Dan Kaminsky

PHREEBIRD SUITE 1.0: INTRODUCING THE DOMAIN KEY INFRASTRUCTURE

The Vagaries Of Talking Defense

- Security Conferences are normally about discussing offense
  - Necessary: We spent the 90's thinking all security could be accomplished with Crypto™ and Java™
  - Understanding the full scope of vulnerabilities is critical if we're to fix anything
  - However…

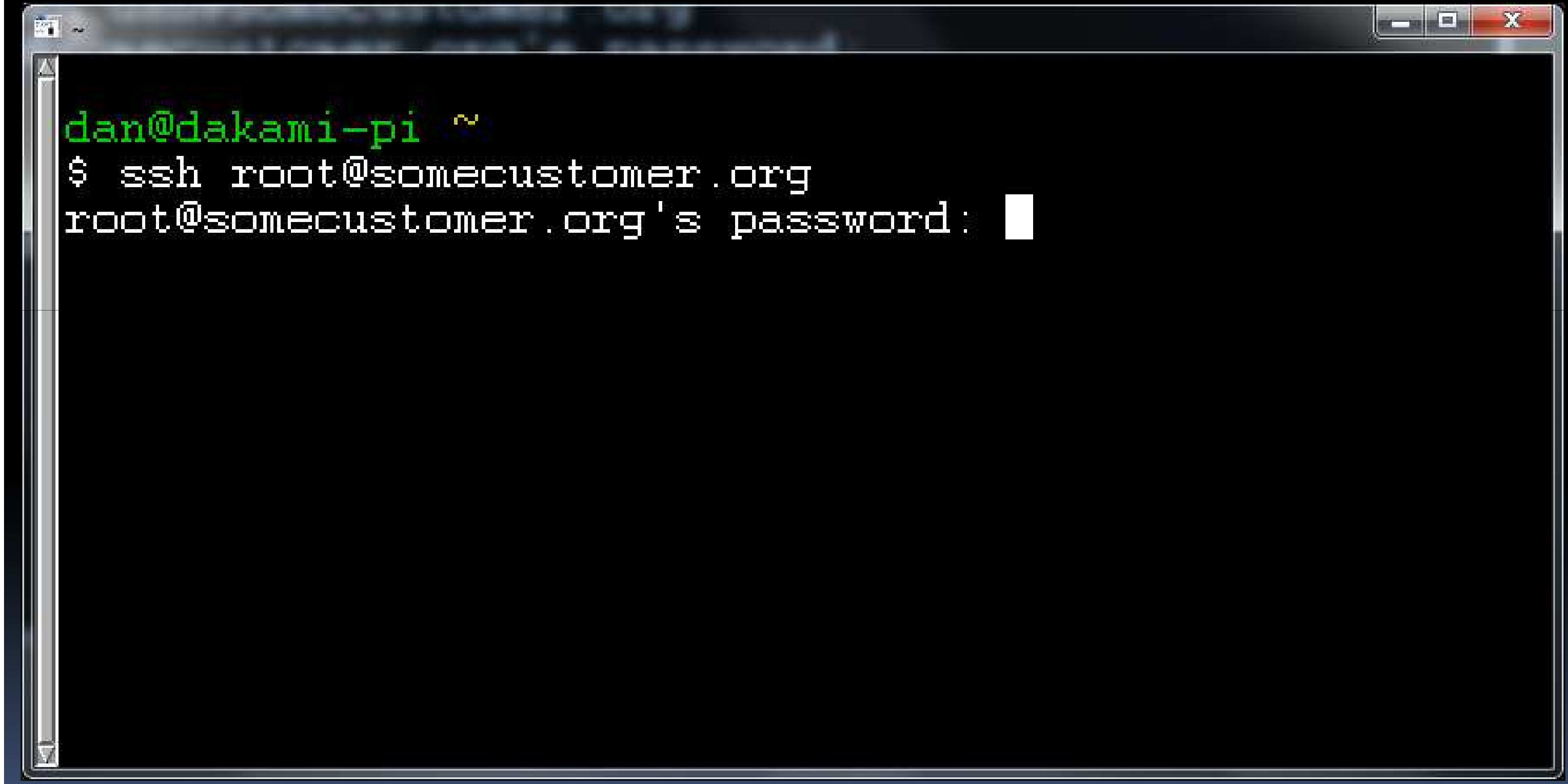
We actually need to fix things every once in a while

- We're finding lots and lots of new bugs
- The old bugs aren't actually going away
- We have a choice
  - Either keep finding the same problems for the next ten years
  - Change the ground rules

My New Rule

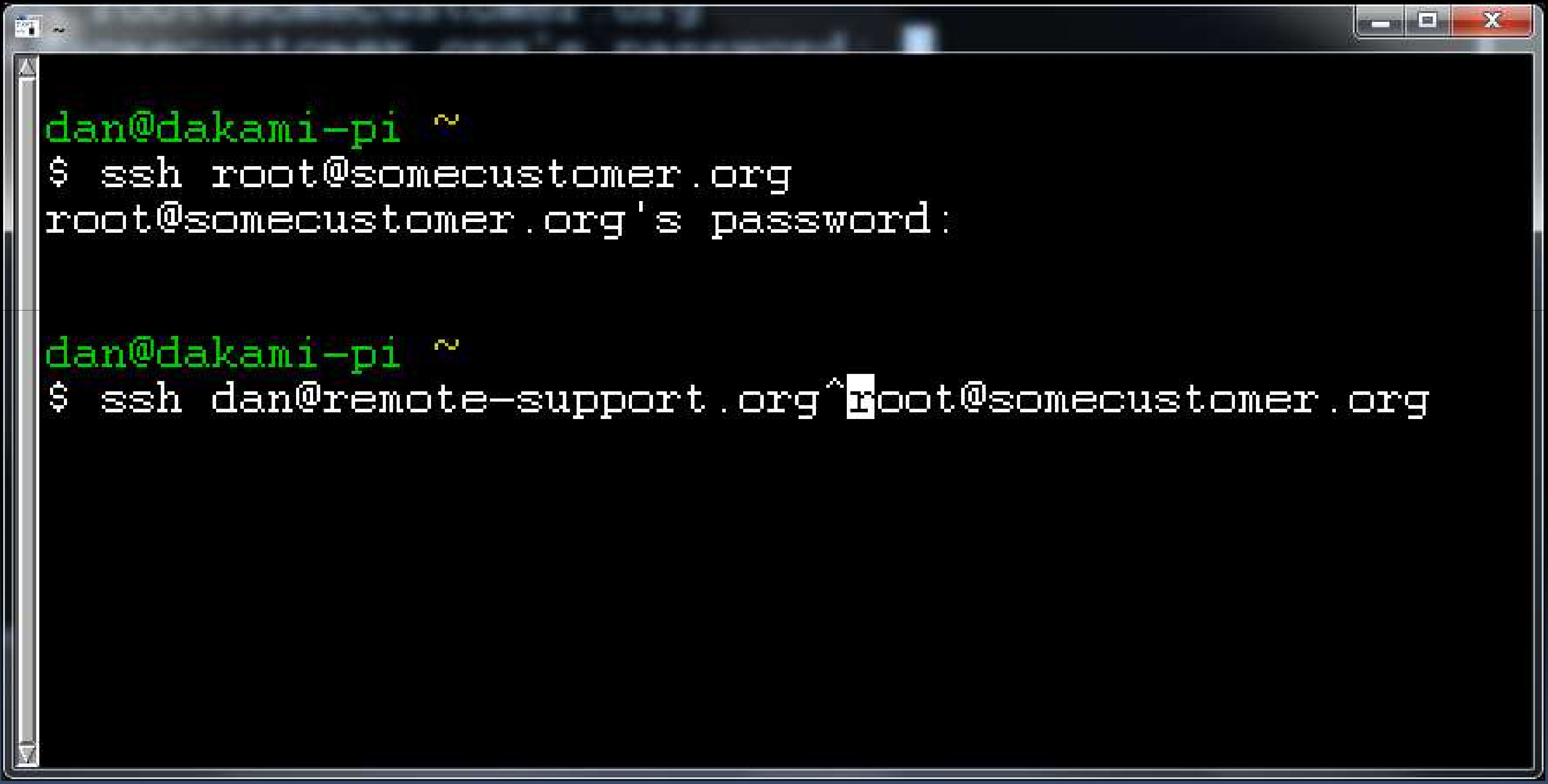
- It’s not enough to make security better
- We have to make security <u>cheaper</u>
  - Security needs to drop in cost by two orders of magnitude (100x)
  - That’s OK – security needs to increase in effectiveness by five orders of magnitude (10,000x)
    - We’re building our economies on this foundation
    - It doesn’t feel particularly solid, does it?
  - Hard Truth: We are competing with insecure systems
    - They’re (sometimes/appear) cheaper than getting owned
    - But they’re not exactly cheap right now
    - **We can do a better job if we care to!**

A Very Common Scene



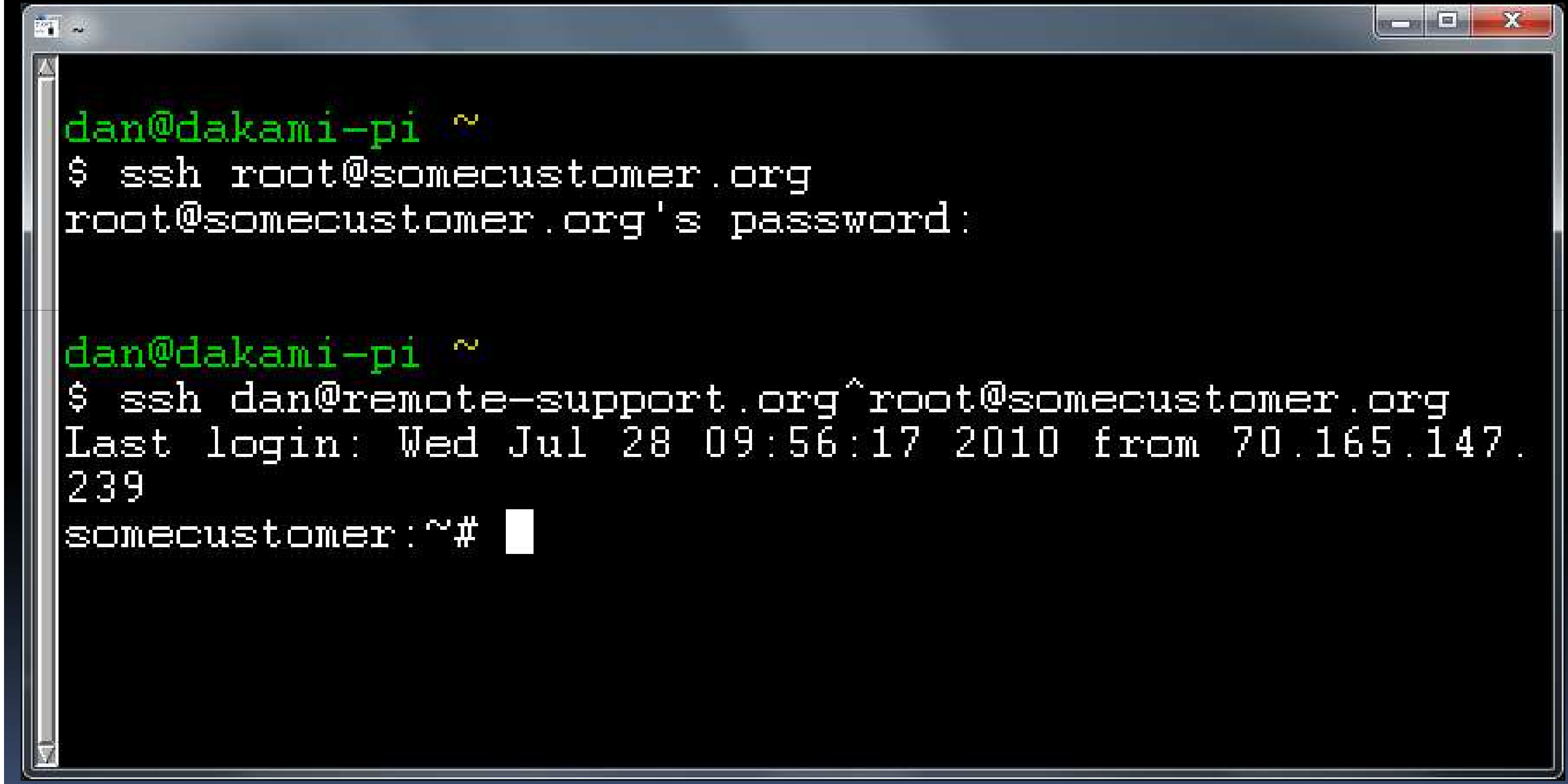
```
dan@dakami-pi ~
$ ssh root@somecustomer.org
root@somecustomer.org's password:
```

[That's A Strange Username…]



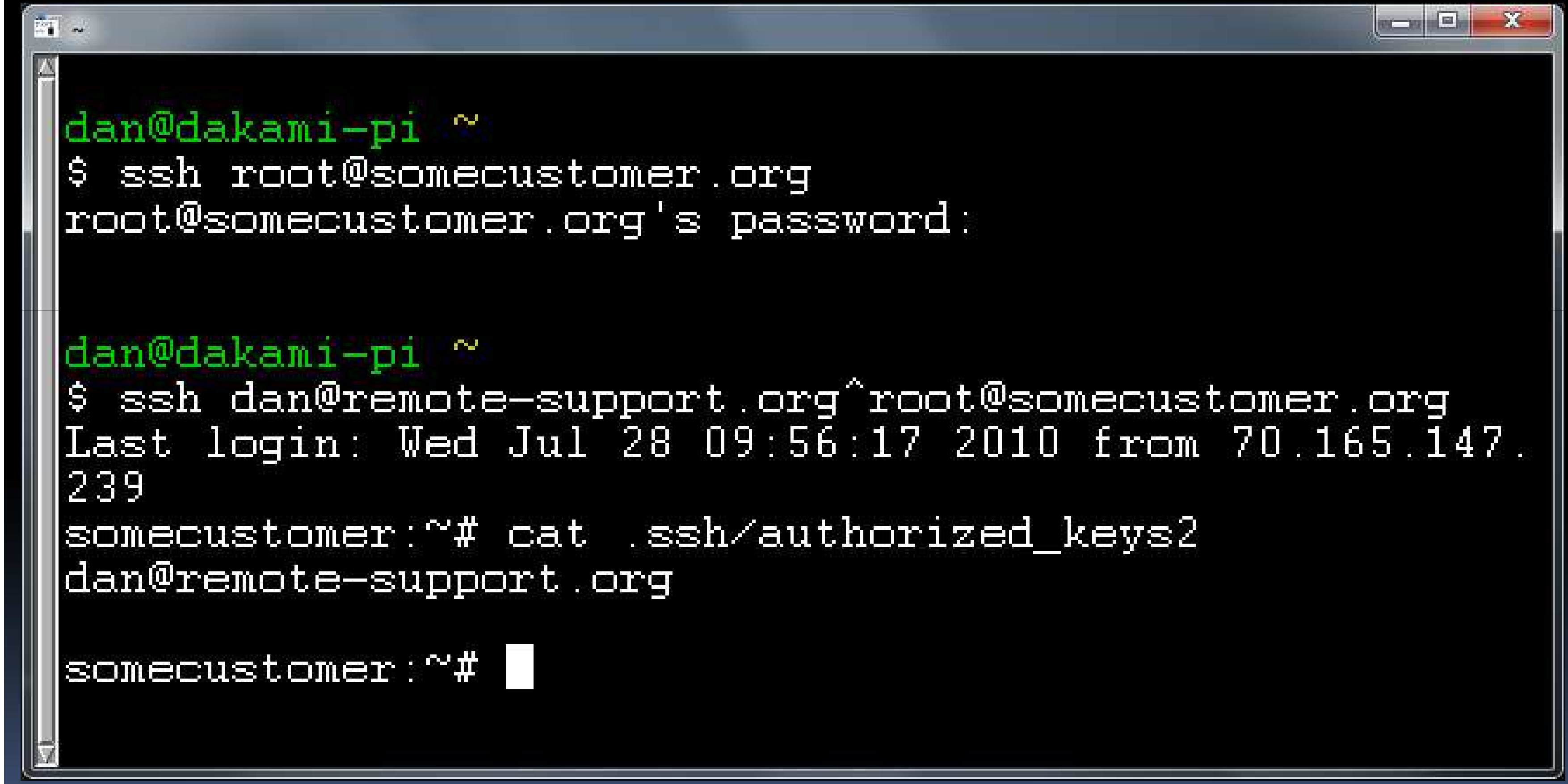
```
dan@dakami-pi ~
$ ssh root@somecustomer.org
root@somecustomer.org's password:


dan@dakami-pi ~
$ ssh dan@remote-support.org^root@somecustomer.org
```

[Heh Wait, That Worked?]



```
dan@dakami-pi ~
$ ssh root@somecustomer.org
root@somecustomer.org's password:


dan@dakami-pi ~
$ ssh dan@remote-support.org^root@somecustomer.org
Last login: Wed Jul 28 09:56:17 2010 from 70.165.147.
239
somecustomer:~# 
```

[What’s that in authorized_keys2?!]



```
dan@dakami-pi ~
$ ssh root@somecustomer.org
root@somecustomer.org's password:


dan@dakami-pi ~
$ ssh dan@remote-support.org^root@somecustomer.org
Last login: Wed Jul 28 09:56:17 2010 from 70.165.147.
239
somecustomer:~# cat .ssh/authorized_keys2
dan@remote-support.org

somecustomer:~#
```

Redefining The Possible

- We’ve been trying to authenticate (federate) from one domain to another for years
  - DNSSEC makes it easy.
  - This is the power of the Domain Key Infrastructure
- We can’t do this if DNSSEC is hard to deploy
  - So is it possible to make DNSSEC easy?
  - Yes.

What I’m Releasing

- Phreebird Suite 1.0
  - Demonstration Toolkit for DNSSEC
  - Phreebird: Zero Configuration DNSSEC Server
  - Phreeload: Automatic DNSSEC Integration Engine
  - Sample Code for End-To-End DNSSEC Integration
    - Including the Phreeshell Federated Identity Code
  - BSD Licensed – Lets get working on apps
- Why?

Introduction

- While we continue to fight the war against implementation flaws…
- …authentication continues to haunt us
  - Verizon Business: Majority of compromises linked to credential failure
- Authentication, by in large, remains synonymous with one technology:
  PASSWORDS
  - No passwords
  - Default passwords
  - Shared passwords
  - Stolen passwords

Why?

- They work.
  - More importantly, they work cross organizationally
  - “Anyone who thinks a large company is one organization, has never worked at a large company”
  - Passwords are based on strings of text – we’ve figured out how to make them cross boundaries

The Problem

- From Workgroups, to Domains, to Forests, the model is based on an internal hierarchy, where authentication for outsiders is a special case
  - Workgroups beget Domains (up)
  - Domains beget Forests (up)
  - Forests beget manually established Federations / Cross Forest Trusts (out)
- However, **authenticating outsiders is not actually a special case**

Reality

- Groups Outside Your Hierarchy
  - Clients
  - Customers
  - Vendors
  - Partners
  - Contractors
  - Outsourcers
  - Governments (and not necessarily your own)
- A lot of people still need to be authenticated
  - Couldn’t there be a hierarchy that sits above all of them?

The Three (Bad) Choices

- M-to-1: "Everybody Trust Me!"
  - Keeps getting tried
  - Almost always leads to the guy in charge seeking rents
  - **Always** leads to guys not in charge trying to get in charge, so they can seek rents
- M-to-Any: "Everybody Trust The Cabal"
  - Basis of X.509 CAs
  - Always leads to too many people in the Cabal for any serious trust
- M-to-N: "Everybody, Figure Out Who You Trust"
  - Every new major group has to be manually brought into the Federation
  - Doesn't scale

The One Good Choice

- DNS (newly enhanced with DNSSEC)
  - Starts out a M-to-1 system, but…
    - Politically limited – massive governance on ICANN
      - It's so hard to get even *legitimate* content into the root, that imagine getting bad content in
    - Technically limited
      - The root hosts such a small subset of the final data, that it's a weak point to attack anyway
  - Huge install base
    - All customers, vendors, partners, contractors, etc are already in it – they're receiving emails, aren't they?
  - Already trusted
    - DNS is *how* they're receiving emails

DNS is actually pretty simple

- DNS:
  - Ask a question, get an answer
  - Ask a question, get a referral
    - Alice: Jenny's number? Ask Travis.
    - Travis: Jenny's number? Ask Charlie.
    - Charlie: Jenny's number? 876-5309

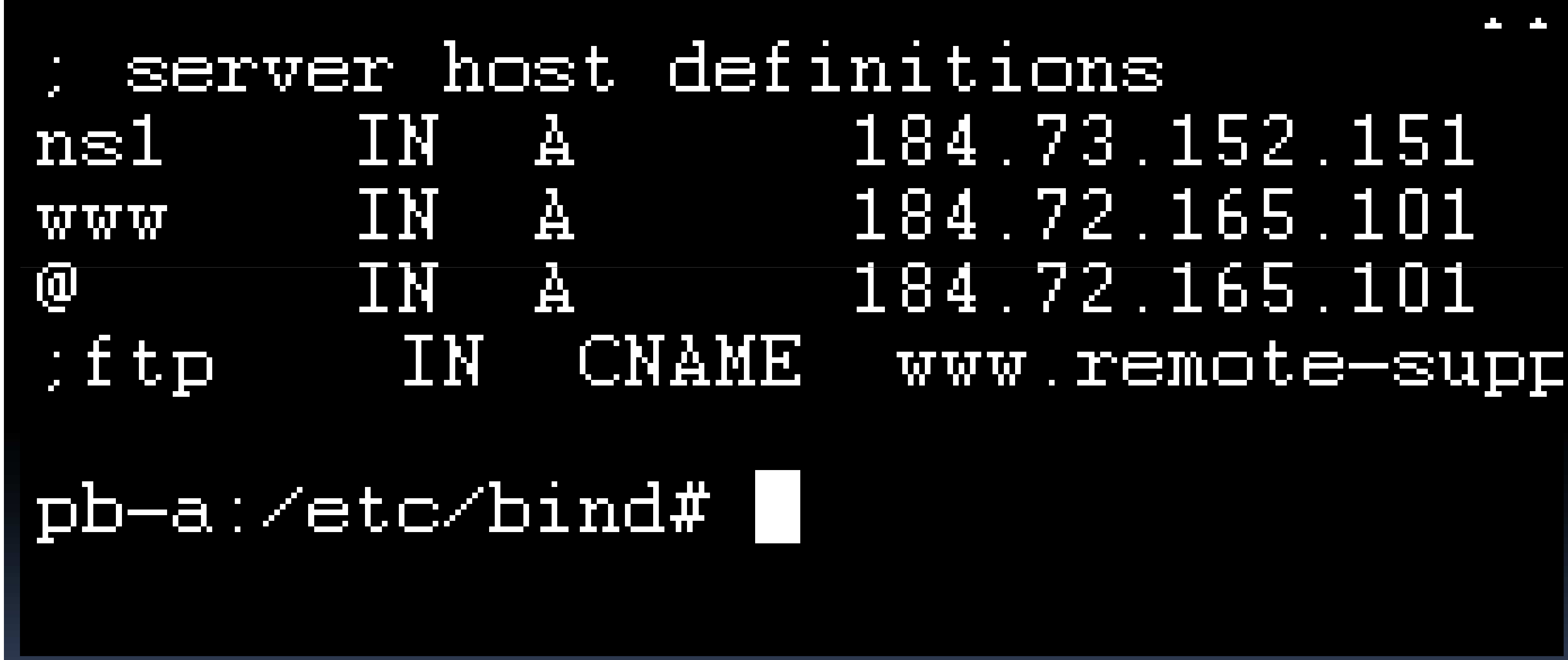
DNSSEC CHANGES EVERYTHING (No, it’s simple too)

- DNSSEC
  - Ask a question, get an answer *and a signature*
  - Ask a question, get a referral *and a signature*
    - Alice: Jenny’s number? Ask Travis™
    - Travis: Jenny’s number? Ask Charlie™
    - Charlie: Jenny’s number? 867-5309™
- Yes, that’s mostly it.
  - A lot of the complexity came from *optional magic*

The General Idea

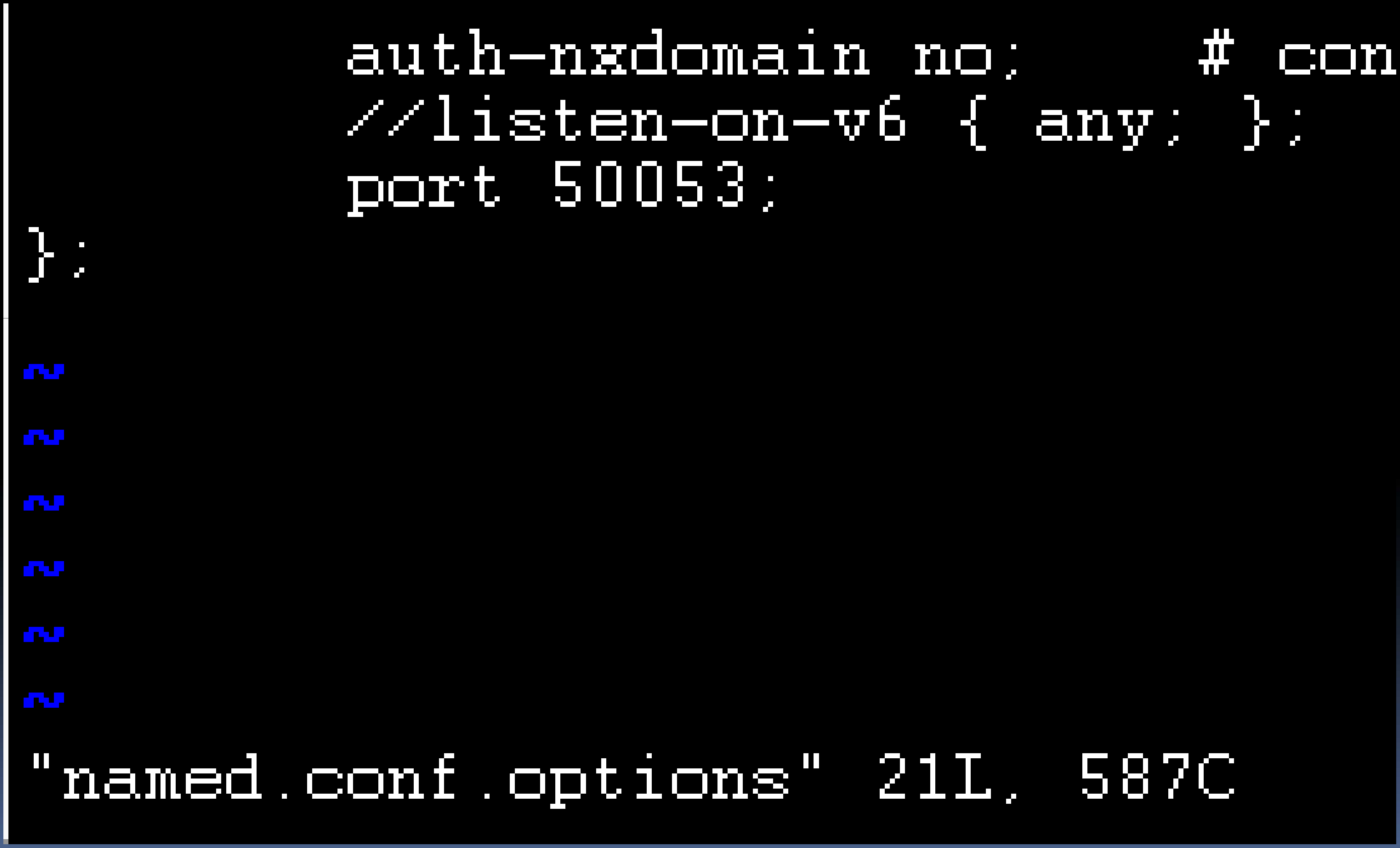
- DNSSEC lets DNS store trusted data
  - Use this data to bootstrap trust in various protocols, as per a globally shared namespace
- Open Questions
  - Can DNSSEC be deployed easily?
  - Will it function end to end?
  - Does it actually help real world applications?
- Lets see some working code…
  - First: Can DNSSEC be deployed easily?

A Simple Bind9 Install With A Handful Of Small Zones



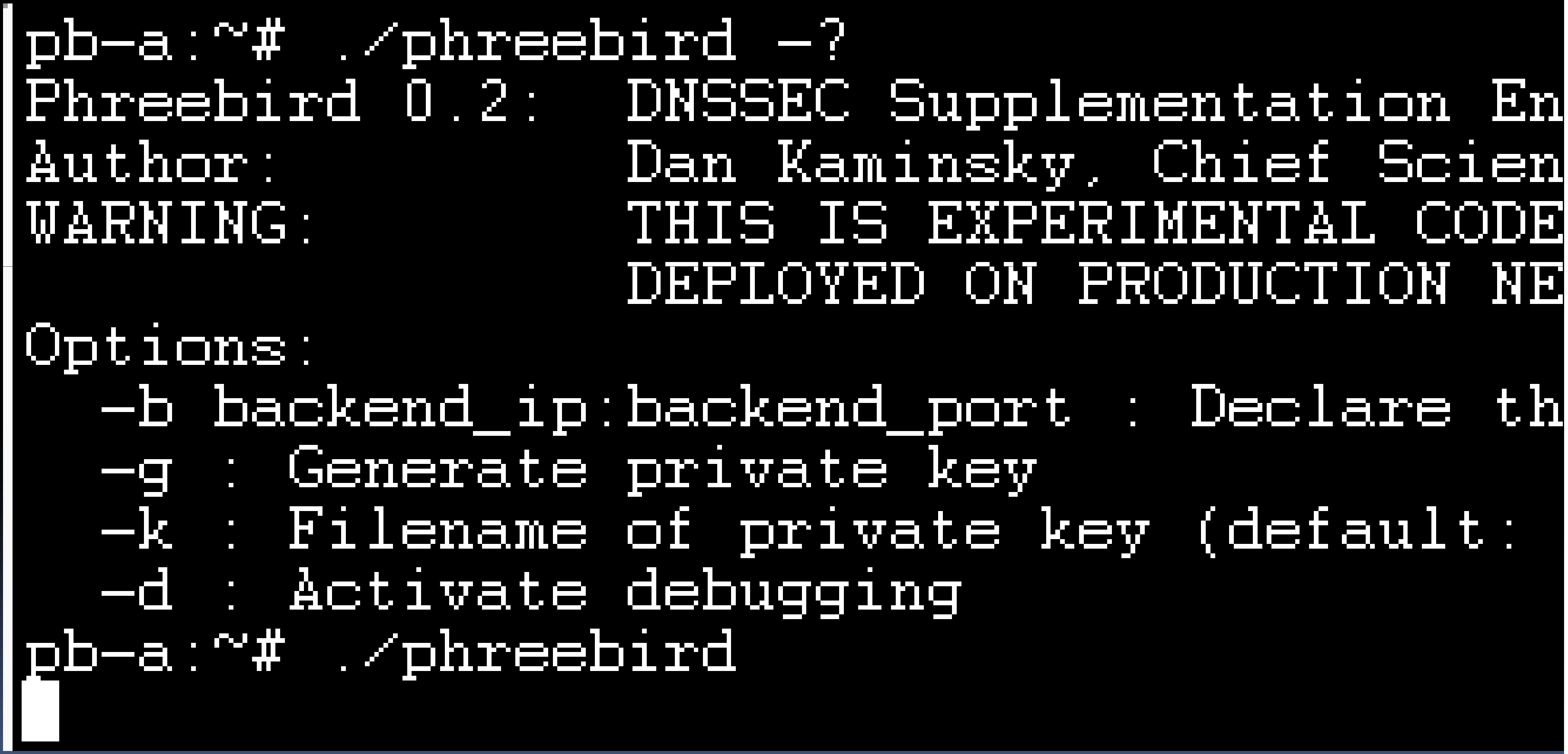
```
; server host definitions
ns1       IN   A          184.73.152.151
www       IN   A          184.72.165.101
@         IN   A          184.72.165.101
;ftp       IN   CNAME     www.remote-supp

pb-a:/etc/bind# █
```

Step 1: Change The Port To 50053



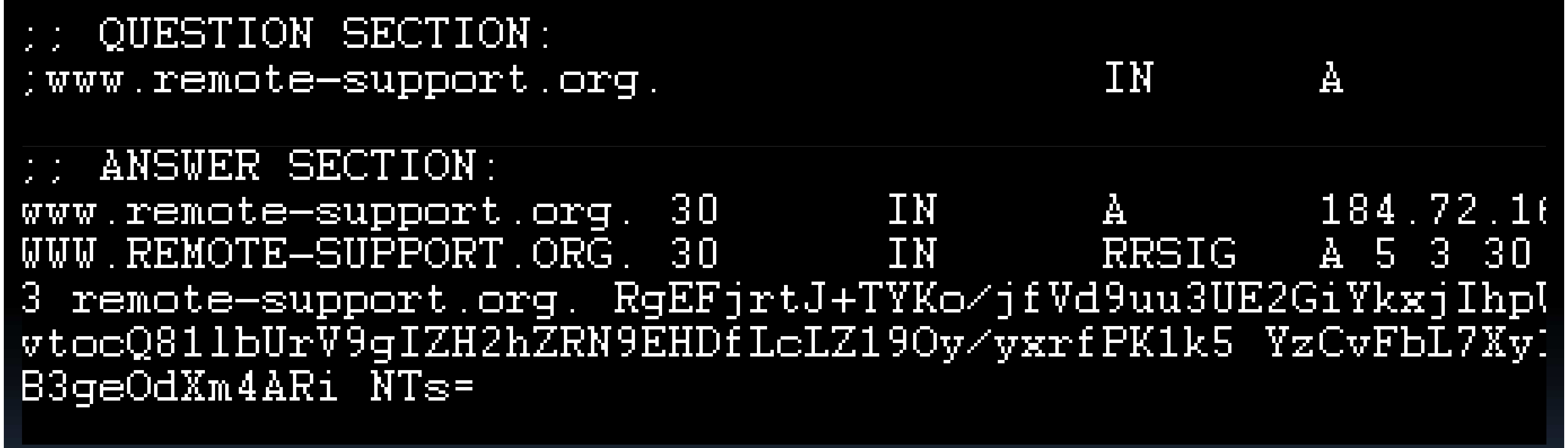
```
        auth-nxdomain no;    # con
        //listen-on-v6 { any; };
        port 50053;
};
~
~
~
~
~
~
"named.conf.options" 21L, 587C
```

Step 2: Launch Phreebird



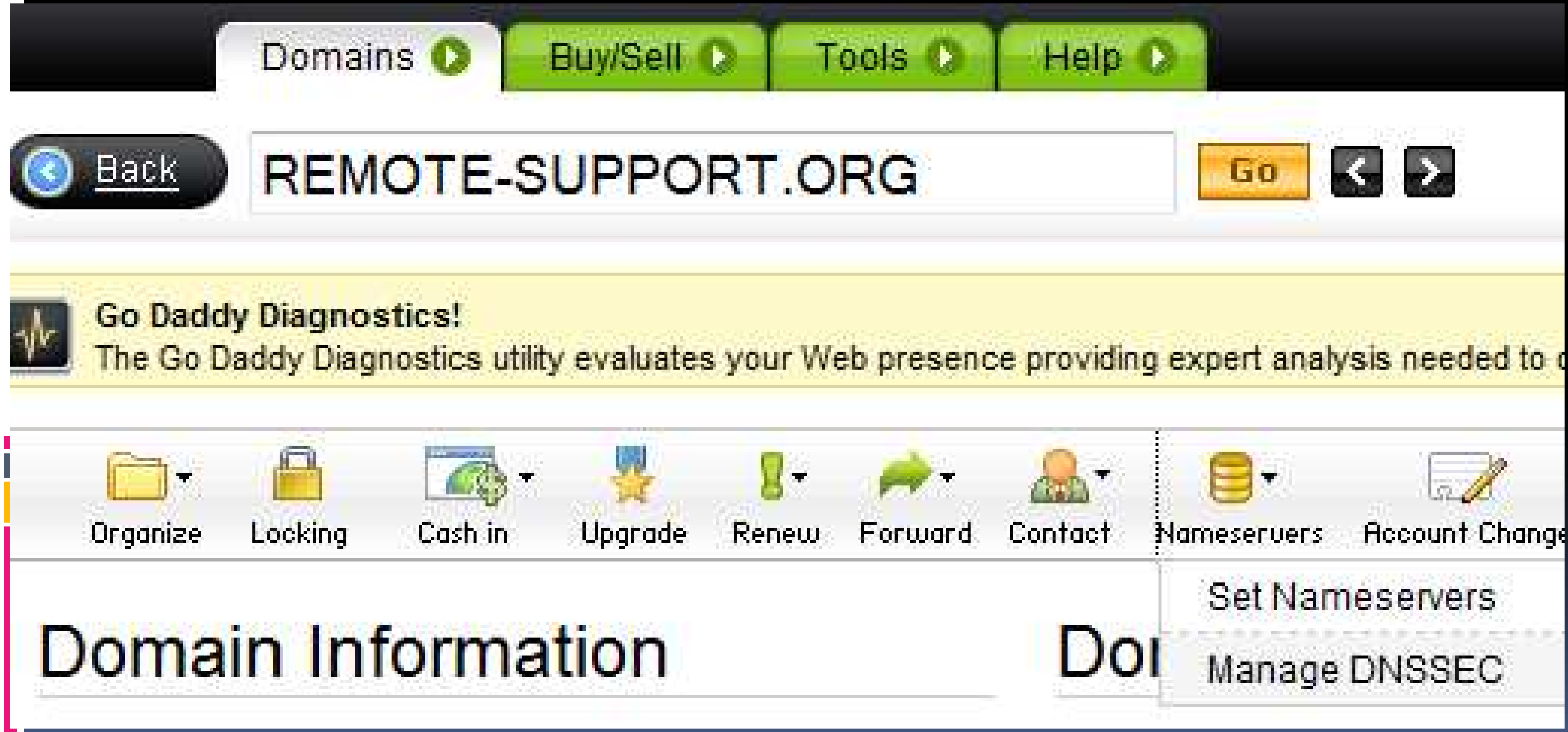
```
pb-a:~# ./phreebird -?
Phreebird 0.2:   DNSSEC Supplementation En
Author:          Dan Kaminsky, Chief Scien
WARNING:         THIS IS EXPERIMENTAL CODE
                 DEPLOYED ON PRODUCTION NE
Options:
   -b backend_ip:backend_port : Declare th
   -g : Generate private key
   -k : Filename of private key (default:
   -d : Activate debugging
pb-a:~# ./phreebird
```

Step 3: There is no step 3



```
;; QUESTION SECTION:
;www.remote-support.org.                        IN      A

;; ANSWER SECTION:
www.remote-support.org. 30      IN      A       184.72.1
WWW.REMOTE-SUPPORT.ORG. 30      IN      RRSIG   A 5 3 30
3 remote-support.org. RgEFjrtJ+TYKo/jfVd9uu3UE2GiYkxjIhp
vtocQ811bUrV9gIZH2hZRN9EHDfLcLZ19Oy/yxrfPK1k5 YzCvFbL7Xy
B3geOdXm4ARi NTs=
```

OK, you have to tell your registrar… (GoDaddy for now)



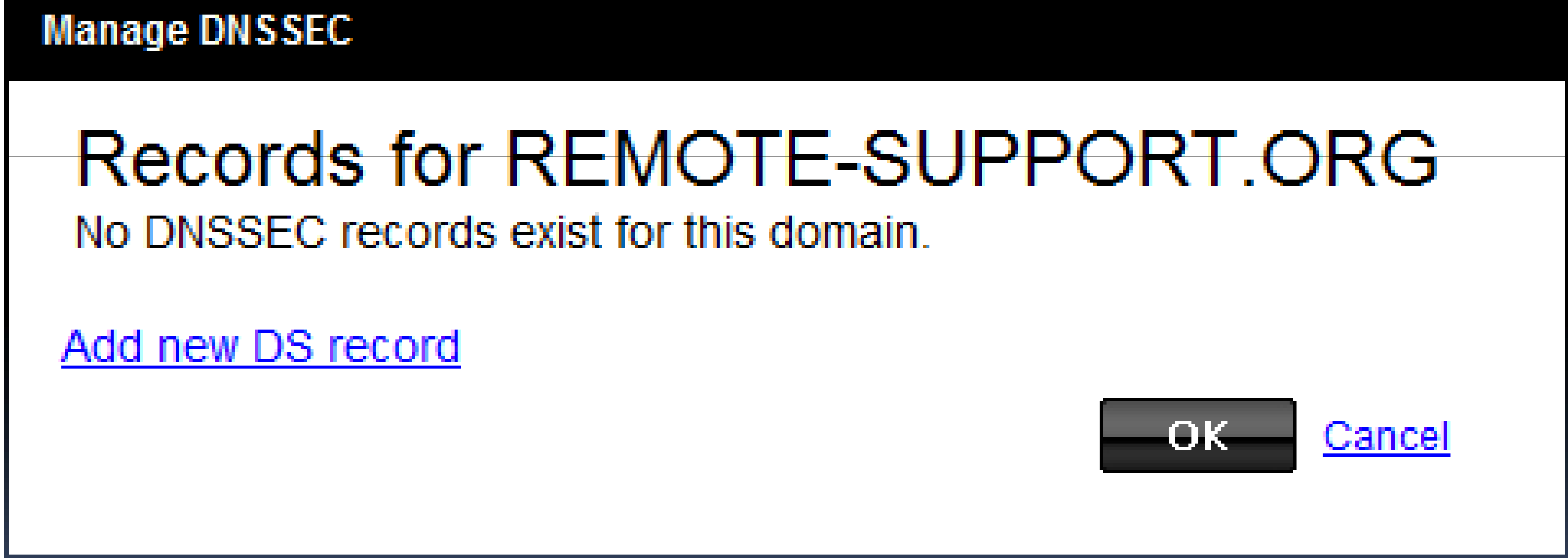
REMOTE-SUPPORT.ORG

Go Daddy Diagnostics!
The Go Daddy Diagnostics utility evaluates your Web presence providing expert analysis needed to

Domain Information

Do

It wants a DS record…



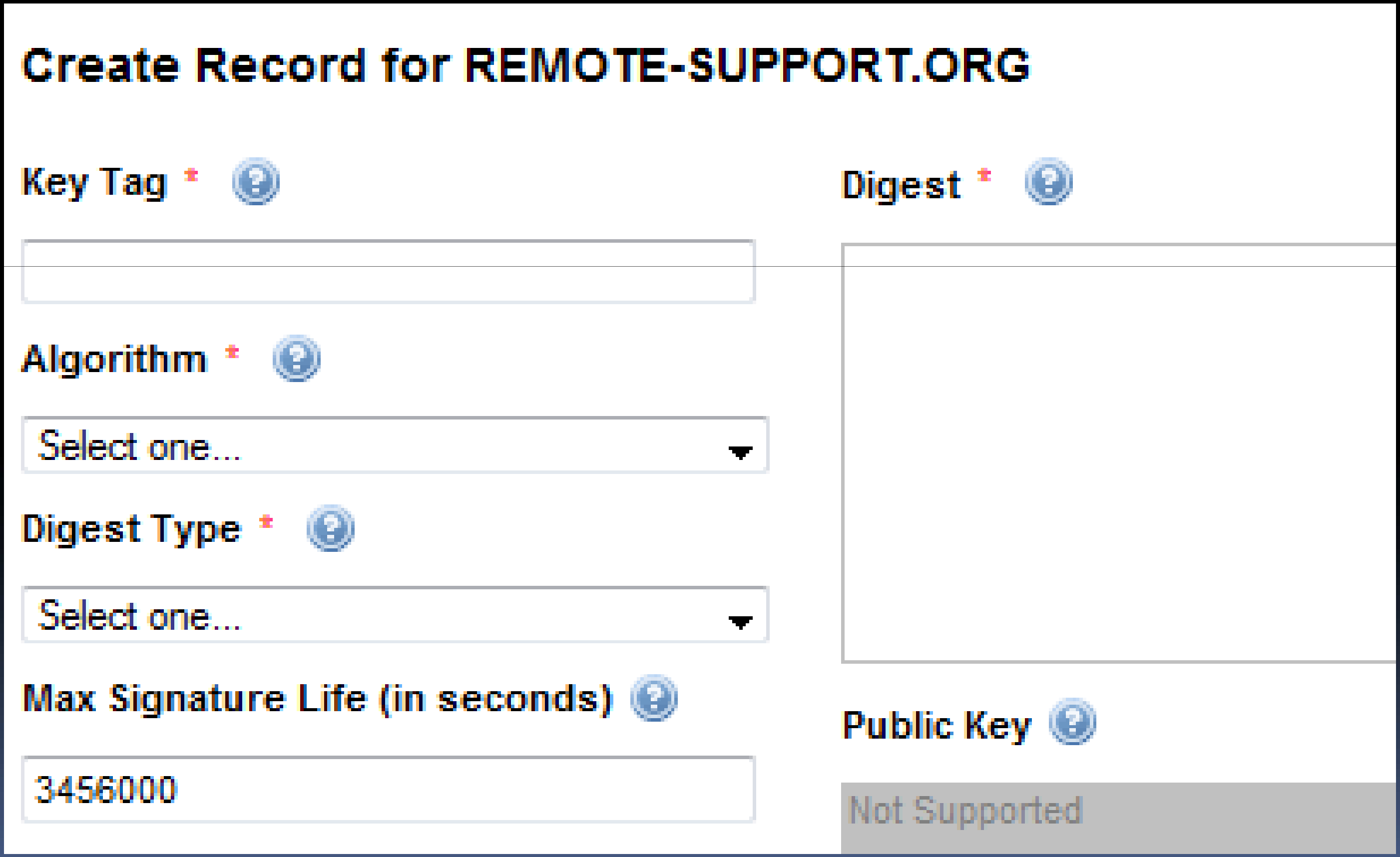
Manage DNSSEC

Records for REMOTE-SUPPORT.ORG

No DNSSEC records exist for this domain.

Where can we get these values?

Add DS Record



Create Record for REMOTE-SUPPORT.ORG

Key Tag *

Digest *

Algorithm *

Select one...

Digest Type *

Select one...

Max Signature Life (in seconds)

3456000

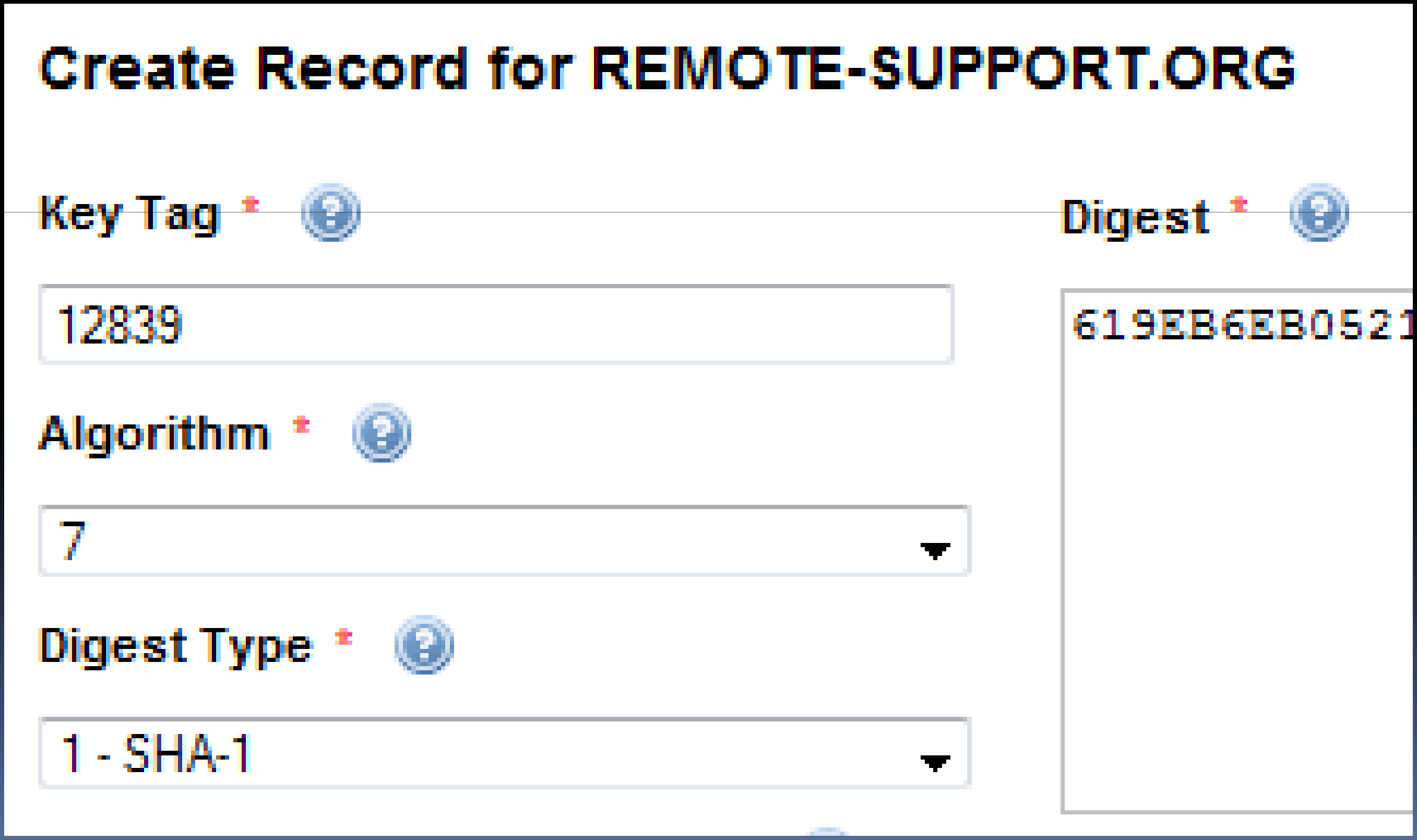
Public Key

Not Supported

Just Run Dig…

- # dig +short @127.0.0.1 remote-support.org ds
  12839 7 1
  619EB6EB0521605393FA60353660 7949AE58EDD6

Just paste 'em in…



Create Record for REMOTE-SUPPORT.ORG

Key Tag *

12839

Algorithm *

7

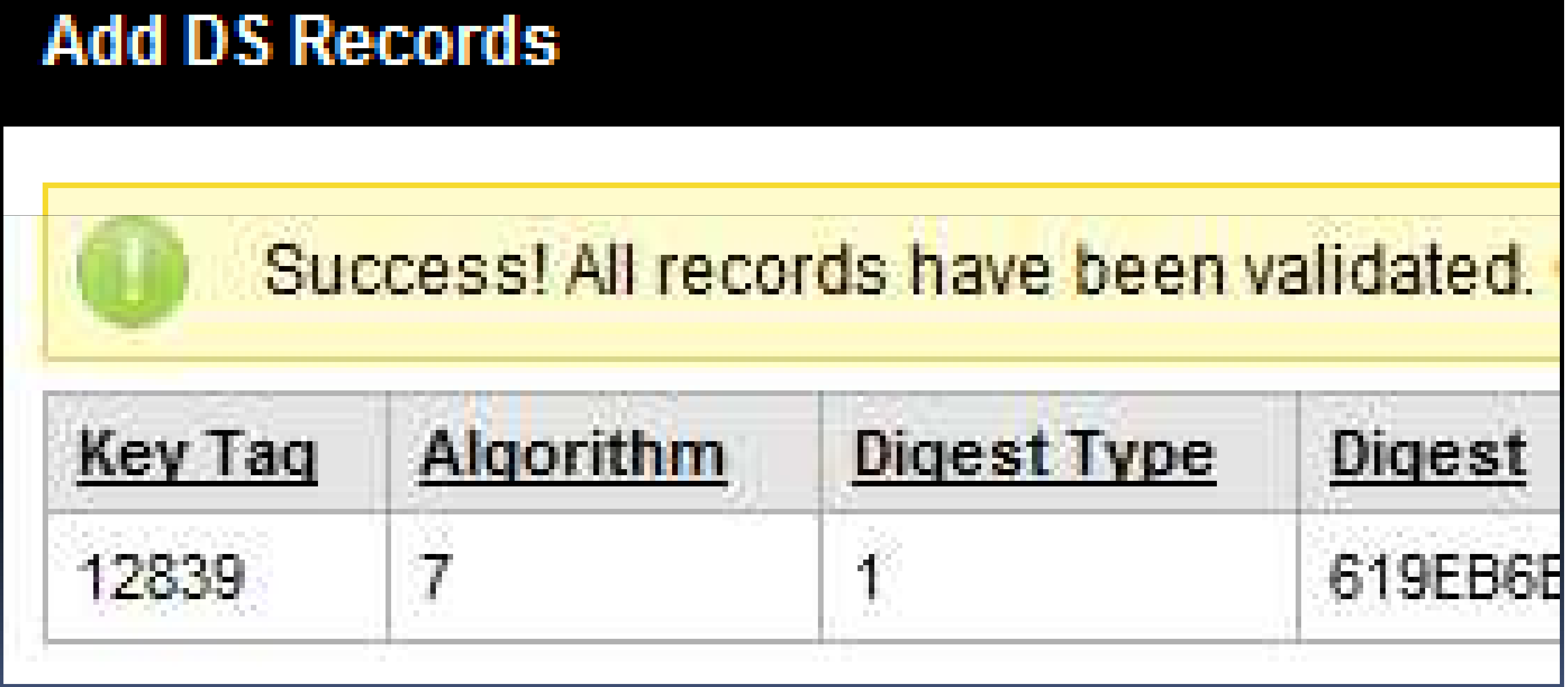
Digest Type *

1 - SHA-1

Digest *

619EB6EB0521

And, that's it!

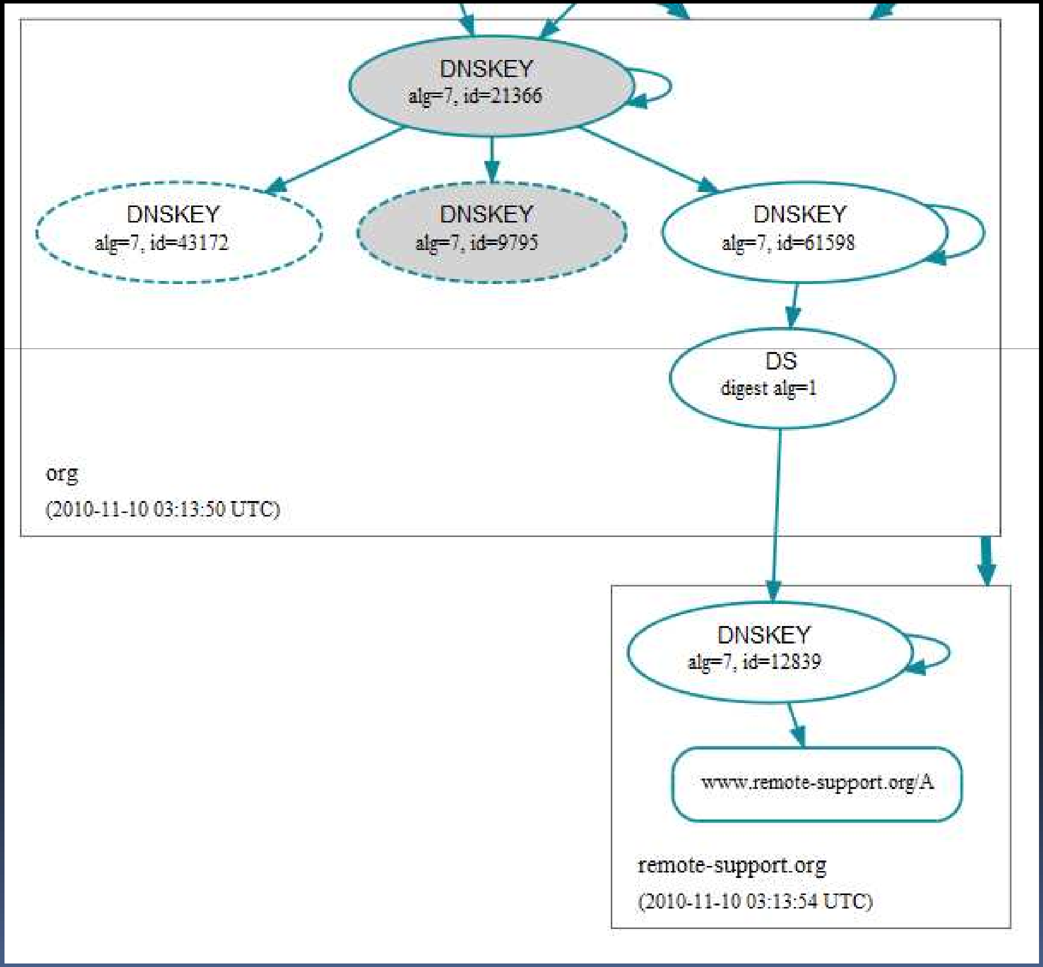


Add DS Records

Success! All records have been validated.

Key Tag	Algorithm	Digest Type	Digest
12839	7	1	619EB6E

DNSSEC is deployed. DONE. (dnsviz.net)



Welcome to Phreebird, Released Today!

- Phreebird: A realtime DNSSEC proxy that sits in front of any DNS server, supplementing its responses with signed answers
  - Most of DNSSEC's problems have come from the requirement to be able to operate *offline*
  - DNSSEC can also be done *online*, like TLS, IPSec, Kerberos, and SSH
- Phreebird can be deployed in minutes
  - No key generation phase
  - No zone signing phase
    - Doesn't care how many zones you have
  - No configuration
  - It Just Works™

Phreebird Efficiency

- Signatures are cached as they're generated
  - Answers are sacred – whatever answer happens to be delivered, based on time/geo/load/mood, will still get delivered
- Nonexistence records are dynamically generated
  - "White Lies", only for NSEC3 instead of NSEC
  - In NSEC3, names are turned into numbers
  - "There are no names between 1 and 3"
- Under heavy load or attack, server prioritizes positive replies over NSEC3 sigs
  - Under overload conditions, SERVFAIL is returned
  - NSEC3 does not actually stop SERVFAIL – but servers don't cache it

Isn’t Online Keysigning Dangerous?

- Many protocols use online keysigning
  - SSL
  - SSH
  - IPSec
- DNSSEC needed to be able to support offline keys
  - The root should not have the keys online
  - Massive TLDs shouldn’t need to have key material in every location/jurisdiction they have hosting
- But *supporting* online keys != *requiring* online keys

The Cost Of Offline Operation

- PGP/GPG
  - What happens when you receive mail from someone not on your keyring?
  - What happens when you have to send mail to someone not on your keyring?
  - What happens when a key expires?
  - What happens when a key is lost?
  - What happens when a key is stolen?

If you can’t handle failures, you can’t succeed

- *Offline key management is unable to handle special cases well*
  - In DNSSEC, you can quickly publish new data, and client can quickly retrieve it. The special cases get first class support.
  - **Revocation stops being an exceptions. Keys expire all the time, get over it!**

Is There Other Magic?

- There is a lot of obscura in the DNSSEC realm that we've been filtering through
  - How do we tunnel trusted records to registries when the registrars in front of them don't implement DNSSEC?
  - How do we manage rollover and expiration?
  - How do we keep clocks in sync, especially given the chicken-and-egg relationship between NTP and DNSSEC?
- If you're interested – ask me after this talk. Right now, I want to focus on applications.

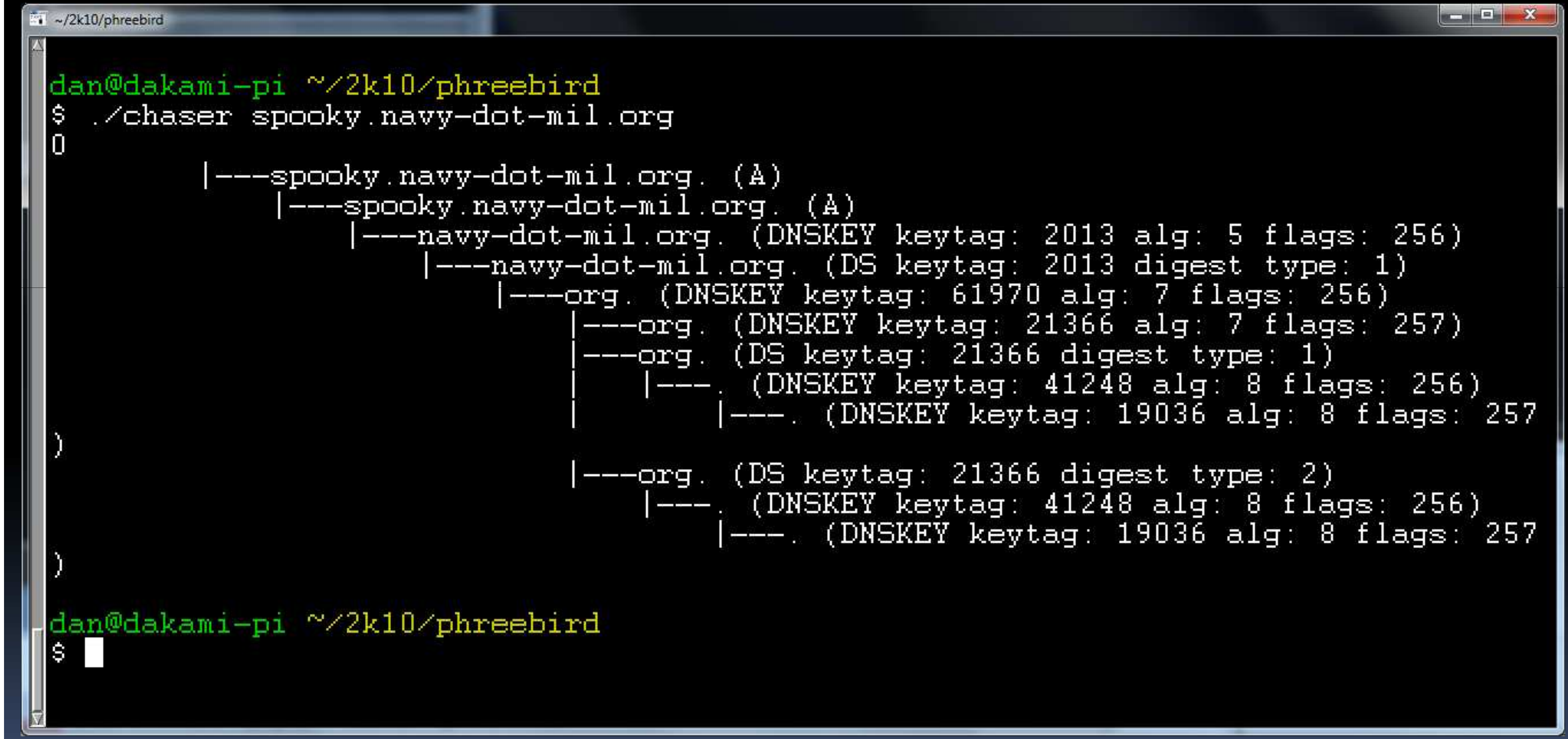
What App Developers Need

- App developers don't want to be crypto developers!
  - They don't have to be masters of distributed databases, but they get to benefit from one every day when they resolve DNS names
- App developers need to be able to ***easily*** authenticate entities outside their organization
  - It's no drama to look up a user's mail server.
  - It's epic drama to recognize a user's smartcard
  - Can DNSSEC fix this?

End-to-End Security for DNSSEC

- Problem: DNSSEC was originally envisioned to allow *name servers* (not desktops) to be able to verify data
  - Desktops would just get a “bit” declaring everything safe
  - Reality: I like Starbucks, but I’m not trusting their name server to tell me anything is safe
- Is it possible to efficiently determine a trust chain via DNSSEC, at the desktop?

Yes, in about 10 lines of code using LDNS



```
dan@dakami-pi ~/2k10/phreebird
$ ./chaser spooky.navy-dot-mil.org
0
              |---spooky.navy-dot-mil.org. (A)
                     |---spooky.navy-dot-mil.org. (A)
                            |---navy-dot-mil.org. (DNSKEY keytag: 2013 alg: 5 flags: 256)
                                   |---navy-dot-mil.org. (DS keytag: 2013 digest type: 1)
                                          |---org. (DNSKEY keytag: 61970 alg: 7 flags: 256)
                                                 |---org. (DNSKEY keytag: 21366 alg: 7 flags: 257)
                                                 |---org. (DS keytag: 21366 digest type: 1)
                                                 |      |---. (DNSKEY keytag: 41248 alg: 8 flags: 256)
                                                 |             |---. (DNSKEY keytag: 19036 alg: 8 flags: 257
)
                                                 |---org. (DS keytag: 21366 digest type: 2)
                                                        |---. (DNSKEY keytag: 41248 alg: 8 flags: 256)
                                                               |---. (DNSKEY keytag: 19036 alg: 8 flags: 257
)

dan@dakami-pi ~/2k10/phreebird
$
```

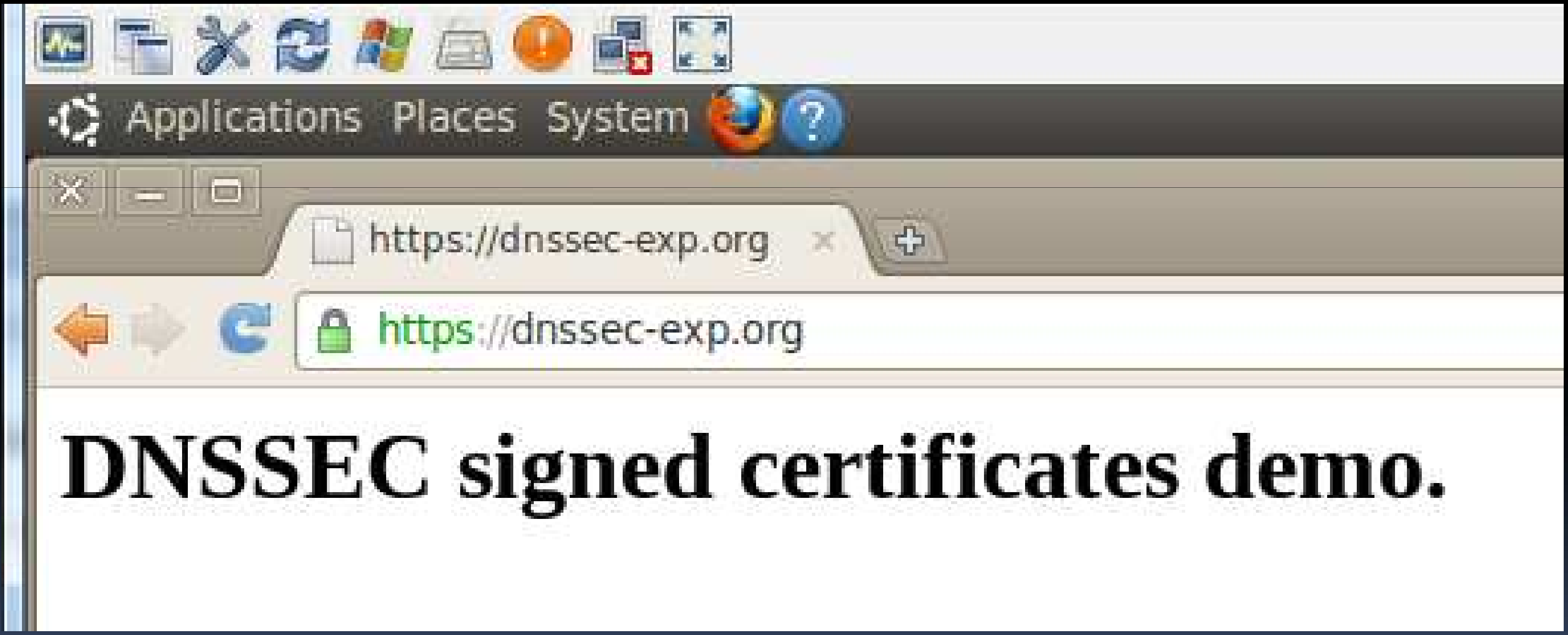
Approaches for End-To-End Trust (All Implemented Here)

- Chase (via ldns)
  - Given the signature for www.foo.com, discover the signature up from foo.com, then com, then the root.
- Trace (via libunbound)
  - Given the signature for the root, discover the signature down from the root, then com, then foo.com.
  - Basically, just embed a recursive DNS resolver in client
    - Load issues!
- Wrap (via Phreebird-modified ldns)
  - Encapsulate DNS in an HTTP request to a compliant server
  - Useful when behind inclement firewalls (common!)
- Pack…

X.509 Packing

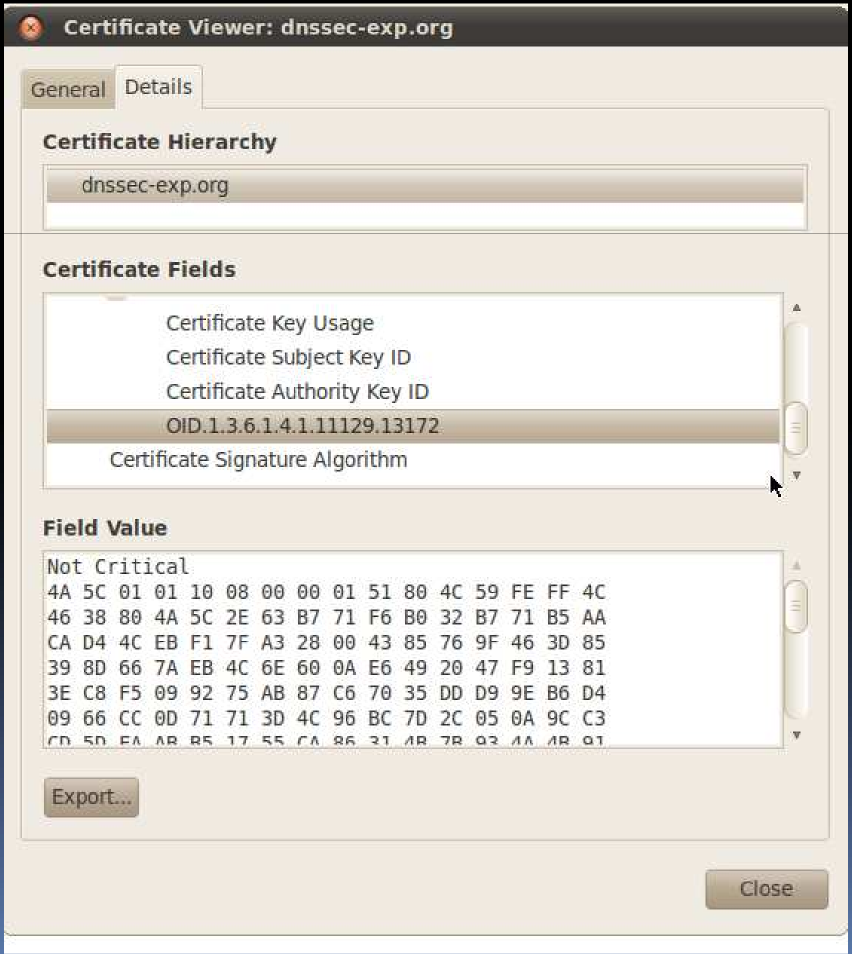
- Inspired by Brett Watson's quote
  - "You have to be willing to separate the *content* of DNS from the *transport* of DNS"
- X.509 as a chain delivery mechanism is pretty broken (see 2009 Black Ops of X.509 for details)
- X.509 as a way to transfer arbitrary payloads as part of a chain bound to a TLS session…is pretty solid
  - Why not tunnel DNSSEC data re: a TLS endpoint through DNSSEC?

So Adam Langley at Google sent me a **private unofficial** build of Chrome…



DNSSEC signed certificates demo.

That certificate was self signed.....with a DNSSEC chain embedded.



Certificate Hierarchy

dnssec-exp.org

Certificate Fields

Certificate Key Usage
Certificate Subject Key ID
Certificate Authority Key ID
OID.1.3.6.1.4.1.11129.13172
Certificate Signature Algorithm

Field Value

```
Not Critical
4A 5C 01 01 10 08 00 00 01 51 80 4C 59 FE FF 4C
46 38 80 4A 5C 2E 63 B7 71 F6 B0 32 B7 71 B5 AA
CA D4 4C EB F1 7F A3 28 00 43 85 76 9F 46 3D 85
39 8D 66 7A EB 4C 6E 60 0A E6 49 20 47 F9 13 81
3E C8 F5 09 92 75 AB 87 C6 70 35 DD D9 9E B6 D4
09 66 CC 0D 71 71 3D 4C 96 BC 7D 2C 05 0A 9C C3
CD 5D FA AB B5 17 55 CA 86 31 4B 7B 93 4A 4B 91
```

An Interesting Variant

- Host DNSSEC chains over HTTP, at well known addresses
  - http://www.foo.com/.well_known/dns-http
    - This is actually RFC compliant
  - Can even host over HTTPS, letting the endpoint self-authenticate via the chain for www.foo.com

So, do we add ldns/libunbound to each package, one by one?

- Eventually, possibly
  - Works very well for PhreeShell, the Federated OpenSSH Demo at the start of the talk
  - Sample code for this also part of Phreebird Suite
- But in the short term? To prove value?
- On Linux/Unix, SSL is handled via OpenSSL
  - Specifically, X509_verify_cert
  - A nice and self contained library call…hmm…

PhreeLoad: Integrating DNSSEC into OpenSSL via LD_PRELOAD

- Prior Work: libval_shim from Russ Mundy @ Sparta
  - Great work!
  - Two major differentiators
    - 1) Written before the root was signed, so no provisions for chasing a signature down to the root
    - 2) Validated the results of a DNS query – which might just be an IP address, attackable via other means
- PhreeLoad operates at a different layer
  - Given software that's attempting to achieve end-to-end security, replace/augment the auth layer with DNSSEC

CURL to a self signed certificate

- # curl https://www.hospital-link.org

```
curl: (60) SSL certificate problem, verify that the CA cert is OK. Details:
error:14090086:SSL routines:SSL3_GET_SERVER_CERTIFICATE:certificate verify failed
...
```

If you'd like to turn off curl's verification of the certificate, use the -k (or --insecure) option.

After Loading Phreeload

- # phreeload curl https://www.hospital-link.org
- <pre>
  Now this is a story
  all about how my life
  got twisted upside down
  and id like to take a minute
  just sit right there
  ill tell you how i became the prince
  of a town called Bel-Air

Enabling Debug

- # phreeload curl https://www.hospital-link.org
- Resolving www.hospital-link.org
  Secure result recieved.
- V:1 Hash Algorithm:sha1
  Hash:5e0905b0eafd35d59f1b178727d4eaadd06c415d
  STS:0. Secure Reneg:0 Livehash:0 Hash Range:(null)
  Hash detected: sha1
  5e0905b0eafd35d59f1b178727d4eaadd06c415d
  Hash Validated
  DNSSEC validated
  <pre>
  Now this is a story
  all about how my life
  got twisted upside down

What Are We Actually Putting Into DNS?

- www.hospital-link.org IN TXT "v=key1 ha=sha1 h=5e0905b0eafd35d5…"
  - v=KEY1
    Version is KEY1
  - ha=sha1
    Hash Algorithm is SHA-1
  - h=5e0905b0eafd35d5…
    Hash of certificate is h=5e0905b0eafd35d5…

Alternate Key Retrievals

- lh=[0|1]:   LiveHash
  - Whether, upon a failed resolution for www.foo.com, a second lookup to _tlshash-f1d2d2f924e986ac86fdf7b36c94bcdf32beec15.www.foo.com should be attempted. (Not done by default due to performance implications.)
- hr=[cert|pubkey]: Hash Range
  - If set to "cert" (or unset), the hash validated is the hash of the entire certificate. If set to "pubkey", the hash validated is the hash of the public key in the certificate.

Extensible Metadata Support

- sts=[0|1]: Strict-Transport-Security:
  - Whether insecure (http) access to www.foo.com should be allowed
  - **This is how we address Firesheep!**
    - It's too expensive / tricky right now to get certs for everything
    - It's too expensive / tricky right now to shut down insecure channels after secure ones exist
    - **DNSSEC fixes things by making security cheaper.**
- sn=[0|1]: Secure Negotiation:
  - Whether secure renegotiation will be present at this HTTPS endpoint

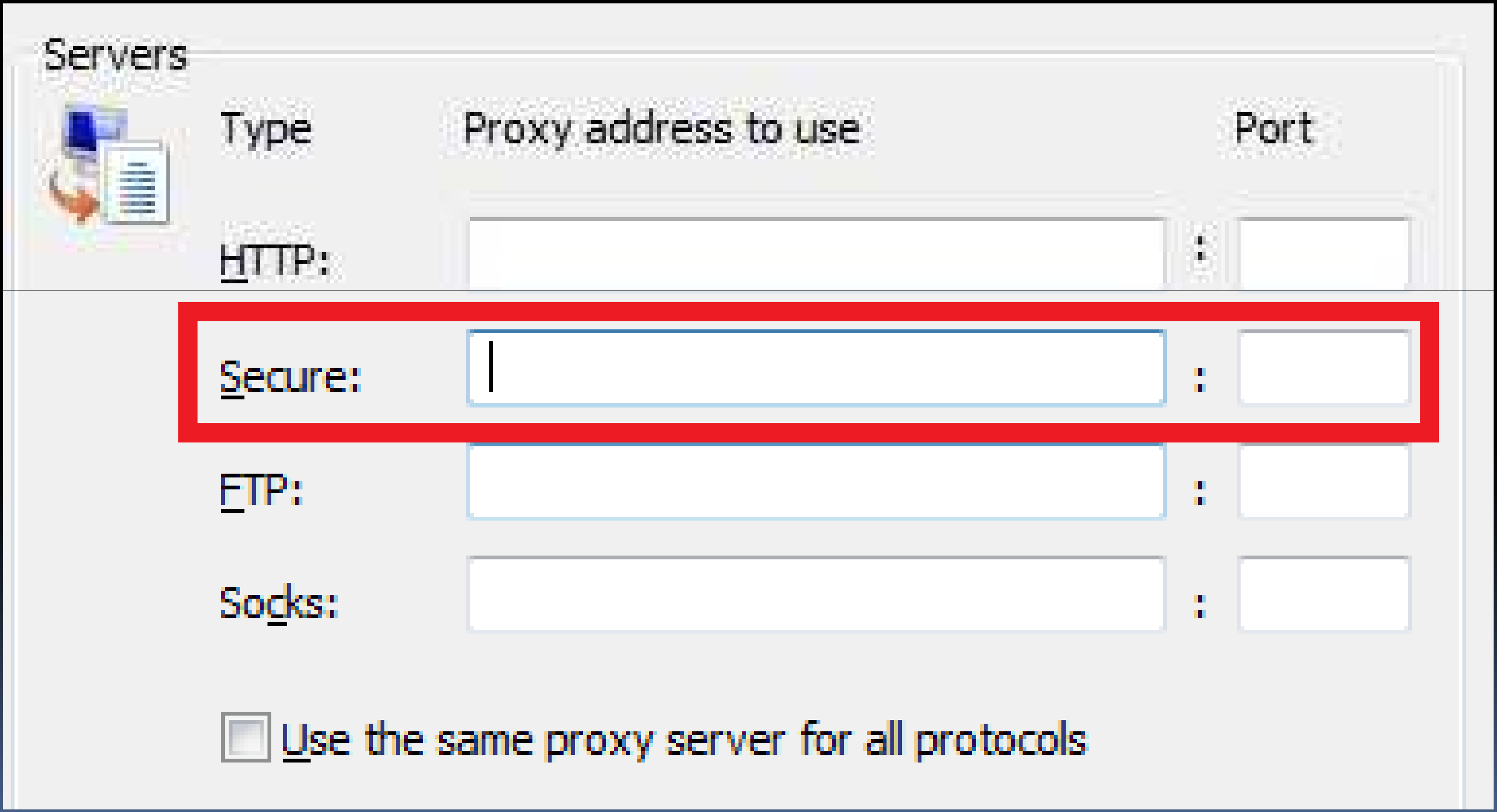
Why This Particular Schema?

- Have to go live with something – this should *not be seen as canonical or consensus*
  - However…
- Last four protocols to try to do complex things in DNS went TXT
  - DKIM
  - SPF
  - HPA (in GPG)
  - IPSec
- Don't want a record compiler
- Don't want to require upgrading servers / web Uis
- **Really** don't want another binary protocol
  - If you notice, we've sort of abandoned ASN.1 for XML/JSON
  - For a reason

All OpenSSL apps means *All OpenSSL Apps*

- Postfix
- Postgres
- MySQL
- Apache
  - Want to start working on client certificates that actually work? Much easier now that we have a signed root
  - Welcome to DKI
- But how do we get this working on browsers?
  - Most not running on Linux
  - Most not running with OpenSSL

Phoxie: Remote SSL Validation For All Browsers

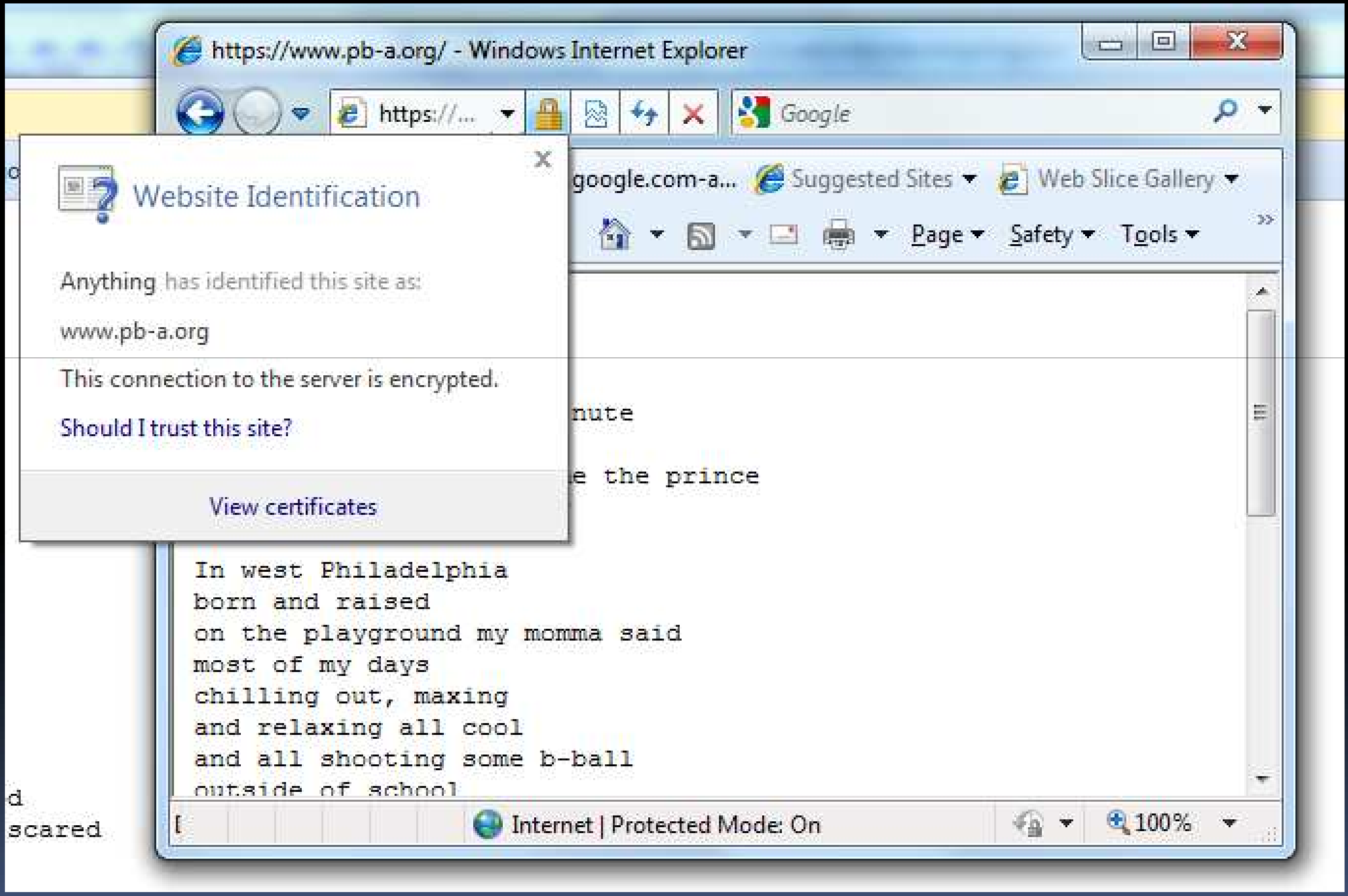


Servers

Type	Proxy address to use		Port
HTTP:		:	
Secure:		:	
FTP:		:	
Socks:		:	

[ ] Use the same proxy server for all protocols

Browser Lock In IE



Website Identification

Anything has identified this site as:

www.pb-a.org

This connection to the server is encrypted.

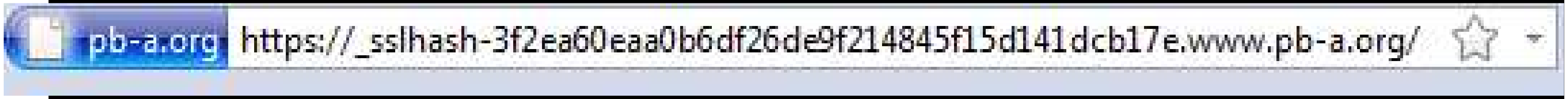
Should I trust this site?

View certificates

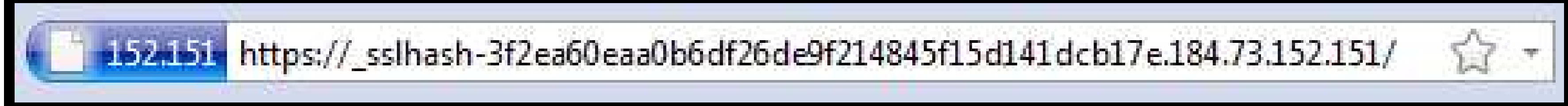
```
nute

e the prince

In west Philadelphia
born and raised
on the playground my momma said
most of my days
chilling out, maxing
and relaxing all cool
and all shooting some b-ball
outside of school
```

```
d
scared
```

Also: A Neat Toy! Self Certifying URLs (Inspired by SFS)


pb-a.org https://_sslhash-3f2ea60eaa0b6df26de9f214845f15d141dcb17e.www.pb-a.org/

It even works by IP!


152.151 https://_sslhash-3f2ea60eaa0b6df26de9f214845f15d141dcb17e.184.73.152.151/

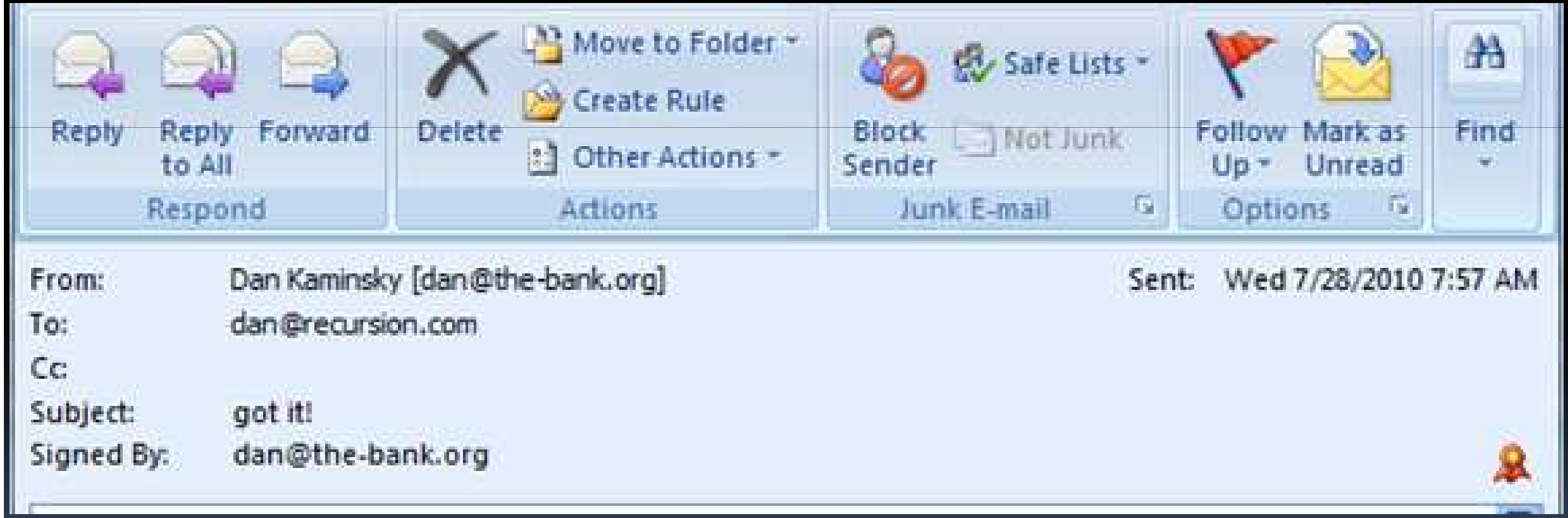
Self Certification Modes: Useful?

- Possibly – it'd be nice if applications had a clean way to directly declare the actual key they wanted to use.
    - This approach adds the key to the domain being connected to
    - Works well for HTTP-based APIs
    - Works poorly for OS sockets
- Admittedly, those are ugly domain names
    - But they work for free

What About Windows?

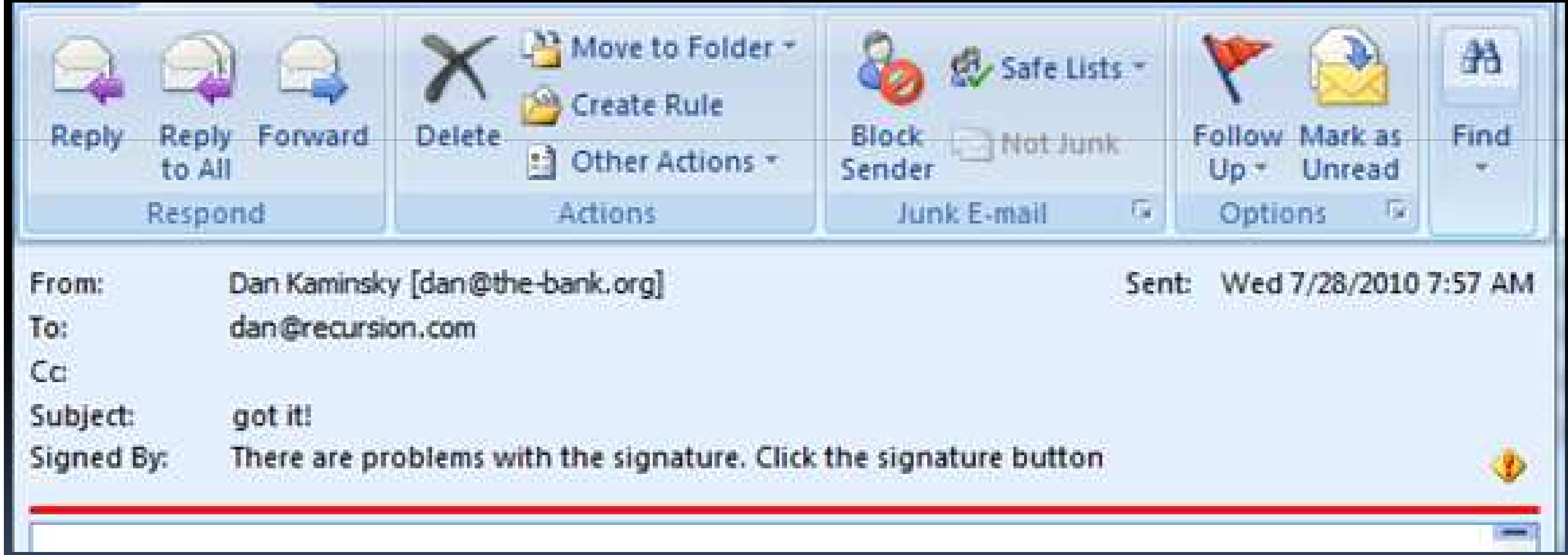
- Linux makes it very clean, to hook individual functions inside of major APIs
- Windows makes it harder, but not impossible
  - PhreeCAPI: A DLL Injector for CryptoAPI
  - Target Application: Outlook 2007
    - Problem: S/MIME certificates are free and automatically issued. Not much confidence in them.
    - Could we use DNSSEC to achieve exclusion, the primary property that makes DNSSEC better than X.509?

So, we have this signed email from dan@the-bank.org
(Not exactly the most compelling image)



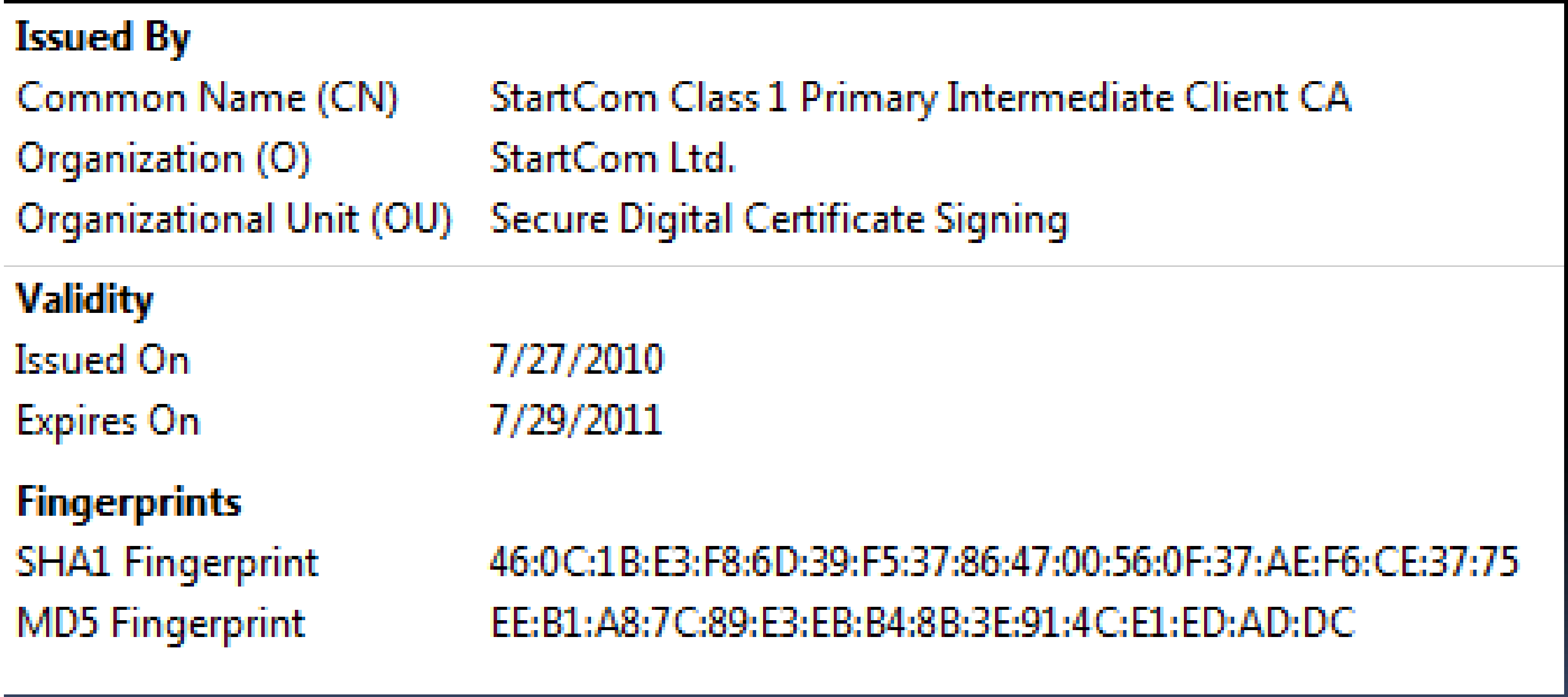
From: Dan Kaminsky [dan@the-bank.org]
Sent: Wed 7/28/2010 7:57 AM
To: dan@recursion.com
Cc:
Subject: got it!
Signed By: dan@the-bank.org

We just added the checker, didn't put any records into the DKI



From: Dan Kaminsky [dan@the-bank.org]
Sent: Wed 7/28/2010 7:57 AM
To: dan@recursion.com
Cc:
Subject: got it!
Signed By: There are problems with the signature. Click the signature button

Well, there’s the fingerprint of the canonical certificate…



Issued By

Common Name (CN)	StartCom Class 1 Primary Intermediate Client CA
Organization (O)	StartCom Ltd.
Organizational Unit (OU)	Secure Digital Certificate Signing

Validity

Issued On	7/27/2010
Expires On	7/29/2011

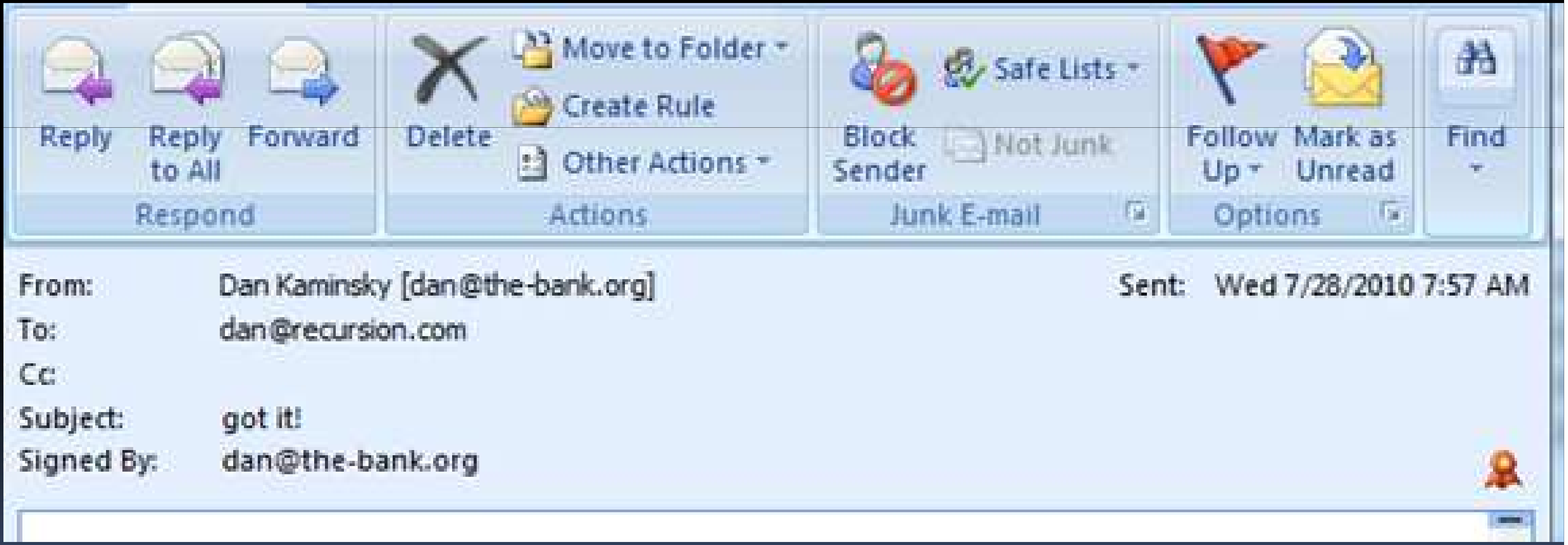
Fingerprints

SHA1 Fingerprint	46:0C:1B:E3:F8:6D:39:F5:37:86:47:00:56:0F:37:AE:F6:CE:37:75
MD5 Fingerprint	EE:B1:A8:7C:89:E3:EB:B4:8B:3E:91:4C:E1:ED:AD:DC

So, lets put this (DELIBERATELY OVERSIMPLISTIC) thing in the DKI

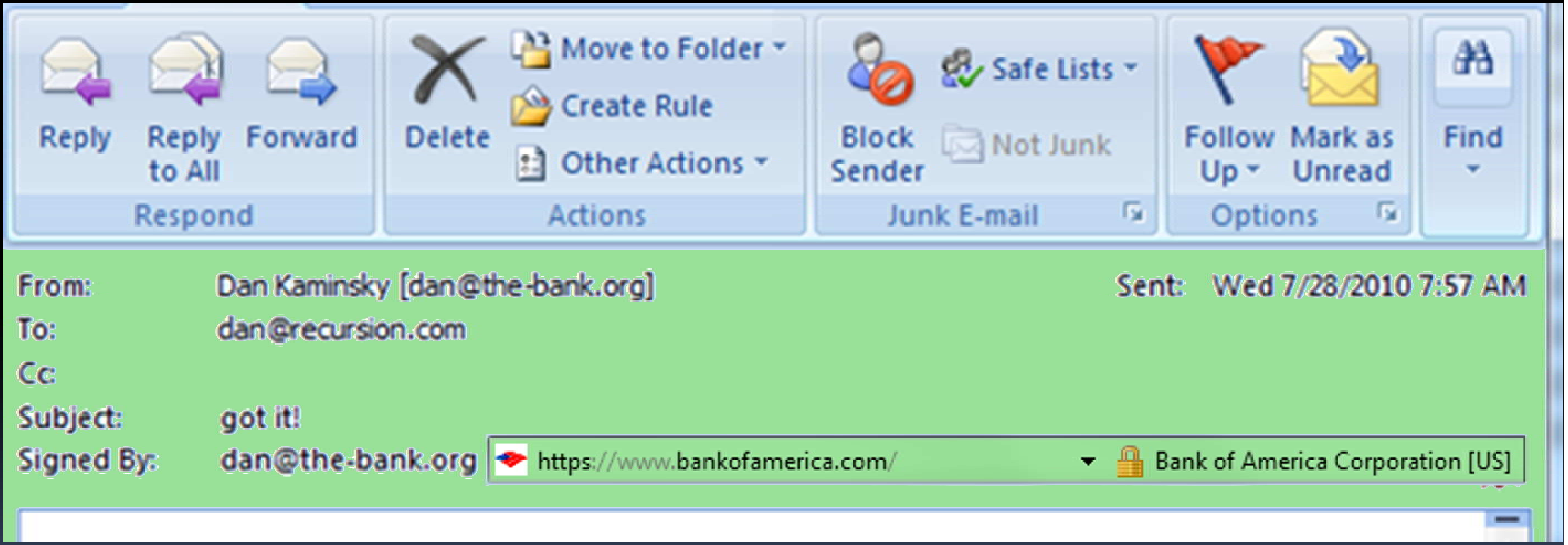
- dan._smpka.the-bank.org. IN TXT ‘v=KEY1 ha=sha1 h=460c1be3f86d39f53786470056of37aef6ce3775'

Now we have mail from the bank, signed by the only key in the world that can sign it.



From: Dan Kaminsky [dan@the-bank.org]   Sent: Wed 7/28/2010 7:57 AM
To: dan@recursion.com
Cc:
Subject: got it!
Signed By: dan@the-bank.org

So…why not make it look even better?



From: Dan Kaminsky [dan@the-bank.org]
Sent: Wed 7/28/2010 7:57 AM
To: dan@recursion.com
Cc:
Subject: got it!
Signed By: dan@the-bank.org https://www.bankofamerica.com/ Bank of America Corporation [US]

We’ve Been Promising Secure Email For Over A Decade

- DNSSEC is how we can deliver it
- CAs remain useful – DNSSEC only says that this is the-bank.org. It doesn’t say that this is “The Bank.”
  - That is what EV is for
  - We can (and should) port EV to email
- **With added confidence in the source of email, even cross organizationally, we could reasonably implement meaningful UI around message security**
  - We are blocked from doing that today because we have so little confidence in either success *or* failure

Conclusion

- The Domain Key Infrastructure is real
  - Federated OpenSSH works
  - Browser locks work
  - Easy application integration works
  - Email works
  - The CA's still matter!
- This is real, and this is a big deal
  - Phreebird Suite 1.0 is on blackhat.com's website! Enjoy!