

Safety in Domain Name Service

Internet Systems Consortium, 30 March 2009

What is DNS?

- Whenever a computer needs information from an Internet source, it uses DNS to find that source
- For example, the name “www.ras.ru” is recorded in DNS as 83.149.246.161
- The computer actually gets its information from 83.149.246.161
- DNS lookup turns a name into a number

Secure DNS

- Usually it is important to know that the DNS answer you have received is authentic
- Secure DNS, usually called DNSSEC, is a protocol standard that allows an internet user to authenticate an answer
- The query and reply are not encrypted. The “Secure” part is not secrecy or privacy, but authenticity.

DNSSEC

- DNSSEC protocols are an Internet communication standard
- Like all protocol standards, the two parties must both obey the protocol or communication will not succeed
- If both parties in a DNS transaction support DNSSEC, then a DNSSEC-enabled query receives a DNSSEC-verified response

What DNSSEC does

- DNSSEC allows a client to verify that the answer it received is authentic
- This verification is performed by the client, and does not involve the server
- If verification succeeds, the client knows
 - The responding server is authentic
 - The reply has not been altered in transit

DNSSEC does not

- Does not encrypt *anything*, either the query or the response
- Does not attempt to hide or conceal anything
- Does not replace regular DNS

DNSSEC processing

- Verify that the signed response to the query is valid and has not been tampered with
- Verify that the key used to sign the query is authentic

Verifying a response

- DNSSEC uses digital signatures
- To verify a query response, the client verifies that its digital signature is valid
- And verifies that the key used for the digital signature is authentic

Digital Signatures

- Digital signatures are a form of cryptography
- You cannot decrypt a digital signature to recover cleartext
- A digital signature gives you a yes/no answer and nothing more

Digital Signatures

- Digital signatures are possible because of public-key cryptographic algorithms
- Classical encryption, in which the same key is used to encode and decode, does not work for digital signatures
- Text is signed with a private key; the signature is verified with the public key

Digital Signatures

- Verifying a digital signature has two parts:
 - Verify that the signature matches the signed text
 - Verify that the public key is authentic
- To verify a digital signature you must use the cryptographic algorithm that signed it

Digital Signatures

- Any public-key cryptographic algorithm can be used for digital signatures
- But if the server signs with an algorithm unknown to the client, the signature cannot be verified
- Client must know which algorithm to use
- And must contain an implementation of it

Key Management

- To authenticate a digital signature, you need the public key
- You can get it from anywhere you trust
- Internet standards provide guidance for finding it
- But you have to be able to trust all of the servers involved

Key Management

- To find a public key, you must start at a trusted place — a “trust anchor”
- Then follow a “chain of trust” until it reaches your destination
- You can trust a public key from that destination because you can trace its pedigree back to a trust anchor

Key Management

- If keys for Secure DNS are retrieved from DNS
- Then a signed zone must have a signed parent, ...
- All the way back to a universally trusted root or anchor

Key Management

- If example.com.ru is signed
- then com.ru must be signed
- and ru must be signed
- and the root must be signed
 - (or ru used as a trust anchor)

DNSSEC Standards

- DNSSEC must be fast, universal, and simple to implement
 - Otherwise no one would use it
- Programmers who create clients must know all encryption algorithms that could be used
 - So that the client can verify any signature

DNSSEC Standards

- Standards include many factors not relevant to an architectural understanding
 - Keys reserved for specific purposes
 - Key rollover
 - Signature expiration
 - Verifying that a name does not exist

DNSSEC Standards

- DNSSEC signature algorithms today
 - RSA with SHA-1
 - DSA (which incorporates SHA-1)
 - Transition soon to RSA with SHA-2
- Future signature algorithms
 - Any public-key algorithm is possible

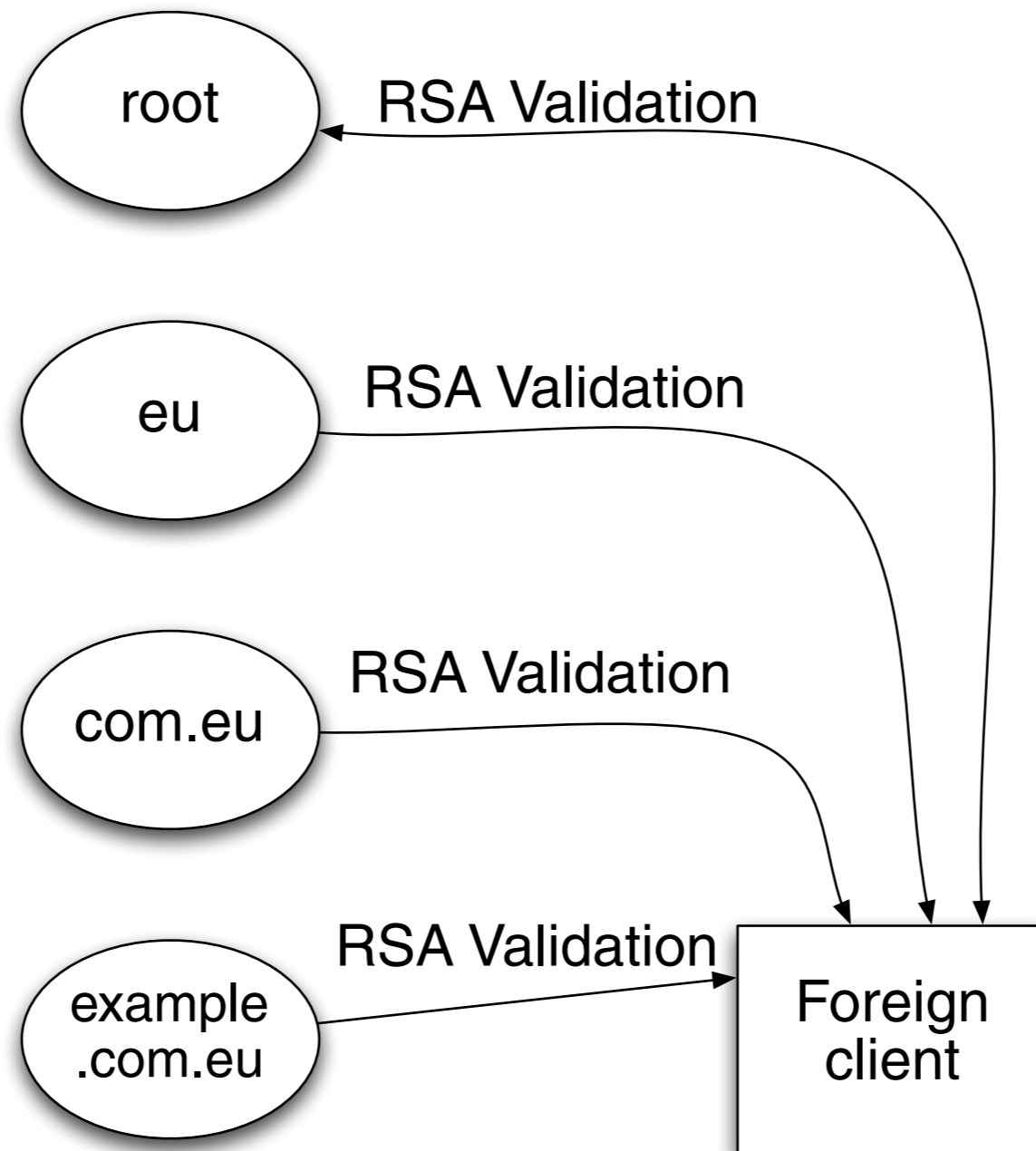
Interoperability

DNSSEC implemented?	Server yes	Server no
Client yes	Authentication	Rejected (treated as an attack)
Client no	Classic behavior	Classic behavior

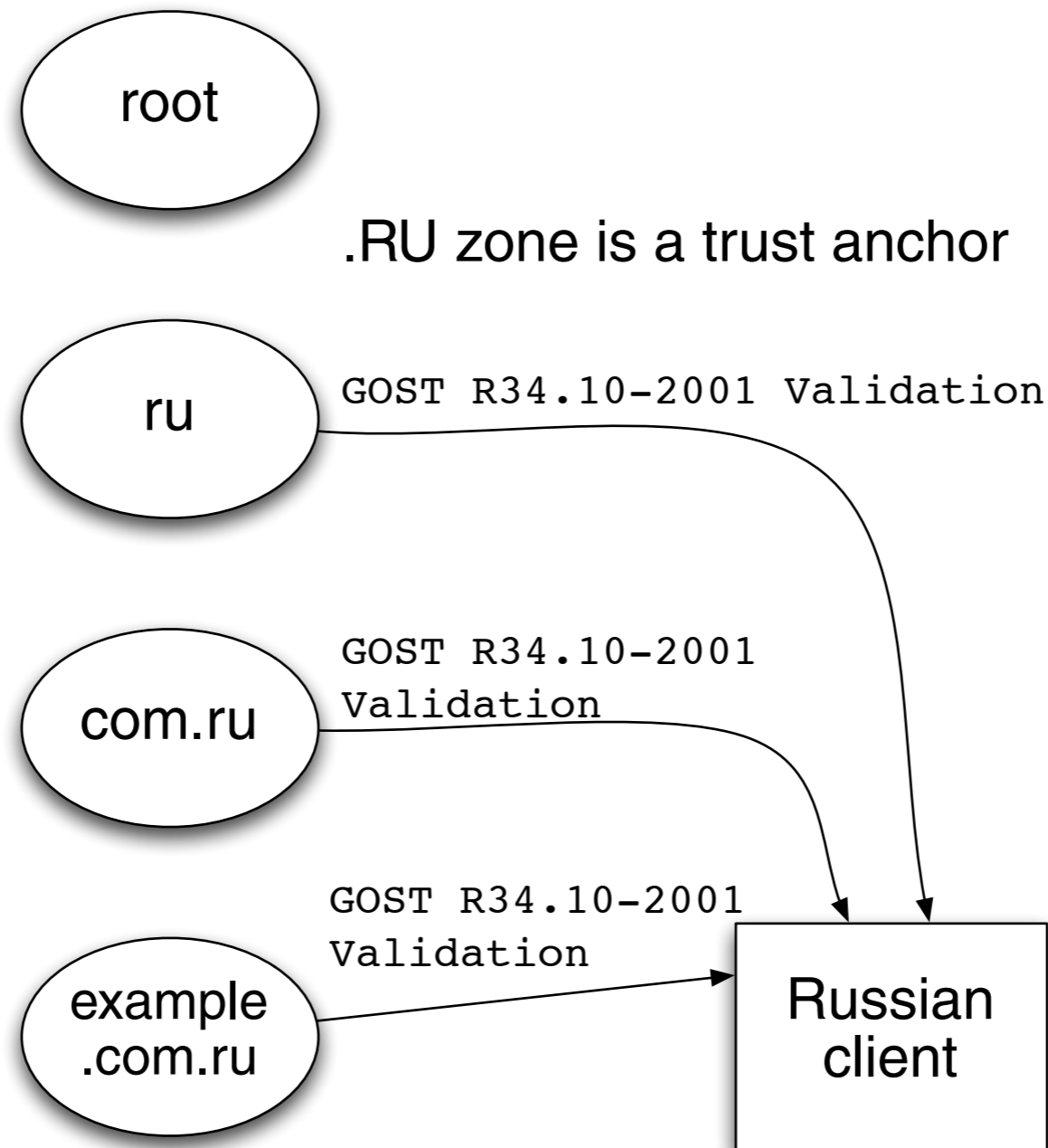
Interoperability with multiple algorithms

DNSSEC implemented?	Server yes	Server no
Client yes this algorithm	Authentication	Rejected
Client yes algorithm no	Treat as insecure	Classic behavior
Client no	Classic behavior	Classic behavior

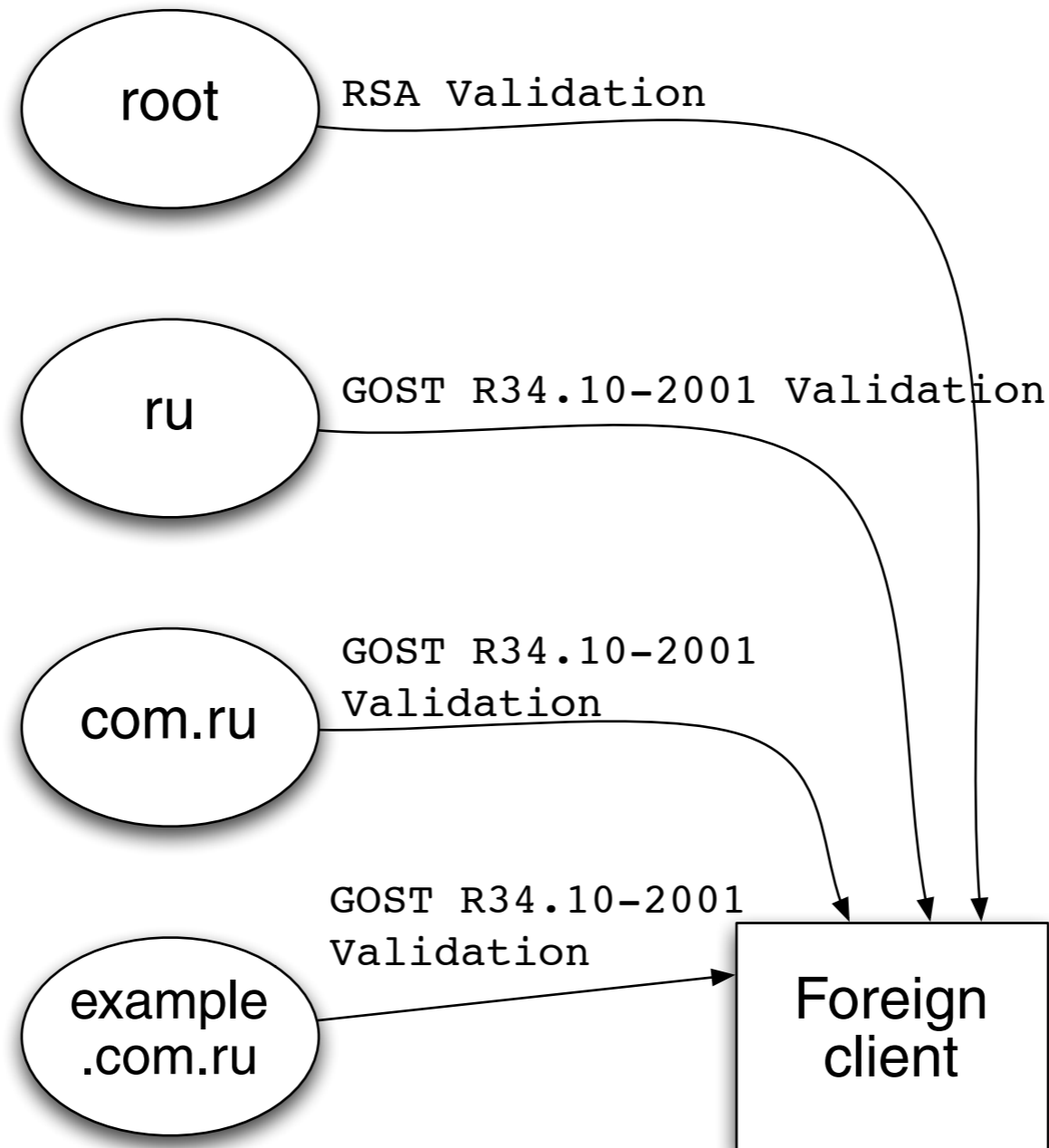
Foreign client validating foreign reply



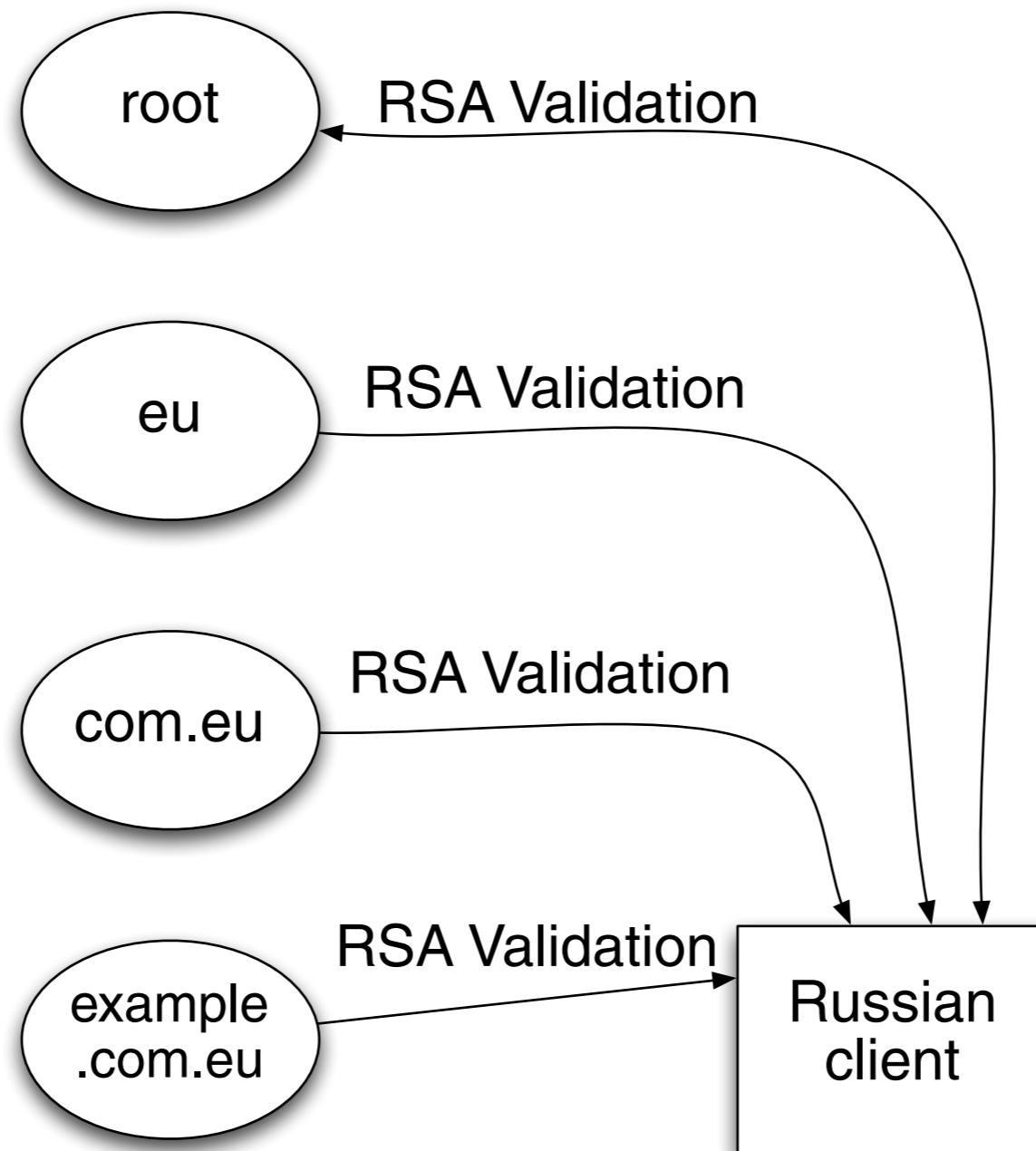
Russian site validating Russian reply



Foreign site validating Russian reply



Russian site validating Foreign reply



Interoperability

- If server implements DNSSEC
- and client does not
- then query/response is classic non-secure DNS

Interoperability

- Digital signatures stored in separate records
- Client won't get them unless it asks for them
- A client without DNSSEC will not ask for them
- So will get classic non-signed data, as always

Interoperability

- If client implements DNSSEC
- and server does not
- then client software decides whether to proceed or to stop
- Only the client knows whether it can use non-secured response

Interoperability

- Existing DNSSEC protocol can encode multiple signature algorithms
- A query for a signature record will get reply containing multiple signatures
- Client must decide which signature to use
- and what to do if client does not support any of those algorithms

Interoperability

Most internet software uses DNS like this:

```
answer = lookup(DNS-name);  
if (answer == NoSuchName)  
    signal appropriate error;  
else  
    continue with processing;
```

Interoperability

A DNSSEC-aware client uses DNS like this:

```
answer = lookup(DNS-name);  
check the signatures;  
figure out what you've got: InvalidResponse,  
AuthenticAnswer, AuthenticError,  
NoSuchName, NoSuchType, etc;  
Take appropriate action;
```

Interoperability

- Merely adding DNSSEC to the server response will not provide the expanded client logic needed to respond to the many possible error conditions
- Having multiple signature algorithms increases the number of cases that the client must deal with

Interoperability

- Use of new algorithm introduces new forms of client failure (don't understand this algorithm)
- The logic to process it must be put into client software, not into DNS library
- But use of a new algorithm within a community that expects it is possible today

Interoperability

- If a zone is signed multiple times,
 - each DNSSEC response will contain more data; network traffic will increase
 - An RFC draft (draft-crocker-dnssec-algo-signal-02) proposes a solution to this problem
- Clients not aware of new algorithm will not be able to authenticate and will treat it as insecure

Conclusions

- New DNSSEC algorithms can be used today inside sub-groups
- DNSSEC Clients outside those groups will not have the software needed to validate
- Non-DNSSEC clients will see same behavior as before
- Internet community is producing draft standards to make all this easier

Conclusions

- DNSSEC algorithms represent an agreement between server and client
- DNSSEC standard specifies a default algorithm so that arbitrary clients can authenticate arbitrary servers
- Client software must do most of the work to adapt to new algorithm

Current deployment

- Several TLDs signed, including .bg and .br
- DNSSEC support in many DNS software implementations
 - Open source: BIND (ISC), nsd (NLnet labs)
 - Best practices under development