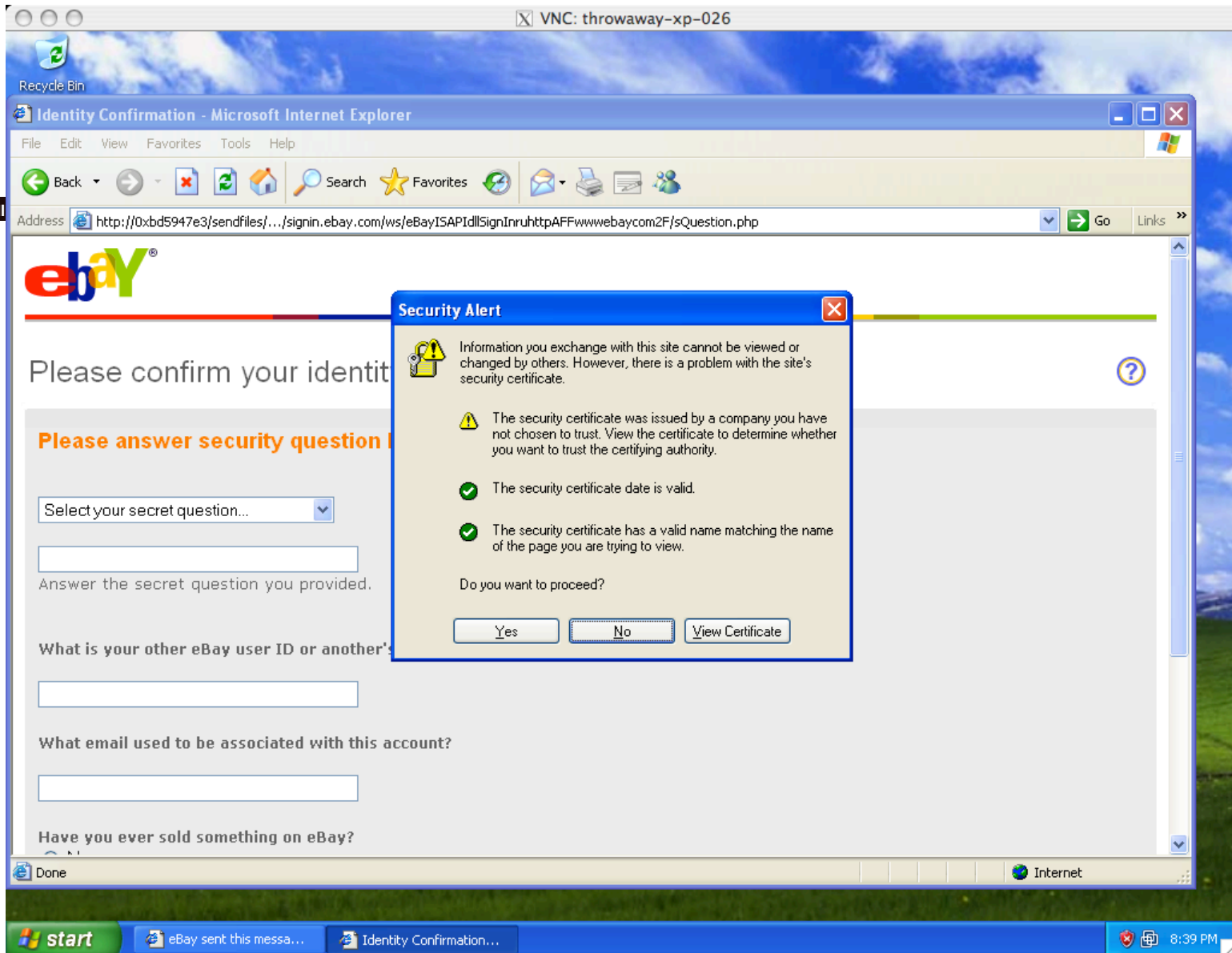
- Problems that SSL / TLS does not take care of ?
- Censorship
- SQL injection / XSS / server-side coding/logic flaws
- Vulnerabilities introduced by server inconsistencies
- Browser and server bugs
- Bad passwords
- What about the trust?

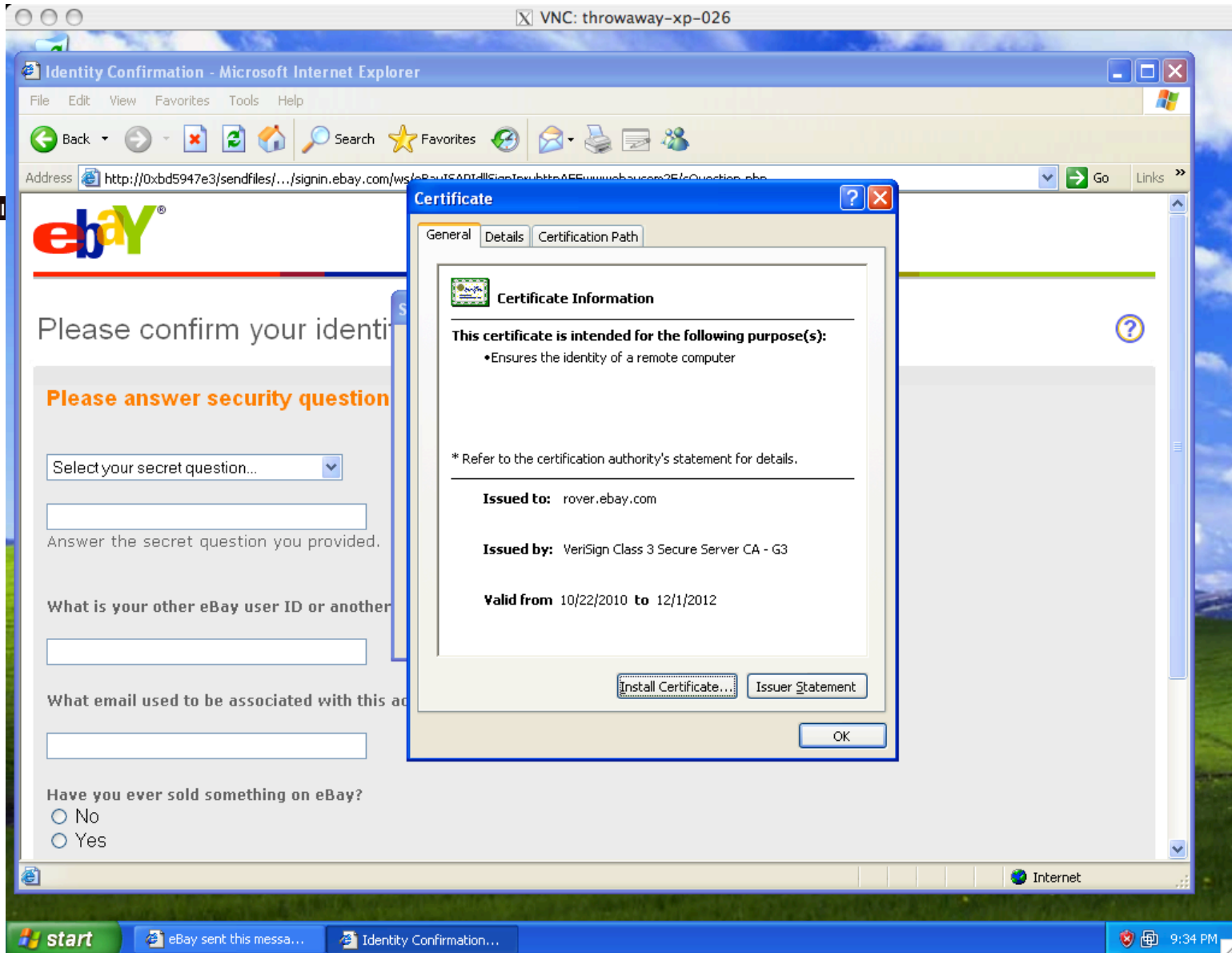
TLS/SSL Trust Issues

- User has to make correct trust decisions ...









The screenshot shows a Windows XP desktop environment. At the top, a VNC window title bar reads "VNC: throwaway-xp-026". Below it is a Microsoft Internet Explorer browser window titled "Identity Confirmation - Microsoft Internet Explorer". The address bar shows the URL: "http://0xbd5947e3/sendfiles/.../signin.ebay.com/ws/eBayISAPI.dll?SignInruhttpAFFwww.ebay.com2F/sQuestion.php". The page content includes the eBay logo and a form titled "Please confirm your identity" with the instruction "Please answer security questions". The form has several input fields and radio buttons for "No" and "Yes".

Overlaid on the browser window is a "Certificate" dialog box. It has tabs for "General", "Details", and "Certification Path". The "Details" tab is active, showing a table of certificate fields and values:

Field	Value
Version	V3
Serial number	4d ab c9 a6 0a 30 20 57 f9 23 ...
Signature algorithm	sha1RSA
Issuer	VeriSign Class 3 Secure Server...
Valid from	Friday, October 22, 2010 4:00...
Valid to	Saturday, December 01, 2012...
Subject	rover.ebay.com, Site Operatio...
Public key	RSA (1024 Bits)

At the bottom of the dialog box are buttons for "Edit Properties...", "Copy to File...", and "OK". The Windows XP taskbar at the bottom shows the Start button, taskbar buttons for "eBay sent this messa..." and "Identity Confirmation...", and a system tray with the Internet icon and the time "9:36 PM".

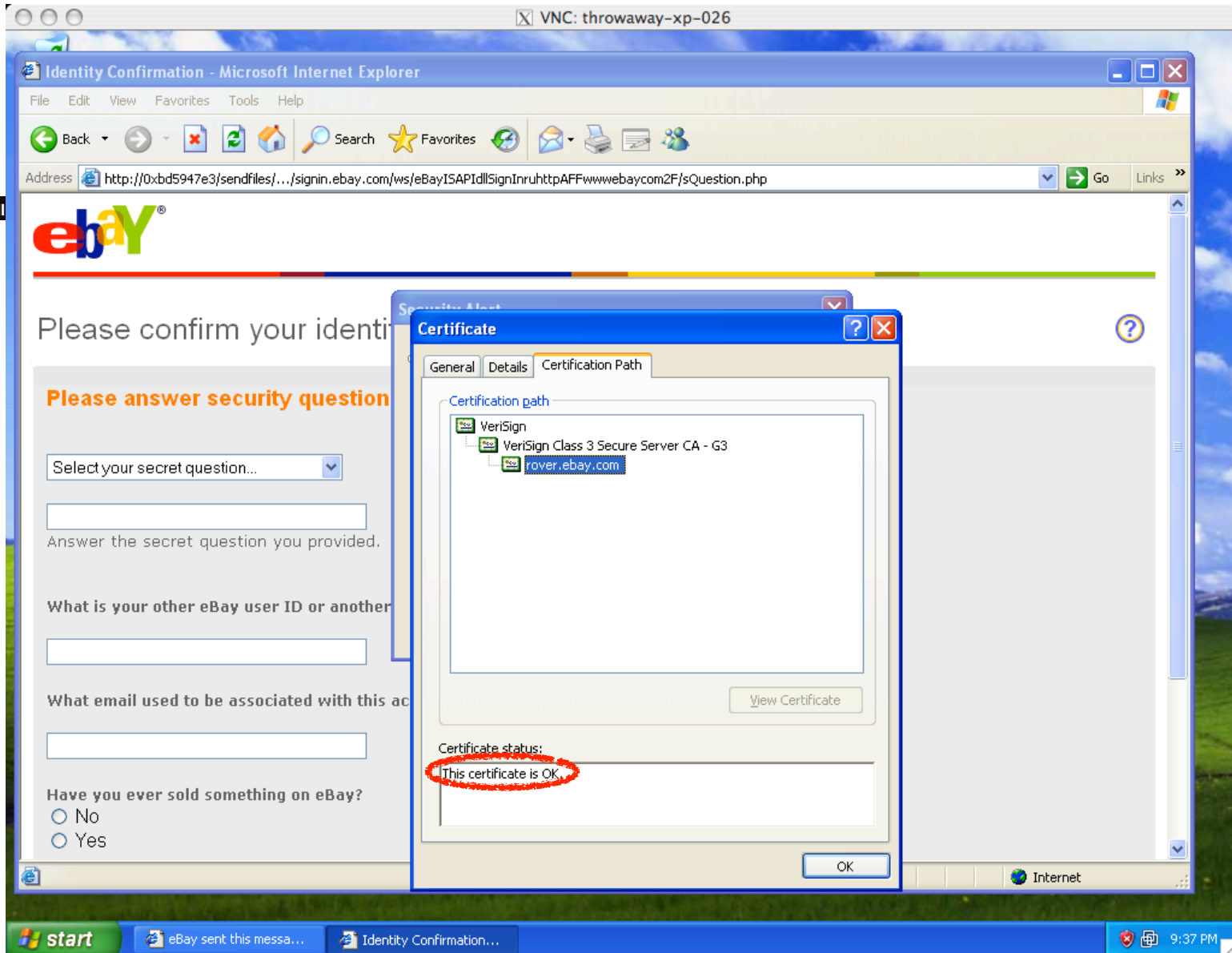
The screenshot shows a Microsoft Internet Explorer window titled "Identity Confirmation - Microsoft Internet Explorer". The address bar contains the URL: `http://0xbd5947e3/sendfiles/.../signin.ebay.com/ws/eBayISAPI.dll?SignInruhttpAFFwww.ebay.com2F/sQuestion.php`. The page content includes the eBay logo and a form titled "Please confirm your identity". The form has several sections:

- Please answer security question**: A dropdown menu labeled "Select your secret question..." and a text input field.
- Answer the secret question you provided.**: A text input field.
- What is your other eBay user ID or another**: A text input field.
- What email used to be associated with this ac**: A text input field.
- Have you ever sold something on eBay?**: Radio buttons for "No" and "Yes".

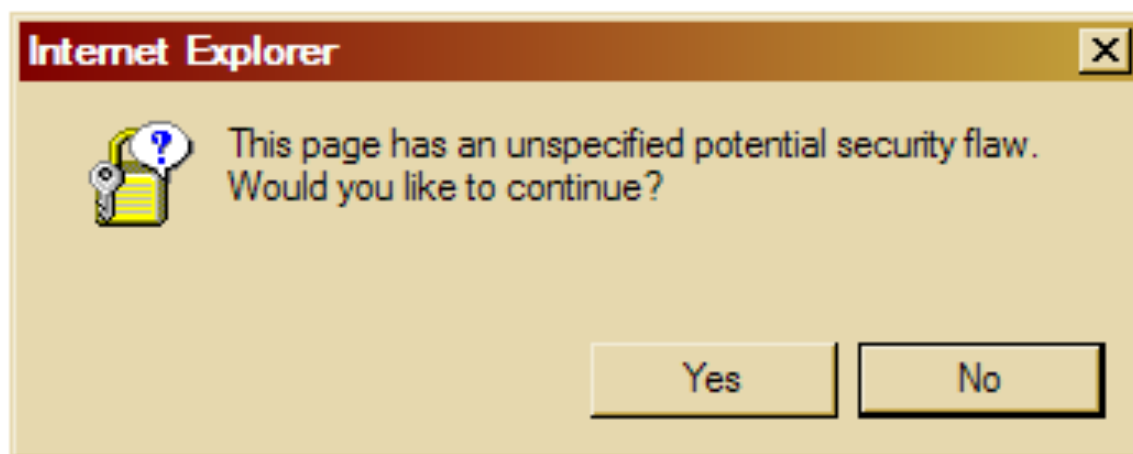
A "Certificate" dialog box is overlaid on the page, showing the "General" tab. It displays a table of certificate fields and values:

Field	Value
Subject Alternative Name	DNS Name=rover.ebay.com, ...
Basic Constraints	Subject Type=End Entity, Pat...
Key Usage	Digital Signature, Key Encipher...
CRL Distribution Points	[1]CRL Distribution Point: Distr...
Certificate Policies	[1]Certificate Policy:Policy Ide...
Enhanced Key Usage	Server Authentication (1.3.6....
Authority Key Identifier	KeyID=0d 44 5c 16 53 44 c1 8...
Authority Information Access	[1]Authority Info Access: Acc...

The dialog box also includes buttons for "Edit Properties...", "Copy to File...", and "OK". The Windows XP taskbar at the bottom shows the Start button, taskbar buttons for "eBay sent this messa..." and "Identity Confirmation...", and a system tray with the Internet icon and the time "9:36 PM".



The equivalent as seen by most Internet users:



(note: an actual Windows error message!)

TLS/SSL Trust Issues, cont.

- *“Commercial certificate authorities protect you from anyone from whom they are unwilling to take money.”*
- Matt Blaze, circa 2001
- So how many CAs do we have to worry about, anyway?

Keychain Access

Click to unlock the System Roots keychain.

Keychains

- login
- iCloud
- System
- System Roots**

Category

- All Items
- Passwords
- Secure Notes
- My Certificates
- Keys
- Certificates

AAA Certificate Services
Root certificate authority
Expires: Sunday, December 31, 2028 at 3:59:59 PM Pacific Standard Time
✔ This certificate is valid

Name	Kind	Date Modified	Expires	Keychain
AAA Certificate Services	certificate	--	Dec 31, 2028, 3:59:59 PM	System Roots
Actalis Authentication Root CA	certificate	--	Sep 22, 2030, 4:22:02 AM	System Roots
AddTrust Class 1 CA Root	certificate	--	May 30, 2020, 3:38:31 AM	System Roots
AddTrust External CA Root	certificate	--	May 30, 2020, 3:48:38 AM	System Roots
Admin-Root-CA	certificate	--	Nov 9, 2021, 11:51:07 PM	System Roots
AffirmTrust Commercial	certificate	--	Dec 31, 2030, 6:06:06 AM	System Roots
AffirmTrust Networking	certificate	--	Dec 31, 2030, 6:08:24 AM	System Roots
AffirmTrust Premium	certificate	--	Dec 31, 2040, 6:10:36 AM	System Roots
AffirmTrust Premium ECC	certificate	--	Dec 31, 2040, 6:20:24 AM	System Roots
ANF Global Root CA	certificate	--	Jun 5, 2033, 10:45:38 AM	System Roots
Apple Root CA	certificate	--	Feb 9, 2035, 1:40:36 PM	System Roots
Apple Root CA - G2	certificate	--	Apr 30, 2039, 11:10:09 AM	System Roots
Apple Root CA - G3	certificate	--	Apr 30, 2039, 11:19:06 AM	System Roots
Apple Root Certificate Authority	certificate	--	Feb 9, 2025, 4:18:14 PM	System Roots
ApplicationCA	certificate	--	Dec 12, 2017, 7:00:00 AM	System Roots
ApplicationCA2 Root	certificate	--	Mar 12, 2033, 7:00:00 AM	System Roots
Atos TrustedRoot 2011	certificate	--	Dec 31, 2030, 3:59:59 PM	System Roots
Autoridad de...nal CIF A62634068	certificate	--	Dec 31, 2030, 12:38:15 AM	System Roots
Autoridad de...Estado Venezolano	certificate	--	Dec 17, 2030, 3:59:59 PM	System Roots

168 items

TLS/SSL Trust Issues

- *“Commercial certificate authorities protect you from anyone from whom they are unwilling to take money.”*
- Matt Blaze, circa 2001
- So how many CAs do we have to worry about, anyway?
- Of course, it’s not just their greed that matters ...

News

Solo Iranian hacker takes credit for Comodo certificate attack

Security researchers split on whether 'ComodoHacker' is the real deal

By **Gregg Keizer**

March 27, 2011 08:39 PM ET

 [Comments \(5\)](#)

 [Recommended \(37\)](#)

 [Like](#) 84

Computerworld - A solo Iranian hacker on Saturday claimed responsibility for stealing multiple SSL certificates belonging to some of the Web's biggest sites, including Google, Microsoft, Skype and Yahoo.

Early reaction from security experts was mixed, with some believing the hacker's claim, while others were dubious.

Fraudulent Google certificate points to Internet attack

Weaver

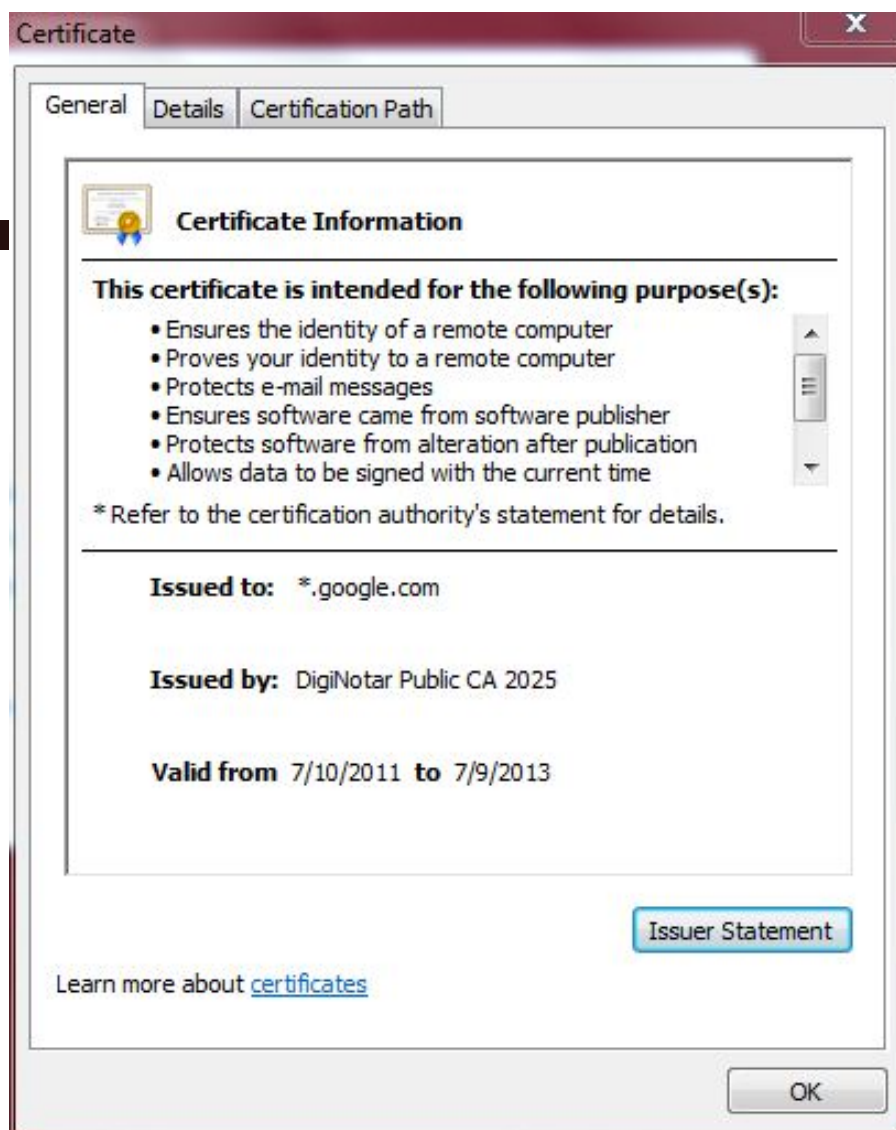
Is Iran behind a fraudulent Google.com digital certificate? The situation is similar to one that happened in March in which spoofed certificates were traced back to Iran.



by [Elinor Mills](#) | August 29, 2011 1:22 PM PDT

A Dutch company appears to have issued a digital certificate for Google.com to someone other than Google, who may be using it to try to re-direct traffic of users based in Iran.

Yesterday, someone reported on a Google support site that when attempting to log in to Gmail the browser issued a warning for the digital certificate used as proof that the site is legitimate, according to [this thread](#) on a Google support forum site.



This appears to be a fully valid cert using normal browser validation rules.

Only detected by Chrome due to its introduction of cert “pinning” – requiring that certs for certain domains must be signed by specific CAs rather than any generally trusted CA

October 31, 2012, 10:49AM

Final Report on DigiNotar Hack Shows Total Compromise of CA Servers

The attacker who penetrated the Dutch CA DigiNotar last year had complete control of all eight of the company's certificate-issuing servers during the operation and he may also have issued some rogue certificates that have not yet been identified. The final report from a

Evidence Suggests DigiNotar, Who Issued Fraudulent Google Certificate, Was Hacked *Years Ago*

from the *diginot* dept

The big news in the security world, obviously, is the fact that a **fraudulent Google certificate made its way out into the wild**, apparently targeting internet users in Iran. The Dutch company DigiNotar has put out a statement saying that **it discovered a breach** back on July 19th during a security audit, and that fraudulent certificates were generated for "several dozen" websites. The only one known to have gotten out into the wild is the Google one.

The DigiNotar Fallout

- The result was the “CA Death Sentence”:
 - Web browsers removed it from the trusted root certificate store
- This happened again with “WoSign”
 - A Chinese CA
- WoSign would allow an interesting attack
 - If I controlled `nweaver.github.com`...
 - WoSign would allow me to create a certificate for `*.github.com!?!?`
 - And a bunch of other shady shenanigans

TLS/SSL Trust Issues

- “Commercial certificate authorities protect you from anyone from whom they are unwilling to take money.”
 - Matt Blaze, circa 2001
- So how many CAs do we have to worry about, anyway?
- Of course, it’s not just their greed that matters ...
- ... and it’s not just their diligence & security that matters ...
- *“A decade ago, I observed that commercial certificate authorities protect you from anyone from whom they are unwilling to take money. That turns out to be wrong; they don't even do that much.”* - Matt Blaze, circa 2010

So the Modern Solution: Invoke Ronald Reagan, “Trust, but Verify”

- Static Certificate Pinning:
The chrome browser has a list of certificates or certificate authorities that it trusts for given sites
 - Now creating a fake certificate requires attacking a *particular* CA
- HPKP Certificate Pinning:
The web server provides hashes of certificates that should be trusted
 - This is “Leap of Faith”: The first time you assume it is honest but you will catch future changes
- Transparency mechanisms:
 - Public logs provided by certificate authorities
 - Browser extensions (EFF’s TLS observatory)
 - Backbone monitors (ICSI’s TLS notary)

And Making It Cheap: LetsEncrypt...

- Coupled to the depreciation of unencrypted HTTP...
 - Need to be able to have HTTPS be just about the same complexity...
- Idea: Make it easy to "prove" you own a web site:
 - Can you write an arbitrary cookie at an arbitrary location?
- Build ***automated*** infrastructure to do this
 - Script to create a private key
 - Generate a certificate signing request
 - PKI authority says "here's a file, put it on the server"
 - Script puts it on the server