High-speed high-security cryptography: encrypting and authenticating the whole Internet

D. J. Bernstein
University of Illinois at Chicago

```
wget -m -k -I / \
  secspider.cs.ucla.edu
cd secspider.cs.ucla.edu
awk ’
  /GREEN.*GREEN.*GREEN.*Yes/ {
    split($0,x,/<TD>/)
    sub(/<\/TD>/,"",x[5])
    print x[5]
  }
’ ./*--zone.html \
| sort -u | wc -l
```

A brief history of DNSSEC server deployment:

1993.11: DNSSEC design begins.

A brief history of
DNSSEC server deployment:

1993.11: DNSSEC design begins.

2008.07: Kaminsky announces apocalypse, saves the world.

A brief history of
DNSSEC server deployment:

1993.11: DNSSEC design begins.

2008.07: Kaminsky announces apocalypse, saves the world.
$\Rightarrow$ New focus on DNSSEC.

A brief history of
DNSSEC server deployment:

1993.11: DNSSEC design begins.

2008.07: Kaminsky announces apocalypse, saves the world.
$\Rightarrow$ New focus on DNSSEC.

2009.08.09:
941 IP addresses worldwide are running DNSSEC servers.

A brief history of
DNSSEC server deployment:

1993.11: DNSSEC design begins.

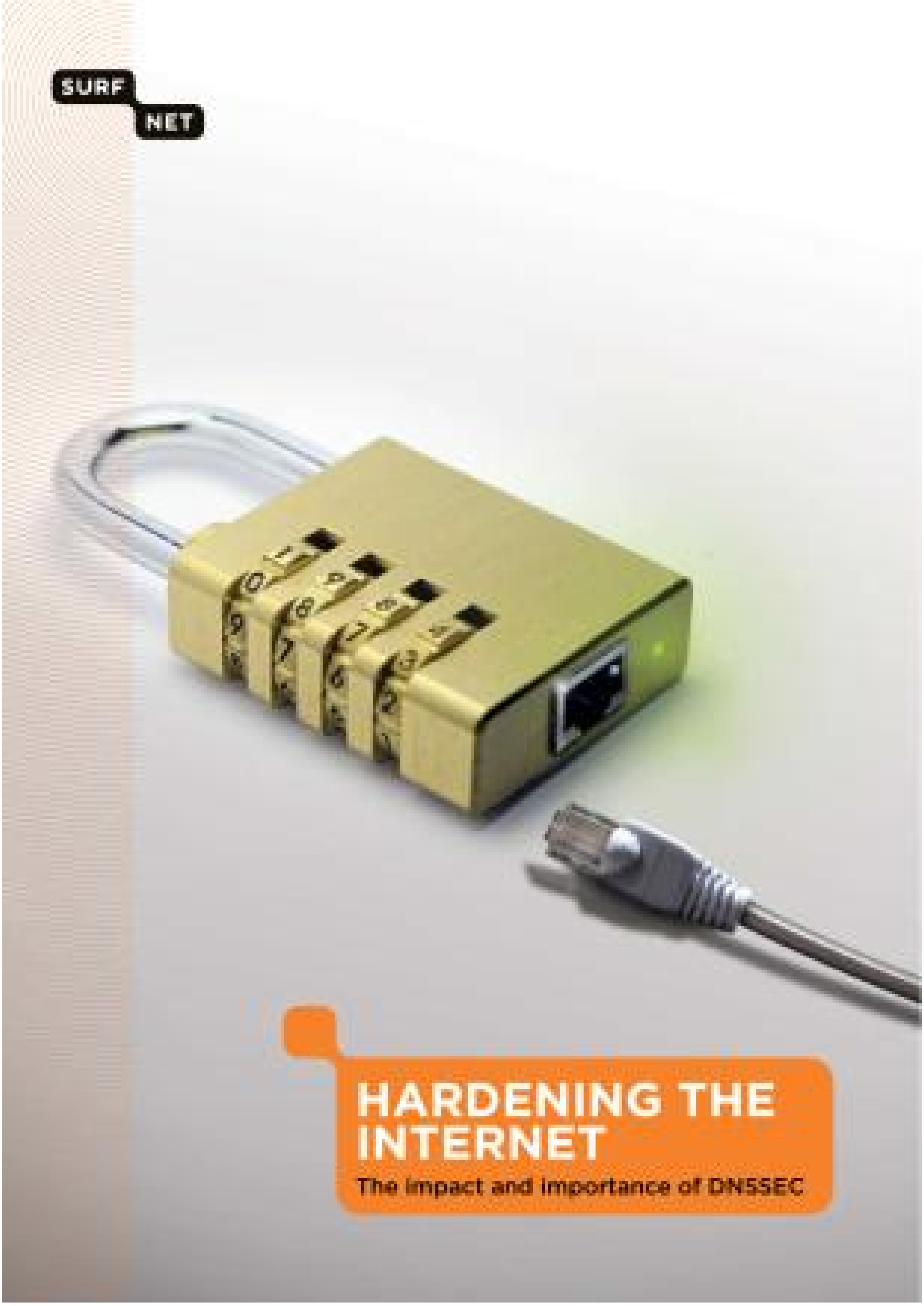
2008.07: Kaminsky announces apocalypse, saves the world.
$\Rightarrow$ New focus on DNSSEC.

2009.08.09:
941 IP addresses worldwide are running DNSSEC servers.

2010.12.24:
2536 IP addresses worldwide are running DNSSEC servers.

What is DNSSEC?

What is DNSSEC?
Is it a lock for the Internet?



What is DNSSEC?

Is it a lock for the Internet?

Or is it more like this?

What is DNSSEC?

Is it a lock for the Internet?

Or is it more like this?

Let's see what DNSSEC can do
as an amplification tool for
denial-of-service attacks.

Make list of DNSSEC domains:

```
( cd secspider.cs.ucla.edu
  awk '
    /^Zone <STRONG>/ { z = $2
      sub(/<STRONG>/,"",z)
      sub(/<\/STRONG>/,"",z)
    }
    /GREEN.*GREEN.*GREEN.*Yes/ {
      split($0,x,/<TD>/)
      sub(/<\/TD>/,"",x[5])
      print x[5],z,rand()
    }' ./*--zone.html
) | sort -k3n \
| awk '{print $1,$2}' \
> SERVERS
```

For each domain: Try query, estimate DNSSEC amplification.

```
while read ip z
do
  dig +dnssec +ignore +tries=1 \
  +time=1 any "$z" "@$ip" | \
  awk -v "z=$z" -v "ip=$ip" '{
    if ($1 != ";;") next
    if ($2 != "MSG") next
    if ($3 != "SIZE") next
    if ($4 != "rcvd:") next
    est = (22+$5)/(40+length(z))
    print est,ip,z
  }'
done < SERVERS > AMP
```

For each DNSSEC server,
find domain estimated to have
maximum DNSSEC amplification:

```
sort -nr AMP | awk '{
  if (seen[$2]) next
  if ($1 < 30) next
  print $1,$2,$3
  seen[$2] = 1
}' > MAXAMP
head -1 MAXAMP
wc -l MAXAMP
```

Output:

```
95.6279 156.154.102.26 fi.
2326 MAXAMP
```

Can that really be true?
$>$ 2000 DNSSEC servers
around the Internet, each
providing $> 30\times$ amplification
of incoming UDP packets?

Can that really be true?
$>$ 2000 DNSSEC servers
around the Internet, each
providing $> 30\times$ amplification
of incoming UDP packets?

Let's verify this.

Choose quiet test machines
on two different networks
(without egress filters).

e.g. Sender: 1.2.3.4.
Receiver: 5.6.7.8.

Run network-traffic monitors on 1.2.3.4 and 5.6.7.8.

On 1.2.3.4, set response address to 5.6.7.8, and send 1 query/second:

```
ifconfig eth0:1 \
  5.6.7.8 \
  netmask 255.255.255.255
while read est ip z
do
  dig -b 5.6.7.8 \
  +dnssec +ignore +tries=1 \
  +time=1 any "$z" "@$ip"
done < MAXAMP >/dev/null 2>&1
```

I sustained 51$\times$ amplification
of actual network traffic
in a US-to-Europe experiment
on typical university computers.

I sustained 51× amplification of actual network traffic in a US-to-Europe experiment on typical university computers.

Attacker sending 10Mbps can trigger 500Mbps flood from the DNSSEC drone pool, taking down typical site.

I sustained 51× amplification
of actual network traffic
in a US-to-Europe experiment
on typical university computers.

Attacker sending 10Mbps
can trigger 500Mbps flood from
the DNSSEC drone pool,
taking down typical site.

Attacker sending 200Mbps
can trigger 10Gbps flood,
taking down very large site.

I sustained 51× amplification
of actual network traffic
in a US-to-Europe experiment
on typical university computers.

Attacker sending 10Mbps
can trigger 500Mbps flood from
the DNSSEC drone pool,
taking down typical site.

Attacker sending 200Mbps
can trigger 10Gbps flood,
taking down very large site.

Want even more: 100Gbps?
Tell people to install DNSSEC!

Cryptographic failure patterns

Alice and Bob are communicating.
Eve is eavesdropping.

Alice and Bob have several standard security goals:

Confidentiality despite espionage.
Maybe Eve wants to acquire data.

Integrity despite corruption.
Maybe Eve wants to change data.

Availability despite sabotage.
Maybe Eve wants to destroy data.

Failure pattern #1: “The attacker isn’t sniffing our network packets so we’re secure.”

Example of this “security”:
Typical HTTP user cookies.

Failure pattern #1: "The attacker isn't sniffing our network packets so we're secure."

Example of this "security": Typical HTTP user cookies.

Failure pattern #2: "The attacker isn't forging network packets so we're secure."

Examples of this "security":

- TCP checking IP address.
- DNS checking IP address.
- New: Tcpcrypt.

Failure pattern #2: "The attacker isn't forging network packets so we're secure."

Examples of this "security":

- TCP checking IP address.
- DNS checking IP address.
- New: Tcpcrypt.

"Compare this tcpdump output, which appears encrypted . . . with the cleartext packets you would see without tcpcryptd running. . . . Active attacks are much harder as they require listening and modifying network traffic."

Failure pattern #3: "We detect corrupt data so we're secure."

Failure pattern #3: "We detect corrupt data so we're secure."

What about confidentiality? DNSSEC encrypts nothing, and broadcasts private DNS names (such as `acadmedpa.org.br`).

`dnscurve.org/nsec3walker.html`

Failure pattern #3: “We detect corrupt data so we’re secure.”

What about confidentiality?
DNSSEC encrypts nothing, and broadcasts private DNS names (such as `acadmedpa.org.br`).
dnscurve.org/nsec3walker.html

What about availability?
Eve destroys an SSH connection or an HTTPS connection or a DNSSEC lookup by forging one packet.
Eve uses the DNSSEC drones to amplify DDoS attacks.

Failure pattern #4: “The attacker doesn’t control these trusted third parties so we’re secure.”

Failure pattern #4: “The attacker doesn’t control these trusted third parties so we’re secure.”

Are the HTTPS certificate authorities all trustworthy?

Failure pattern #4: "The attacker doesn't control these trusted third parties so we're secure."

Are the HTTPS certificate authorities all trustworthy?
Is the DNS root trustworthy?

Failure pattern #5: "We're cryptographically protecting X so we're secure."

Failure pattern #5: “We’re cryptographically protecting X so we’re secure.”

Is X the complete communication from Alice to Bob, all the way from Alice to Bob?

Failure pattern #5: "We're cryptographically protecting X so we're secure."

Is X the complete communication from Alice to Bob, all the way from Alice to Bob?

Often X doesn't reach Bob.

Failure pattern #5: "We're cryptographically protecting X so we're secure."

Is X the complete communication from Alice to Bob, all the way from Alice to Bob?

Often X doesn't reach Bob.
Example: Bob views Alice's web page on his Android phone.
Phone asked hotel DNS cache for web server's address.
Eve forged the DNS response!
DNS cache checked DNSSEC but the phone didn't.

Often X isn't Alice's data.

Often X isn't Alice's data.

".ORG becomes the first open TLD to sign their zone with DNSSEC … Today we reached a significant milestone in our effort to bolster online security for the .ORG community. We are the first open generic Top-Level Domain to successfully sign our zone with Domain Name Security Extensions (DNSSEC). To date, the .ORG zone is the largest domain registry to implement this needed security measure."

What did `.org` actually sign?

2010.12.25 test:
Look up `wikipedia.org`.

The response has a *signed* statement "There might be names with hashes between `hh91kmqm332a7m6egn74ln9afi3fgk84`, `hheprfsv14o44rv9pgcndkt4thnraomv` but we haven't signed any of those names. Sincerely, `.org`"

Plus an *unsigned* statement "The `wikipedia.org` servers are 208.80.152.130, 208.80.152.142, 91.198.174.4."

Often X is horribly incomplete.

Often X is horribly incomplete.

Example: X is a server address, with a DNSSEC signature.

What Alice is sending to Bob are web pages, email, etc. Those aren't the same as X!

Often X is horribly incomplete.

Example: X is a server address, with a DNSSEC signature.

What Alice is sending to Bob are web pages, email, etc. Those aren't the same as X!

Alice can use HTTPS to protect her web pages . . . but then what attack is stopped by DNSSEC?

DNSSEC purists criticize HTTPS:
"Alice can't trust her servers."

DNSSEC signers are offline
(preferably in guarded rooms).
DNSSEC precomputes signatures.
DNSSEC doesn't trust servers.

DNSSEC purists criticize HTTPS:
“Alice can’t trust her servers.”

DNSSEC signers are offline
(preferably in guarded rooms).
DNSSEC precomputes signatures.
DNSSEC doesn’t trust servers.

... but X is still wrong!
Alice’s servers still control
all of Alice’s web pages,
unless Alice uses PGP.

With or without PGP, what
attack is stopped by DNSSEC?

Interlude: Signatures

Are precomputed signatures fundamentally a good idea?

1. They can't sign answers that are generated dynamically. Those need security too!

Interlude: Signatures

Are precomputed signatures fundamentally a good idea?

1. They can't sign answers that are generated dynamically. Those need security too!

DNSSEC purists say "Answers should always be static."

Interlude: Signatures

Are precomputed signatures fundamentally a good idea?

1. They can't sign answers that are generated dynamically. Those need security too!

DNSSEC purists say "Answers should always be static."

Imagine the web with only statically generated content: no more database integration, no more PHP, no more fun.

Interlude: Signatures

Are precomputed signatures fundamentally a good idea?

1. They can't sign answers that are generated dynamically. Those need security too!

DNSSEC purists say "Answers should always be static."

Imagine the web with only statically generated content: no more database integration, no more PHP, no more fun. As boring as `cr.yp.to`.

2. They can't sign answers to unpredictable questions.

Ask DNSSEC for `qptidszl.de`. Signed response: “There are no DNSSEC names with hashes between . . . and . . . .”

2. They can't sign answers to unpredictable questions.

Ask DNSSEC for `qptidszl.de`. Signed response: "There are no DNSSEC names with hashes between … and … ."

Attacker downloads hashes of all 457657 DNSSEC names in `.de` with $<$ 457657 queries.

Invert the hashes to find, e.g., `wedemotors.de`. Software from Ruben Niederhagen checks 1700 billion names/day on a PC with two GTX 295 graphics cards.

3. They need to be stored.

Huge deployment problems.

3. They need to be stored.
Huge deployment problems.

4. They aren't fresh.
Is an attacker replaying
obsolete signed data?

3. They need to be stored.
Huge deployment problems.

4. They aren't fresh.
Is an attacker replaying obsolete signed data?

If clocks are synchronized then signatures can include expiration times.
But frequent re-signing is an administrative disaster.

Some DNSSEC suicide examples:
2010.09.02: `.us` killed itself.
2010.10.07: `.be` killed itself.

More cryptographic failure patterns

Failure pattern #6: "We're using a cryptographic standard so we're secure."

Examples of this "security":

- DES.
- 512-bit RSA.
- 768-bit RSA.
- MD5-based certificates.

More cryptographic failure patterns

Failure pattern #6: "We're using a cryptographic standard so we're secure."

Examples of this "security":

- DES.
- 512-bit RSA.
- 768-bit RSA.
- MD5-based certificates.

Fact: By 1996, a few years after the introduction of MD5, prominent cryptographers such as Preneel and Dobbertin were calling for MD5 to be scrapped.

Failure pattern #7: “2^{80} operations are infeasible so we’re secure.”

Examples of this “security”:

- 1024-bit RSA.
- 160-bit ECC.

Failure pattern #7: “2^{80} operations are infeasible so we’re secure.”

Examples of this “security”:

- 1024-bit RSA.
- 160-bit ECC.

Is 2^{80} such a big number?
Multi-university ECC2K-130 attack is $> 10\%$ done.
Will be $\approx 2^{77}$ bit operations.

Failure pattern #7: "2^{80} operations are infeasible so we're secure."

Examples of this "security":

- 1024-bit RSA.
- 160-bit ECC.

Is 2^{80} such a big number? Multi-university ECC2K-130 attack is $> 10\%$ done. Will be $\approx 2^{77}$ bit operations.

One GTX 295 graphics card: $> 2^{69}$ bit operations/year.
2048 GTX 295 graphics cards: $> 2^{80}$ bit operations/year.

Failure pattern #8: "Even if the attacker can do 2^{80} operations, our data isn't worth that much, so we're secure."

Failure pattern #8: "Even if the attacker can do 2^{80} operations, our data isn't worth that much, so we're secure."

1. Does the attack cost so much? Radeon 5970; FPGAs; ASICs.

Failure pattern #8: "Even if the attacker can do 2^{80} operations, our data isn't worth that much, so we're secure."

1. Does the attack cost so much? Radeon 5970; FPGAs; ASICs.

2. Are *you* the only target? Can attack many keys at once, spreading costs over those keys: batch NFS, batch ECDL, etc.

Failure pattern #8: "Even if the attacker can do 2^{80} operations, our data isn't worth that much, so we're secure."

1. Does the attack cost so much? Radeon 5970; FPGAs; ASICs.

2. Are *you* the only target? Can attack many keys at once, spreading costs over those keys: batch NFS, batch ECDL, etc.

3. Is the attacker paying? Conficker broke into $> 2^{23}$ PCs.

Failure pattern #9: "This is so complicated that it must be secure."

Failure pattern #9: "This is so complicated that it must be secure." . . . and so complicated that software implementations never get it right.

Failure pattern #9: “This is so complicated that it must be secure.” . . . and so complicated that software implementations never get it right.

CVE-2009-0265: BIND DNSSEC bug $\Rightarrow$ Forge DSA-signed data.

Failure pattern #9: “This is so complicated that it must be secure.” . . . and so complicated that software implementations never get it right.

CVE-2009-0265: BIND DNSSEC bug $\Rightarrow$ Forge DSA-signed data.

CVE-2009-4022: BIND DNSSEC bug $\Rightarrow$ Forge all data.

Failure pattern #9: "This is so complicated that it must be secure." . . . and so complicated that software implementations never get it right.

CVE-2009-0265: BIND DNSSEC bug $\Rightarrow$ Forge DSA-signed data.

CVE-2009-4022: BIND DNSSEC bug $\Rightarrow$ Forge all data.

CVE-2010-0097: BIND DNSSEC bug $\Rightarrow$ Forge all data.

Failure pattern #9: "This is so complicated that it must be secure." . . . and so complicated that software implementations never get it right.

CVE-2009-0265: BIND DNSSEC bug ⇒ Forge DSA-signed data.

CVE-2009-4022: BIND DNSSEC bug ⇒ Forge all data.

CVE-2010-0097: BIND DNSSEC bug ⇒ Forge all data.

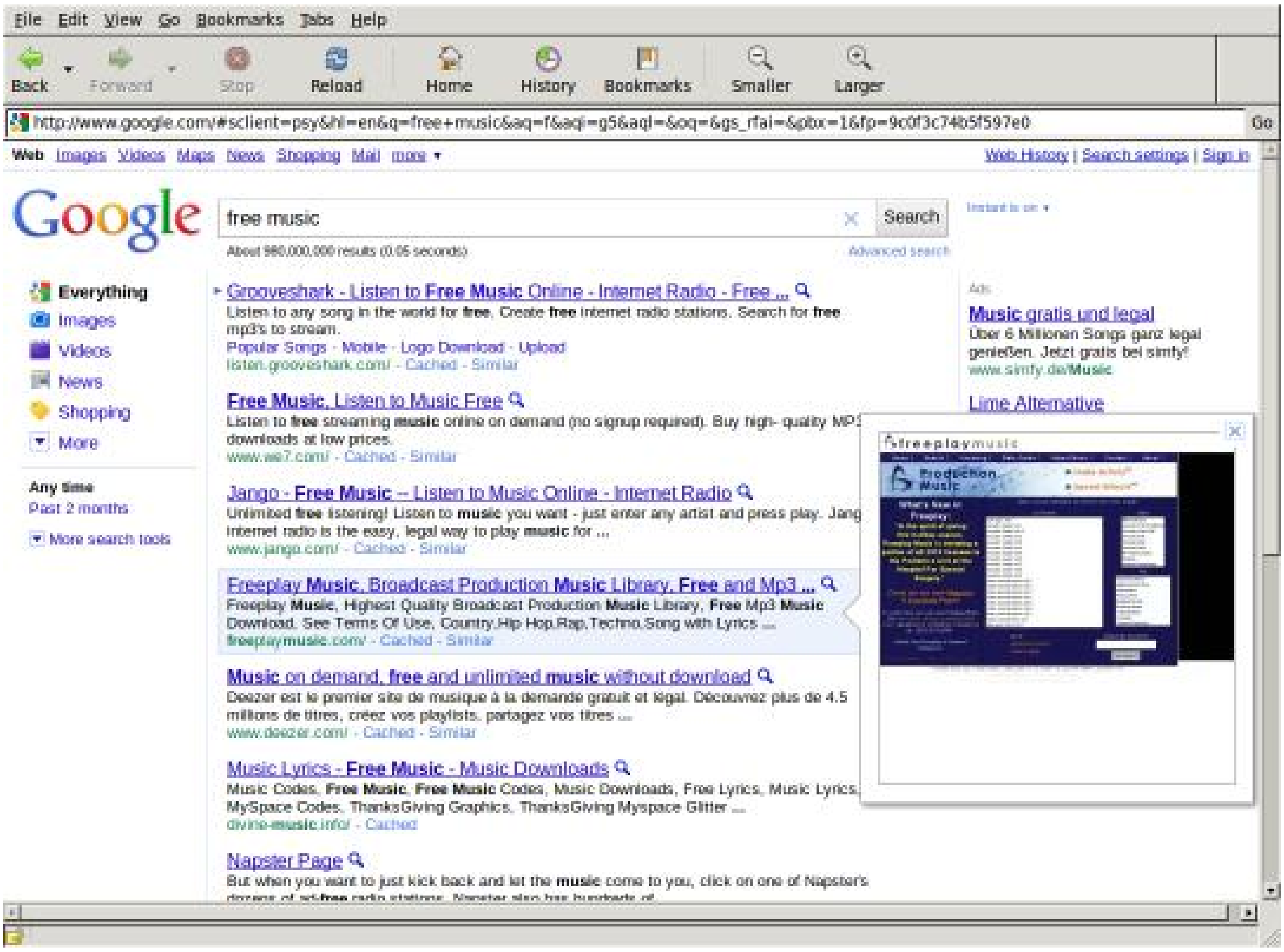
CVE-2010-0290: BIND DNSSEC bug ⇒ Forge all data.

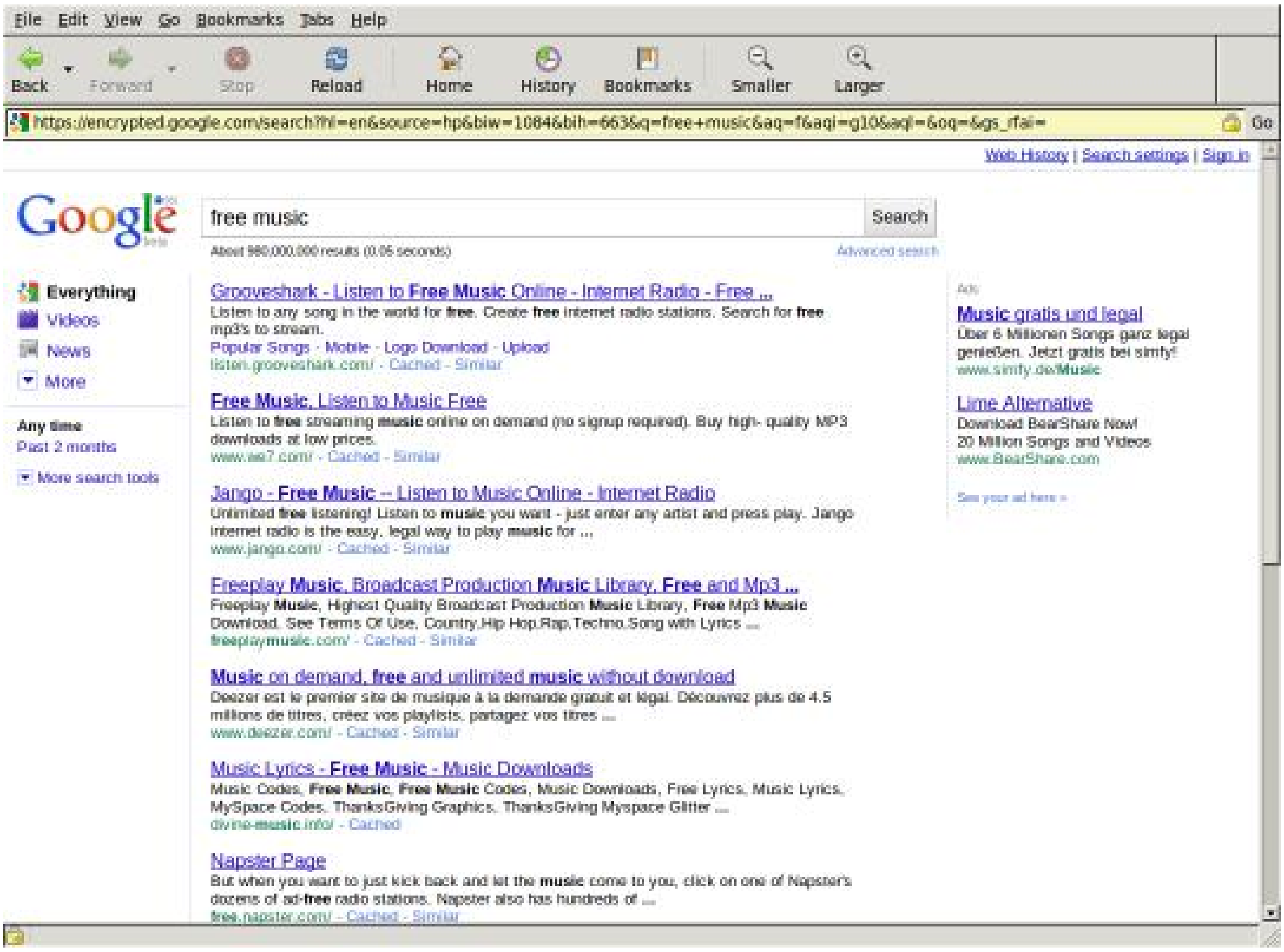
Failure pattern #10:
"Cryptography is valuable so people will deploy it."

Failure pattern #10:
"Cryptography is valuable so people will deploy it." . . . but too slow to be deployed.

Google has installed HTTPS and has fully configured it:
`https://www.google.com`
encrypts normal text search, news search, etc.

But Google doesn't allow encryption for high-volume data: images, maps, etc.


File Edit View Go Bookmarks Tabs Help
Back Forward Stop Reload Home History Bookmarks Smaller Larger
http://www.google.com/#sclient=psy&hl=en&q=free+music&aq=f&aqi=g5&aql=&oq=&gs_rfai=&pbx=1&fp=9c0f3c74b5f597e0
Go
Web Images Videos Maps News Shopping Mail more
Web History | Search settings | Sign in
free music
Search
Instant is on
About 980,000,000 results (0.05 seconds)
Advanced search
Everything
Images
Videos
News
Shopping
More
Any time
Past 2 months
More search tools
Grooveshark - Listen to Free Music Online - Internet Radio - Free ...
Listen to any song in the world for free. Create free internet radio stations. Search for free mp3's to stream.
Popular Songs - Mobile - Logo Download - Upload
listen.grooveshark.com/ - Cached - Similar
Free Music, Listen to Music Free
Listen to free streaming music online on demand (no signup required). Buy high- quality MP3 downloads at low prices.
www.we7.com/ - Cached - Similar
Jango - Free Music -- Listen to Music Online - Internet Radio
Unlimited free listening! Listen to music you want - just enter any artist and press play. Jango internet radio is the easy, legal way to play music for ...
www.jango.com/ - Cached - Similar
Freeplay Music, Broadcast Production Music Library, Free and Mp3 ...
Freeplay Music, Highest Quality Broadcast Production Music Library, Free Mp3 Music Download. See Terms Of Use, Country,Hip Hop,Rap,Techno,Song with Lyrics ...
freeplaymusic.com/ - Cached - Similar
Music on demand, free and unlimited music without download
Deezer est le premier site de musique à la demande gratuit et légal. Découvrez plus de 4.5 millions de titres, créez vos playlists, partagez vos titres ...
www.deezer.com/ - Cached - Similar
Music Lyrics - Free Music - Music Downloads
Music Codes, Free Music, Free Music Codes, Music Downloads, Free Lyrics, Music Lyrics, MySpace Codes, ThanksGiving Graphics, ThanksGiving Myspace Glitter ...
divine-music.info/ - Cached
Napster Page
But when you want to just kick back and let the music come to you, click on one of Napster's
Ads
Music gratis und legal
Über 6 Millionen Songs ganz legal genießen. Jetzt gratis bei simfy!
www.simfy.de/Music
Lime Alternative
freeplaymusic
Production Music


File Edit View Go Bookmarks Tabs Help
Back Forward Stop Reload Home History Bookmarks Smaller Larger
https://encrypted.google.com/search?hl=en&source=hp&biw=1084&bih=663&q=free+music&aq=f&aqi=g10&aql=&oq=&gs_rfai=
Go
Web History | Search settings | Sign in
free music
Search
About 980,000,000 results (0.05 seconds)
Advanced search
Everything
Videos
News
More
Any time
Past 2 months
More search tools
Grooveshark - Listen to Free Music Online - Internet Radio - Free ...
Listen to any song in the world for free. Create free internet radio stations. Search for free mp3's to stream.
Popular Songs - Mobile - Logo Download - Upload
listen.grooveshark.com/ - Cached - Similar
Free Music, Listen to Music Free
Listen to free streaming music online on demand (no signup required). Buy high- quality MP3 downloads at low prices.
www.we7.com/ - Cached - Similar
Jango - Free Music -- Listen to Music Online - Internet Radio
Unlimited free listening! Listen to music you want - just enter any artist and press play. Jango internet radio is the easy, legal way to play music for ...
www.jango.com/ - Cached - Similar
Freeplay Music, Broadcast Production Music Library, Free and Mp3 ...
Freeplay Music, Highest Quality Broadcast Production Music Library, Free Mp3 Music Download. See Terms Of Use, Country,Hip Hop,Rap,Techno,Song with Lyrics ...
freeplaymusic.com/ - Cached - Similar
Music on demand, free and unlimited music without download
Deezer est le premier site de musique à la demande gratuit et légal. Découvrez plus de 4.5 millions de titres, créez vos playlists, partagez vos titres ...
www.deezer.com/ - Cached - Similar
Music Lyrics - Free Music - Music Downloads
Music Codes, Free Music, Free Music Codes, Music Downloads, Free Lyrics, Music Lyrics, MySpace Codes, ThanksGiving Graphics, ThanksGiving Myspace Glitter ...
divine-music.info/ - Cached
Napster Page
But when you want to just kick back and let the music come to you, click on one of Napster's dozens of ad-free radio stations. Napster also has hundreds of ...
free.napster.com/ - Cached - Similar
Ads
Music gratis und legal
Über 6 Millionen Songs ganz legal genießen. Jetzt gratis bei simfy!
www.simfy.de/Music
Lime Alternative
Download BearShare Now!
20 Million Songs and Videos
www.BearShare.com
See your ad here »

A different approach

Focus on security. Assume that crypto is instantaneous.

How easily can we deploy high-security cryptography?

A different approach

Focus on security. Assume that crypto is instantaneous.

How easily can we deploy high-security cryptography?

It's safe for the moment to assume that the attacker can't do 2^{128} operations and doesn't have quantum computers. (Future: see `pqcrypto.org`.)

Safe, conservative crypto:
Strong 256-bit elliptic curve.
No degradation since 1985.

What cryptography does for us:

Alice encrypts and authenticates a message using her secret key and Bob's public key.

Bob verifies and decrypts a message using his secret key and Alice's public key.

Attacker can't understand the encrypted message and can't forge a verifiable message.

What cryptography does for us:

Alice encrypts and authenticates a packet using her secret key and Bob's public key.

Bob verifies and decrypts a packet using his secret key and Alice's public key.

Attacker can't understand the encrypted packet and can't forge a verifiable packet.

What cryptography does for us:

Alice encrypts and authenticates a packet using her secret key and Bob's public key.

Bob verifies and decrypts a packet using his secret key and Alice's public key.

Attacker can't understand the encrypted packet and can't forge a verifiable packet.

Split long messages into separately verified packets to improve availability.

Put these protected packets
inside a TCP connection,
as in SSH and HTTPS?

No. Much better availability
and much better speed:
Send packets through UDP.
Discard unverifiable packets.

Put these protected packets
inside a TCP connection,
as in SSH and HTTPS?

No. Much better availability
and much better speed:
Send packets through UDP.
Discard unverifiable packets.

“UDP is unreliable!
We want a reliable stream!”

No problem: Imitate TCP
inside the protected packets.
Simple new protocol: CurveCP.

How do we protect HTTP?

Alice starts with Bob's URL.

Alice knows her own secret key.

How does Alice learn

Bob's public key?

How do we protect HTTP?

Alice starts with Bob's URL.
Alice knows her own secret key.
How does Alice learn
Bob's public key?

"Nym" case: URL has a key!
Recognize magic number 123 in
`http://`
`   1238675309.twitter.com`
and extract key 8675309.

(Technical note: Keys are
actually longer than this,
but still fit into names.)

Normal case: URL is
`http://www.twitter.com`.

`twitter.com` DNS server
says `www.twitter.com CNAME 1238675309.twitter.com`.
Again extract key 8675309.

Long CNAME chains are bad
but short chains are okay
and very easy to deploy.

Normal case: URL is
`http://www.twitter.com`.

`twitter.com` DNS server
says `www.twitter.com CNAME`
`1238675309.twitter.com`.
Again extract key 8675309.

Long CNAME chains are bad
but short chains are okay
and very easy to deploy.

CNAME can't overlap NS.
What if URL is
`http://twitter.com`?
Answer: `twitter.com NS`
`1238675309.twitter.com`.

Alice obtains this DNS data for free as part of looking up server address.

Alice uses CurveCP to contact Bob’s web server. As fast as HTTP, but secure!

Alice obtains this DNS data
for free as part of
looking up server address.

Alice uses CurveCP to
contact Bob's web server.
As fast as HTTP, but secure!

Simplifying deployment:
Bob actually installs
a CurveCP forwarder
on UDP port 53
talking to his existing
HTTP server on TCP port 80.

How did Alice talk to
`twitter.com` DNS server?

The DNS server also has
a DNSCurve public key:
`twitter.com NS ...`

Alice obtains this DNS data
for free as part of
receiving DNS server
address from `.com` server.

Alice uses DNSCurve to
contact the DNS server.
As fast as DNS, but secure!

Standard final step:
Obtain `.com` server key
from root server.
Well-known root key.

But now I think it's better
for DNS software to know
the keys for `.com`, `.de`, etc.
Ultra-powerful root is bad.

Standard final step:
Obtain `.com` server key
from root server.
Well-known root key.

But now I think it's better
for DNS software to know
the keys for `.com`, `.de`, etc.
Ultra-powerful root is bad.

What if `.com` misbehaves?
Easily protect integrity of
web pages from the URL
`1238675309.twitter.com`
but availability is harder.
Perhaps P2P DNS can help.

Summary of deployment cost:

Alice installs DNS cache that understands DNSCurve, and installs HTTP proxy that understands CurveCP. These are small and fast and run on her laptop/phone/etc.

Bob installs small forwarder and updates his DNS records. Simple, robust, easy to use.

No changes to DNS servers, DNS databases, HTTP servers, web browsers, firewalls, etc.

Is speed a problem?

Wild speculation by Kaminsky:
Secure link from Alice's computer to Bob's DNS server means "abandoning caching . . . $100\times$ increase in load."

Is speed a problem?

Wild speculation by Kaminsky:
Secure link from Alice's computer to Bob's DNS server means "abandoning caching . . . 100× increase in load."

Reality check:

1. Measured increase: 1.15×.

Is speed a problem?

Wild speculation by Kaminsky:
Secure link from Alice's computer to Bob's DNS server means "abandoning caching . . . 100$\times$ increase in load."

Reality check:

1. Measured increase: 1.15$\times$.

2. Big DNS server operators have much higher capacity.
Why? Survive DDoS floods!

Is speed a problem?

Wild speculation by Kaminsky: Secure link from Alice's computer to Bob's DNS server means "abandoning caching . . . 100× increase in load."

Reality check:

1. Measured increase: 1.15×.

2. Big DNS server operators have much higher capacity. Why? Survive DDoS floods!

3. HTTPS can't be cached and is much bigger than DNS.

What about CPU time?

Simple `crypto_box` API from
`nacl.cace-project.eu`:

High-security curve (Curve25519).
High-security implementation
(e.g., no secret array indices).
Extensive code validation.

Very high speed:
Client and server handle
10000000 new public keys
in $<$ 10 minutes on typical CPUs.
Each public-key computation
is shared by many packets.

Post-quantum cryptography:
pqcrypto.org

Measuring DNSSEC amplification and DNSSEC privacy violations:
dnscurve.org/dnssecamp.html
dnscurve.org/nsec3walker.html

General DNSCurve information:
dnscurve.org

Installing a DNSCurve forwarder:
curvedns.on2it.net

New CurveCP mailing list:
curvecp-subscribe@
list.cr.yp.to