Cryptography in DNS

D. J. Bernstein
University of Illinois at Chicago

Exercise: How big is the
`dig +dnssec -t any se @a.ns.se` response packet?
How big was the query packet?

Some general questions

Why doesn't the Internet use cryptography?

Cryptography in DNS

D. J. Bernstein
University of Illinois at Chicago

Exercise: How big is the
`dig +dnssec -t any se @a.ns.se` response packet?
How big was the query packet?

Some general questions

Why doesn't the Internet use cryptography?

"The Internet does use cryptography! I just made an SSL connection to my bank."

Cryptography in DNS

D. J. Bernstein
University of Illinois at Chicago

Exercise: How big is the
`dig +dnssec -t any se @a.ns.se` response packet?
How big was the query packet?

Some general questions

Why doesn't the Internet use cryptography?

"The Internet does use cryptography! I just made an SSL connection to my bank."

Indeed, many connections use SSL, Skype, etc.
But *most* connections don't.

s on DNS

graphy in DNS

Bernstein
sity of Illinois at Chicago

se: How big is the
`nssec -t any se`
`.se` response packet?
ig was the query packet?

Some general questions

Why doesn't the Internet
use cryptography?

"The Internet does
use cryptography! I just made
an SSL connection to my bank."

Indeed, many connections
use SSL, Skype, etc.
But *most* connections don't.

Why is
Interne

DNS

ois at Chicago

g is the
any se
onse packet?
query packet?

Some general questions

Why doesn't the Internet
use cryptography?

"The Internet does
use cryptography! I just made
an SSL connection to my bank."

Indeed, many connections
use SSL, Skype, etc.

But *most* connections don't.

Why is there so
Internet commun

cago

et?

cket?

Some general questions

Why doesn't the Internet
use cryptography?

"The Internet does
use cryptography! I just made
an SSL connection to my bank."

Indeed, many connections
use SSL, Skype, etc.
But *most* connections don't.

Why is there so much unpr
Internet communication?

Some general questions

Why doesn't the Internet
use cryptography?

"The Internet does
use cryptography! I just made
an SSL connection to my bank."

Indeed, many connections
use SSL, Skype, etc.

But *most* connections don't.

Why is there so much unprotected
Internet communication?

Some general questions

Why doesn't the Internet
use cryptography?

"The Internet does
use cryptography! I just made
an SSL connection to my bank."

Indeed, many connections
use SSL, Skype, etc.

But *most* connections don't.

Why is there so much unprotected
Internet communication?

"Because nobody cares.
Cryptography is pointless.
Attackers are exploiting
buffer overflows; they aren't
intercepting or forging packets."

Some general questions

Why doesn't the Internet
use cryptography?

"The Internet does
use cryptography! I just made
an SSL connection to my bank."

Indeed, many connections
use SSL, Skype, etc.
But *most* connections don't.

Why is there so much unprotected
Internet communication?

"Because nobody cares.
Cryptography is pointless.
Attackers are exploiting
buffer overflows; they aren't
intercepting or forging packets."

In fact, attackers
are forging packets
and exploiting buffer overflows
and doing much more. Users
want *all* of these problems fixed.

general questions

oesn't the Internet
yptography?

nternet does
yptography! I just made
connection to my bank."

many connections
L, Skype, etc.
ost connections don't.

Why is there so much unprotected Internet communication?

"Because nobody cares.
Cryptography is pointless.
Attackers are exploiting
buffer overflows; they aren't
intercepting or forging packets."

In fact, attackers
are forging packets
and exploiting buffer overflows
and doing much more. Users
want *all* of these problems fixed.

Why ar
unencr

estions

Internet
?

es
! I just made
on to my bank."

nnections
etc.
ctions don't.

Why is there so much unprotected Internet communication?

"Because nobody cares.
Cryptography is pointless.
Attackers are exploiting
buffer overflows; they aren't
intercepting or forging packets."

In fact, attackers
are forging packets
and exploiting buffer overflows
and doing much more. Users
want *all* of these problems fixed.

Why are typical
unencrypted and

ade
bank."

't.

Why is there so much unprotected Internet communication?

"Because nobody cares.
Cryptography is pointless.
Attackers are exploiting
buffer overflows; they aren't
intercepting or forging packets."

In fact, attackers
are forging packets
and exploiting buffer overflows
and doing much more. Users
want *all* of these problems fixed.

Why are typical Internet pa
unencrypted and unauthen

Why is there so much unprotected Internet communication?

"Because nobody cares. Cryptography is pointless. Attackers are exploiting buffer overflows; they aren't intercepting or forging packets."

In fact, attackers are forging packets *and* exploiting buffer overflows *and* doing much more. Users want *all* of these problems fixed.

Why are typical Internet packets unencrypted and unauthenticated?

Why is there so much unprotected Internet communication?

"Because nobody cares. Cryptography is pointless. Attackers are exploiting buffer overflows; they aren't intercepting or forging packets."

In fact, attackers are forging packets *and* exploiting buffer overflows *and* doing much more. Users want *all* of these problems fixed.

Why are typical Internet packets unencrypted and unauthenticated?

"It's too easy to write Internet software that exchanges data without any cryptographic protection. Most Internet clients and servers don't know how to make cryptographic connections."

Why is there so much unprotected Internet communication?

"Because nobody cares. Cryptography is pointless. Attackers are exploiting buffer overflows; they aren't intercepting or forging packets."

In fact, attackers are forging packets *and* exploiting buffer overflows *and* doing much more. Users want *all* of these problems fixed.

Why are typical Internet packets unencrypted and unauthenticated?

"It's too easy to write Internet software that exchanges data without any cryptographic protection. Most Internet clients and servers don't know how to make cryptographic connections."

True for most protocols. But let's focus on HTTP. Most HTTP servers and browsers (Apache, Internet Explorer, Firefox, etc.) support SSL.

there so much unprotected
t communication?

use nobody cares.
graphy is pointless.
ers are exploiting
overflows; they aren't
pting or forging packets."

, attackers
ging packets
ploiting buffer overflows
oing much more. Users
// of these problems fixed.

Why are typical Internet packets unencrypted and unauthenticated?

"It's too easy to write Internet software that exchanges data without any cryptographic protection. Most Internet clients and servers don't know how to make cryptographic connections."

True for most protocols.
But let's focus on HTTP.
Most HTTP servers and browsers (Apache, Internet Explorer, Firefox, etc.) support SSL.

Why is
fraction

much unprotected
nication?

y cares.
pointless.
ploiting
they aren't
orging packets."
s

ets
uffer overflows
more. Users
problems fixed.

Why are typical Internet packets unencrypted and unauthenticated?

"It's too easy to write Internet software that exchanges data without any cryptographic protection. Most Internet clients and servers don't know how to make cryptographic connections."

True for most protocols.
But let's focus on HTTP.
Most HTTP servers and browsers (Apache, Internet Explorer, Firefox, etc.) support SSL.

Why is SSL used
fraction of all HT

rotected

't
kets."

lows
ers
fixed.

Why are typical Internet packets unencrypted and unauthenticated?

"It's too easy to write Internet software that exchanges data without any cryptographic protection. Most Internet clients and servers don't know how to make cryptographic connections."

True for most protocols.
But let's focus on HTTP.
Most HTTP servers and browsers (Apache, Internet Explorer, Firefox, etc.) support SSL.

Why is SSL used for only a fraction of all HTTP conne

Why are typical Internet packets unencrypted and unauthenticated?

"It's too easy to write Internet software that exchanges data without any cryptographic protection. Most Internet clients and servers don't know how to make cryptographic connections."

Why is SSL used for only a tiny fraction of all HTTP connections?

True for most protocols.

But let's focus on HTTP.

Most HTTP servers and browsers (Apache, Internet Explorer, Firefox, etc.) support SSL.

Why are typical Internet packets unencrypted and unauthenticated?

"It's too easy to write Internet software that exchanges data without any cryptographic protection. Most Internet clients and servers don't know how to make cryptographic connections."

True for most protocols.
But let's focus on HTTP.
Most HTTP servers and browsers (Apache, Internet Explorer, Firefox, etc.) support SSL.

Why is SSL used for only a tiny fraction of all HTTP connections?

"Have you ever tried to set up SSL? Do you want to go through all these extra Apache configuration steps? Do you want to pay for a certificate? Do you want to annoy your web-site visitors with self-signed certificates?"

Why are typical Internet packets unencrypted and unauthenticated?

"It's too easy to write Internet software that exchanges data without any cryptographic protection. Most Internet clients and servers don't know how to make cryptographic connections."

True for most protocols.
But let's focus on HTTP.
Most HTTP servers and browsers (Apache, Internet Explorer, Firefox, etc.) support SSL.

Why is SSL used for only a tiny fraction of all HTTP connections?

"Have you ever tried to set up SSL? Do you want to go through all these extra Apache configuration steps? Do you want to pay for a certificate? Do you want to annoy your web-site visitors with self-signed certificates?"

Indeed, usability is a major issue.
Only $\approx$1% of the Apache servers on the Internet have SSL enabled.

re typical Internet packets
ypted and unauthenticated?

oo easy to write Internet
re that exchanges data
t any cryptographic
tion. Most Internet clients
rvers don't know how to
cryptographic connections."

or most protocols.
t's focus on HTTP.
HTTP servers and browsers
ne, Internet Explorer,
, etc.) support SSL.

Why is SSL used for only a tiny fraction of all HTTP connections?

"Have you ever tried to set up SSL? Do you want to go through all these extra Apache configuration steps? Do you want to pay for a certificate? Do you want to annoy your web-site visitors with self-signed certificates?"

Indeed, usability is a major issue. Only $\approx$1% of the Apache servers on the Internet have SSL enabled.

But let

Google
paid fo
Google
`https`

Internet packets
unauthenticated?

write Internet
changes data
tographic
Internet clients
know how to
hic connections."

otocols.
n HTTP.
vers and browsers
t Explorer,
pport SSL.

Why is SSL used for only a tiny fraction of all HTTP connections?

"Have you ever tried to set up SSL? Do you want to go through all these extra Apache configuration steps? Do you want to pay for a certificate? Do you want to annoy your web-site visitors with self-signed certificates?"

Indeed, usability is a major issue. Only $\approx$1% of the Apache servers on the Internet have SSL enabled.

But let's focus o

Google has alrea
paid for a certific
Google uses SSL
`https://mail.g`

ackets
ticated?

rnet
ata

clients
w to
ctions."

rowsers
,

Why is SSL used for only a tiny fraction of all HTTP connections?

"Have you ever tried to set up SSL? Do you want to go through all these extra Apache configuration steps? Do you want to pay for a certificate? Do you want to annoy your web-site visitors with self-signed certificates?"

Indeed, usability is a major issue. Only ≈1% of the Apache servers on the Internet have SSL enabled.

But let's focus on Google.

Google has already paid for a certificate. Google uses SSL for `https://mail.google.c`

Why is SSL used for only a tiny fraction of all HTTP connections?

"Have you ever tried to set up SSL? Do you want to go through all these extra Apache configuration steps? Do you want to pay for a certificate? Do you want to annoy your web-site visitors with self-signed certificates?"

Indeed, usability is a major issue. Only $\approx$1% of the Apache servers on the Internet have SSL enabled.

But let's focus on Google.

Google has already paid for a certificate. Google uses SSL for `https://mail.google.com`.

Why is SSL used for only a tiny fraction of all HTTP connections?

"Have you ever tried to set up SSL? Do you want to go through all these extra Apache configuration steps? Do you want to pay for a certificate? Do you want to annoy your web-site visitors with self-signed certificates?"

Indeed, usability is a major issue. Only $\approx$1% of the Apache servers on the Internet have SSL enabled.

But let's focus on Google.

Google has already paid for a certificate. Google uses SSL for `https://mail.google.com`.

If you connect to `https://www.google.com`, Google redirects your browser to `http://www.google.com`.

SSL used for only a tiny
n of all HTTP connections?

you ever tried to set
L? Do you want to go
h all these extra Apache
uration steps? Do you
o pay for a certificate?
u want to annoy your
te visitors with self-signed
ates?"

, usability is a major issue.
≈1% of the Apache servers
Internet have SSL enabled.

But let's focus on Google.

Google has already
paid for a certificate.
Google uses SSL for
`https://mail.google.com`.

If you connect to
`https://www.google.com`,
Google redirects your browser to
`http://www.google.com`.

Why d
turn of

for only a tiny
TTP connections?

tried to set
want to go
extra Apache
ps? Do you
a certificate?
annoy your
with self-signed

is a major issue.
e Apache servers
have SSL enabled.

But let's focus on Google.

Google has already
paid for a certificate.
Google uses SSL for
`https://mail.google.com`.

If you connect to
`https://www.google.com`,
Google redirects your browser to
`http://www.google.com`.

Why does Google
turn off cryptogr

a tiny
ections?

t
go
ache
ou
te?
r
igned

issue.
servers
enabled.

But let's focus on Google.

Google has already
paid for a certificate.
Google uses SSL for
`https://mail.google.com`.

If you connect to
`https://www.google.com`,
Google redirects your browser to
`http://www.google.com`.

Why does Google actively
turn off cryptographic prot

But let's focus on Google.

Google has already
paid for a certificate.
Google uses SSL for
`https://mail.google.com`.

If you connect to
`https://www.google.com`,
Google redirects your browser to
`http://www.google.com`.

Why does Google actively
turn off cryptographic protection?

But let's focus on Google.

Google has already
paid for a certificate.
Google uses SSL for
`https://mail.google.com`.

If you connect to
`https://www.google.com`,
Google redirects your browser to
`http://www.google.com`.

Why does Google actively
turn off cryptographic protection?

"Enabling SSL
for more than a small fraction
of Google connections would
overload the Google servers.
Google doesn't want to pay for
a bunch of extra computers.
Too slow ⇒ unusable."

But let's focus on Google.

Google has already
paid for a certificate.
Google uses SSL for
`https://mail.google.com`.

If you connect to
`https://www.google.com`,
Google redirects your browser to
`http://www.google.com`.

Why does Google actively
turn off cryptographic protection?

"Enabling SSL
for more than a small fraction
of Google connections would
overload the Google servers.
Google doesn't want to pay for
a bunch of extra computers.
Too slow ⇒ unusable."

Many companies sell
SSL-acceleration hardware,
but that costs money too.

's focus on Google.

has already
r a certificate.
uses SSL for
://mail.google.com.

connect to
://www.google.com,
redirects your browser to
//www.google.com.

Why does Google actively
turn off cryptographic protection?

"Enabling SSL
for more than a small fraction
of Google connections would
overload the Google servers.
Google doesn't want to pay for
a bunch of extra computers.
Too slow $\Rightarrow$ unusable."

Many companies sell
SSL-acceleration hardware,
but that costs money too.

Why ar
comput

n Google.

dy
cate.
for
google.com.

oogle.com,
your browser to
ogle.com.

Why does Google actively
turn off cryptographic protection?

"Enabling SSL
for more than a small fraction
of Google connections would
overload the Google servers.
Google doesn't want to pay for
a bunch of extra computers.
Too slow ⇒ unusable."

Many companies sell
SSL-acceleration hardware,
but that costs money too.

Why are cryptog
computations so

om.

m,

ser to

.

Why does Google actively
turn off cryptographic protection?

"Enabling SSL
for more than a small fraction
of Google connections would
overload the Google servers.
Google doesn't want to pay for
a bunch of extra computers.
Too slow ⇒ unusable."

Many companies sell
SSL-acceleration hardware,
but that costs money too.

Why are cryptographic
computations so expensive

Why does Google actively
turn off cryptographic protection?

"Enabling SSL
for more than a small fraction
of Google connections would
overload the Google servers.
Google doesn't want to pay for
a bunch of extra computers.
Too slow ⇒ unusable."

Many companies sell
SSL-acceleration hardware,
but that costs money too.

Why are cryptographic
computations so expensive?

Why does Google actively
turn off cryptographic protection?

"Enabling SSL
for more than a small fraction
of Google connections would
overload the Google servers.
Google doesn't want to pay for
a bunch of extra computers.
Too slow ⇒ unusable."

Many companies sell
SSL-acceleration hardware,
but that costs money too.

Why are cryptographic
computations so expensive?

Can crypto be faster,
without being easy to break?

Why does Google actively
turn off cryptographic protection?

"Enabling SSL
for more than a small fraction
of Google connections would
overload the Google servers.
Google doesn't want to pay for
a bunch of extra computers.
Too slow $\Rightarrow$ unusable."

Many companies sell
SSL-acceleration hardware,
but that costs money too.

Why are cryptographic
computations so expensive?

Can crypto be faster,
without being easy to break?

Can crypto be fast enough
to solidly protect all of
Google's communications?

Why does Google actively
turn off cryptographic protection?

"Enabling SSL
for more than a small fraction
of Google connections would
overload the Google servers.
Google doesn't want to pay for
a bunch of extra computers.
Too slow ⇒ unusable."

Many companies sell
SSL-acceleration hardware,
but that costs money too.

Why are cryptographic
computations so expensive?

Can crypto be faster,
without being easy to break?

Can crypto be fast enough
to solidly protect all of
Google's communications?

Can crypto be fast enough
to protect every Internet packet?

Why does Google actively
turn off cryptographic protection?

"Enabling SSL
for more than a small fraction
of Google connections would
overload the Google servers.
Google doesn't want to pay for
a bunch of extra computers.
Too slow $\Rightarrow$ unusable."

Many companies sell
SSL-acceleration hardware,
but that costs money too.

Why are cryptographic
computations so expensive?

Can crypto be faster,
without being easy to break?

Can crypto be fast enough
to solidly protect all of
Google's communications?

Can crypto be fast enough
to protect every Internet packet?

Can universal crypto be *usable*?

oes Google actively
ff cryptographic protection?

ing SSL
re than a small fraction
gle connections would
d the Google servers.
doesn't want to pay for
h of extra computers.
ow $\Rightarrow$ unusable."

companies sell
celeration hardware,
at costs money too.

Why are cryptographic
computations so expensive?

Can crypto be faster,
without being easy to break?

Can crypto be fast enough
to solidly protect all of
Google's communications?

Can crypto be fast enough
to protect every Internet packet?

Can universal crypto be *usable*?

What c

Crypto
stop sn
by scra

Crypto
as prot
attacke
the scr

Can als
attacke
a prope

e actively
raphic protection?

small fraction
ctions would
ogle servers.
vant to pay for
computers.
sable."

sell
hardware,
oney too.

Why are cryptographic computations so expensive?

Can crypto be faster, without being easy to break?

Can crypto be fast enough to solidly protect all of Google's communications?

Can crypto be fast enough to protect every Internet packet?

Can universal crypto be *usable*?

What cryptograp

Cryptography ca
stop sniffing atta
by scrambling leg

Cryptography is
as protecting co
attackers can't u
the scrambled pa

Can also protect
attackers can't fi
a properly scram

tection?

tion
ld
s.
y for
s.

Why are cryptographic
computations so expensive?

Can crypto be faster,
without being easy to break?

Can crypto be fast enough
to solidly protect all of
Google's communications?

Can crypto be fast enough
to protect every Internet packet?

Can universal crypto be *usable*?

What cryptography can do

Cryptography can
stop sniffing attackers
by scrambling legitimate pa

Cryptography is often desc
as protecting confidentialit
attackers can't understand
the scrambled packets.

Can also protect integrity:
attackers can't figure out
a properly scrambled forge

Why are cryptographic computations so expensive?

Can crypto be faster, without being easy to break?

Can crypto be fast enough to solidly protect all of Google's communications?

Can crypto be fast enough to protect every Internet packet?

Can universal crypto be *usable*?

What cryptography can do

Cryptography can stop sniffing attackers by scrambling legitimate packets.

Cryptography is often described as protecting confidentiality: attackers can't understand the scrambled packets.

Can also protect integrity: attackers can't figure out a properly scrambled forgery.

re cryptographic
tations so expensive?

ypto be faster,
t being easy to break?

ypto be fast enough
dly protect all of
's communications?

ypto be fast enough
ect every Internet packet?

niversal crypto be *usable*?

What cryptography can do

Cryptography can
stop sniffing attackers
by scrambling legitimate packets.

Cryptography is often described
as protecting confidentiality:
attackers can't understand
the scrambled packets.

Can also protect integrity:
attackers can't figure out
a properly scrambled forgery.

Traditi
each le
to shar

Public-
has mu
(1976 D
many s

Each p
Two pa
securel
the oth

1993:
project
signatu

raphic
expensive?

ster,
sy to break?

st enough
all of
nications?

st enough
Internet packet?

ypto be *usable*?

What cryptography can do

Cryptography can
stop sniffing attackers
by scrambling legitimate packets.

Cryptography is often described
as protecting confidentiality:
attackers can't understand
the scrambled packets.

Can also protect integrity:
attackers can't figure out
a properly scrambled forgery.

Traditional crypt
each legitimate c
to share a secret

Public-key crypto
has much lower r
(1976 Diffie–Hell
many subsequent

Each party has o
Two parties can
securely if each p
the other party's

1993: IETF begi
project to add pu
signatures to DN

?

k?

acket?

able?

What cryptography can do

Cryptography can
stop sniffing attackers
by scrambling legitimate packets.

Cryptography is often described
as protecting confidentiality:
attackers can't understand
the scrambled packets.

Can also protect integrity:
attackers can't figure out
a properly scrambled forgery.

Traditional cryptography re
each legitimate client-serve
to share a secret key.

Public-key cryptography
has much lower requiremen
(1976 Diffie–Hellman;
many subsequent refineme

Each party has one public
Two parties can communic
securely if each party know
the other party's public key

1993: IETF begins "DNSS
project to add public-key
signatures to DNS.

What cryptography can do

Cryptography can
stop sniffing attackers
by scrambling legitimate packets.

Cryptography is often described
as protecting confidentiality:
attackers can't understand
the scrambled packets.

Can also protect integrity:
attackers can't figure out
a properly scrambled forgery.

Traditional cryptography requires
each legitimate client-server pair
to share a secret key.

Public-key cryptography
has much lower requirements.
(1976 Diffie–Hellman;
many subsequent refinements)

Each party has one public key.
Two parties can communicate
securely if each party knows
the other party's public key.

1993: IETF begins "DNSSEC"
project to add public-key
signatures to DNS.

cryptography can do

graphy can
iffing attackers
mbling legitimate packets.

graphy is often described
ecting confidentiality:
ers can't understand
ambled packets.

so protect integrity:
ers can't figure out
erly scrambled forgery.

Traditional cryptography requires
each legitimate client-server pair
to share a secret key.

Public-key cryptography
has much lower requirements.
(1976 Diffie–Hellman;
many subsequent refinements)

Each party has one public key.
Two parties can communicate
securely if each party knows
the other party's public key.

1993: IETF begins "DNSSEC"
project to add public-key
signatures to DNS.

Paul V

This
deep
in b
impl
take
a se
solu
anot
in w
and
befo
the

BIND 8

[Top
DNS

BIND 9

[Top

<u>phy can do</u>

n

ckers

gitimate packets.

often described

nfidentiality:

nderstand

ackets.

integrity:

gure out

bled forgery.

Traditional cryptography requires each legitimate client-server pair to share a secret key.

Public-key cryptography has much lower requirements. (1976 Diffie–Hellman; many subsequent refinements)

Each party has one public key. Two parties can communicate securely if each party knows the other party's public key.

1993: IETF begins "DNSSEC" project to add public-key signatures to DNS.

Paul Vixie, 1995.

> This sounds sim
> deep reaching co
> in both the prot
> implementation–
> taken more than
> a security model
> solution. We ex
> another year bef
> in wide use on t
> and at least a y
> before its use is
> the Internet.

BIND 8.2 blurb,

> [Top feature:] P
> DNSSEC.

BIND 9 blurb, 20

> [Top feature:] D

ackets.

ribed

y:

ry.

Traditional cryptography requires each legitimate client-server pair to share a secret key.

Public-key cryptography has much lower requirements. (1976 Diffie–Hellman; many subsequent refinements)

Each party has one public key. Two parties can communicate securely if each party knows the other party's public key.

1993: IETF begins "DNSSEC" project to add public-key signatures to DNS.

Paul Vixie, 1995.06:

> This sounds simple but it ha deep reaching consequences in both the protocol and the implementation—which is wh taken more than a year to c a security model and design solution. We expect it to be another year before DNSSEC in wide use on the leading e and at least a year after tha before its use is commonplac the Internet.

BIND 8.2 blurb, 1999.03:

> [Top feature:] Preliminary DNSSEC.

BIND 9 blurb, 2000.09:

> [Top feature:] DNSSEC.

Traditional cryptography requires each legitimate client-server pair to share a secret key.

Public-key cryptography has much lower requirements. (1976 Diffie–Hellman; many subsequent refinements)

Each party has one public key. Two parties can communicate securely if each party knows the other party's public key.

1993: IETF begins "DNSSEC" project to add public-key signatures to DNS.

Paul Vixie, 1995.06:

> This sounds simple but it has deep reaching consequences in both the protocol and the implementation—which is why it's taken more than a year to choose a security model and design a solution. We expect it to be another year before DNSSEC is in wide use on the leading edge, and at least a year after that before its use is commonplace on the Internet.

BIND 8.2 blurb, 1999.03:

> [Top feature:] Preliminary DNSSEC.

BIND 9 blurb, 2000.09:

> [Top feature:] DNSSEC.

onal cryptography requires
gitimate client-server pair
e a secret key.

-key cryptography
uch lower requirements.
Diffie–Hellman;
subsequent refinements)

arty has one public key.
arties can communicate
y if each party knows
her party's public key.

IETF begins "DNSSEC"
to add public-key
ures to DNS.

Paul Vixie, 1995.06:

> This sounds simple but it has deep reaching consequences in both the protocol and the implementation—which is why it's taken more than a year to choose a security model and design a solution. We expect it to be another year before DNSSEC is in wide use on the leading edge, and at least a year after that before its use is commonplace on the Internet.

BIND 8.2 blurb, 1999.03:

> [Top feature:] Preliminary DNSSEC.

BIND 9 blurb, 2000.09:

> [Top feature:] DNSSEC.

Paul V

> We
> on w
> work
> thre
> "NO
> TIM
> som
> . . .
> . . .
>
> It's
> mor
> it's
> sure
> and
> spec
> don'
>
> 253
> The
> start

ography requires
client-server pair
key.

ography
requirements.
man;
t refinements)

ne public key.
communicate
party knows
public key.

ns “DNSSEC”
ublic-key
IS.

Paul Vixie, 1995.06:

> This sounds simple but it has deep reaching consequences in both the protocol and the implementation—which is why it's taken more than a year to choose a security model and design a solution. We expect it to be another year before DNSSEC is in wide use on the leading edge, and at least a year after that before its use is commonplace on the Internet.

BIND 8.2 blurb, 1999.03:

> [Top feature:] Preliminary DNSSEC.

BIND 9 blurb, 2000.09:

> [Top feature:] DNSSEC.

Paul Vixie, 2002.

> We are still doin
> on what kind of
> work for DNS se
> three or four tim
> “NOW we've go
> TIME for sure”
> some humility in
> . . . “Wonder if
> . . .
>
> It's impossible t
> more flag days w
> it's safe to burn
> sure isn't plain
> and it sure isn't
> specified. When
> don't know. . .
>
> 2535 is already
> There is no inst
> starting from sc

equires
er pair

nts.

nts)

key.

cate
ys
y.

EC”

Paul Vixie, 1995.06:

> This sounds simple but it has deep reaching consequences in both the protocol and the implementation—which is why it's taken more than a year to choose a security model and design a solution. We expect it to be another year before DNSSEC is in wide use on the leading edge, and at least a year after that before its use is commonplace on the Internet.

BIND 8.2 blurb, 1999.03:

> [Top feature:] Preliminary DNSSEC.

BIND 9 blurb, 2000.09:

> [Top feature:] DNSSEC.

Paul Vixie, 2002.11:

> We are still doing basic rese
> on what kind of data model
> work for DNS security. Afte
> three or four times of saying
> “NOW we've got it, THIS
> TIME for sure” there's finall
> some humility in the picture
> . . . “Wonder if THIS'll work
> . . .
>
> It's impossible to know how
> more flag days we'll have be
> it's safe to burn ROMs . . .
> sure isn't plain old SIG+KE
> and it sure isn't DS as curre
> specified. When will it be?
> don't know. . . .
>
> 2535 is already dead and bu
> There is no installed base. W
> starting from scratch.

Paul Vixie, 1995.06:

> This sounds simple but it has deep reaching consequences in both the protocol and the implementation—which is why it's taken more than a year to choose a security model and design a solution. We expect it to be another year before DNSSEC is in wide use on the leading edge, and at least a year after that before its use is commonplace on the Internet.

BIND 8.2 blurb, 1999.03:

> [Top feature:] Preliminary DNSSEC.

BIND 9 blurb, 2000.09:

> [Top feature:] DNSSEC.

Paul Vixie, 2002.11:

> We are still doing basic research on what kind of data model will work for DNS security. After three or four times of saying "NOW we've got it, THIS TIME for sure" there's finally some humility in the picture . . . "Wonder if THIS'll work?" . . .
>
> It's impossible to know how many more flag days we'll have before it's safe to burn ROMs . . . It sure isn't plain old SIG+KEY, and it sure isn't DS as currently specified. When will it be? We don't know. . . .
>
> 2535 is already dead and buried. There is no installed base. We're starting from scratch.

ixie, 1995.06:

s sounds simple but it has
p reaching consequences
oth the protocol and the
ementation—which is why it's
n more than a year to choose
curity model and design a
tion. We expect it to be
ther year before DNSSEC is
ide use on the leading edge,
at least a year after that
re its use is commonplace on
Internet.

8.2 blurb, 1999.03:

p feature:] Preliminary
SSEC.

9 blurb, 2000.09:

p feature:] DNSSEC.

Paul Vixie, 2002.11:

> We are still doing basic research on what kind of data model will work for DNS security. After three or four times of saying "NOW we've got it, THIS TIME for sure" there's finally some humility in the picture . . . "Wonder if THIS'll work?" . . .
>
> It's impossible to know how many more flag days we'll have before it's safe to burn ROMs . . . It sure isn't plain old SIG+KEY, and it sure isn't DS as currently specified. When will it be? We don't know. . . .
>
> 2535 is already dead and buried. There is no installed base. We're starting from scratch.

Paul V
announ

BIN

06:

ple but it has
onsequences
tocol and the
—which is why it's
a year to choose
l and design a
pect it to be
fore DNSSEC is
the leading edge,
ear after that
commonplace on

1999.03:

Preliminary

000.09:

ONSSEC.

Paul Vixie, 2002.11:

> We are still doing basic research on what kind of data model will work for DNS security. After three or four times of saying "NOW we've got it, THIS TIME for sure" there's finally some humility in the picture . . . "Wonder if THIS'll work?" . . .
>
> It's impossible to know how many more flag days we'll have before it's safe to burn ROMs . . . It sure isn't plain old SIG+KEY, and it sure isn't DS as currently specified. When will it be? We don't know. . . .
>
> 2535 is already dead and buried. There is no installed base. We're starting from scratch.

Paul Vixie, 2004.

announcing BIND

BIND 9.3 will sh

as
e
hy it's
choose
a
e
C is
edge,
nt
ce on

Paul Vixie, 2002.11:

> We are still doing basic research on what kind of data model will work for DNS security. After three or four times of saying "NOW we've got it, THIS TIME for sure" there's finally some humility in the picture . . . "Wonder if THIS'll work?" . . .
>
> It's impossible to know how many more flag days we'll have before it's safe to burn ROMs . . . It sure isn't plain old SIG+KEY, and it sure isn't DS as currently specified. When will it be? We don't know. . . .
>
> 2535 is already dead and buried. There is no installed base. We're starting from scratch.

Paul Vixie, 2004.04.20, announcing BIND 9.3 beta

> BIND 9.3 will ship with DN

Paul Vixie, 2002.11:

> We are still doing basic research on what kind of data model will work for DNS security. After three or four times of saying "NOW we've got it, THIS TIME for sure" there's finally some humility in the picture . . . "Wonder if THIS'll work?" . . .
>
> It's impossible to know how many more flag days we'll have before it's safe to burn ROMs . . . It sure isn't plain old SIG+KEY, and it sure isn't DS as currently specified. When will it be? We don't know. . . .
>
> 2535 is already dead and buried. There is no installed base. We're starting from scratch.

Paul Vixie, 2004.04.20, announcing BIND 9.3 beta:

> BIND 9.3 will ship with DNSSEC

Paul Vixie, 2002.11:

> We are still doing basic research on what kind of data model will work for DNS security. After three or four times of saying "NOW we've got it, THIS TIME for sure" there's finally some humility in the picture . . . "Wonder if THIS'll work?" . . .
>
> It's impossible to know how many more flag days we'll have before it's safe to burn ROMs . . . It sure isn't plain old SIG+KEY, and it sure isn't DS as currently specified. When will it be? We don't know. . . .
>
> 2535 is already dead and buried. There is no installed base. We're starting from scratch.

Paul Vixie, 2004.04.20, announcing BIND 9.3 beta:

> BIND 9.3 will ship with DNSSEC support turned off by default in the configuration file.

Paul Vixie, 2002.11:

> We are still doing basic research on what kind of data model will work for DNS security. After three or four times of saying "NOW we've got it, THIS TIME for sure" there's finally some humility in the picture . . . "Wonder if THIS'll work?" . . .
>
> It's impossible to know how many more flag days we'll have before it's safe to burn ROMs . . . It sure isn't plain old SIG+KEY, and it sure isn't DS as currently specified. When will it be? We don't know. . . .
>
> 2535 is already dead and buried. There is no installed base. We're starting from scratch.

Paul Vixie, 2004.04.20, announcing BIND 9.3 beta:

> BIND 9.3 will ship with DNSSEC support turned off by default in the configuration file.
>
> . . .
>
> ISC will also begin offering direct support to users of BIND through the sale of annual support contracts.

ixie, 2002.11:

are still doing basic research
what kind of data model will
k for DNS security. After
e or four times of saying
OW we've got it, THIS
IE for sure" there's finally
e humility in the picture
"Wonder if THIS'll work?"

impossible to know how many
e flag days we'll have before
safe to burn ROMs . . . It
isn't plain old SIG+KEY,
it sure isn't DS as currently
cified. When will it be? We
't know. . . .

5 is already dead and buried.
re is no installed base. We're
ting from scratch.

Paul Vixie, 2004.04.20, announcing BIND 9.3 beta:

> BIND 9.3 will ship with DNSSEC support turned off by default in the configuration file.
>
> . . .
>
> ISC will also begin offering direct support to users of BIND through the sale of annual support contracts.

Paul V

Had
ten
not
inte
laws
we
NSE
as f
actu

11:

ng basic research
data model will
ecurity. After
nes of saying
t it, THIS
there's finally
the picture
THIS'll work?"

o know how many
we'll have before
ROMs . . . It
old SIG+KEY,
DS as currently
will it be? We
.

dead and buried.
alled base. We're
ratch.

Paul Vixie, 2004.04.20,

announcing BIND 9.3 beta:

> BIND 9.3 will ship with DNSSEC support turned off by default in the configuration file.
> . . .
>
> ISC will also begin offering direct support to users of BIND through the sale of annual support contracts.

Paul Vixie, 2005.

> Had we done a
> ten years ago . .
> not have noticed
> intersect their na
> laws or business
> we might still ha
> NSEC3 juggerna
> as far off the ra
> actually are now

arch
will
r
g
ly
?”
many
efore
lt
Y,
ently
We
ried.
We’re

Paul Vixie, 2004.04.20,

announcing BIND 9.3 beta:

> BIND 9.3 will ship with DNSSEC support turned off by default in the configuration file.
> . . .
>
> ISC will also begin offering direct support to users of BIND through the sale of annual support contracts.

Paul Vixie, 2005.11.01:

> Had we done a requirements
> ten years ago . . . they migh
> not have noticed that it wou
> intersect their national priva
> laws or business requirement
> we might still have run into
> NSEC3 juggernaut and be ju
> as far off the rails now as w
> actually are now.

Paul Vixie, 2004.04.20,

announcing BIND 9.3 beta:

> BIND 9.3 will ship with DNSSEC support turned off by default in the configuration file.
>
> . . .
>
> ISC will also begin offering direct support to users of BIND through the sale of annual support contracts.

Paul Vixie, 2005.11.01:

> Had we done a requirements doc ten years ago . . . they might not have noticed that it would intersect their national privacy laws or business requirements, we might still have run into the NSEC3 juggernaut and be just as far off the rails now as we actually are now.

ixie, 2004.04.20,

ncing BIND 9.3 beta:

D 9.3 will ship with DNSSEC
port turned off by default in
configuration file.

will also begin offering
ct support to users of BIND
ugh the sale of annual support
tracts.

Paul Vixie, 2005.11.01:

> Had we done a requirements doc ten years ago . . . they might not have noticed that it would intersect their national privacy laws or business requirements, we might still have run into the NSEC3 juggernaut and be just as far off the rails now as we actually are now.

After fi

dollars

(e.g., D

NSF to

Softwa

how su

The In

780000

.04.20,

D 9.3 beta:

hip with DNSSEC
off by default in
n file.

gin offering
o users of BIND
e of annual support

Paul Vixie, 2005.11.01:

> Had we done a requirements doc ten years ago . . . they might not have noticed that it would intersect their national privacy laws or business requirements, we might still have run into the NSEC3 juggernaut and be just as far off the rails now as we actually are now.

After fifteen year
dollars of U.S. go
(e.g., DISA to B
NSF to UCLA; D
Software Corpora
how successful is

The Internet has
78000000 `*.com`

:

SSEC
t in

IND
support

Paul Vixie, 2005.11.01:

> Had we done a requirements doc ten years ago . . . they might not have noticed that it would intersect their national privacy laws or business requirements, we might still have run into the NSEC3 juggernaut and be just as far off the rails now as we actually are now.

After fifteen years and mill
dollars of U.S. government
(e.g., DISA to BIND comp
NSF to UCLA; DHS to Sec
Software Corporation),
how successful is DNSSEC

The Internet has about
78000000 `*.com` names.

Paul Vixie, 2005.11.01:

> Had we done a requirements doc ten years ago . . . they might not have noticed that it would intersect their national privacy laws or business requirements, we might still have run into the NSEC3 juggernaut and be just as far off the rails now as we actually are now.

After fifteen years and millions of dollars of U.S. government grants (e.g., DISA to BIND company; NSF to UCLA; DHS to Secure64 Software Corporation), how successful is DNSSEC?

The Internet has about 78000000 `*.com` names.

Paul Vixie, 2005.11.01:

> Had we done a requirements doc ten years ago . . . they might not have noticed that it would intersect their national privacy laws or business requirements, we might still have run into the NSEC3 juggernaut and be just as far off the rails now as we actually are now.

After fifteen years and millions of dollars of U.S. government grants (e.g., DISA to BIND company; NSF to UCLA; DHS to Secure64 Software Corporation), how successful is DNSSEC?

The Internet has about 78000000 `*.com` names.

Surveys by DNSSEC developers, last updated 2009.02.28, have found 251 `*.com` names with DNSSEC signatures. 116 on 2008.08.20; $251 > 116$.

ixie, 2005.11.01:

we done a requirements doc
years ago . . . they might
have noticed that it would
rsect their national privacy
s or business requirements,
might still have run into the
EC3 juggernaut and be just
ar off the rails now as we
ually are now.

After fifteen years and millions of dollars of U.S. government grants (e.g., DISA to BIND company; NSF to UCLA; DHS to Secure64 Software Corporation), how successful is DNSSEC?

The Internet has about 78000000 `*.com` names.

Surveys by DNSSEC developers, last updated 2009.02.28, have found 251 `*.com` names with DNSSEC signatures. 116 on 2008.08.20; $251 > 116$.

Why is

Some o
servers
the roo
the goo

DNSSE
server-s
signatu

Signatu
saved;
Hopefu
to sign

.11.01:

requirements doc
. . they might
d that it would
ational privacy
requirements,
ave run into the
ut and be just
ils now as we
.

After fifteen years and millions of dollars of U.S. government grants (e.g., DISA to BIND company; NSF to UCLA; DHS to Secure64 Software Corporation), how successful is DNSSEC?

The Internet has about 78000000 `*.com` names.

Surveys by DNSSEC developers, last updated 2009.02.28, have found 251 `*.com` names with DNSSEC signatures. 116 on 2008.08.20; $251 > 116$.

Why is nobody u

Some of the Inte
servers are extren
the root servers,
the `google.com`

DNSSEC tries to
server-side costs
signatures of DN

Signature is com
saved; sent to ma
Hopefully the ser
to sign each DNS

s doc
ht
uld
cy
ts,
the
ust
ve

After fifteen years and millions of dollars of U.S. government grants (e.g., DISA to BIND company; NSF to UCLA; DHS to Secure64 Software Corporation), how successful is DNSSEC?

The Internet has about 78000000 `*.com` names.

Surveys by DNSSEC developers, last updated 2009.02.28, have found 251 `*.com` names with DNSSEC signatures. 116 on 2008.08.20; $251 > 116$.

Why is nobody using DNS

Some of the Internet's DN servers are extremely busy: the root servers, the `.com` the `google.com` servers.

DNSSEC tries to minimize server-side costs by *precom* signatures of DNS records.

Signature is computed onc saved; sent to many clients Hopefully the server can af to sign each DNS record o

After fifteen years and millions of dollars of U.S. government grants (e.g., DISA to BIND company; NSF to UCLA; DHS to Secure64 Software Corporation), how successful is DNSSEC?

The Internet has about 78000000 `*.com` names.

Surveys by DNSSEC developers, last updated 2009.02.28, have found 251 `*.com` names with DNSSEC signatures. 116 on 2008.08.20; $251 > 116$.

Why is nobody using DNSSEC?

Some of the Internet's DNS servers are extremely busy: e.g., the root servers, the `.com` servers, the `google.com` servers.

DNSSEC tries to minimize server-side costs by *precomputing* signatures of DNS records.

Signature is computed once; saved; sent to many clients. Hopefully the server can afford to sign each DNS record once.

ifteen years and millions of
of U.S. government grants
DISA to BIND company;
o UCLA; DHS to Secure64
re Corporation),
ccessful is DNSSEC?

ternet has about
000 `*.com` names.

s by DNSSEC developers,
dated 2009.02.28,
ound 251 `*.com`
with DNSSEC signatures.
2008.08.20; $251 > 116$.

Why is nobody using DNSSEC?

Some of the Internet's DNS servers are extremely busy: e.g., the root servers, the `.com` servers, the `google.com` servers.

DNSSEC tries to minimize server-side costs by *precomputing* signatures of DNS records.

Signature is computed once; saved; sent to many clients. Hopefully the server can afford to sign each DNS record once.

Clients
of *verif*

DNSSE
client-s
choice

DNSSE
say DS
slow fo
recomm
preferr
suggest
of only
for "lea

rs and millions of
overnment grants
IND company;
DHS to Secure64
ation),
DNSSEC?

about
names.

SEC developers,
9.02.28,
*.com
SEC signatures.
20; 251 $>$ 116.

Why is nobody using DNSSEC?

Some of the Internet's DNS servers are extremely busy: e.g., the root servers, the `.com` servers, the `google.com` servers.

DNSSEC tries to minimize server-side costs by *precomputing* signatures of DNS records.

Signature is computed once; saved; sent to many clients. Hopefully the server can afford to sign each DNS record once.

Clients don't sha
of *verifying* a sig

DNSSEC tries to
client-side costs
choice of crypto

DNSSEC RFCs
say DSA is "10 t
slow for verificati
recommend RSA
preferred algorith
suggest RSA key
of only 1024 bits
for "leaf nodes in

ions of
grants
any;
cure64

?

opers,

atures.
116.

Why is nobody using DNSSEC?

Some of the Internet's DNS servers are extremely busy: e.g., the root servers, the `.com` servers, the `google.com` servers.

DNSSEC tries to minimize server-side costs by *precomputing* signatures of DNS records.

Signature is computed once; saved; sent to many clients. Hopefully the server can afford to sign each DNS record once.

Clients don't share the wor
of *verifying* a signature.

DNSSEC tries to reduce client-side costs through choice of crypto primitive.

DNSSEC RFCs
say DSA is "10 to 40 times
slow for verification" as RS
recommend RSA "as the
preferred algorithm" for DN
suggest RSA key size
of only 1024 bits
for "leaf nodes in the DNS

Why is nobody using DNSSEC?

Some of the Internet's DNS servers are extremely busy: e.g., the root servers, the `.com` servers, the `google.com` servers.

DNSSEC tries to minimize server-side costs by *precomputing* signatures of DNS records.

Signature is computed once; saved; sent to many clients. Hopefully the server can afford to sign each DNS record once.

Clients don't share the work of *verifying* a signature.

DNSSEC tries to reduce client-side costs through choice of crypto primitive.

DNSSEC RFCs
say DSA is "10 to 40 times as slow for verification" as RSA;
recommend RSA "as the preferred algorithm" for DNSSEC;
suggest RSA key size
of only 1024 bits
for "leaf nodes in the DNS."

nobody using DNSSEC?

of the Internet's DNS
are extremely busy: e.g.,
ot servers, the `.com` servers,
`ogle.com` servers.

EC tries to minimize
side costs by *precomputing*
ures of DNS records.

ure is computed once;
sent to many clients.
ully the server can afford
each DNS record once.

Clients don't share the work
of *verifying* a signature.

DNSSEC tries to reduce
client-side costs through
choice of crypto primitive.

DNSSEC RFCs
say DSA is "10 to 40 times as
slow for verification" as RSA;
recommend RSA "as the
preferred algorithm" for DNSSEC;
suggest RSA key size
of only 1024 bits
for "leaf nodes in the DNS."

I say:
1024-b

2003:
conclu
was al
large c

2003:
recomm
2048-b
of this
made t

sing DNSSEC?

rnet's DNS
mely busy: e.g.,
the `.com` servers,
servers.

minimize
by *precomputing*
S records.

puted once;
any clients.
rver can afford
S record once.

Clients don't share the work
of *verifying* a signature.

DNSSEC tries to reduce
client-side costs through
choice of crypto primitive.

DNSSEC RFCs
say DSA is "10 to 40 times as
slow for verification" as RSA;
recommend RSA "as the
preferred algorithm" for DNSSEC;
suggest RSA key size
of only 1024 bits
for "leaf nodes in the DNS."

I say:
1024-bit RSA is

2003: Shamir–Tr
concluded that 1
was already breal
large companies

2003: RSA Labo
recommended a
2048-bit keys "ov
of this decade."
made the same r

SEC?

S

e.g.,
servers,

nputing

e;
s.
fford
nce.

Clients don't share the work
of *verifying* a signature.

DNSSEC tries to reduce
client-side costs through
choice of crypto primitive.

DNSSEC RFCs
say DSA is "10 to 40 times as
slow for verification" as RSA;
recommend RSA "as the
preferred algorithm" for DNSSEC;
suggest RSA key size
of only 1024 bits
for "leaf nodes in the DNS."

I say:
1024-bit RSA is irresponsib

2003: Shamir–Tromer et a
concluded that 1024-bit RS
was already breakable by
large companies and botne

2003: RSA Laboratories
recommended a transition
2048-bit keys "over the ne
of this decade." 2007: NIS
made the same recommend

Clients don't share the work
of *verifying* a signature.

DNSSEC tries to reduce
client-side costs through
choice of crypto primitive.

DNSSEC RFCs
say DSA is "10 to 40 times as
slow for verification" as RSA;
recommend RSA "as the
preferred algorithm" for DNSSEC;
suggest RSA key size
of only 1024 bits
for "leaf nodes in the DNS."

I say:
1024-bit RSA is irresponsible.

2003: Shamir–Tromer et al.
concluded that 1024-bit RSA
was already breakable by
large companies and botnets.

2003: RSA Laboratories
recommended a transition to
2048-bit keys "over the remainder
of this decade." 2007: NIST
made the same recommendation.

Clients don't share the work
of *verifying* a signature.

DNSSEC tries to reduce
client-side costs through
choice of crypto primitive.

DNSSEC RFCs
say DSA is "10 to 40 times as
slow for verification" as RSA;
recommend RSA "as the
preferred algorithm" for DNSSEC;
suggest RSA key size
of only 1024 bits
for "leaf nodes in the DNS."

I say:
1024-bit RSA is irresponsible.

2003: Shamir–Tromer et al.
concluded that 1024-bit RSA
was already breakable by
large companies and botnets.

2003: RSA Laboratories
recommended a transition to
2048-bit keys "over the remainder
of this decade." 2007: NIST
made the same recommendation.

But *most users don't know this*.
Why aren't they using DNSSEC?

don't share the work
fying a signature.

EC tries to reduce
side costs through
of crypto primitive.

EC RFCs
A is "10 to 40 times as
r verification" as RSA;
mend RSA "as the
ed algorithm" for DNSSEC;
t RSA key size
1024 bits
af nodes in the DNS."

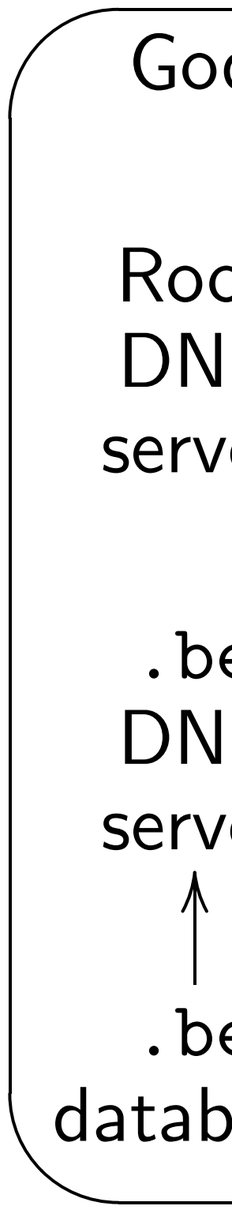
I say:
1024-bit RSA is irresponsible.

2003: Shamir–Tromer et al.
concluded that 1024-bit RSA
was already breakable by
large companies and botnets.

2003: RSA Laboratories
recommended a transition to
2048-bit keys "over the remainder
of this decade." 2007: NIST
made the same recommendation.

But *most users don't know this*.
Why aren't they using DNSSEC?

Recall



at Inte
Centra

re the work
nature.

reduce
through
primitive.

o 40 times as
ion" as RSA;
"as the
m" for DNSSEC;
size

n the DNS."

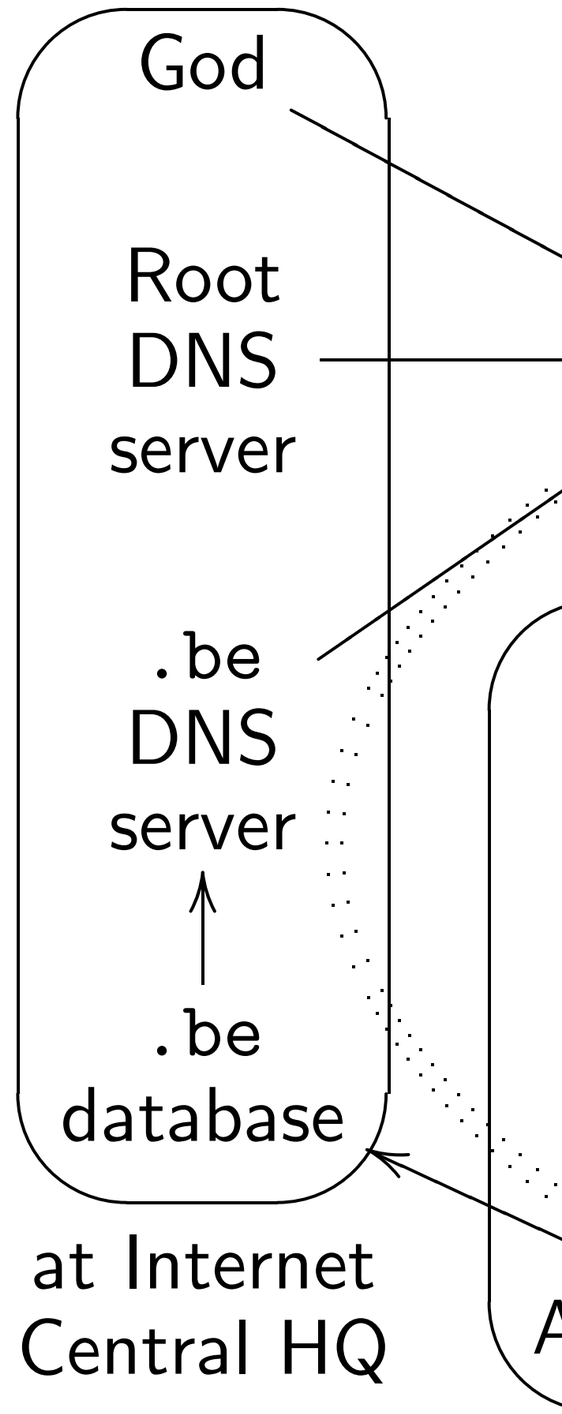
I say:
1024-bit RSA is irresponsible.

2003: Shamir–Tromer et al. concluded that 1024-bit RSA was already breakable by large companies and botnets.

2003: RSA Laboratories recommended a transition to 2048-bit keys "over the remainder of this decade." 2007: NIST made the same recommendation.

But *most users don't know this*.
Why aren't they using DNSSEC?

Recall the DNS a



rk

s as
SA;

NSSEC;

."

I say:

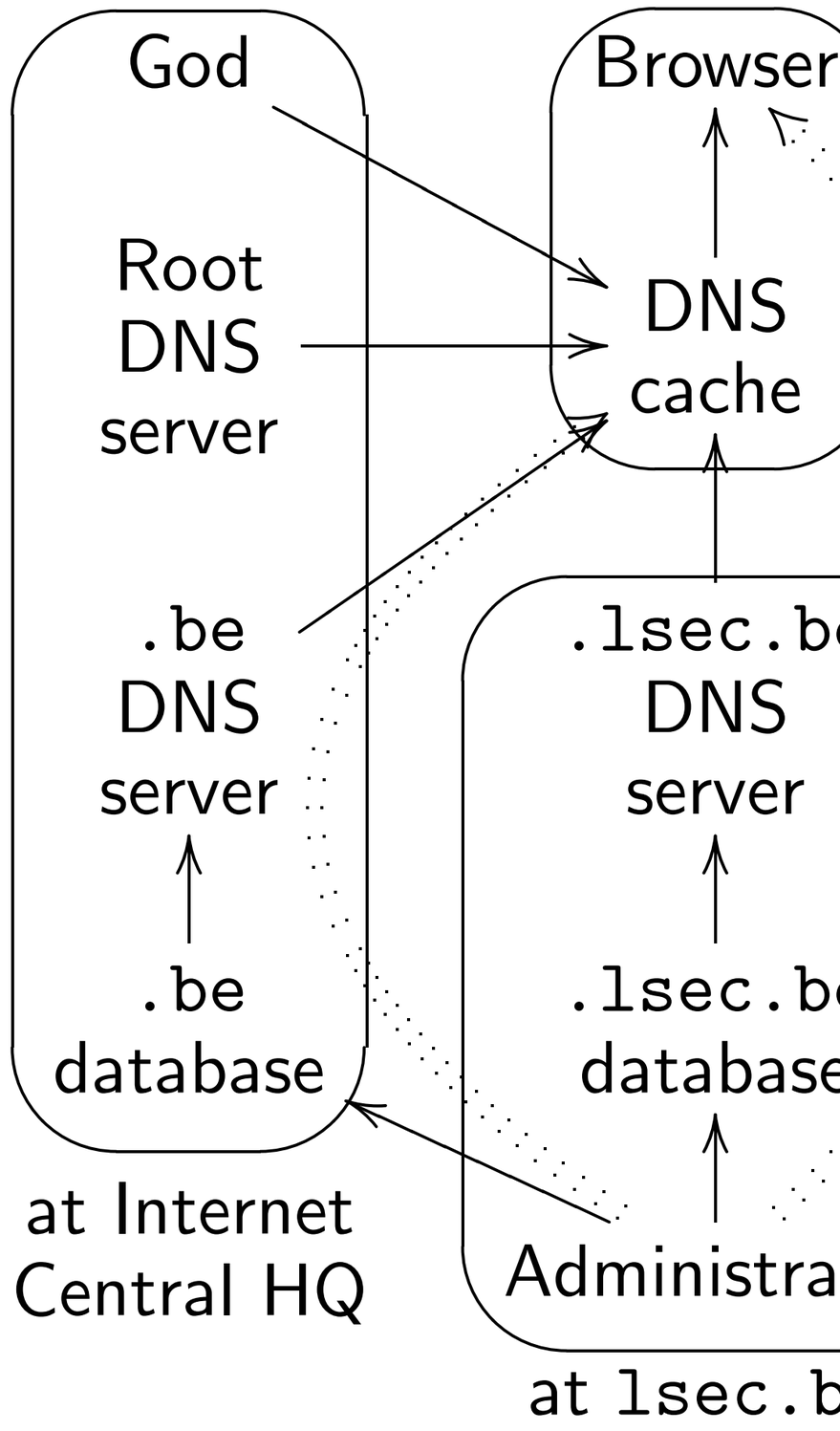
1024-bit RSA is irresponsible.

2003: Shamir–Tromer et al. concluded that 1024-bit RSA was already breakable by large companies and botnets.

2003: RSA Laboratories recommended a transition to 2048-bit keys "over the remainder of this decade." 2007: NIST made the same recommendation.

But *most users don't know this*. Why aren't they using DNSSEC?

Recall the DNS architectur



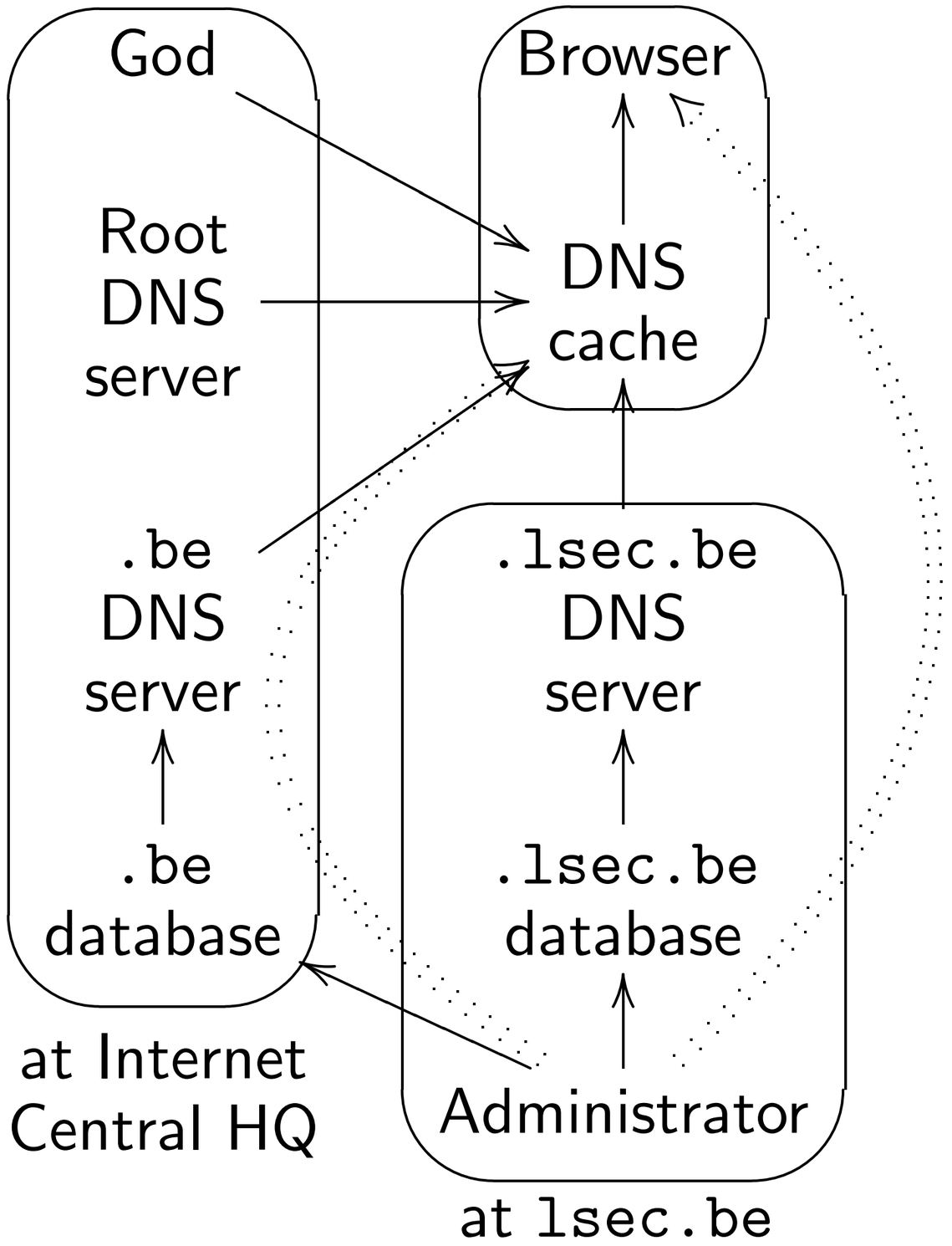
I say:
1024-bit RSA is irresponsible.

2003: Shamir–Tromer et al. concluded that 1024-bit RSA was already breakable by large companies and botnets.

2003: RSA Laboratories recommended a transition to 2048-bit keys "over the remainder of this decade." 2007: NIST made the same recommendation.

But *most users don't know this*.
Why aren't they using DNSSEC?

Recall the DNS architecture:



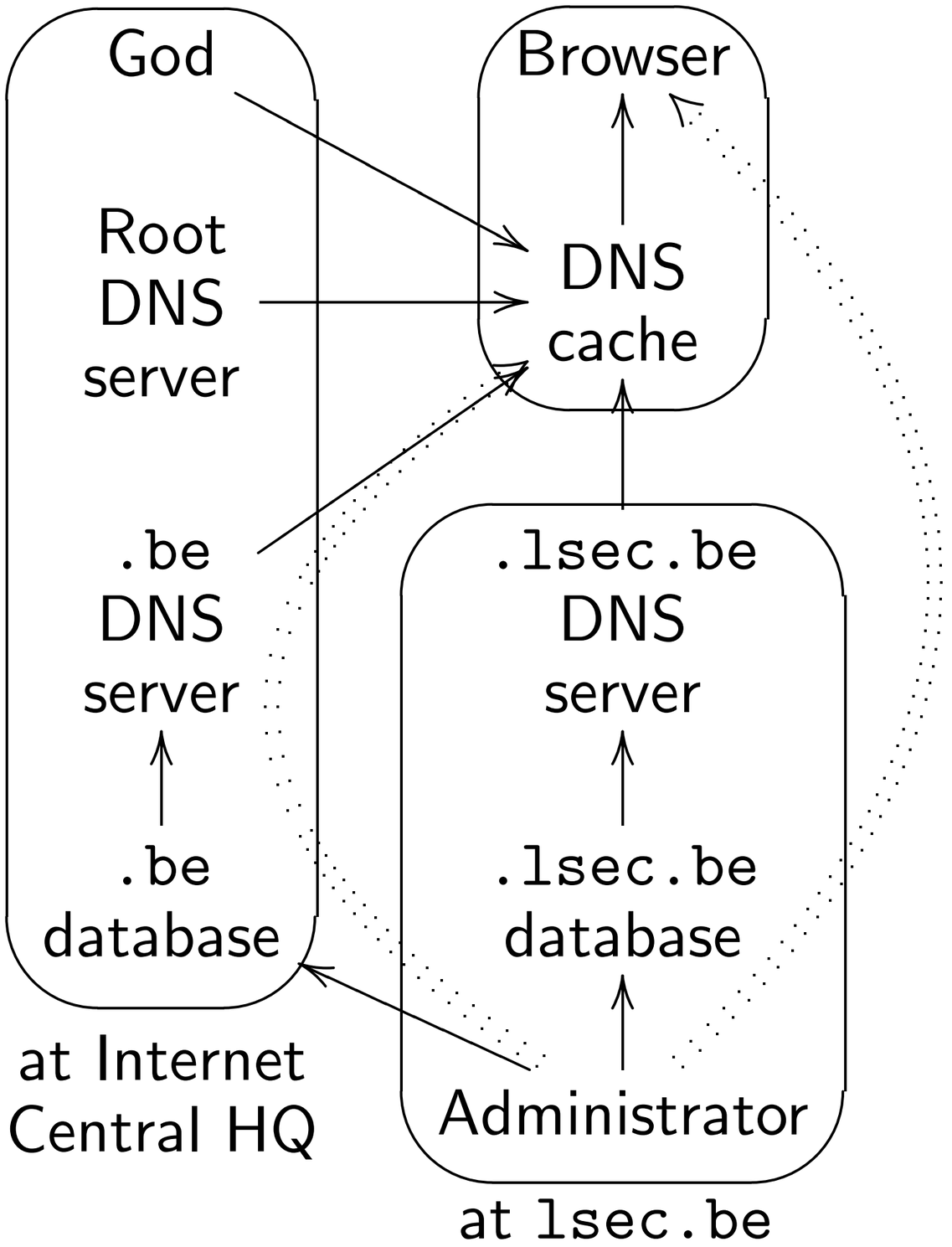
it RSA is irresponsible.

Shamir–Tromer et al.
ded that 1024-bit RSA
eady breakable by
ompanies and botnets.

RSA Laboratories
mended a transition to
it keys "over the remainder
decade." 2007: NIST
he same recommendation.

ost users don't know this.
ren't they using DNSSEC?

Recall the DNS architecture:



DNS se
Wikipe
DNS, c
DNS P
MaraD
Secure

DNS d
tools li
BPP, D
gencic
NSC, n
update
webdns
homegr
DNS re

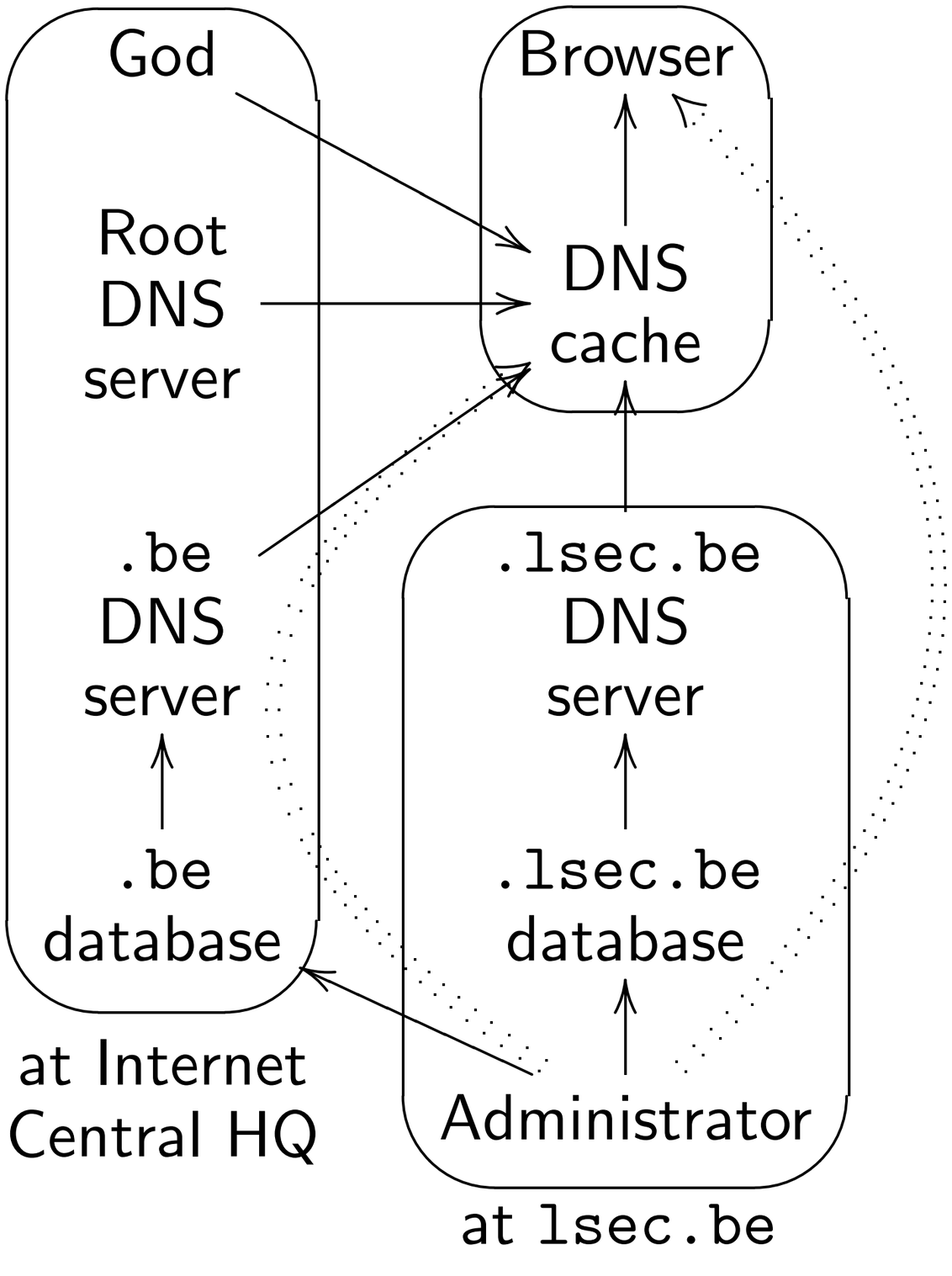
irresponsible.

romer et al.
024-bit RSA
kable by
and botnets.

ratories
transition to
ver the remainder
2007: NIST
ecommendation.

don't know this.
using DNSSEC?

Recall the DNS architecture:



DNS server softw
Wikipedia: BIND
DNS, djbdns, Dn
DNS Plus, NSD,
MaraDNS, ANS,
Secure64 DNS.

DNS database-m
tools listed by 20
BPP, DNS Boss,
`gencidrzone`, `h`
NSC, `nsupdate`,
`updatehosts`, U
`webdns`, `zsu`. Pl
homegrown tools
DNS registrars et

ble.

l.

SA

ts.

to

mainder

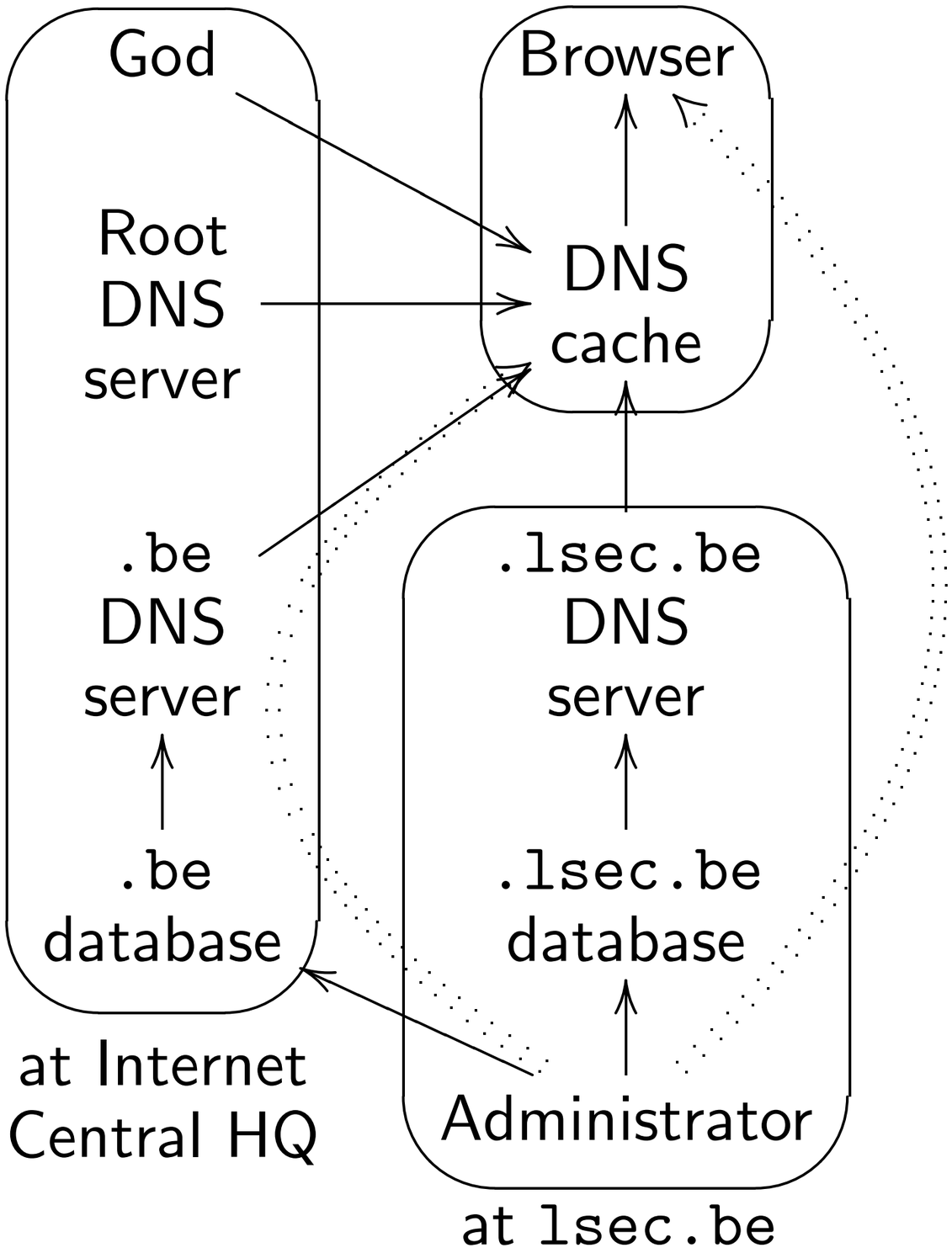
ST

dation.

this.

SSEC?

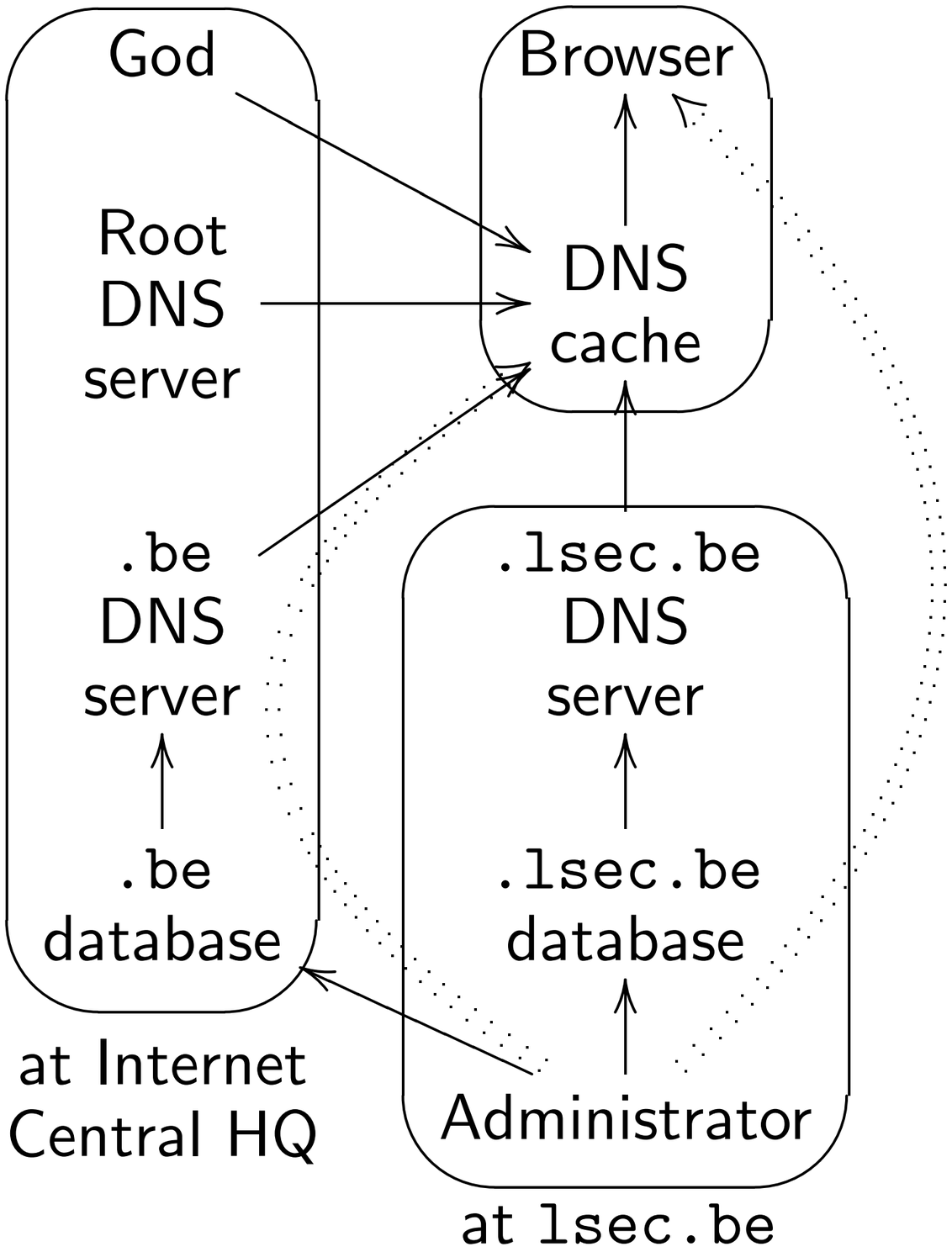
Recall the DNS architecture:



DNS server software listed
Wikipedia: BIND, Microso
DNS, djbdns, Dnsmasq, Si
DNS Plus, NSD, PowerDN
MaraDNS, ANS, Posadis,
Secure64 DNS.

DNS database-managemen
tools listed by 2008 Salome
BPP, DNS Boss, DNStool,
`gencidrzone`, `h2n`, `makez`
NSC, `nsupdate`, SENDS,
`updatehosts`, Utah Tools,
`webdns`, `zsu`. Plus hundre
homegrown tools written b
DNS registrars etc.

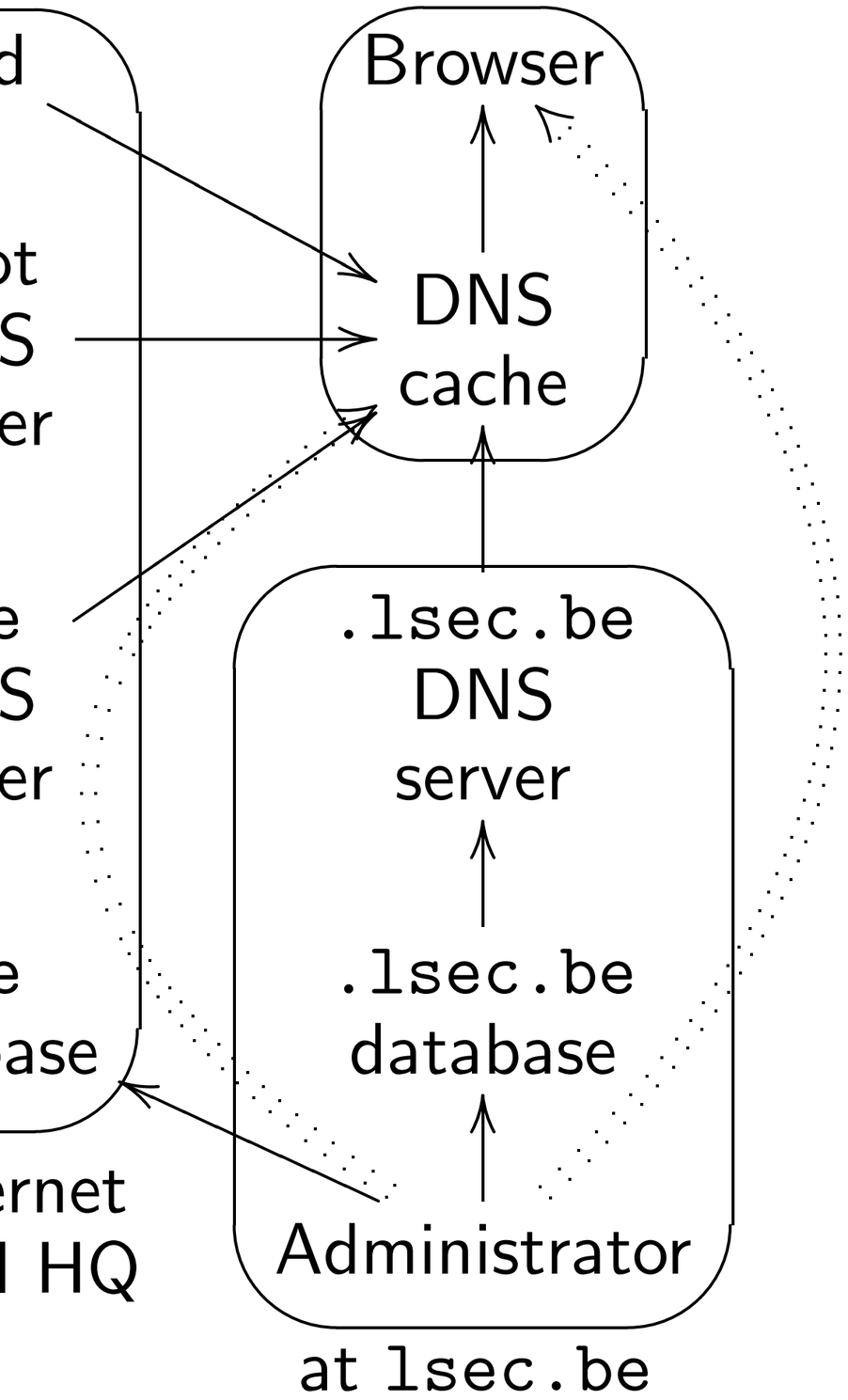
Recall the DNS architecture:



DNS server software listed in Wikipedia: BIND, Microsoft DNS, djbdns, Dnsmasq, Simple DNS Plus, NSD, PowerDNS, MaraDNS, ANS, Posadis, Secure64 DNS.

DNS database-management tools listed by 2008 Salomon: BPP, DNS Boss, DNStool, `gencidrzone`, `h2n`, `makezones`, NSC, `nsupdate`, SENDS, `updatehosts`, Utah Tools, `webdns`, `zsu`. Plus hundreds of homegrown tools written by DNS registrars etc.

the DNS architecture:



DNS server software listed in Wikipedia: BIND, Microsoft DNS, djbdns, Dnsmasq, Simple DNS Plus, NSD, PowerDNS, MaraDNS, ANS, Posadis, Secure64 DNS.

DNS database-management tools listed by 2008 Salomon: BPP, DNS Boss, DNStool, `gencidrzone`, `h2n`, `makezones`, NSC, `nsupdate`, SENDS, `updatehosts`, Utah Tools, `webdns`, `zsu`. Plus hundreds of homegrown tools written by DNS registrars etc.

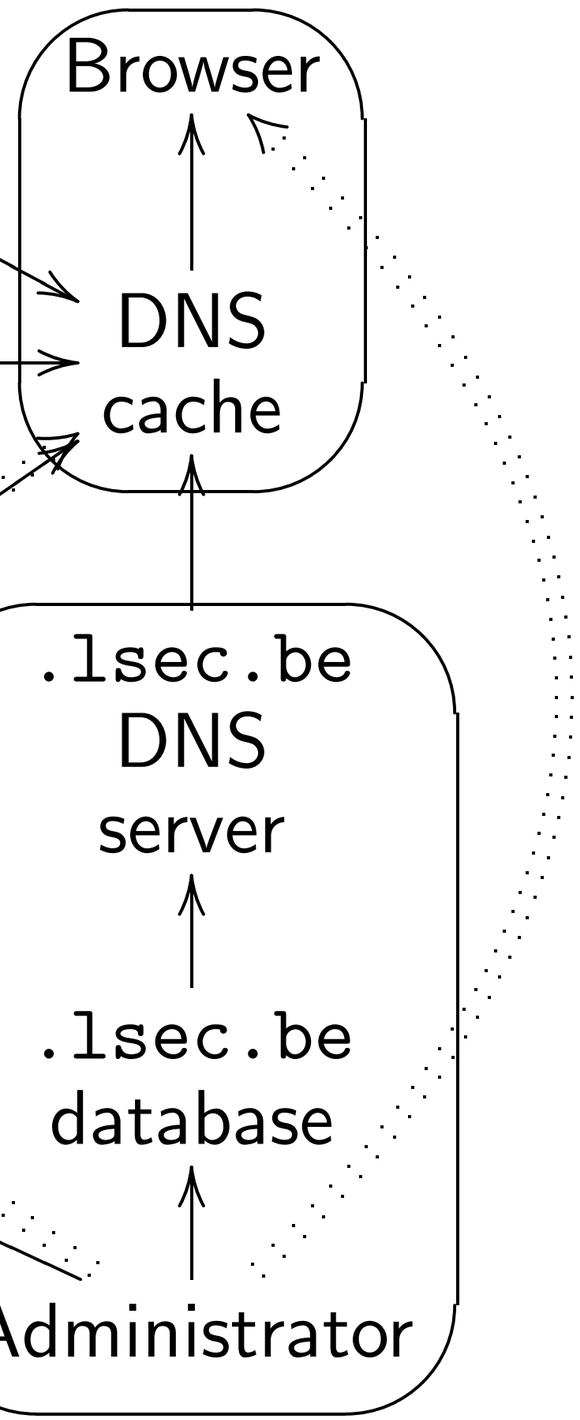
DNSSE
every D

Whene
a DNS
precom
signatu

Often
for the

Examp
can pro
(2005
Tool re
probab

architecture:



at `lsec.be`

DNS server software listed in Wikipedia: BIND, Microsoft DNS, djbdns, Dnsmasq, Simple DNS Plus, NSD, PowerDNS, MaraDNS, ANS, Posadis, Secure64 DNS.

DNS database-management tools listed by 2008 Salomon: BPP, DNS Boss, DNStool, `gencidrzone`, `h2n`, `makezones`, NSC, `nsupdate`, SENDS, `updatehosts`, Utah Tools, `webdns`, `zsu`. Plus hundreds of homegrown tools written by DNS registrars etc.

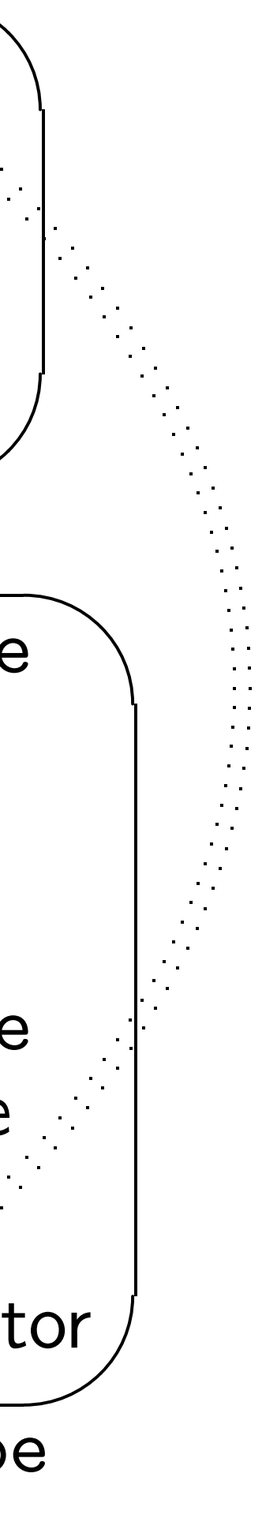
DNSSEC require
every DNS-mana

Whenever a tool
a DNS record, al
precompute and
signature for the

Often considerab
for the tool prog

Example: Signing
can produce 10G
(2005 NIST stud
Tool reading data
probably has to b

re:



DNS server software listed in Wikipedia: BIND, Microsoft DNS, djbdns, Dnsmasq, Simple DNS Plus, NSD, PowerDNS, MaraDNS, ANS, Posadis, Secure64 DNS.

DNS database-management tools listed by 2008 Salomon: BPP, DNS Boss, DNStool, `gencidrzone`, `h2n`, `makezones`, NSC, `nsupdate`, SENDS, `updatehosts`, Utah Tools, `webdns`, `zsu`. Plus hundreds of homegrown tools written by DNS registrars etc.

DNSSEC requires new cod
every DNS-management to

Whenever a tool adds or c
a DNS record, also has to
precompute and store a DN
signature for the new reco

Often considerable effort
for the tool programmers.

Example: Signing 2GB dat
can produce 10GB databas
(2005 NIST study).
Tool reading database into
probably has to be reengin

DNS server software listed in Wikipedia: BIND, Microsoft DNS, djbdns, Dnsmasq, Simple DNS Plus, NSD, PowerDNS, MaraDNS, ANS, Posadis, Secure64 DNS.

DNS database-management tools listed by 2008 Salomon: BPP, DNS Boss, DNStool, `gencidrzone`, `h2n`, `makezones`, NSC, `nsupdate`, SENDS, `updatehosts`, Utah Tools, `webdns`, `zsu`. Plus hundreds of homegrown tools written by DNS registrars etc.

DNSSEC requires new code in every DNS-management tool.

Whenever a tool adds or changes a DNS record, also has to precompute and store a DNSSEC signature for the new record.

Often considerable effort for the tool programmers.

Example: Signing 2GB database can produce 10GB database (2005 NIST study).
Tool reading database into RAM probably has to be reengineered.

erver software listed in
edia: BIND, Microsoft
djbdns, Dnsmasq, Simple
Plus, NSD, PowerDNS,
NS, ANS, Posadis,
64 DNS.

atabase-management
sted by 2008 Salomon:
ONS Boss, DNStool,
`drzone`, `h2n`, `makezones`,
`nsupdate`, SENDS,
`ehosts`, Utah Tools,
`s`, `zsu`. Plus hundreds of
rown tools written by
egistrars etc.

DNSSEC requires new code in
every DNS-management tool.

Whenever a tool adds or changes
a DNS record, also has to
precompute and store a DNSSEC
signature for the new record.

Often considerable effort
for the tool programmers.

Example: Signing 2GB database
can produce 10GB database
(2005 NIST study).
Tool reading database into RAM
probably has to be reengineered.

Becaus
and re
few da
have a

Admini
mix ex
with se
for eve

vare listed in
D, Microsoft
nsmasq, Simple
PowerDNS,
Posadis,

anagement
008 Salomon:
DNStool,
2n, `makezones`,
SENDS,
tah Tools,
us hundreds of
s written by
tc.

DNSSEC requires new code in every DNS-management tool.

Whenever a tool adds or changes a DNS record, also has to precompute and store a DNSSEC signature for the new record.

Often considerable effort for the tool programmers.

Example: Signing 2GB database can produce 10GB database (2005 NIST study).
Tool reading database into RAM probably has to be reengineered.

Because of engin
and redeploymen
few database-ma
have added DNS

Administrator ha
mix existing man
with separate sig
for every change

in
ft
mple
S,

nt
on:

ones,

,
ds of
by

DNSSEC requires new code in every DNS-management tool.

Whenever a tool adds or changes a DNS record, also has to precompute and store a DNSSEC signature for the new record.

Often considerable effort for the tool programmers.

Example: Signing 2GB database can produce 10GB database (2005 NIST study).
Tool reading database into RAM probably has to be reengineered.

Because of engineering cos
and redeployment costs, ve
few database-management
have added DNSSEC supp

Administrator has to manu
mix existing management t
with separate signature ge
for every change to DNS d

DNSSEC requires new code in every DNS-management tool.

Whenever a tool adds or changes a DNS record, also has to precompute and store a DNSSEC signature for the new record.

Often considerable effort for the tool programmers.

Example: Signing 2GB database can produce 10GB database (2005 NIST study).
Tool reading database into RAM probably has to be reengineered.

Because of engineering costs and redeployment costs, very few database-management tools have added DNSSEC support.

Administrator has to manually mix existing management tools with separate signature generation for every change to DNS data.

DNSSEC requires new code in every DNS-management tool.

Whenever a tool adds or changes a DNS record, also has to precompute and store a DNSSEC signature for the new record.

Often considerable effort for the tool programmers.

Example: Signing 2GB database can produce 10GB database (2005 NIST study).
Tool reading database into RAM probably has to be reengineered.

Because of engineering costs and redeployment costs, very few database-management tools have added DNSSEC support.

Administrator has to manually mix existing management tools with separate signature generation for every change to DNS data.

2008 slideshow "DNSSEC in six minutes" (79 pages): "Any time you modify a zone ... you must re-run `dnssec-signzone`."

EC requires new code in
DNS-management tool.

ver a tool adds or changes
record, also has to
npute and store a DNSSEC
ure for the new record.

considerable effort
tool programmers.

le: Signing 2GB database
oduce 10GB database
NIST study).
eading database into RAM
ly has to be reengineered.

Because of engineering costs and redeployment costs, very few database-management tools have added DNSSEC support.

Administrator has to manually mix existing management tools with separate signature generation for every change to DNS data.

2008 slideshow "DNSSEC in six minutes" (79 pages): "Any time you modify a zone . . . you must re-run `dnssec-signzone`."

Admini
public

The `.b`
and da
and we
need to
to acce
and to

Big zon
refuse
Full DN
much t

s new code in
gement tool.

adds or changes
so has to
store a DNSSEC
new record.

le effort
rammers.

g 2GB database
B database
y).
abase into RAM
be reengineered.

Because of engineering costs and redeployment costs, very few database-management tools have added DNSSEC support.

Administrator has to manually mix existing management tools with separate signature generation for every change to DNS data.

2008 slideshow "DNSSEC in six minutes" (79 pages): "Any time you modify a zone ... you must re-run `dnssec-signzone`."

Administrator als
public key to `.be`

The `.be` server
and database sof
and web interfac
need to be updat
to accept these p
and to sign every

Big zones such a
refuse to sign co
Full DNSSEC sig
much too slow a

e in
ool.

hanges

NSSEC
rd.

tabase
se

RAM
eered.

Because of engineering costs and redeployment costs, very few database-management tools have added DNSSEC support.

Administrator has to manually mix existing management tools with separate signature generation for every change to DNS data.

2008 slideshow "DNSSEC in six minutes" (79 pages): "Any time you modify a zone ... you must re-run `dnssec-signzone`."

Administrator also has to s
public key to `.be`.

The `.be` server
and database software
and web interface
need to be updated
to accept these public keys
and to sign everything.

Big zones such as `.com`
refuse to sign complete dat
Full DNSSEC signing woul
much too slow and much t

Because of engineering costs and redeployment costs, very few database-management tools have added DNSSEC support.

Administrator has to manually mix existing management tools with separate signature generation for every change to DNS data.

2008 slideshow "DNSSEC in six minutes" (79 pages): "Any time you modify a zone ... you must re-run `dnssec-signzone`."

Administrator also has to send public key to `.be`.

The `.be` server
and database software
and web interface
need to be updated
to accept these public keys
and to sign everything.

Big zones such as `.com`
refuse to sign complete database.
Full DNSSEC signing would be
much too slow and much too big.

se of engineering costs
deployment costs, very
tabase-management tools
dded DNSSEC support.

istrator has to manually
isting management tools
eparate signature generation
ry change to DNS data.

lideshow “DNSSEC in six
s” (79 pages): “Any time
odify a zone . . . you must
`dnssec-signzone`.”

Administrator also has to send
public key to `.be`.

The `.be` server
and database software
and web interface
need to be updated
to accept these public keys
and to sign everything.

Big zones such as `.com`
refuse to sign complete database.
Full DNSSEC signing would be
much too slow and much too big.

DNS ca
to fetch
and ver

Often r
than or
Higher
More fr

Also, m
attacke
Official
RFC 40
provide
against

eering costs
t costs, very
nagement tools
SEC support.

s to manually
agement tools
nature generation
to DNS data.

DNSSEC in six
ges): “Any time
ne . . . you must
`ignzone`.”

Administrator also has to send
public key to `.be`.

The `.be` server
and database software
and web interface
need to be updated
to accept these public keys
and to sign everything.

Big zones such as `.com`
refuse to sign complete database.
Full DNSSEC signing would be
much too slow and much too big.

DNS cache needs
to fetch keys, fet
and verify signatu

Often many more
than original DN
Higher latency fo
More frequent fa

Also, much easie
attacker to deny
Official DNSSEC
RFC 4033: “DNS
provides no prote
against denial of

sts
ery
tools
ort.

ually
tools
neration
ata.

in six
y time
must
”

Administrator also has to send
public key to `.be`.

The `.be` server
and database software
and web interface
need to be updated
to accept these public keys
and to sign everything.

Big zones such as `.com`
refuse to sign complete database.
Full DNSSEC signing would be
much too slow and much too big.

DNS cache needs new soft
to fetch keys, fetch signatu
and verify signatures.

Often many more packets
than original DNS.
Higher latency for user.
More frequent failures.

Also, much easier for
attacker to deny service.
Official DNSSEC response,
RFC 4033: “DNSSEC
provides no protection
against denial of service at

Administrator also has to send
public key to `.be`.

The `.be` server
and database software
and web interface
need to be updated
to accept these public keys
and to sign everything.

Big zones such as `.com`
refuse to sign complete database.
Full DNSSEC signing would be
much too slow and much too big.

DNS cache needs new software
to fetch keys, fetch signatures,
and verify signatures.

Often many more packets
than original DNS.
Higher latency for user.
More frequent failures.

Also, much easier for
attacker to deny service.
Official DNSSEC response,
RFC 4033: “DNSSEC
provides no protection
against denial of service attacks.”

istrator also has to send
key to .be.

be server
tabase software
eb interface
o be updated
ept these public keys
sign everything.

nes such as .com
to sign complete database.
NSSEC signing would be
too slow and much too big.

DNS cache needs new software
to fetch keys, fetch signatures,
and verify signatures.

Often many more packets
than original DNS.
Higher latency for user.
More frequent failures.

Also, much easier for
attacker to deny service.
Official DNSSEC response,
RFC 4033: “DNSSEC
provides no protection
against denial of service attacks.”

Replay

Attack
signatu

lsec.b
acquire
change

so has to send
e.

ftware
e
ted
ublic keys
ything.

s .com
mplete database.
gning would be
nd much too big.

DNS cache needs new software
to fetch keys, fetch signatures,
and verify signatures.

Often many more packets
than original DNS.
Higher latency for user.
More frequent failures.

Also, much easier for
attacker to deny service.
Official DNSSEC response,
RFC 4033: “DNSSEC
provides no protection
against denial of service attacks.”

Replay attack on

Attacker inspects
signatures from I

`1sec.be` changes
acquires new IP
changes DNS rec

send

tabase.
d be
too big.

DNS cache needs new software to fetch keys, fetch signatures, and verify signatures.

Often many more packets than original DNS.
Higher latency for user.
More frequent failures.

Also, much easier for attacker to deny service.
Official DNSSEC response, RFC 4033: "DNSSEC provides no protection against denial of service attacks."

Replay attack on DNSSEC

Attacker inspects DNSSEC signatures from `lsec.be`.

`lsec.be` changes location, acquires new IP addresses, changes DNS records.

DNS cache needs new software
to fetch keys, fetch signatures,
and verify signatures.

Often many more packets
than original DNS.
Higher latency for user.
More frequent failures.

Also, much easier for
attacker to deny service.
Official DNSSEC response,
RFC 4033: "DNSSEC
provides no protection
against denial of service attacks."

Replay attack on DNSSEC:

Attacker inspects DNSSEC
signatures from `lsec.be`.

`lsec.be` changes location,
acquires new IP addresses,
changes DNS records.

DNS cache needs new software
to fetch keys, fetch signatures,
and verify signatures.

Often many more packets
than original DNS.
Higher latency for user.
More frequent failures.

Also, much easier for
attacker to deny service.
Official DNSSEC response,
RFC 4033: "DNSSEC
provides no protection
against denial of service attacks."

Replay attack on DNSSEC:

Attacker inspects DNSSEC
signatures from `lsec.be`.

`lsec.be` changes location,
acquires new IP addresses,
changes DNS records.

Attacker buys the old addresses,
forges DNS responses
with the *old* DNS records
and the *old* signatures.
Successfully steals mail!

ache needs new software
h keys, fetch signatures,
rify signatures.

many more packets
riginal DNS.
latency for user.
requent failures.

nuch easier for
er to deny service.
l DNSSEC response,
033: “DNSSEC
es no protection
t denial of service attacks.”

Replay attack on DNSSEC:

Attacker inspects DNSSEC signatures from `lsec.be`.

`lsec.be` changes location, acquires new IP addresses, changes DNS records.

Attacker buys the old addresses, forges DNS responses with the *old* DNS records and the *old* signatures.
Successfully steals mail!

DNSSE
Signatu
normal

Not ve
replay
for up

s new software
tch signatures,
ures.

e packets
S.
or user.
ilures.

r for
service.
response,
SSEC
ection
service attacks."

Replay attack on DNSSEC:

Attacker inspects DNSSEC signatures from `lsec.be`.

`lsec.be` changes location, acquires new IP addresses, changes DNS records.

Attacker buys the old addresses, forges DNS responses with the *old* DNS records and the *old* signatures.
Successfully steals mail!

DNSSEC has a p
Signature has an
normally signing

Not very good se
replay attack con
for up to 30 days

ware
ures,

tacks."

Replay attack on DNSSEC:

Attacker inspects DNSSEC signatures from `lsec.be`.

`lsec.be` changes location, acquires new IP addresses, changes DNS records.

Attacker buys the old addresses, forges DNS responses with the *old* DNS records and the *old* signatures. Successfully steals mail!

DNSSEC has a partial defe
Signature has an expiration
normally signing date + 30

Not very good security:
replay attack continues to
for up to 30 days!

Replay attack on DNSSEC:

Attacker inspects DNSSEC signatures from `lsec.be`.

`lsec.be` changes location, acquires new IP addresses, changes DNS records.

Attacker buys the old addresses, forges DNS responses with the *old* DNS records and the *old* signatures. Successfully steals mail!

DNSSEC has a partial defense. Signature has an expiration date, normally signing date + 30 days.

Not very good security: replay attack continues to work for up to 30 days!

Replay attack on DNSSEC:

Attacker inspects DNSSEC signatures from `lsec.be`.

`lsec.be` changes location, acquires new IP addresses, changes DNS records.

Attacker buys the old addresses, forges DNS responses with the *old* DNS records and the *old* signatures. Successfully steals mail!

DNSSEC has a partial defense. Signature has an expiration date, normally signing date + 30 days.

Not very good security: replay attack continues to work for up to 30 days!

Also a major administrative hassle: administrator must generate new signatures before old signatures expire.

If administrator forgets, domain is destroyed. "DNSSEC suicide."

attack on DNSSEC:

er inspects DNSSEC
ures from `lsec.be`.

be changes location,
es new IP addresses,
s DNS records.

er buys the old addresses,
DNS responses
he *old* DNS records
e *old* signatures.
sfully steals mail!

DNSSEC has a partial defense.
Signature has an expiration date,
normally signing date + 30 days.

Not very good security:
replay attack continues to work
for up to 30 days!

Also a major administrative
hassle: administrator must
generate new signatures
before old signatures expire.

If administrator forgets,
domain is destroyed.
"DNSSEC suicide."

Imagin
that w

DNSSEC:

s DNSSEC
lsec.be.

s location,
addresses,
cords.

e old addresses,
onses
S records
atures.
ls mail!

DNSSEC has a partial defense.
Signature has an expiration date,
normally signing date + 30 days.

Not very good security:
replay attack continues to work
for up to 30 days!

Also a major administrative
hassle: administrator must
generate new signatures
before old signatures expire.

If administrator forgets,
domain is destroyed.
"DNSSEC suicide."

Imagine an "HTT
that works like D

:

esses,

DNSSEC has a partial defense.
Signature has an expiration date,
normally signing date + 30 days.

Not very good security:
replay attack continues to work
for up to 30 days!

Also a major administrative
hassle: administrator must
generate new signatures
before old signatures expire.

If administrator forgets,
domain is destroyed.
"DNSSEC suicide."

Imagine an "HTTPSEC"
that works like DNSSEC.

DNSSEC has a partial defense.
Signature has an expiration date,
normally signing date + 30 days.

Not very good security:
replay attack continues to work
for up to 30 days!

Also a major administrative
hassle: administrator must
generate new signatures
before old signatures expire.

If administrator forgets,
domain is destroyed.
"DNSSEC suicide."

Imagine an "HTTPSEC"
that works like DNSSEC.

DNSSEC has a partial defense.
Signature has an expiration date,
normally signing date + 30 days.

Not very good security:
replay attack continues to work
for up to 30 days!

Also a major administrative
hassle: administrator must
generate new signatures
before old signatures expire.

If administrator forgets,
domain is destroyed.
"DNSSEC suicide."

Imagine an "HTTPSEC"
that works like DNSSEC.

Installing HTTPSEC software
and setting up a public key
is just the beginning.

After every change to web pages,
have to run `httpsec-signpages`
to precompute new signatures.

DNSSEC has a partial defense.
Signature has an expiration date,
normally signing date + 30 days.

Not very good security:
replay attack continues to work
for up to 30 days!

Also a major administrative
hassle: administrator must
generate new signatures
before old signatures expire.

If administrator forgets,
domain is destroyed.
"DNSSEC suicide."

Imagine an "HTTPSEC"
that works like DNSSEC.

Installing HTTPSEC software
and setting up a public key
is just the beginning.

After every change to web pages,
have to run `httpsec-signpages`
to precompute new signatures.

Replay attacks work for 30 days.

Have to run `httpsec-signpages`
before 30-day expiration or your
web pages are destroyed.

EC has a partial defense.
ure has an expiration date,
ly signing date + 30 days.

ry good security:
attack continues to work
to 30 days!

major administrative
administrator must
te new signatures
old signatures expire.

inistrator forgets,
n is destroyed.
SEC suicide."

Imagine an "HTTPSEC"
that works like DNSSEC.

Installing HTTPSEC software
and setting up a public key
is just the beginning.

After every change to web pages,
have to run `httpsec-signpages`
to precompute new signatures.

Replay attacks work for 30 days.

Have to run `httpsec-signpages`
before 30-day expiration or your
web pages are destroyed.

But wa

NXDO
Attack
from g
`citro`
doesn't

partial defense.
expiration date,
date + 30 days.

ecurity:
ntinues to work
s!

ministrative
rator must
natures
ures expire.

forgets,
yed.
e."

Imagine an "HTTPSEC"
that works like DNSSEC.

Installing HTTPSEC software
and setting up a public key
is just the beginning.

After every change to web pages,
have to run `httpsec-signpages`
to precompute new signatures.

Replay attacks work for 30 days.

Have to run `httpsec-signpages`
before 30-day expiration or your
web pages are destroyed.

But wait, there's

NXDOMAIN atta
Attacker forges D
from `google.co`
`citronella.goo`
doesn't exist.

ense.
n date,
0 days.

work

e

e.

Imagine an "HTTPSEC"
that works like DNSSEC.

Installing HTTPSEC software
and setting up a public key
is just the beginning.

After every change to web pages,
have to run `httpsec-signpages`
to precompute new signatures.

Replay attacks work for 30 days.

Have to run `httpsec-signpages`
before 30-day expiration or your
web pages are destroyed.

But wait, there's more!

NXDOMAIN attack on DN
Attacker forges DNS respo
from `google.com` saying t
`citronella.google.com`
doesn't exist.

Imagine an "HTTPSEC"
that works like DNSSEC.

Installing HTTPSEC software
and setting up a public key
is just the beginning.

After every change to web pages,
have to run `httpsec-signpages`
to precompute new signatures.

Replay attacks work for 30 days.

Have to run `httpsec-signpages`
before 30-day expiration or your
web pages are destroyed.

But wait, there's more!

NXDOMAIN attack on DNSSEC:
Attacker forges DNS response
from `google.com` saying that
`citronella.google.com`
doesn't exist.

Imagine an "HTTPSEC"
that works like DNSSEC.

Installing HTTPSEC software
and setting up a public key
is just the beginning.

After every change to web pages,
have to run `httpsec-signpages`
to precompute new signatures.

Replay attacks work for 30 days.

Have to run `httpsec-signpages`
before 30-day expiration or your
web pages are destroyed.

But wait, there's more!

NXDOMAIN attack on DNSSEC:
Attacker forges DNS response
from `google.com` saying that
`citronella.google.com`
doesn't exist.

Cache can't accept this
without a signature:
otherwise attacker can
knock names off the Internet.

Imagine an "HTTPSEC"
that works like DNSSEC.

Installing HTTPSEC software
and setting up a public key
is just the beginning.

After every change to web pages,
have to run `httpsec-signpages`
to precompute new signatures.

Replay attacks work for 30 days.

Have to run `httpsec-signpages`
before 30-day expiration or your
web pages are destroyed.

But wait, there's more!

NXDOMAIN attack on DNSSEC:
Attacker forges DNS response
from `google.com` saying that
`citronella.google.com`
doesn't exist.

Cache can't accept this
without a signature:
otherwise attacker can
knock names off the Internet.

When is the signature
precomputed? Does Google
precompute signatures for
all possible names? Too many!

e an "HTTPSEC"
orks like DNSSEC.

ng HTTPSEC software
tting up a public key
the beginning.

very change to web pages,
o run `httpsec-signpages`
ompute new signatures.

attacks work for 30 days.

o run `httpsec-signpages`
30-day expiration or your
ges are destroyed.

But wait, there's more!

NXDOMAIN attack on DNSSEC:
Attacker forges DNS response
from `google.com` saying that
`citronella.google.com`
doesn't exist.

Cache can't accept this
without a signature:
otherwise attacker can
knock names off the Internet.

When is the signature
precomputed? Does Google
precompute signatures for
all possible names? Too many!

DNSSE
multi-N
"there
`chrome`
`code.g`

DNSSE
data in
betwee
Implem
but the

TPSEC"
DNSSEC.

SEC software
public key
ning.

ge to web pages,
`osec-signpages`
ew signatures.

ork for 30 days.

`psec-signpages`
piration or your
estroyed.

But wait, there's more!

NXDOMAIN attack on DNSSEC:
Attacker forges DNS response
from `google.com` saying that
`citronella.google.com`
doesn't exist.

Cache can't accept this
without a signature:
otherwise attacker can
knock names off the Internet.

When is the signature
precomputed? Does Google
precompute signatures for
all possible names? Too many!

DNSSEC solution
multi-NXDOMAI
"there are no na
`chrome.google`
`code.google.co`

DNSSEC server i
data in response
between `chrome`
Implementing "b
but theoretically

are
y

pages,
npages
ures.

days.

npages
your

But wait, there's more!

NXDOMAIN attack on DNSSEC: Attacker forges DNS response from `google.com` saying that `citronella.google.com` doesn't exist.

Cache can't accept this without a signature: otherwise attacker can knock names off the Internet.

When is the signature precomputed? Does Google precompute signatures for *all* possible names? Too many!

DNSSEC solution: Sign multi-NXDOMAIN such as "there are no names betwe `chrome.google.com` and `code.google.com`."

DNSSEC server issues this data in response to any na between `chrome` and `code` Implementing "between" is but theoretically possible.

But wait, there's more!

NXDOMAIN attack on DNSSEC: Attacker forges DNS response from `google.com` saying that `citronella.google.com` doesn't exist.

Cache can't accept this without a signature: otherwise attacker can knock names off the Internet.

When is the signature precomputed? Does Google precompute signatures for *all* possible names? Too many!

DNSSEC solution: Sign multi-NXDOMAIN such as "there are no names between `chrome.google.com` and `code.google.com`."

DNSSEC server issues this signed data in response to any name between `chrome` and `code`. Implementing "between" is tricky but theoretically possible.

But wait, there's more!

NXDOMAIN attack on DNSSEC: Attacker forges DNS response from `google.com` saying that `citronella.google.com` doesn't exist.

Cache can't accept this without a signature: otherwise attacker can knock names off the Internet.

When is the signature precomputed? Does Google precompute signatures for *all* possible names? Too many!

DNSSEC solution: Sign multi-NXDOMAIN such as "there are no names between `chrome.google.com` and `code.google.com`."

DNSSEC server issues this signed data in response to any name between `chrome` and `code`. Implementing "between" is tricky but theoretically possible.

Consequence: If you deploy DNSSEC then you are exposing *all* of your DNS names!

it, there's more!

MAIN attack on DNSSEC:
er forges DNS response
oogle.com saying that
nella.google.com
t exist.

can't accept this
t a signature:
ise attacker can
names off the Internet.

is the signature
nputed? Does Google
npute signatures for
sible names? Too many!

DNSSEC solution: Sign multi-NXDOMAIN such as "there are no names between `chrome.google.com` and `code.google.com`."

DNSSEC server issues this signed data in response to any name between `chrome` and `code`.

Implementing "between" is tricky but theoretically possible.

Consequence: If you deploy DNSSEC then you are exposing *all* of your DNS names!

Newest
"NSEC
exposin
Hash is

Hash-e
Attacke
comput
compar
exposed

Small
$\approx 2^{44}$ g

Large c
$\approx 2^{64}$ g

more!

ack on DNSSEC:
DNS response
m saying that
ogle.com

ept this
ure:
er can
the Internet.

ature
oes Google
atures for
es? Too many!

DNSSEC solution: Sign multi-NXDOMAIN such as "there are no names between `chrome.google.com` and `code.google.com`."

DNSSEC server issues this signed data in response to any name between `chrome` and `code`. Implementing "between" is tricky but theoretically possible.

Consequence: If you deploy DNSSEC then you are exposing *all* of your DNS names!

Newest DNSSEC
"NSEC3" (2008
exposing *hashes*
Hash is 150 SHA

Hash-enumeratio
Attacker guesses
computes their h
compares to the
exposed by DNS

Small 10-comput
$\approx 2^{44}$ guesses/ye

Large company c
$\approx 2^{64}$ guesses/ye

NSSEC:
nse
hat

net.

e

nany!

DNSSEC solution: Sign multi-NXDOMAIN such as "there are no names between `chrome.google.com` and `code.google.com`."

DNSSEC server issues this signed data in response to any name between `chrome` and `code`. Implementing "between" is tricky but theoretically possible.

Consequence: If you deploy DNSSEC then you are exposing *all* of your DNS names!

Newest DNSSEC variant: "NSEC3" (2008 Laurie), exposing *hashes* of DNS na Hash is 150 SHA-1 iteratio

Hash-enumeration attack: Attacker guesses many nam computes their hashes, compares to the hashes exposed by DNSSEC+NSE

Small 10-computer cluster: $\approx 2^{44}$ guesses/year.

Large company or botnet: $\approx 2^{64}$ guesses/year.

DNSSEC solution: Sign multi-NXDOMAIN such as "there are no names between `chrome.google.com` and `code.google.com`."

DNSSEC server issues this signed data in response to any name between `chrome` and `code`. Implementing "between" is tricky but theoretically possible.

Consequence: If you deploy DNSSEC then you are exposing *all* of your DNS names!

Newest DNSSEC variant: "NSEC3" (2008 Laurie), exposing *hashes* of DNS names. Hash is 150 SHA-1 iterations.

Hash-enumeration attack: Attacker guesses many names, computes their hashes, compares to the hashes exposed by DNSSEC+NSEC3.

Small 10-computer cluster: $\approx 2^{44}$ guesses/year.

Large company or botnet: $\approx 2^{64}$ guesses/year.

EC solution: Sign
NXDOMAIN such as
are no names between
e.google.com and
google.com."

EC server issues this signed
response to any name
n chrome and code.
henting "between" is tricky
eoretically possible.

quence: If you deploy
EC then you are exposing
your DNS names!

Newest DNSSEC variant:
"NSEC3" (2008 Laurie),
exposing *hashes* of DNS names.
Hash is 150 SHA-1 iterations.

Hash-enumeration attack:
Attacker guesses many names,
computes their hashes,
compares to the hashes
exposed by DNSSEC+NSEC3.

Small 10-computer cluster:
$\approx 2^{44}$ guesses/year.

Large company or botnet:
$\approx 2^{64}$ guesses/year.

Withou
attacke
for eac

Floodin
$\approx 2^{37}$ g

Compa
DNSSE
makes
makes
for a w

DNSSE
as bein
but it s
compar

n: Sign
IN such as
mes between
.com and
om."

issues this signed
to any name
and code.
etween" is tricky
possible.

you deploy
ou are exposing
names!

Newest DNSSEC variant:
"NSEC3" (2008 Laurie),
exposing *hashes* of DNS names.
Hash is 150 SHA-1 iterations.

Hash-enumeration attack:
Attacker guesses many names,
computes their hashes,
compares to the hashes
exposed by DNSSEC+NSEC3.

Small 10-computer cluster:
$\approx 2^{44}$ guesses/year.

Large company or botnet:
$\approx 2^{64}$ guesses/year.

Without DNSSE
attacker has to s
for each guessed

Flooding a 4Mbp
$\approx 2^{37}$ guesses/ye

Compared to nor
DNSSEC+NSEC
makes guessing *s*
makes it *millions*
for a well-equipp

DNSSEC+NSEC
as being better t
but it still loses p
compared to nor

s
een

signed
me
.
s tricky

y
osing

Newest DNSSEC variant:
"NSEC3" (2008 Laurie),
exposing *hashes* of DNS names.
Hash is 150 SHA-1 iterations.

Hash-enumeration attack:
Attacker guesses many names,
computes their hashes,
compares to the hashes
exposed by DNSSEC+NSEC3.

Small 10-computer cluster:
$\approx 2^{44}$ guesses/year.

Large company or botnet:
$\approx 2^{64}$ guesses/year.

Without DNSSEC,
attacker has to send query
for each guessed name.

Flooding a 4Mbps connect
$\approx 2^{37}$ guesses/year.

Compared to normal DNS,
DNSSEC+NSEC3
makes guessing *silent* and
makes it *millions of times*
for a well-equipped attacke

DNSSEC+NSEC3 is adver
as being better than DNS
but it still loses privacy
compared to normal DNS.

Newest DNSSEC variant:
"NSEC3" (2008 Laurie),
exposing *hashes* of DNS names.
Hash is 150 SHA-1 iterations.

Hash-enumeration attack:
Attacker guesses many names,
computes their hashes,
compares to the hashes
exposed by DNSSEC+NSEC3.

Small 10-computer cluster:
$\approx 2^{44}$ guesses/year.

Large company or botnet:
$\approx 2^{64}$ guesses/year.

Without DNSSEC,
attacker has to send query
for each guessed name.

Flooding a 4Mbps connection:
$\approx 2^{37}$ guesses/year.

Compared to normal DNS,
DNSSEC+NSEC3
makes guessing *silent* and
makes it *millions of times faster*
for a well-equipped attacker.

DNSSEC+NSEC3 is advertised
as being better than DNSSEC;
but it still loses privacy
compared to normal DNS.

t DNSSEC variant:
C3" (2008 Laurie),
ng *hashes* of DNS names.
s 150 SHA-1 iterations.

numeration attack:
er guesses many names,
tes their hashes,
res to the hashes
d by DNSSEC+NSEC3.

10-computer cluster:
guesses/year.

company or botnet:
guesses/year.

Without DNSSEC,
attacker has to send query
for each guessed name.

Flooding a 4Mbps connection:
$\approx 2^{37}$ guesses/year.

Compared to normal DNS,
DNSSEC+NSEC3
makes guessing *silent* and
makes it *millions of times faster*
for a well-equipped attacker.

DNSSEC+NSEC3 is advertised
as being better than DNSSEC;
but it still loses privacy
compared to normal DNS.

Precom

DNSSE
Hundre
all cach
and all
need to
precom

DNSSE
admini
simple

DNSSE
DNSSE
DNSSE

C variant:
Laurie),
of DNS names.
A-1 iterations.

on attack:
many names,
ashes,
hashes
SEC+NSEC3.

ter cluster:
ear.

or botnet:
ear.

Without DNSSEC,
attacker has to send query
for each guessed name.

Flooding a 4Mbps connection:
$\approx 2^{37}$ guesses/year.

Compared to normal DNS,
DNSSEC+NSEC3
makes guessing *silent* and
makes it *millions of times faster*
for a well-equipped attacker.

DNSSEC+NSEC3 is advertised
as being better than DNSSEC;
but it still loses privacy
compared to normal DNS.

Precomputation

DNSSEC is pain
Hundreds of DN
all caches, all se
and all managem
need to be modif
precompute and

DNSSEC is pain
administrators, fa
simple upgrade.

DNSSEC hurts p
DNSSEC hurts re
DNSSEC aids de

ames.
ons.

mes,

EC3.

Without DNSSEC,
attacker has to send query
for each guessed name.

Flooding a 4Mbps connection:
$\approx 2^{37}$ guesses/year.

Compared to normal DNS,
DNSSEC+NSEC3
makes guessing *silent* and
makes it *millions of times faster*
for a well-equipped attacker.

DNSSEC+NSEC3 is advertised
as being better than DNSSEC;
but it still loses privacy
compared to normal DNS.

Precomputation impact su

DNSSEC is pain for implen
Hundreds of DNS program
all caches, all servers,
and all management tools–
need to be modified to
precompute and store signa

DNSSEC is pain for
administrators, far beyond
simple upgrade.

DNSSEC hurts privacy.
DNSSEC hurts reliability.
DNSSEC aids denial of ser

Without DNSSEC,
attacker has to send query
for each guessed name.

Flooding a 4Mbps connection:
$\approx 2^{37}$ guesses/year.

Compared to normal DNS,
DNSSEC+NSEC3
makes guessing *silent* and
makes it *millions of times faster*
for a well-equipped attacker.

DNSSEC+NSEC3 is advertised
as being better than DNSSEC;
but it still loses privacy
compared to normal DNS.

Precomputation impact summary:

DNSSEC is pain for implementors.
Hundreds of DNS programs—
all caches, all servers,
and all management tools—
need to be modified to
precompute and store signatures.

DNSSEC is pain for
administrators, far beyond a
simple upgrade.

DNSSEC hurts privacy.
DNSSEC hurts reliability.
DNSSEC aids denial of service.

ut DNSSEC,
er has to send query
h guessed name.

ng a 4Mbps connection:
guesses/year.

red to normal DNS,
EC+NSEC3
guessing *silent* and
it *millions of times faster*
ell-equipped attacker.

EC+NSEC3 is advertised
g better than DNSSEC;
still loses privacy
red to normal DNS.

Precomputation impact summary:

DNSSEC is pain for implementors.
Hundreds of DNS programs—
all caches, all servers,
and all management tools—
need to be modified to
precompute and store signatures.

DNSSEC is pain for
administrators, far beyond a
simple upgrade.

DNSSEC hurts privacy.
DNSSEC hurts reliability.
DNSSEC aids denial of service.

Rethin

Conver
DNSSE
sacrific
creatin
is nece

Can we
withou
Let's c

C,
end query
name.

os connection:
ear.

rmal DNS,
3
silent and
of times faster
ed attacker.

3 is advertised
han DNSSEC;
privacy
mal DNS.

Precomputation impact summary:

DNSSEC is pain for implementors.
Hundreds of DNS programs—
all caches, all servers,
and all management tools—
need to be modified to
precompute and store signatures.

DNSSEC is pain for
administrators, far beyond a
simple upgrade.

DNSSEC hurts privacy.
DNSSEC hurts reliability.
DNSSEC aids denial of service.

Rethinking signat

Conventional wis
DNSSEC's preco
sacrificing securit
creating severe u
is necessary for s

Can we achieve a
without precomp
Let's change the

ion:

faster

er.

tised

SEC;

Precomputation impact summary:

DNSSEC is pain for implementors.
Hundreds of DNS programs—
all caches, all servers,
and all management tools—
need to be modified to
precompute and store signatures.

DNSSEC is pain for
administrators, far beyond a
simple upgrade.

DNSSEC hurts privacy.
DNSSEC hurts reliability.
DNSSEC aids denial of service.

<u>Rethinking signatures</u>

Conventional wisdom:
DNSSEC's precomputation
sacrificing security while
creating severe usability pr
is necessary for speed.

Can we achieve adequate s
without precomputation?
Let's change the design.

Precomputation impact summary:

DNSSEC is pain for implementors.
Hundreds of DNS programs—
all caches, all servers,
and all management tools—
need to be modified to
precompute and store signatures.

DNSSEC is pain for
administrators, far beyond a
simple upgrade.

DNSSEC hurts privacy.
DNSSEC hurts reliability.
DNSSEC aids denial of service.

Rethinking signatures

Conventional wisdom:
DNSSEC's precomputation,
sacrificing security while
creating severe usability problems,
is necessary for speed.

Can we achieve adequate speed
without precomputation?
Let's change the design.

Precomputation impact summary:

DNSSEC is pain for implementors.
Hundreds of DNS programs—
all caches, all servers,
and all management tools—
need to be modified to
precompute and store signatures.

DNSSEC is pain for
administrators, far beyond a
simple upgrade.

DNSSEC hurts privacy.
DNSSEC hurts reliability.
DNSSEC aids denial of service.

Rethinking signatures

Conventional wisdom:
DNSSEC's precomputation,
sacrificing security while
creating severe usability problems,
is necessary for speed.

Can we achieve adequate speed
without precomputation?
Let's change the design.

1. Add encryption.
Want to protect against sabotage
and against espionage.
So use public-key signatures
and public-key encryption.

nputation impact summary:

EC is pain for implementors.
eds of DNS programs—
nes, all servers,
management tools—
o be modified to
npute and store signatures.

EC is pain for
strators, far beyond a
upgrade.

EC hurts privacy.
EC hurts reliability.
EC aids denial of service.

Rethinking signatures

Conventional wisdom:
DNSSEC's precomputation,
sacrificing security while
creating severe usability problems,
is necessary for speed.

Can we achieve adequate speed
without precomputation?
Let's change the design.

1. Add encryption.
Want to protect against sabotage
and against espionage.
So use public-key signatures
and public-key encryption.

2. Me
encryp

"Public
against
in one

"Public
encrypt

No nee
partitio
an encr
an auth
Combi

impact summary:

for implementors.
S programs—
rvers,
nent tools—
fied to
store signatures.

for
ar beyond a

rivacy.
eliability.
nial of service.

Rethinking signatures

Conventional wisdom:
DNSSEC's precomputation,
sacrificing security while
creating severe usability problems,
is necessary for speed.

Can we achieve adequate speed
without precomputation?
Let's change the design.

1. Add encryption.
Want to protect against sabotage
and against espionage.
So use public-key signatures
and public-key encryption.

**2. Merge signin
encryption.**

"Public-key signc
against forgery a
in one step.

"Public-key auth
encryption" is ev

No need to
partition the algo
an encryption co
an authentication
Combined algorit

mmary:

mentors.

s—

—

atures.

a

vice.

Rethinking signatures

Conventional wisdom:
DNSSEC's precomputation,
sacrificing security while
creating severe usability problems,
is necessary for speed.

Can we achieve adequate speed
without precomputation?
Let's change the design.

1. Add encryption.
Want to protect against sabotage
and against espionage.
So use public-key signatures
and public-key encryption.

2. Merge signing with encryption.

"Public-key signcryption" p
against forgery and eavesd
in one step.

"Public-key authenticated
encryption" is even faster.

No need to
partition the algorithms int
an encryption component a
an authentication compone
Combined algorithms are fa

Rethinking signatures

Conventional wisdom:
DNSSEC's precomputation,
sacrificing security while
creating severe usability problems,
is necessary for speed.

Can we achieve adequate speed
without precomputation?
Let's change the design.

1. Add encryption.
Want to protect against sabotage
and against espionage.
So use public-key signatures
and public-key encryption.

2. Merge signing with encryption.

"Public-key signcryption" protects
against forgery and eavesdropping
in one step.

"Public-key authenticated
encryption" is even faster.

No need to
partition the algorithms into
an encryption component and
an authentication component.
Combined algorithms are faster.

king signatures

ntional wisdom:
EC's precomputation,
ing security while
g severe usability problems,
ssary for speed.

e achieve adequate speed
t precomputation?
hange the design.

d encryption.
to protect against sabotage
gainst espionage.
public-key signatures
ublic-key encryption.

2. Merge signing with encryption.

"Public-key signcryption" protects against forgery and eavesdropping in one step.

"Public-key authenticated encryption" is even faster.

No need to
partition the algorithms into
an encryption component and
an authentication component.
Combined algorithms are faster.

3. Me across

It's sill
to auth
to the

"Hybri
is muc

Examp
generat
authcry
uses th
authen

tures

dom:
mputation,
ty while
sability problems,
peed.

adequate speed
utation?
design.

ion.
against sabotage
onage.
y signatures
ncryption.

2. Merge signing with encryption.

"Public-key signcryption" protects against forgery and eavesdropping in one step.

"Public-key authenticated encryption" is even faster.

No need to
partition the algorithms into
an encryption component and
an authentication component.
Combined algorithms are faster.

3. Merge public across multiple

It's silly for a sen
to authcrypt two
to the same recip

"Hybrid cryptogr
is much faster.

Example: Sender
generates a rand
authcrypts the A
uses the AES key
authenticate both

n,

oblems,

speed

botage

es

2. Merge signing with encryption.

"Public-key signcryption" protects against forgery and eavesdropping in one step.

"Public-key authenticated encryption" is even faster.

No need to partition the algorithms into an encryption component and an authentication component. Combined algorithms are faster.

3. Merge public-key ope across multiple messages

It's silly for a sender to authcrypt two messages to the same recipient.

"Hybrid cryptography" is much faster.

Example: Sender generates a random AES k authcrypts the AES key, uses the AES key to encryp authenticate both message

2. Merge signing with encryption.

"Public-key signcryption" protects against forgery and eavesdropping in one step.

"Public-key authenticated encryption" is even faster.

No need to partition the algorithms into an encryption component and an authentication component. Combined algorithms are faster.

3. Merge public-key operations across multiple messages.

It's silly for a sender to authcrypt two messages to the same recipient.

"Hybrid cryptography" is much faster.

Example: Sender generates a random AES key, authcrypts the AES key, uses the AES key to encrypt and authenticate both messages.

rge signing with
ption.

c-key signcryption" protects
t forgery and eavesdropping
step.

c-key authenticated
tion" is even faster.

ed to
on the algorithms into
ryption component and
hentication component.
hed algorithms are faster.

3. Merge public-key operations across multiple messages.

It's silly for a sender
to authcrypt two messages
to the same recipient.

"Hybrid cryptography"
is much faster.

Example: Sender
generates a random AES key,
authcrypts the AES key,
uses the AES key to encrypt and
authenticate both messages.

4. Cho

256-bit
using p

489069
new co

5355 cy
authen

6786 cy
a legiti

3465 cy
510-by

ng with

cryption" protects
nd eavesdropping

enticated
en faster.

orithms into
mponent and
n component.
thms are faster.

3. Merge public-key operations across multiple messages.

It's silly for a sender
to authcrypt two messages
to the same recipient.

"Hybrid cryptography"
is much faster.

Example: Sender
generates a random AES key,
authcrypts the AES key,
uses the AES key to encrypt and
authenticate both messages.

4. Choose sensi

256-bit elliptic-cu
using public-dom

489069 Core 2 cy
new communicat

5355 cycles to en
authenticate a 51

6786 cycles to ve
a legitimate 510-

3465 cycles to re
510-byte message

protects
ropping

to
and
ent.
aster.

3. Merge public-key operations across multiple messages.

It's silly for a sender
to authcrypt two messages
to the same recipient.

"Hybrid cryptography"
is much faster.

Example: Sender
generates a random AES key,
authcrypts the AES key,
uses the AES key to encrypt and
authenticate both messages.

4. Choose sensible primi

256-bit elliptic-curve crypto
using public-domain softwa

489069 Core 2 cycles to ha
new communication partne

5355 cycles to encrypt and
authenticate a 510-byte me

6786 cycles to verify and d
a legitimate 510-byte mess

3465 cycles to reject a forg
510-byte message.

3. Merge public-key operations across multiple messages.

It's silly for a sender
to authcrypt two messages
to the same recipient.

"Hybrid cryptography"
is much faster.

Example: Sender
generates a random AES key,
authcrypts the AES key,
uses the AES key to encrypt and
authenticate both messages.

4. Choose sensible primitives.

256-bit elliptic-curve cryptography
using public-domain software:

489069 Core 2 cycles to handle a
new communication partner.

5355 cycles to encrypt and
authenticate a 510-byte message.

6786 cycles to verify and decrypt
a legitimate 510-byte message.

3465 cycles to reject a forged
510-byte message.

rge public-key operations multiple messages.

y for a sender
ncrypt two messages
same recipient.

d cryptography"
h faster.

le: Sender
tes a random AES key,
ypts the AES key,
e AES key to encrypt and
ticate both messages.

4. Choose sensible primitives.

256-bit elliptic-curve cryptography using public-domain software:

489069 Core 2 cycles to handle a new communication partner.

5355 cycles to encrypt and authenticate a 510-byte message.

6786 cycles to verify and decrypt a legitimate 510-byte message.

3465 cycles to reject a forged 510-byte message.

A 2.5G
Q9300
Comple

This C
Each c
2.5 bill

On this
this sof
second
commu
and jus
100000
100000

c-key operations
messages.

nder
messages
pient.

raphy"

r

om AES key,
ES key,
y to encrypt and
h messages.

4. Choose sensible primitives.

256-bit elliptic-curve cryptography using public-domain software:

489069 Core 2 cycles to handle a new communication partner.

5355 cycles to encrypt and authenticate a 510-byte message.

6786 cycles to verify and decrypt a legitimate 510-byte message.

3465 cycles to reject a forged 510-byte message.

A 2.5GHz Intel C
Q9300 CPU cost
Complete compu

This CPU has 4
Each core carries
2.5 billion cycles/

On this compute
this software tak
seconds to handl
communication p
and just 12 secon
10000000 incomi
10000000 outgoi

rations

s.

ey,

pt and

s.

4. Choose sensible primitives.

256-bit elliptic-curve cryptography using public-domain software:

489069 Core 2 cycles to handle a new communication partner.

5355 cycles to encrypt and authenticate a 510-byte message.

6786 cycles to verify and decrypt a legitimate 510-byte message.

3465 cycles to reject a forged 510-byte message.

A 2.5GHz Intel Core 2 Qua
Q9300 CPU costs $225.
Complete computer: $400.

This CPU has 4 cores.
Each core carries out
2.5 billion cycles/second.

On this computer,
this software takes just 49
seconds to handle 1000000
communication partners,
and just 12 seconds to han
10000000 incoming packet
10000000 outgoing packets

4. Choose sensible primitives.

256-bit elliptic-curve cryptography using public-domain software:

489069 Core 2 cycles to handle a new communication partner.

5355 cycles to encrypt and authenticate a 510-byte message.

6786 cycles to verify and decrypt a legitimate 510-byte message.

3465 cycles to reject a forged 510-byte message.

A 2.5GHz Intel Core 2 Quad Q9300 CPU costs $225. Complete computer: $400.

This CPU has 4 cores. Each core carries out 2.5 billion cycles/second.

On this computer, this software takes just 49 seconds to handle 1000000 new communication partners, and just 12 seconds to handle 10000000 incoming packets *and* 10000000 outgoing packets.

oose sensible primitives.

t elliptic-curve cryptography
ublic-domain software:

9 Core 2 cycles to handle a
mmunication partner.

ycles to encrypt and
ticate a 510-byte message.

ycles to verify and decrypt
mate 510-byte message.

ycles to reject a forged
te message.

A 2.5GHz Intel Core 2 Quad
Q9300 CPU costs $225.
Complete computer: $400.

This CPU has 4 cores.
Each core carries out
2.5 billion cycles/second.

On this computer,
this software takes just 49
seconds to handle 1000000 new
communication partners,
and just 12 seconds to handle
10000000 incoming packets *and*
10000000 outgoing packets.

VeriSig
>$100
Interne

In a ty
togethe
from 5

ible primitives.

urve cryptography
ain software:

ycles to handle a
tion partner.

ncrypt and
10-byte message.

erify and decrypt
-byte message.

ject a forged
e.

A 2.5GHz Intel Core 2 Quad
Q9300 CPU costs $225.
Complete computer: $400.

This CPU has 4 cores.
Each core carries out
2.5 billion cycles/second.

On this computer,
this software takes just 49
seconds to handle 1000000 new
communication partners,
and just 12 seconds to handle
10000000 incoming packets *and*
10000000 outgoing packets.

VeriSign is spend
>$100000000 to
Internet's `.com` D

In a typical day,
together handle 3
from 5 million cli

tives.

ography
are:

andle a
er.

essage.

ecrypt
age.

ged

A 2.5GHz Intel Core 2 Quad Q9300 CPU costs $225. Complete computer: $400.

This CPU has 4 cores. Each core carries out 2.5 billion cycles/second.

On this computer, this software takes just 49 seconds to handle 1000000 new communication partners, and just 12 seconds to handle 10000000 incoming packets *and* 10000000 outgoing packets.

VeriSign is spending >$100000000 to upgrade t Internet's `.com` DNS serve

In a typical day, these serv together handle 35 billion q from 5 million clients.

A 2.5GHz Intel Core 2 Quad Q9300 CPU costs $225. Complete computer: $400.

This CPU has 4 cores. Each core carries out 2.5 billion cycles/second.

On this computer, this software takes just 49 seconds to handle 1000000 new communication partners, and just 12 seconds to handle 10000000 incoming packets *and* 10000000 outgoing packets.

VeriSign is spending >$100000000 to upgrade the Internet's `.com` DNS servers.

In a typical day, these servers together handle 35 billion queries from 5 million clients.

A 2.5GHz Intel Core 2 Quad Q9300 CPU costs $225. Complete computer: $400.

This CPU has 4 cores. Each core carries out 2.5 billion cycles/second.

On this computer, this software takes just 49 seconds to handle 1000000 new communication partners, and just 12 seconds to handle 10000000 incoming packets *and* 10000000 outgoing packets.

VeriSign is spending >$100000000 to upgrade the Internet's `.com` DNS servers.

In a typical day, these servers together handle 35 billion queries from 5 million clients.

Total cryptographic cost: about half a day on a single $400 computer with this software.

A 2.5GHz Intel Core 2 Quad Q9300 CPU costs $225. Complete computer: $400.

This CPU has 4 cores. Each core carries out 2.5 billion cycles/second.

On this computer, this software takes just 49 seconds to handle 1000000 new communication partners, and just 12 seconds to handle 10000000 incoming packets *and* 10000000 outgoing packets.

VeriSign is spending >$100000000 to upgrade the Internet's `.com` DNS servers.

In a typical day, these servers together handle 35 billion queries from 5 million clients.

Total cryptographic cost: about half a day on a single $400 computer with this software.

Verisign says that it wants to be prepared for 4 trillion packets/day. Cryptographic cost of 4 trillion partners/day with this software: < 3000 computers.

Hz Intel Core 2 Quad
CPU costs $225.
ete computer: $400.

PU has 4 cores.
ore carries out
ion cycles/second.

s computer,
ftware takes just 49
s to handle 1000000 new
unication partners,
st 12 seconds to handle
000 incoming packets *and*
000 outgoing packets.

VeriSign is spending
>$100000000 to upgrade the
Internet's `.com` DNS servers.

In a typical day, these servers
together handle 35 billion queries
from 5 million clients.

Total cryptographic cost:
about half a day on a single
$400 computer with this software.

Verisign says that it wants to be
prepared for 4 trillion packets/day.
Cryptographic cost of
4 trillion partners/day with this
software: < 3000 computers.

This so
"NaCl"
Crypto
within
(Comp
Engine

Core 2 Quad
s $225.
ter: $400.

cores.
s out
/second.

r,
es just 49
e 1000000 new
partners,
nds to handle
ng packets *and*
ng packets.

VeriSign is spending
$>$\$100000000 to upgrade the
Internet's `.com` DNS servers.

In a typical day, these servers
together handle 35 billion queries
from 5 million clients.

Total cryptographic cost:
about half a day on a single
\$400 computer with this software.

Verisign says that it wants to be
prepared for 4 trillion packets/day.
Cryptographic cost of
4 trillion partners/day with this
software: < 3000 computers.

This software is a
"NaCl" (Network
Cryptography lib
within the EU FP
(Computer Aided
Engineering) proj

ad

new

ndle

s *and*

s.

VeriSign is spending
$>$\$100000000 to upgrade the
Internet's `.com` DNS servers.

In a typical day, these servers
together handle 35 billion queries
from 5 million clients.

Total cryptographic cost:
about half a day on a single
\$400 computer with this software.

Verisign says that it wants to be
prepared for 4 trillion packets/day.
Cryptographic cost of
4 trillion partners/day with this
software: $<$ 3000 computers.

This software is a new libra
"NaCl" (Networking and
Cryptography library) deve
within the EU FP7 "CACE
(Computer Aided Cryptogr
Engineering) project.

VeriSign is spending
>$100000000 to upgrade the
Internet's `.com` DNS servers.

In a typical day, these servers
together handle 35 billion queries
from 5 million clients.

Total cryptographic cost:
about half a day on a single
$400 computer with this software.

Verisign says that it wants to be
prepared for 4 trillion packets/day.
Cryptographic cost of
4 trillion partners/day with this
software: < 3000 computers.

This software is a new library
"NaCl" (Networking and
Cryptography library) developed
within the EU FP7 "CACE"
(Computer Aided Cryptography
Engineering) project.

VeriSign is spending
$>$\$100000000 to upgrade the
Internet's `.com` DNS servers.

In a typical day, these servers
together handle 35 billion queries
from 5 million clients.

Total cryptographic cost:
about half a day on a single
\$400 computer with this software.

Verisign says that it wants to be
prepared for 4 trillion packets/day.
Cryptographic cost of
4 trillion partners/day with this
software: $<$ 3000 computers.

This software is a new library
"NaCl" (Networking and
Cryptography library) developed
within the EU FP7 "CACE"
(Computer Aided Cryptography
Engineering) project.

News: All parts of NaCl
needed for DNS are done!

VeriSign is spending
$>$\$100000000 to upgrade the
Internet's `.com` DNS servers.

In a typical day, these servers
together handle 35 billion queries
from 5 million clients.

Total cryptographic cost:
about half a day on a single
\$400 computer with this software.

Verisign says that it wants to be
prepared for 4 trillion packets/day.
Cryptographic cost of
4 trillion partners/day with this
software: $<$ 3000 computers.

This software is a new library
"NaCl" (Networking and
Cryptography library) developed
within the EU FP7 "CACE"
(Computer Aided Cryptography
Engineering) project.

News: All parts of NaCl
needed for DNS are done!

Actually done three months ago.
Subsequent time has been QA.
This month QA will finish
and NaCl will be released.

VeriSign is spending
$>$\$100000000 to upgrade the
Internet's `.com` DNS servers.

In a typical day, these servers
together handle 35 billion queries
from 5 million clients.

Total cryptographic cost:
about half a day on a single
\$400 computer with this software.

Verisign says that it wants to be
prepared for 4 trillion packets/day.
Cryptographic cost of
4 trillion partners/day with this
software: $<$ 3000 computers.

This software is a new library
"NaCl" (Networking and
Cryptography library) developed
within the EU FP7 "CACE"
(Computer Aided Cryptography
Engineering) project.

News: All parts of NaCl
needed for DNS are done!

Actually done three months ago.
Subsequent time has been QA.
This month QA will finish
and NaCl will be released.

Want to peek?
`http://nacl.cace-project.eu`

gn is spending
000000 to upgrade the
t's .com DNS servers.

pical day, these servers
er handle 35 billion queries
million clients.

cryptographic cost:
half a day on a single
omputer with this software.

n says that it wants to be
ed for 4 trillion packets/day.
graphic cost of
on partners/day with this
re: $<$ 3000 computers.

This software is a new library "NaCl" (Networking and Cryptography library) developed within the EU FP7 "CACE" (Computer Aided Cryptography Engineering) project.

News: All parts of NaCl needed for DNS are done!

Actually done three months ago.
Subsequent time has been QA.
This month QA will finish
and NaCl will be released.

Want to peek?
http://nacl.cace-project.eu

The DN

DNSCu
to add
(RSA-1
DNSCu
and so
and av
to the

Despite
DNSCu
softwar
and ve
adminis

lling
upgrade the
DNS servers.

these servers
35 billion queries
ients.

hic cost:
on a single
with this software.

t it wants to be
illion packets/day.
ost of
s/day with this
0 computers.

This software is a new library "NaCl" (Networking and Cryptography library) developed within the EU FP7 "CACE" (Computer Aided Cryptography Engineering) project.

News: All parts of NaCl needed for DNS are done!

Actually done three months ago.
Subsequent time has been QA.
This month QA will finish
and NaCl will be released.

Want to peek?
http://nacl.cace-project.eu

The DNSCurve p

DNSCurve uses N
to add heavy-dut
(RSA-1024 has 8
DNSCurve has 1
and some confid
and availability
to the Domain N

Despite all this s
DNSCurve is ver
software authors
and very easy for
administrators to

the
rs.

ers
queries

e
oftware.

to be
ets/day.

this
rs.

This software is a new library
"NaCl" (Networking and
Cryptography library) developed
within the EU FP7 "CACE"
(Computer Aided Cryptography
Engineering) project.

News: All parts of NaCl
needed for DNS are done!

Actually done three months ago.
Subsequent time has been QA.
This month QA will finish
and NaCl will be released.

Want to peek?
http://nacl.cace-project.eu

The DNSCurve project

DNSCurve uses NaCl
to add heavy-duty integrity
(RSA-1024 has 80-bit secu
DNSCurve has 128-bit secu
and some confidentiality
and availability
to the Domain Name Syste

Despite all this security,
DNSCurve is very easy for
software authors to implem
and very easy for
administrators to deploy.

This software is a new library "NaCl" (Networking and Cryptography library) developed within the EU FP7 "CACE" (Computer Aided Cryptography Engineering) project.

News: All parts of NaCl needed for DNS are done!

Actually done three months ago. Subsequent time has been QA. This month QA will finish and NaCl will be released.

Want to peek?
http://nacl.cace-project.eu

The DNSCurve project

DNSCurve uses NaCl to add heavy-duty integrity (RSA-1024 has 80-bit security; DNSCurve has 128-bit security) *and* some confidentiality *and* availability to the Domain Name System.

Despite all this security, DNSCurve is very easy for DNS software authors to implement and very easy for administrators to deploy.

oftware is a new library
" (Networking and
graphy library) developed
the EU FP7 "CACE"
uter Aided Cryptography
ering) project.

All parts of NaCl
for DNS are done!

ly done three months ago.
uent time has been QA.
onth QA will finish
aCl will be released.

to peek?
//nacl.cace-project.eu

The DNSCurve project

DNSCurve uses NaCl
to add heavy-duty integrity
(RSA-1024 has 80-bit security;
DNSCurve has 128-bit security)
and some confidentiality
and availability
to the Domain Name System.

Despite all this security,
DNSCurve is very easy for DNS
software authors to implement
and very easy for
administrators to deploy.

Admini
the `ls`
to supp
or insta
alongsi

Admini
change
does *n*
for upc
does *n*

a new library
king and
rary) developed
P7 "CACE"
d Cryptography
ject.

of NaCl
are done!

ree months ago.
has been QA.
will finish
released.

ace-project.eu

The DNSCurve project

DNSCurve uses NaCl
to add heavy-duty integrity
(RSA-1024 has 80-bit security;
DNSCurve has 128-bit security)
and some confidentiality
and availability
to the Domain Name System.

Despite all this security,
DNSCurve is very easy for DNS
software authors to implement
and very easy for
administrators to deploy.

Administrator ha
the `lsec.be` serv
to support DNSC
or install a DNS
alongside the ser

Administrator do
change database
does *not* need to
does *not* need ne
for updating DN
does *not* risk DN

ary

loped
”
raphy

s ago.
QA.

ect.eu

The DNSCurve project

DNSCurve uses NaCl
to add heavy-duty integrity
(RSA-1024 has 80-bit security;
DNSCurve has 128-bit security)
and some confidentiality
and availability
to the Domain Name System.

Despite all this security,
DNSCurve is very easy for DNS
software authors to implement
and very easy for
administrators to deploy.

Administrator has to chang
the `1sec.be` server
to support DNSCurve,
or install a DNSCurve forw
alongside the server.

Administrator does *not* nee
change database software,
does *not* need to store sign
does *not* need new procedu
for updating DNS records,
does *not* risk DNSSEC suic

The DNSCurve project

DNSCurve uses NaCl
to add heavy-duty integrity
(RSA-1024 has 80-bit security;
DNSCurve has 128-bit security)
and some confidentiality
and availability
to the Domain Name System.

Despite all this security,
DNSCurve is very easy for DNS
software authors to implement
and very easy for
administrators to deploy.

Administrator has to change
the `1sec.be` server
to support DNSCurve,
or install a DNSCurve forwarder
alongside the server.

Administrator does *not* need to
change database software,
does *not* need to store signatures,
does *not* need new procedures
for updating DNS records, and
does *not* risk DNSSEC suicide.

NSCurve project

urve uses NaCl
heavy-duty integrity
1024 has 80-bit security;
urve has 128-bit security)
me confidentiality
ailability
Domain Name System.

e all this security,
urve is very easy for DNS
re authors to implement
ry easy for
strators to deploy.

Administrator has to change
the `lsec.be` server
to support DNSCurve,
or install a DNSCurve forwarder
alongside the server.

Administrator does *not* need to
change database software,
does *not* need to store signatures,
does *not* need new procedures
for updating DNS records, and
does *not* risk DNSSEC suicide.

Admini
server
to a se
the DN

The `.b`
and da
and we
already
`lsec.b`
by the

project

NaCl
ty integrity
80-bit security;
28-bit security)
entiality

lame System.

ecurity,
y easy for DNS
to implement
r

o deploy.

Administrator has to change
the `lsec.be` server
to support DNSCurve,
or install a DNSCurve forwarder
alongside the server.

Administrator does *not* need to
change database software,
does *not* need to store signatures,
does *not* need new procedures
for updating DNS records, and
does *not* risk DNSSEC suicide.

Administrator ch
server name such
to a server name
the DNSCurve pu

The `.be` server
and database sof
and web interfac
already support
`lsec.be` server
by the `lsec.be`

rity;
urity)

em.

DNS
nent

Administrator has to change
the `lsec.be` server
to support DNSCurve,
or install a DNSCurve forwarder
alongside the server.

Administrator does *not* need to
change database software,
does *not* need to store signatures,
does *not* need new procedures
for updating DNS records, and
does *not* risk DNSSEC suicide.

Administrator changes
server name such as `ns2`
to a server name that enco
the DNSCurve public key.

The `.be` server
and database software
and web interface
already support
`lsec.be` server names sele
by the `lsec.be` administra

Administrator has to change
the `lsec.be` server
to support DNSCurve,
or install a DNSCurve forwarder
alongside the server.

Administrator does *not* need to
change database software,
does *not* need to store signatures,
does *not* need new procedures
for updating DNS records, and
does *not* risk DNSSEC suicide.

Administrator changes
server name such as `ns2`
to a server name that encodes
the DNSCurve public key.

The `.be` server
and database software
and web interface
already support
`lsec.be` server names selected
by the `lsec.be` administrator!

istrator has to change
ec.be server
port DNSCurve,
all a DNSCurve forwarder
de the server.

istrator does *not* need to
database software,
ot need to store signatures,
ot need new procedures
dating DNS records, and
ot risk DNSSEC suicide.

Administrator changes
server name such as `ns2`
to a server name that encodes
the DNSCurve public key.

The `.be` server
and database software
and web interface
already support
`lsec.be` server names selected
by the `lsec.be` administrator!

Cache
to supp

Cache
encode
Cache
DNS p
Cache
packets

No ext
Forged
very ef
Denial
much r

s to change
ver
Curve,
Curve forwarder
ver.

es *not* need to
software,
store signatures,
ew procedures
S records, and
NSSEC suicide.

Administrator changes
server name such as `ns2`
to a server name that encodes
the DNSCurve public key.

The `.be` server
and database software
and web interface
already support
`lsec.be` server names selected
by the `lsec.be` administrator!

Cache has to be
to support DNSC

Cache naturally s
encoded DNSCur
Cache encrypts a
DNS packets sen
Cache verifies an
packets received

No extra packets
Forged packets a
very efficiently di
Denial of service
much more diffic

ge

warder

ed to

natures,

ures
and
cide.

Administrator changes
server name such as `ns2`
to a server name that encodes
the DNSCurve public key.

The `.be` server
and database software
and web interface
already support
`lsec.be` server names selected
by the `lsec.be` administrator!

Cache has to be upgraded
to support DNSCurve.

Cache naturally sees the
encoded DNSCurve public
Cache encrypts and authen
DNS packets sent to that s
Cache verifies and decrypts
packets received from that

No extra packets.
Forged packets are
very efficiently discarded.
Denial of service becomes
much more difficult.

Administrator changes
server name such as `ns2`
to a server name that encodes
the DNSCurve public key.

The `.be` server
and database software
and web interface
already support
`lsec.be` server names selected
by the `lsec.be` administrator!

Cache has to be upgraded
to support DNSCurve.

Cache naturally sees the
encoded DNSCurve public key.
Cache encrypts and authenticates
DNS packets sent to that server.
Cache verifies and decrypts DNS
packets received from that server.

No extra packets.
Forged packets are
very efficiently discarded.
Denial of service becomes
much more difficult.

istrator changes
name such as `ns2`
rver name that encodes
NSCurve public key.

be server
tabase software
eb interface
y support
be server names selected
`lsec.be` administrator!

Cache has to be upgraded
to support DNSCurve.

Cache naturally sees the
encoded DNSCurve public key.
Cache encrypts and authenticates
DNS packets sent to that server.
Cache verifies and decrypts DNS
packets received from that server.

No extra packets.
Forged packets are
very efficiently discarded.
Denial of service becomes
much more difficult.

Kamins
actuall
does a
With D
that ca
a secon
ECC op
With 1
1 opera
100% C
before
packet

anges
n as `ns2`
that encodes
ublic key.

ftware
e

names selected
administrator!

Cache has to be upgraded
to support DNSCurve.

Cache naturally sees the
encoded DNSCurve public key.
Cache encrypts and authenticates
DNS packets sent to that server.
Cache verifies and decrypts DNS
packets received from that server.

No extra packets.
Forged packets are
very efficiently discarded.
Denial of service becomes
much more difficult.

Kaminsky, 2008.
actually that fast
does a crypto op
With DJB's samp
that can do 15,0
a second can do
ECC operations
With 1 operation
1 operation outb
100% CPU on 1/
before you've pa
packet"

odes

ected

ator!

Cache has to be upgraded
to support DNSCurve.

Cache naturally sees the
encoded DNSCurve public key.
Cache encrypts and authenticates
DNS packets sent to that server.
Cache verifies and decrypts DNS
packets received from that server.

No extra packets.
Forged packets are
very efficiently discarded.
Denial of service becomes
much more difficult.

Kaminsky, 2008.12: "It's n
actually that fast. . . . DNS
does a crypto operation pe
With DJB's sample code, a
that can do 15,000 DNS q
a second can do maybe 10
ECC operations per second
With 1 operation inbound
1 operation outbound, that
100% CPU on 1/3 the traf
before you've parsed a sing
packet"

Cache has to be upgraded
to support DNSCurve.

Cache naturally sees the
encoded DNSCurve public key.
Cache encrypts and authenticates
DNS packets sent to that server.
Cache verifies and decrypts DNS
packets received from that server.

No extra packets.
Forged packets are
very efficiently discarded.
Denial of service becomes
much more difficult.

Kaminsky, 2008.12: "It's not
actually that fast. . . . DNSCurve
does a crypto operation per query.
With DJB's sample code, a laptop
that can do 15,000 DNS queries
a second can do maybe 10,000
ECC operations per second.
With 1 operation inbound and
1 operation outbound, that's
100% CPU on 1/3 the traffic
before you've parsed a single DNS
packet"

Cache has to be upgraded
to support DNSCurve.

Cache naturally sees the
encoded DNSCurve public key.
Cache encrypts and authenticates
DNS packets sent to that server.
Cache verifies and decrypts DNS
packets received from that server.

No extra packets.
Forged packets are
very efficiently discarded.
Denial of service becomes
much more difficult.

Kaminsky, 2008.12: "It's not
actually that fast. ... DNSCurve
does a crypto operation per query.
With DJB's sample code, a laptop
that can do 15,000 DNS queries
a second can do maybe 10,000
ECC operations per second.
With 1 operation inbound and
1 operation outbound, that's
100% CPU on 1/3 the traffic
before you've parsed a single DNS packet"

Wrong. There is only one ECC
operation *per communication partner*, not per packet.

has to be upgraded
port DNSCurve.

naturally sees the
d DNSCurve public key.
encrypts and authenticates
ackets sent to that server.
verifies and decrypts DNS
s received from that server.

ra packets.
packets are
fficiently discarded.
of service becomes
more difficult.

Kaminsky, 2008.12: "It's not actually that fast. . . . DNSCurve does a crypto operation per query. With DJB's sample code, a laptop that can do 15,000 DNS queries a second can do maybe 10,000 ECC operations per second. With 1 operation inbound and 1 operation outbound, that's 100% CPU on 1/3 the traffic *before you've parsed a single DNS packet*"

Wrong. There is only one ECC operation *per communication partner*, not per packet.

Kamins
End-to
means
100× i

upgraded
Curve.

sees the
rve public key.
nd authenticates
t to that server.
d decrypts DNS
from that server.

.
re
scarded.
becomes
ult.

Kaminsky, 2008.12: "It's not actually that fast. . . . DNSCurve does a crypto operation per query. With DJB's sample code, a laptop that can do 15,000 DNS queries a second can do maybe 10,000 ECC operations per second. With 1 operation inbound and 1 operation outbound, that's 100% CPU on 1/3 the traffic *before you've parsed a single DNS packet*"

Wrong. There is only one ECC operation *per communication partner*, not per packet.

Kaminsky, 2008.
End-to-end trust
means "abandon
100× increase in

key.
nticates
server.
s DNS
server.

Kaminsky, 2008.12: "It's not actually that fast. . . . DNSCurve does a crypto operation per query. With DJB's sample code, a laptop that can do 15,000 DNS queries a second can do maybe 10,000 ECC operations per second. With 1 operation inbound and 1 operation outbound, that's 100% CPU on 1/3 the traffic *before you've parsed a single DNS packet*"

Wrong. There is only one ECC operation *per communication partner*, not per packet.

Kaminsky, 2008.12:
End-to-end trust with DNS
means "abandoning cachin
100× increase in load."

Kaminsky, 2008.12: "It's not actually that fast. . . . DNSCurve does a crypto operation per query. With DJB's sample code, a laptop that can do 15,000 DNS queries a second can do maybe 10,000 ECC operations per second. With 1 operation inbound and 1 operation outbound, that's 100% CPU on 1/3 the traffic *before you've parsed a single DNS packet*"

Wrong. There is only one ECC operation *per communication partner*, not per packet.

Kaminsky, 2008.12:
End-to-end trust with DNSCurve means "abandoning caching . . . 100× increase in load."

Kaminsky, 2008.12: "It's not actually that fast. . . . DNSCurve does a crypto operation per query. With DJB's sample code, a laptop that can do 15,000 DNS queries a second can do maybe 10,000 ECC operations per second. With 1 operation inbound and 1 operation outbound, that's 100% CPU on 1/3 the traffic *before you've parsed a single DNS packet*"

Wrong. There is only one ECC operation *per communication partner*, not per packet.

Kaminsky, 2008.12:
End-to-end trust with DNSCurve means "abandoning caching . . . 100× increase in load."

Wrong. Users *can* and *should* run caches on their own computers.

Security advantage: User running a DNSCurve cache is protected from intermediate ISP modifying the administrator's DNS data.

Speed advantage: Per-user caches are actually *more* effective than a centralized ISP cache, *decreasing* DNS load.

sky, 2008.12: "It's not
y that fast. . . . DNSCurve
crypto operation per query.
JB's sample code, a laptop
n do 15,000 DNS queries
nd can do maybe 10,000
perations per second.
operation inbound and
ation outbound, that's
CPU on 1/3 the traffic
you've parsed a single DNS
"

. There is only one ECC
on *per communication*
r, not per packet.

Kaminsky, 2008.12:
End-to-end trust with DNSCurve
means "abandoning caching . . .
100× increase in load."

Wrong. Users *can* and *should* run
caches on their own computers.

Security advantage: User running
a DNSCurve cache is protected
from intermediate ISP modifying
the administrator's DNS data.

Speed advantage: Per-user caches
are actually *more* effective
than a centralized ISP cache,
decreasing DNS load.

There
that D
and tha

DNSSE
compro
if admi
signatu
that ha

12: "It's not
t. . . . DNSCurve
eration per query.
ple code, a laptop
00 DNS queries
maybe 10,000
per second.
inbound and
ound, that's
/3 the traffic
rsed a single DNS

only one ECC
mmunication
packet.

Kaminsky, 2008.12:
End-to-end trust with DNSCurve means "abandoning caching . . . 100× increase in load."

Wrong. Users *can* and *should* run caches on their own computers.

Security advantage: User running a DNSCurve cache is protected from intermediate ISP modifying the administrator's DNS data.

Speed advantage: Per-user caches are actually *more* effective than a centralized ISP cache, *decreasing* DNS load.

There *is* somethi
that DNSSEC ac
and that DNSCu

DNSSEC can pro
compromise of D
if administrator g
signatures on an
that has not bee

hot
SCurve
er query.
a laptop
ueries
,000
l.
and
t's
ffic
gle DNS

ECC
ion

Kaminsky, 2008.12:
End-to-end trust with DNSCurve
means "abandoning caching . . .
100× increase in load."

Wrong. Users *can* and *should* run
caches on their own computers.

Security advantage: User running
a DNSCurve cache is protected
from intermediate ISP modifying
the administrator's DNS data.

Speed advantage: Per-user caches
are actually *more* effective
than a centralized ISP cache,
decreasing DNS load.

There *is* something
that DNSSEC accomplishe
and that DNSCurve doesn'

DNSSEC can protect again
compromise of DNS server
if administrator generates
signatures on another mac
that has not been compron

Kaminsky, 2008.12:
End-to-end trust with DNSCurve
means "abandoning caching ...
100× increase in load."

Wrong. Users *can* and *should* run
caches on their own computers.

Security advantage: User running
a DNSCurve cache is protected
from intermediate ISP modifying
the administrator's DNS data.

Speed advantage: Per-user caches
are actually *more* effective
than a centralized ISP cache,
decreasing DNS load.

There *is* something
that DNSSEC accomplishes
and that DNSCurve doesn't:

DNSSEC can protect against
compromise of DNS servers
if administrator generates
signatures on another machine
that has not been compromised.

Kaminsky, 2008.12:
End-to-end trust with DNSCurve
means "abandoning caching ...
100× increase in load."

Wrong. Users *can* and *should* run
caches on their own computers.

Security advantage: User running
a DNSCurve cache is protected
from intermediate ISP modifying
the administrator's DNS data.

Speed advantage: Per-user caches
are actually *more* effective
than a centralized ISP cache,
decreasing DNS load.

There *is* something
that DNSSEC accomplishes
and that DNSCurve doesn't:

DNSSEC can protect against
compromise of DNS servers
if administrator generates
signatures on another machine
that has not been compromised.

Analogy: HTTPSEC can protect
against compromise of HTTP
servers if administrator signs
web pages on another machine.
But does this justify the pain of
DNSSEC+HTTPSEC?

sky, 2008.12:
-end trust with DNSCurve
"abandoning caching . . .
ncrease in load."

. Users *can* and *should* run
on their own computers.

y advantage: User running
Curve cache is protected
ntermediate ISP modifying
ministrator's DNS data.

advantage: Per-user caches
ually *more* effective
centralized ISP cache,
sing DNS load.

There *is* something
that DNSSEC accomplishes
and that DNSCurve doesn't:

DNSSEC can protect against
compromise of DNS servers
if administrator generates
signatures on another machine
that has not been compromised.

Analogy: HTTPSEC can protect
against compromise of HTTP
servers if administrator signs
web pages on another machine.
But does this justify the pain of
DNSSEC+HTTPSEC?

More i
See ht

12:
with DNSCurve
ing caching . . .
load."

n and *should* run
own computers.

ge: User running
he is protected
e ISP modifying
r's DNS data.

e: Per-user caches
e effective
d ISP cache,
load.

There *is* something
that DNSSEC accomplishes
and that DNSCurve doesn't:

DNSSEC can protect against
compromise of DNS servers
if administrator generates
signatures on another machine
that has not been compromised.

Analogy: HTTPSEC can protect
against compromise of HTTP
servers if administrator signs
web pages on another machine.
But does this justify the pain of
DNSSEC+HTTPSEC?

More information
See `http://dns`

SCurve
g . . .

uld run
uters.

running
ected
difying
ata.

r caches

ne,

There *is* something
that DNSSEC accomplishes
and that DNSCurve doesn't:

DNSSEC can protect against
compromise of DNS servers
if administrator generates
signatures on another machine
that has not been compromised.

Analogy: HTTPSEC can protect
against compromise of HTTP
servers if administrator signs
web pages on another machine.
But does this justify the pain of
DNSSEC+HTTPSEC?

More information on DNS
See `http://dnscurve.or`

There *is* something
that DNSSEC accomplishes
and that DNSCurve doesn't:

DNSSEC can protect against
compromise of DNS servers
if administrator generates
signatures on another machine
that has not been compromised.

Analogy: HTTPSEC can protect
against compromise of HTTP
servers if administrator signs
web pages on another machine.
But does this justify the pain of
DNSSEC+HTTPSEC?

More information on DNSCurve:
See `http://dnscurve.org`.

There *is* something
that DNSSEC accomplishes
and that DNSCurve doesn't:

DNSSEC can protect against
compromise of DNS servers
if administrator generates
signatures on another machine
that has not been compromised.

Analogy: HTTPSEC can protect
against compromise of HTTP
servers if administrator signs
web pages on another machine.
But does this justify the pain of
DNSSEC+HTTPSEC?

More information on DNSCurve:
See `http://dnscurve.org`.

Thinking beyond DNS:
Can every Internet packet
be protected in a similar way?

There *is* something
that DNSSEC accomplishes
and that DNSCurve doesn't:

DNSSEC can protect against
compromise of DNS servers
if administrator generates
signatures on another machine
that has not been compromised.

Analogy: HTTPSEC can protect
against compromise of HTTP
servers if administrator signs
web pages on another machine.
But does this justify the pain of
DNSSEC+HTTPSEC?

More information on DNSCurve:
See `http://dnscurve.org`.

Thinking beyond DNS:
Can every Internet packet
be protected in a similar way?

Thinking beyond networking:
When people sacrifice security
and usability for the sake of
performance, are they really
improving performance?

is something
NSSEC accomplishes
at DNSCurve doesn't:

EC can protect against
omise of DNS servers
nistrator generates
ures on another machine
as not been compromised.

y: HTTPSEC can protect
t compromise of HTTP
if administrator signs
ges on another machine.
es this justify the pain of
EC+HTTPSEC?

More information on DNSCurve:
See `http://dnscurve.org`.

Thinking beyond DNS:
Can every Internet packet
be protected in a similar way?

Thinking beyond networking:
When people sacrifice security
and usability for the sake of
performance, are they really
improving performance?

ng
ccomplishes
rve doesn't:

otect against
NS servers
generates
other machine
n compromised.

SEC can protect
ise of HTTP
strator signs
other machine.
stify the pain of
PSEC?

More information on DNSCurve:
See `http://dnscurve.org`.

Thinking beyond DNS:
Can every Internet packet
be protected in a similar way?

Thinking beyond networking:
When people sacrifice security
and usability for the sake of
performance, are they really
improving performance?

s
't:
nst
s
hine
mised.
protect
TP
ns
hine.
ain of

More information on DNSCurve:
See `http://dnscurve.org`.

Thinking beyond DNS:
Can every Internet packet
be protected in a similar way?

Thinking beyond networking:
When people sacrifice security
and usability for the sake of
performance, are they really
improving performance?

More information on DNSCurve:
See `http://dnscurve.org`.

Thinking beyond DNS:
Can every Internet packet
be protected in a similar way?

Thinking beyond networking:
When people sacrifice security
and usability for the sake of
performance, are they really
improving performance?