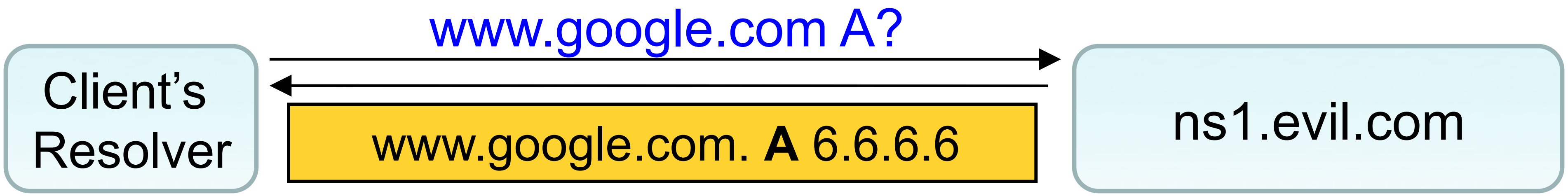
TLS and DNSSEC wrap-up

CS 161: Computer Security

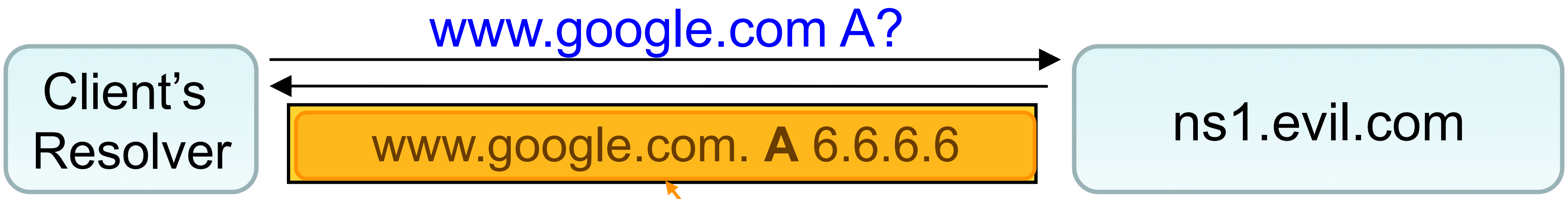
Prof. David Wagner

April 14, 2013

DNSSEC – Mallory attacks!

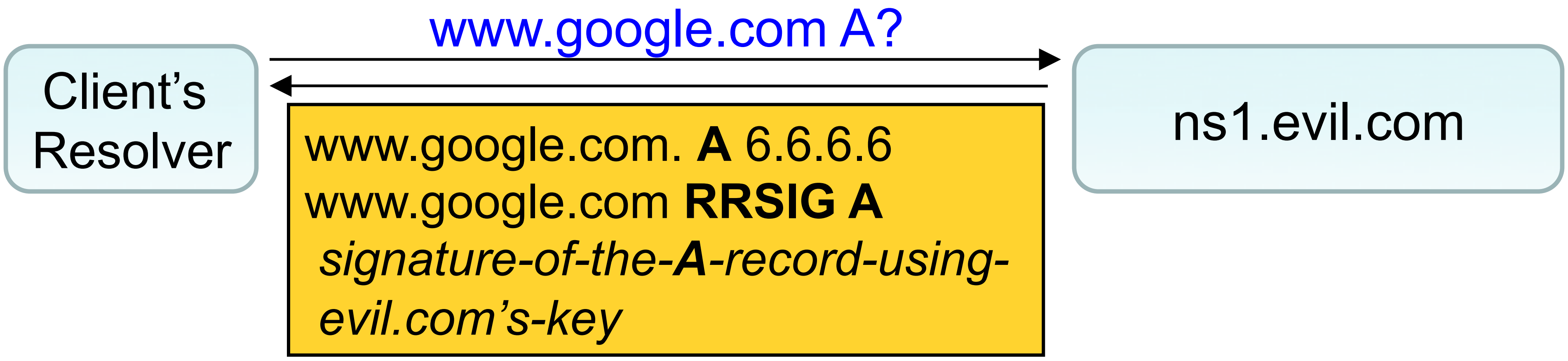


DNSSEC – Mallory attacks!



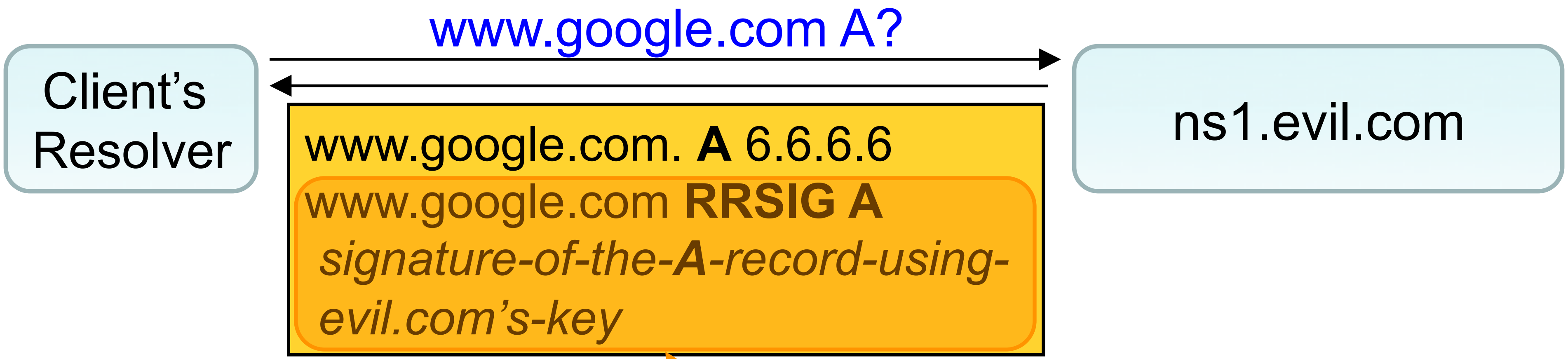
Resolver observes that the reply didn't include a signature, rejects it as insecure

DNSSEC – Mallory attacks!



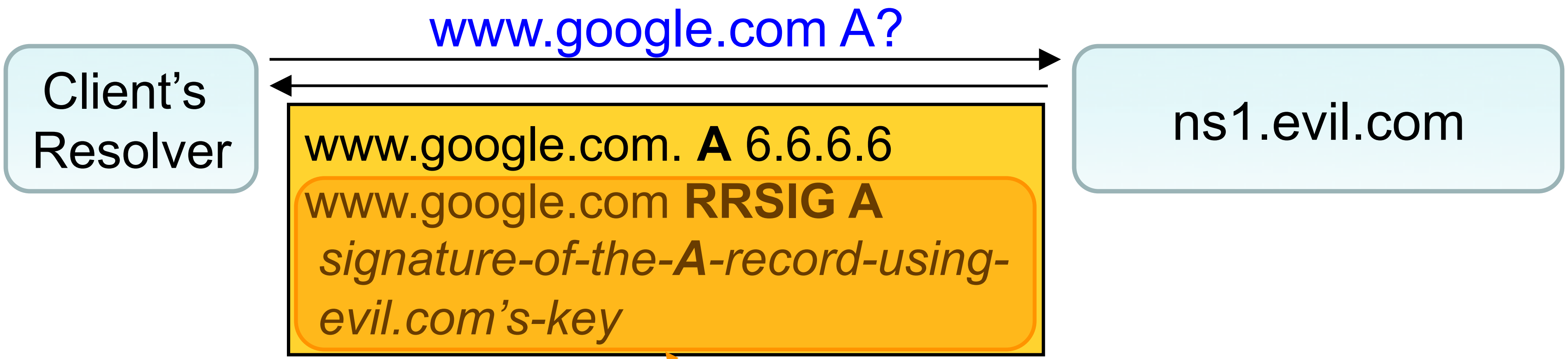
www.google.com. **A** 6.6.6.6
www.google.com **RRSIG A**
*signature-of-the-**A**-record-using-evil.com's-key*

DNSSEC – Mallory attacks!


www.google.com A?
Client's Resolver
ns1.evil.com
www.google.com. **A** 6.6.6.6
www.google.com **RRSIG A**
*signature-of-the-**A**-record-using-evil.com's-key*

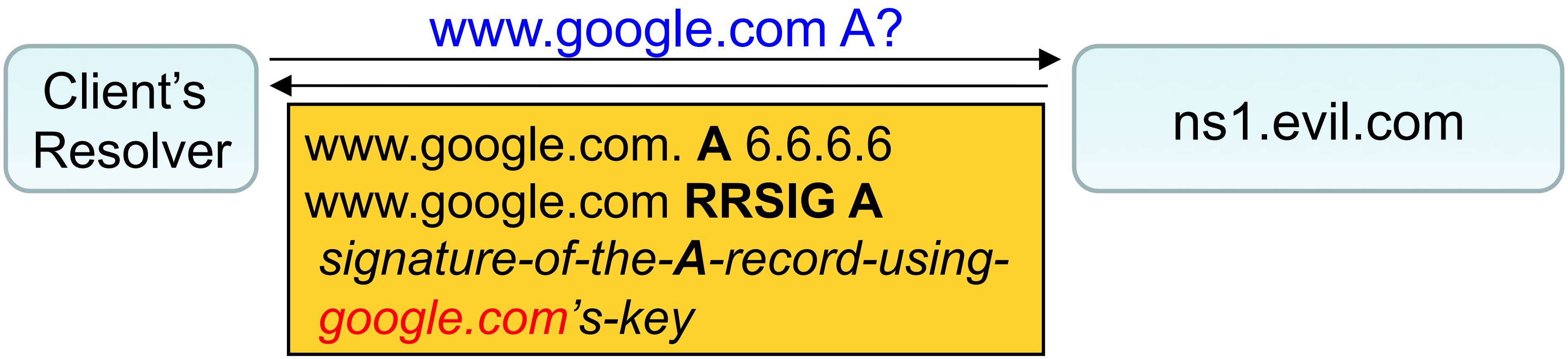
(1) If resolver didn't receive a signature from `.com` for `evil.com`'s key, then it can't validate this signature & ignores reply since it's not properly signed …

DNSSEC – Mallory attacks!

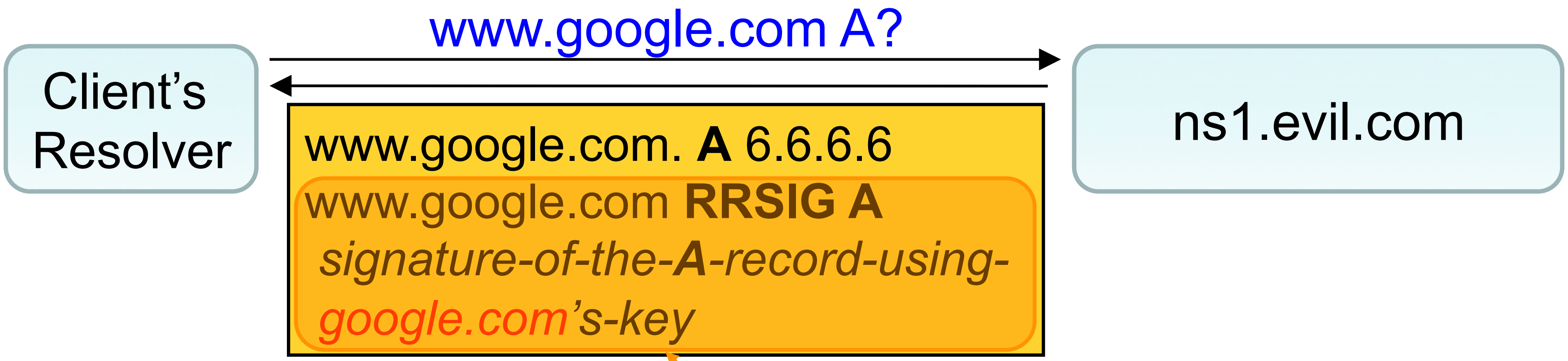

www.google.com A?
Client's Resolver
ns1.evil.com
www.google.com. **A** 6.6.6.6
www.google.com **RRSIG A**
*signature-of-the-**A**-record-using-evil.com's-key*

(2) If resolver *did* receive a signature from `.com` for `evil.com`'s key, then it knows the key is for `evil.com` and not `google.com` … and ignores it

DNSSEC – Mallory attacks!


www.google.com A?
Client's Resolver
ns1.evil.com
www.google.com. A 6.6.6.6
www.google.com RRSIG A
signature-of-the-A-record-using-google.com's-key

DNSSEC – Mallory attacks!


www.google.com A?
Client's Resolver
ns1.evil.com
www.google.com. **A** 6.6.6.6
www.google.com **RRSIG A**
*signature-of-the-**A**-record-using-google.com's-key*

If signature **actually** comes from `google.com`'s key, resolver will believe it …

… but no such signature should exist unless either:

(1) `google.com` *intended* to sign the RR, or

(2) `google.com`'s private key was compromised

Issues With DNSSEC ?

- Issue #1: Replies are Big
  - E.g., "`dig +dnssec berkeley.edu`" can return 2100+ B
  - DoS amplification
  - Increased latency on low-capacity links
  - Headaches w/ older libraries that assume replies < 512B

- Issue #2: *Partial deployment*
  - Suppose `.com` not signing, though `google.com` is
  - Major practical concern. What do we do?
  - Can wire additional key into resolver (doesn't scale)
  - Or: outsource to trusted third party ("lookaside")
    - Wire their key into resolver, they sign numerous early adopters

Issues With DNSSEC, cont.

- Issue #1: *Partial deployment*
  - Suppose `.com` not signing, though `google.com` is. Or, suppose `.com` and `google.com` are signing, but `cnn.com` isn't. Major practical concern. What do we do?
  - What do you do with unsigned/unvalidated results?
  - If you trust them, weakens incentive to upgrade (man-in-the-middle attacker can defeat security even for google.com, by sending forged but unsigned response)
  - If you don't trust them, a whole lot of things break

Issues With DNSSEC, cont.

- Issue #2: Negative results ("no such name")
  - What statement does the nameserver sign?
  - If "`gabluph.google.com`" doesn't exist, then have to do dynamic key-signing (expensive) for any bogus request
  - Instead, sign (off-line) statements about order of names
    - E.g., sign "`gabby.google.com` is followed by `gabrunk.google.com`"
    - Thus, can see that `gabluph.google.com` can't exist
  - But: now attacker can enumerate all names that exist :-(

Issues With DNSSEC, cont.

- Issue #3: *Whom do you really trust?*
  - For your laptop (say), who does all the "grunt work" of fetching keys & validating DNSSEC signatures?
- Your laptop's local resolver?
  - … which you acquire via DHCP in your local coffeeshop
  - I.e., exactly the most-feared potentially untrustworthy part of the DNS resolution process!
- Alternatives?

  ⇒ Your laptop needs to do all the validation work itself :-(

TLS/SSL Trust Issues

- "*Commercial certificate authorities protect you from anyone from whom they are unwilling to take money.*"
  - – Matt Blaze, circa 2001
- … and there are lots of CAs, and we must trust them all.
- Of course, it's not just their greed that matters …

CNET › News › InSecurity Complex › Fraudulent Google certificate points to Internet attack

Fraudulent Google certificate points to Internet attack

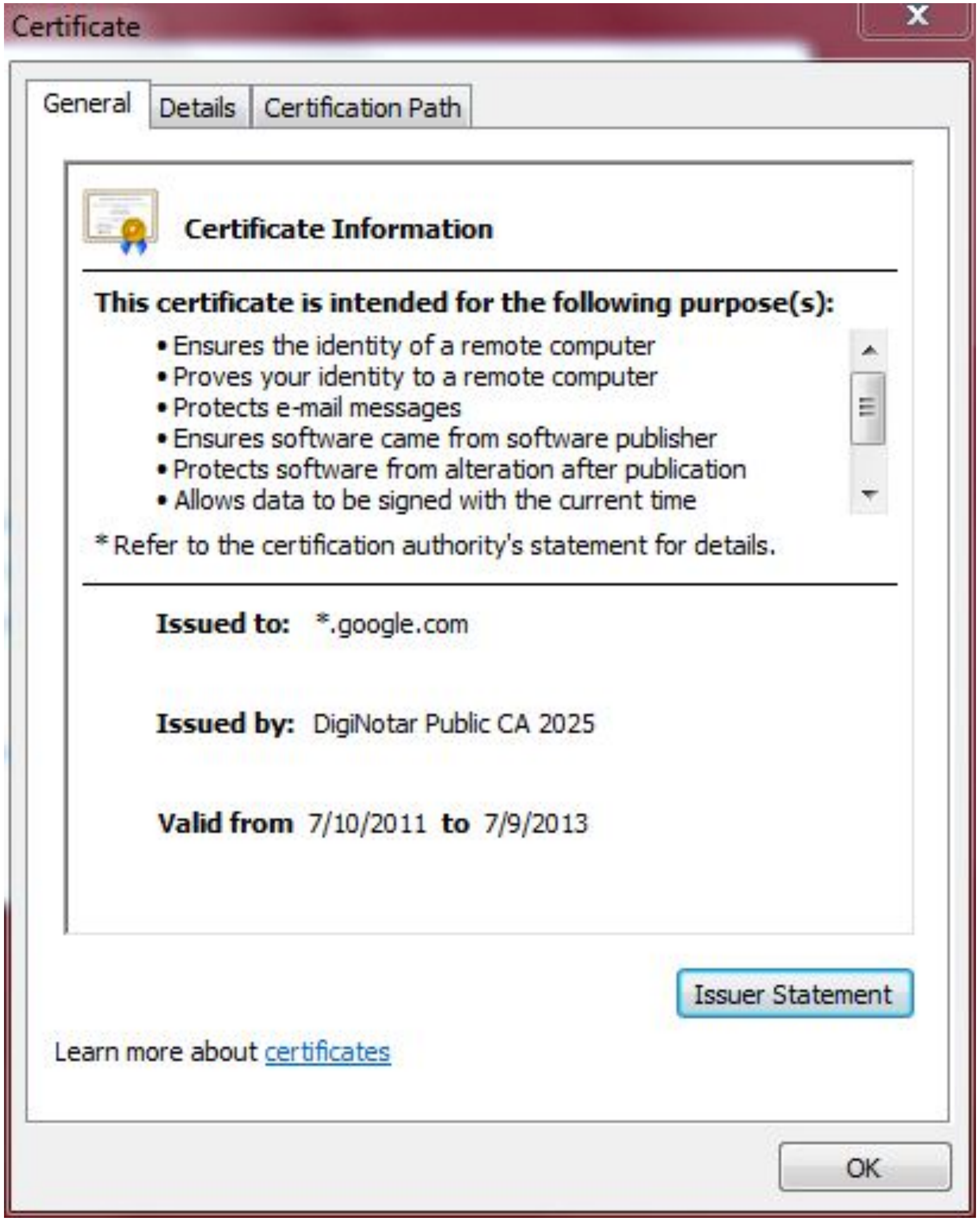
Is Iran behind a fraudulent Google.com digital certificate? The situation is similar to one that happened in March in which spoofed certificates were traced back to Iran.

by Elinor Mills | August 29, 2011 1:22 PM PDT

Follow

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.


Certificate

General | Details | Certification Path

Certificate Information

This certificate is intended for the following purpose(s):

- Ensures the identity of a remote computer
- Proves your identity to a remote computer
- Protects e-mail messages
- Ensures software came from software publisher
- Protects software from alteration after publication
- Allows data to be signed with the current time

* Refer to the certification authority's statement for details.

Issued to: *.google.com

Issued by: DigiNotar Public CA 2025

Valid from 7/10/2011 **to** 7/9/2013

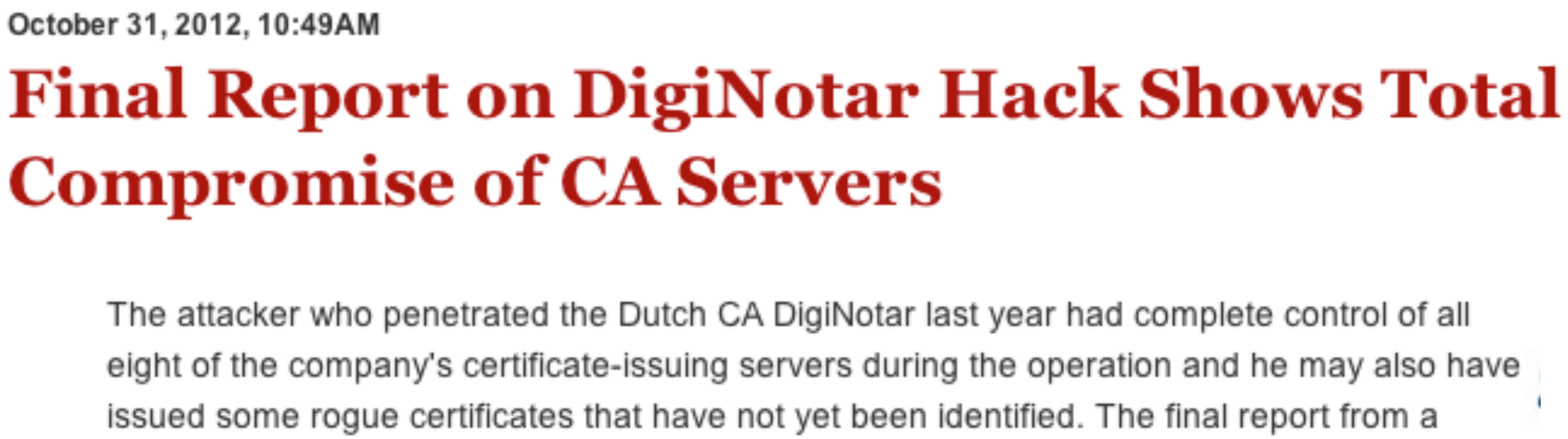
Issuer Statement

Learn more about certificates

OK

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

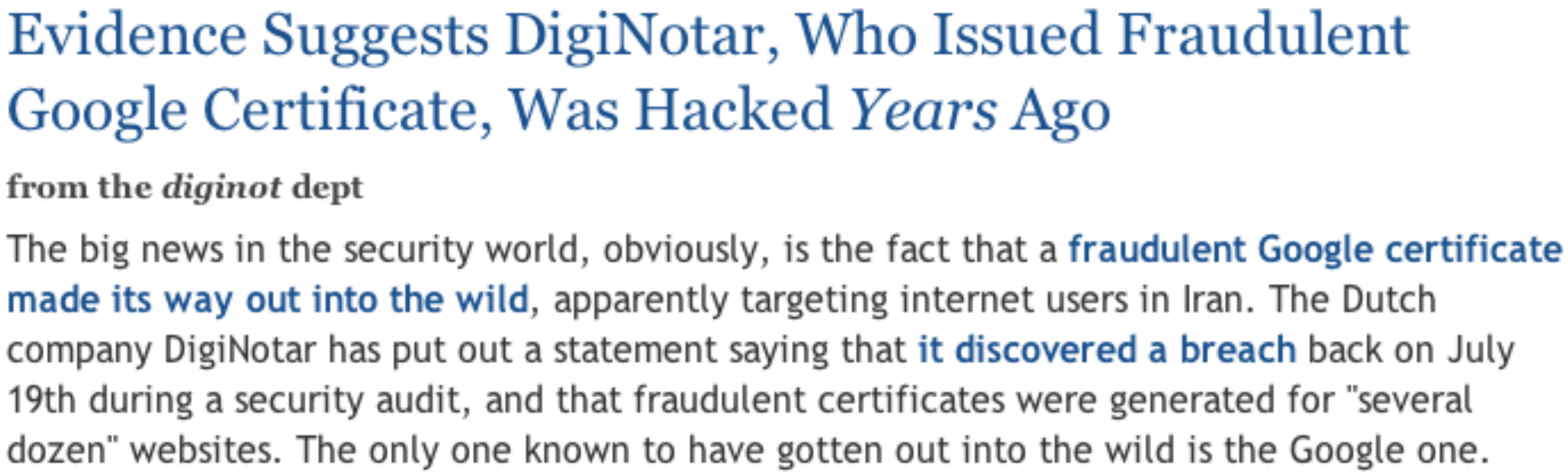
Only detected by Chrome due to its recent introduction of cert "pinning" – requiring that certs for certain domains **must** be signed by specific CAs rather than any generally trusted CA



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.

TLS/SSL Trust Issues

- "*Commercial certificate authorities protect you from anyone from whom they are unwilling to take money.*"
  - Matt Blaze, circa 2001
- … and there are lots of CAs, and we must trust them all.
- 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

Law Enforcement Appliance Subverts SSL

By Ryan Singel March 24, 2010 | 1:55 pm | Categories: Surveillance, Threats

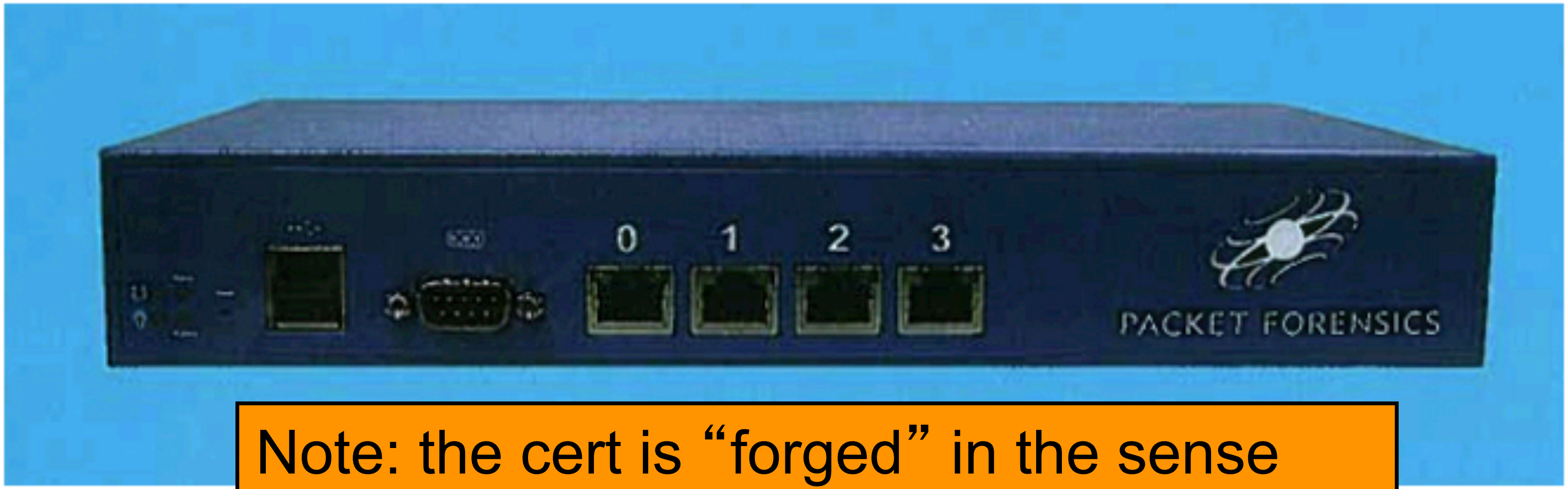
That little lock on your browser window indicating you are communicating securely with your bank or e-mail account may not always mean what you think its means.

Normally when a user visits a secure website, such as Bank of America, Gmail, PayPal or eBay, the browser examines the website's certificate to verify its authenticity.

At a recent wiretapping convention, however, security researcher Chris Soghoian discovered that a small company was marketing internet spying boxes to the feds. The boxes were designed to intercept those communications — without breaking the encryption — by using forged security certificates, instead of the real ones that websites use to verify secure connections. To use the appliance, the government would need to acquire a forged certificate from any one of more than 100 trusted Certificate Authorities.

Law Enforcement Appliance Subverts SSL

By Ryan Singel   March 24, 2010 | 1:55 pm | Categories: Surveillance, Threats

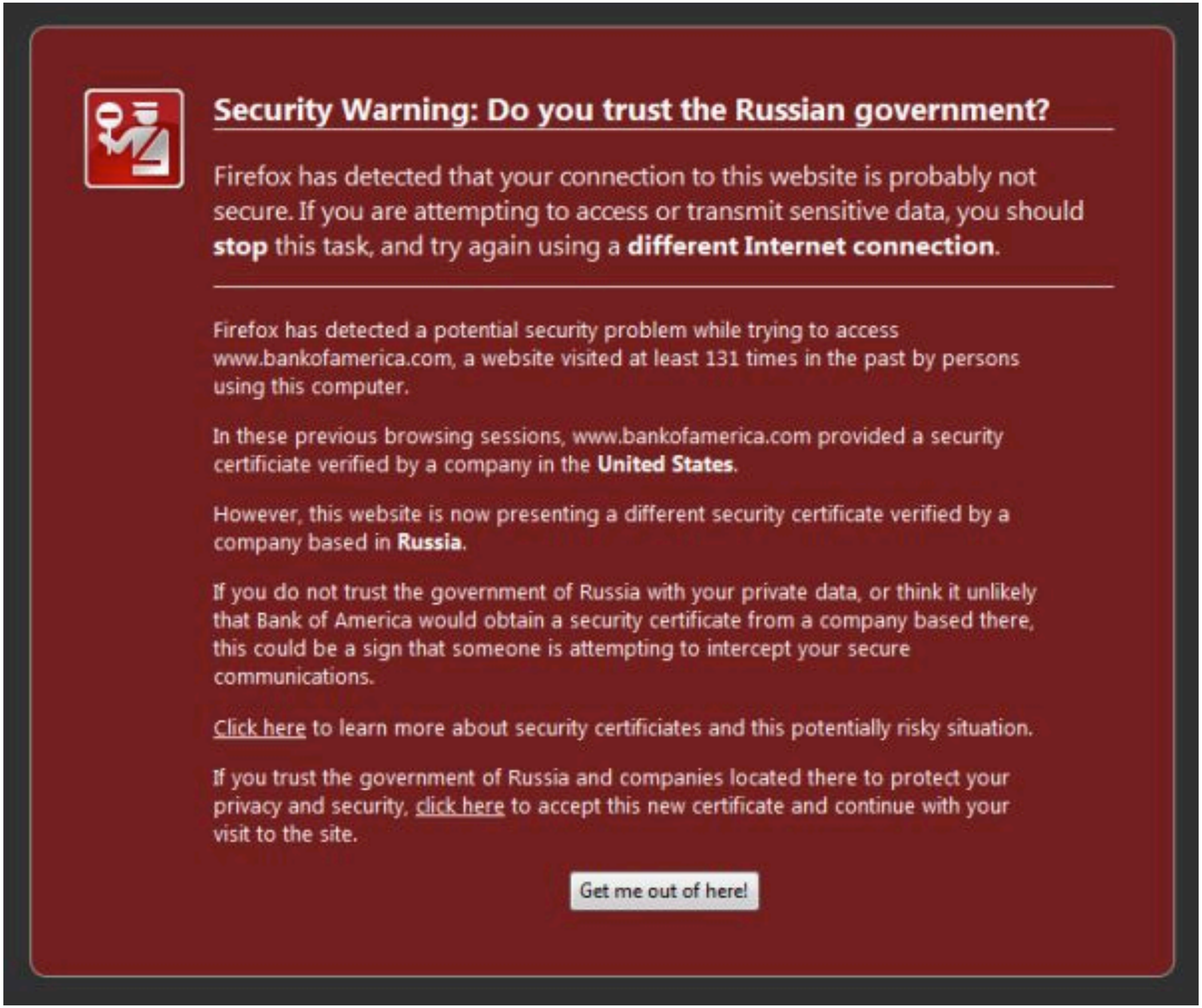


Note: the cert is "forged" in the sense that it doesn't really belong to Gmail, PayPal, or whomever. But it does not *appear* forged because it includes a legitimate signature from a trusted CA.

That little lock o ank or e-
mail account m

Normally when y, the
browser examir

At a recent wiretapping convention, however, security researcher Chris Soghoian discovered that a small company was marketing internet spying boxes to the feds. The boxes were designed to intercept those communications — without breaking the encryption — by using forged security certificates, instead of the real ones that websites use to verify secure connections. To use the appliance, the government would need to acquire a forged certificate from any one of more than 100 trusted Certificate Authorities.



Security Warning: Do you trust the Russian government?

Firefox has detected that your connection to this website is probably not secure. If you are attempting to access or transmit sensitive data, you should **stop** this task, and try again using a **different Internet connection**.

Firefox has detected a potential security problem while trying to access www.bankofamerica.com, a website visited at least 131 times in the past by persons using this computer.

In these previous browsing sessions, www.bankofamerica.com provided a security certificiate verified by a company in the **United States**.

However, this website is now presenting a different security certificate verified by a company based in **Russia**.

If you do not trust the government of Russia with your private data, or think it unlikely that Bank of America would obtain a security certificate from a company based there, this could be a sign that someone is attempting to intercept your secure communications.

Click here to learn more about security certificiates and this potentially risky situation.

If you trust the government of Russia and companies located there to protect your privacy and security, click here to accept this new certificate and continue with your visit to the site.

Summary of TLS & DNSSEC Technologies

- **TLS**: provides channel security (for communication over TCP)
  - Confidentiality, integrity, authentication
  - Client & server agree on crypto, session keys
  - Underlying security dependent on:
    - Trust in Certificate Authorities / decisions to sign keys
    - (as well as implementors)
- **DNSSEC**: provides object security (for DNS results)
  - Just integrity & authentication, not confidentiality
  - No client/server setup "dialog"
  - Tailored to be caching-friendly
  - Underlying security dependent on trust in Root Name Server's key, and all other signing keys

POLICE

TRAINING ACADEMY
TASER

Tamper-Evident Logging

- We work for the police Electronic Records office.
- To ensure that evidence can’t be questioned in court, we want to make sure that evidence can’t be tampered with, after it is logged with the office.
- In other words: a police officer can log an electronic file at any time; after it is logged, no back-dating or after-the-fact changes to evidence should be possible.
- How should we do it? What crypto or data structures could we use?

Design Problem for You

- Idea: Each day, collect all the files (f_1, f_2, …, f_n) that are logged that day. Then, publish something in the next day's newspaper, to commit to these files.
- Question: What should we publish?
  Needs to be short, and ensure after-the-fact changes or backdating are detectable.
- When a file f_i is exhibited into evidence in a trial, how can judge verify it hasn't been modified post-facto?

Your Solution

- Store in database: f_1, Sign(f_1), f_2, Sign(f_2), …, f_n, Sign(f_n)
- Publish: public key
- To verify f_i: reveal f_1, Sign(f_i)

- Critique: Sysadmin can get a copy of the private key, modify database, update the signature, and thus modify old entries or create new backdated entries.

Your Solution

- Publish: $H(f_1, f_2, \ldots, f_n)$
- To verify f_i: reveal $f_1, f_2, \ldots, f_n$

Solution

- Each day, collect all the files (f_1, f_2, …, f_n) that are logged that day. Then, publish $H(f_1, f_2, …, f_n)$ in the next day’s newspaper, to commit to these files.
- When a file f_i is exhibited into evidence in a trial, reveal f_1, f_2, …, f_n to judge. Judge can hash them, check that their hash was in the right day’s newspaper, and check that f_i is in the list.

Better Solution

- Each day, collect all the files (f_1, f_2, …, f_n) that are logged that day. Let f_0 be the previous day's hash. Publish $H(f_0, f_1, f_2, …, f_n)$ in the next day's newspaper, to commit to these files.
- Note that exhibiting file f_i into evidence still requires revealing entire list of other files (f_1, f_2, …, f_n) that were logged that day. Can you think of any way to avoid that?

Take-away

- Using hash chaining, we can provide tamper-evident audit logs that let us detect after-the-fact modifications and backdated entries.