Motivations and Terminology for DNSSEC Operations Handover

Stephen D. Crocker
steve@shinkuro.com and

Ólafur Guðmundsson
ogud@shinkuro.com and

Andrew Sullivan
ajs@shinkuro.com

Shinkuro, Inc.
4922 Fairmont Avenue,
Suite 250
Bethesda MD 20814 U.S.A.

***Abstract*—On the contemporary Internet, the DNS services for many domain names are operated by parties other than the registrant of the domain name. If a registrant wishes to move the domain name's DNS operations from one party to another, several different actors need to coordinate their actions. While this has always been true, the addition of DNSSEC signatures and validation to the DNS means that these operations become more delicate; while DNS without DNSSEC is mostly tolerant of data inconsistencies, DNSSEC is designed to detect such inconsistencies and to stop resolution after detection. In order to create a complete description of a smooth operational transfer without the domain having to go through an insecure phase, it is necessary to develop a taxonomy of all the actors in such a transfer, to understand when and how their roles might be combined and what that might mean, and to understand the effects of any actor's failing to act. We provide this description, and go on to suggest the next steps for a complete procedure for DNS operation transfer.**

***Index Terms*—DNSSEC, DNS, domain name operation transfer**

I. Introduction

THE Domain Name System (DNS) is designed to be loosely coherent. Loose coherence is a consequence of the DNS being a distributed database with distributed caches, where it is possible that the same query issued by different sources (from different parts of the network) at the same moment will get different answers. One of the effects of this loose coherence has been a high tolerance for delayed action and outright misconfiguration on the parts of both DNS operators and the system itself. The introduction of the DNS Security Extensions (DNSSEC) means that some of that tolerance will be lost.

Many DNS domain registrants do not actually operate their own DNS zones, instead preferring to hire someone else to do the job. This creates a market (broadly construed) of competing DNS providers. The competitive market in DNS operation means that it is sometimes necessary to move a zone from one operator to another. The loose coherence of the DNS has prevented those moves from being too difficult, but the advent of DNSSEC promises to complicate matters because the failure to time and coordinate transfer activities (and the occasional unwillingness of vendors who are losing a customer to cooperate) means that a misstep can put a zone into a bogus[1] state, causing resolution failures for that domain name.

In the following sections, we first explain why there is a problem to be solved. We then provide a taxonomy of all the roles involved in the current domain name, DNS, and DNSSEC marketplaces, and then sketch a future complete procedure for transferring the DNS operation of a zone without turning off DNSSEC signatures for that zone.

The DNS is defined in RFC 1034 [10] and RFC 1035 [11], with clarifications in RFC 2181 [6]. The relevant DNSSEC documents for the following discussion are the so-called DNSSECbis RFCs [1], [3], [2], [8]. The reader unfamiliar with those specifications may wish to consult the informal glossary in the appendix.

The authors are grateful for the suggestions of three anonymous reviewers, and for editorial suggestions from Paul Kretkowski.

II. The Competitive Environment

While the DNS offers the opportunity to break the name space up into different pieces, each of which may be administered mostly independently, it does not operate as a closed system. To begin with, there are the questions of how domain names and, underneath them, other DNS records, appear in the DNS. In addition, there is the matter of who actually performs the operation of the DNS for the zone in question.

A. The domain name registration market

In parts of the DNS name space, the registration of domain names has itself become a commercial activity. There are two basic models of this activity: a two-party registration model and a three-party registration model.

In the two-party registration model, someone wishing to register a domain name inside a name space approaches the registration authority, and undertakes an agreement with that authority. As a result, the authority enters the registration into the set of names inside the name space. This may result in delegation consequences; see II-B.

In the three-party registration model, someone wishing to register a domain name inside a name space approaches an authorized agent for that name space. The authorized agent contacts the registration authority, and inquires as to whether

the registration is permitted. If so, the agent registers the domain name on behalf of the initial requester, and then enters the registration information about the individual in its own records.

The registration authority described above has historically been called the "registry," and the original requesting party the "registrant." Strictly speaking, the term "registry" describes any level of registration authority in the DNS, but the concentration of commercial activity at or near the root has caused the term to be understood (in some circles) as referring to TLD and near-TLD registries only.

The authorized agent in the three-party model is usually called the "registrar." For historical reasons, this word is also closely associated with the administration of top-level and near-top-level domains. There is nothing about the model that requires such an interpretation, however. For instance, it would not be surprising for a university to have a central registry of domain names, all operated as one zone, but for each university department to control some portion of the name space, and therefore to be working effectively as a registrar.

The three-party registration model is extremely common among TLDs and is so widely considered "normal" that one frequently hears the two-party model described as "the registry is also the registrar." It is also possible to have a hybrid model, in which the registry performs registration functions directly on behalf of some registrants, but also allows other registrars to compete in performing the registration function.

The three-party model varies from registry to registry in its implementation. For instance, some registries require that a lot of ancillary data about the domain name registered also be deposited with the registry (the so-called "thick registry" model). Other registries require nothing more than the registration, the registrar's (or, in the terminology of one protocol, "sponsor's") identifier, and the data necessary to publish the domain name's data in the DNS. More importantly for our purposes, some registries have very strict rules that all the data in the registry must always and only come from the registrars, whereas others permit various classes of information to be maintained directly in the registry by individuals associated with the domain name (such as the registrant, the administrative contact or the technical contact).

The models outlined above are not quite as clean as suggested, because under neither model does the registry have to be a single organization. Instead, the registry might be made up of the registry operator (which generally sets policies, writes contracts and so on), and the technical operator (which generally performs the actual technical operation of registration and, perhaps, other activities). These various business models might be interesting in themselves, but they have in principle no effect on the basic registry function, and do not affect any conclusion about how to perform DNS operation transfers except as matters of implementation detail. Therefore, we shall ignore this distinction.

B. The delegation market

One may distinguish between the registration and delegation functions in a registry. Simply registering a domain in a registry does not automatically entail that the registration is delegated for operation on the Internet. To see that this is so, consider the case of so-called "reserved" names, which may appear in the registry but which are contractually forbidden from resolving on the Internet. Similarly, there are names registered that are not delegated, because they do not meet the requirements for delegation (such as having some minimum number of name server records). These cases demonstrate that registration and receiving a delegation are not the same thing.[1]

Many (perhaps most) registrants registering in a registry expect a delegation of name space. What one receives, in this case, is the ability to designate certain servers as the authoritative DNS servers for that domain; and (accordingly) to operate any services one likes inside the delegated name space (subject to contractual limitations).

Under most circumstances the delegation market is tightly coupled with the registration market and acquiring a registration also entails the right to a delegation. This distinction is important for our purposes, however, because the parties involved in delegation need not be the same as the parties involved in registration.

One must hold a registration to participate in the delegation market. Both registrants and registrars participate in the delegation market. In the two-party registration model, the registrant simply requests delegation from the registry, which involves submitting some number of name server records for the domain to the registry. The registry adds these records to the registry's zone.

In the three-party registration model, the registrar adds the name server records for the domain. These may be records that the registrant asks to have submitted, or they may be records the registrar submits without prompting; it is not unusual to see domains that have been registered but for which the registrant has not added name server records to resolve (usually to a host that serves HTTP requests and nothing more).

C. The DNS operation market

There is no requirement that any or all of the name servers added by a registrant for a domain be under the direct control of the registrant, and many registrants opt to have their DNS services provided by some third party[2] – someone other than the registrant or persons in his employ. In order for this to work, the registrant causes some DNS records to be registered with the DNS provider in a zone with the registrant's

[1] Moreover, as we noted, not all registries are at the top level, and not everyone who registers inside a lower level registry wants or expects a delegation. For instance, it is useful to think of the service blogspot.com as (partly) a registry. But many of the names inside there are not delegations, and people do not think of setting up a blogspot.com blog as automatically entailing a DNS delegation.

[2] A registrant might use more than one DNS provider for a domain, in an effort to diversify dependencies. If there is more than one operator, then even if one of the operators experiences commercial or operational failure, there will still be another operator offering DNS service for the zone. Of course, adding more DNS operators complicates any changes to the DNS for the zone in question since it increases the number of parties whose actions must be coordinated. This is not usually a major complication, since normally there is one master server where changes happen and where signing occurs. So, apart from the complication of increased coordination, there is not much difference between using multiple DNS operators and using one, and we ignore the difference in what follows.

domain name at the apex, and then adds name server records containing the DNS provider's name servers to the registry. The DNS provider may or may not be the registrar of the three-party registration model.

If a registrant moves the operation of some part of the DNS for a zone from one DNS operator to another, we can speak of these as the losing operator and the gaining operator, respectively.

D. The non-DNS operation market

It might seem that the services offered at a domain name (such as web, email and so on) are irrelevant to considerations of the DNS, but they are worth keeping in mind for three reasons:

1) Nobody would put anything in the DNS if it were not for the other services offered at a host.
2) Resolution failures in a domain will affect the services offered there.
3) The services themselves may be contracted out to a party other than the registrant.

During a change of DNS operators, there are two possible scenarios for other services. The first is that the operation of other services remains unchanged; the servers providing the services remain the same, and no zone data needs to be altered to support changes to these other services. This will be the normal case if the DNS operation and the operation of other services are not linked. In this case, there is no reason that a DNS operator change needs to affect non-DNS operators. Frequently, however, the operation of the DNS is just one part of a bundled "hosting service," where the operation of DNS, web servers, mail servers, etc. is all part of a single package. In this case, it is possible that all the services will need to move at the same time, and the change of DNS operators becomes still more delicate.

E. *Summary:* Dramatis Personae

Given the above, the following actors are potentially involved in any transfer of DNS operation from one operator to another:

- The domain name registrant
- The domain name registry
- The domain name registrar
- Other domain name contacts
  - Administrative contact
  - Technical contact
- The losing DNS operator
- The gaining DNS operator
- The losing operator of "bundled" non-DNS services
- The gaining operator of "bundled" non-DNS services

It is possible that continuing operators of non-DNS services at the domain will also need to be aware of the change, particularly in the event that the service relies on hard-coded IP addresses or network ACLs. Further, it is worth emphasizing that the DNS operator might well be the domain name registrar, and that the registration and DNS resolution services may be linked to one another.

The other domain name contacts are only relevant insofar as they are able to effect changes to the delegation data (either directly in the registry, or indirectly by operating through a registrar, but in either case without the participation of the registrant).

III. Transferring Zone Operation Without DNSSEC

A. Isn't this trivial?

While most changes in the DNS are confined to a single zone, and can therefore be undertaken by that zone operator without paying much attention to other zones,[3] changes at the zone cut (the point where a delegation happens from a parent to a child zone) must be coordinated between the parent and the child. Historically, correct maintenance of the data across the zone cut has been troublesome for three reasons. First, because the database is distributed and loosely coherent by design, it is necessary to tolerate some differences across zone cuts. While RFC 1034 [10] requires that the "administrators of both zones should insure that the NS and glue RRs which mark both sides of the cut are consistent and remain so," it is silent on how to do this. Since data will need to be inconsistent across the zone cut at some times in order to introduce changes[4], tolerance of inconsistency between the apex NS records of a zone and the delegation records at the parent is required for the DNS to operate.

Second, because that tolerance needs to be built in, it is not uncommon for administrators of the DNS not to notice such configuration errors. Administrators' expectations in this regard have, moreover, been helped historically by the efforts of DNS server vendors to make every effort in support of resolution attempts. In particular, BIND 4-era systems were rather promiscuous in accepting glue records whatever their source, which meant that resolution continued to work even in the face of serious misconfiguration. While some of these practices have been tightened considerably over time due to their unfortunate security effects, the basic fact is that some inconsistency across zone cuts is a normal part of the functioning global DNS.

Finally, because DNS is designed for loose consistency, the overall system can be made at once more robust and less subject to performance reductions by caching recently seen data in various places around the Internet. Caches improve robustness because even in the event of a transient name server failure, data might be available to answer a given query. It improves performance, obviously, by allowing answers to be provided immediately by a cache instead of needing to ask for the record from the authoritative server for the name.

[3]The exception, of course, is a domain that is used to provide name service to other domains.

[4]To see why this is so, a simple example will illustrate. Suppose the operator of example.com wants to add a new name server (NS) record to the apex of the zone. The NS set also needs to appear in the parent at the other side of the zone cut in order to effect the delegation. The new name server should be inserted into the apex of the example.com zone first, because if the delegation point offers a name server that is not actually authoritative for the zone, resolution will not work as expected. But as a result, there is a short time of inconsistency necessary for this operation.

From the analysis above, the very things that make the DNS resilient also contribute to DNS operational problems. It is therefore worth outlining exactly how a transfer should happen.

B. The importance of timing

One of the reasons operational transfers are tricky is because of the various timing issues that affect DNS operation. In DNS operation without DNSSEC, there are two kinds of timing issue: those having to do with operating the authoritative DNS servers, and those having to do with the effects of caches (and therefore, the visibility of changed data). The parameters controlling these timings are discussed in detail in "DNSSEC Key Timing Considerations." [12]

Data in the DNS mostly travels in sets of resource records, or "RRsets."[5] There are two exceptions to this rule.

The first is in the transfer of zone data from one authoritative server to another using AXFR [9] and IXFR [14], where entire versions of a zone (as distinguishable by the SOA [10] serial number) are transferred in a single operation. While these protocols do indeed work by moving individual RRsets [16] between the source server (the master) and the client requesting the transfer (the slave), all of the RRsets become visible in the slave at the same time. The protocols do not allow for partial transfer of zone data.

The second is where DNSSEC is involved. Under DNSSEC, the RRSIG [3] for the RRset travels with the RRset. So in one sense, it behaves like any other member of the RRset. Caches hold the value of the RRset for the time defined by the TTL field on the RRset.[6] But a cache might have originally asked for the RRset with DO=0 [4], which would mean that the RRSIG would not be in the cache with the rest of the RRset. In this sense, the RRSIG is separable from the rest of the RRset. This has implications when we turn our attention to performing operations transfers in the presence of DNSSEC.

C. Who must act when transfers are performed: no DNSSEC

To get a complete description of how to transfer DNS operation, we must first work out exactly who must perform each action. For the purposes of this description, let us imagine Adam is a registrant of example.tld. Adam has registered example.tld in a three-party registration model. He used RegistrarCo to be his registrar, which registered the name with the .tld registry. RegistrarCo therefore has a contractual relationship with Adam. Usually, RegistrarCo gives Adam a username and password that allows Adam to manage his domain name registration. This management includes adding and deleting the zone, and also managing the name servers associated with the zone in the registry.

When we begin, Adam has a contract with Bernice to run the DNS for example.tld. Bernice accepts changes to the DNS zone data from Adam and publishes them. Let us suppose that Adam is not sophisticated about the DNS, so changes to the DNS are performed mostly by his operator. Therefore, Adam either gets the NS set for example.tld from Bernice and gives that information to RegistrarCo, or else gives Bernice access to his account at RegistrarCo and allows her to manage the DNS portion of the name herself.

Adam decides to hire Charlie to run the DNS instead, and enters an agreement with Charlie. Adam also informs Bernice that Charlie has been authorized to get copies of the example.tld zone. Now Charlie gets a copy of the zone and starts keeping up to date with the zone data. Charlie's servers answer authoritatively for the zone, but are not in the apex NS set.

Now one of two things happens. Either Charlie asks Bernice to add his name servers to the apex NS set for the zone, or else Charlie logs in to Bernice's system and adds the NS records himself. In either case, Charlie's name servers become part of the apex NS set. No later than this point, change control over DNS data for the zone must no longer reside with Bernice.

Next, Charlie either contacts RegistrarCo on behalf of Adam (having been authorized to do so), or else Adam contacts RegistrarCo directly. Whoever does this adds Charlie's name servers to the name server set for example.tld. RegistrarCo contacts the TLD registry, and adds Charlie's name servers to the NS records for example.tld. At this point, Charlie's name servers are active on the Internet for example.tld, but so are Bernice's.

Next, Charlie contacts RegistrarCo on behalf of Adam, or Adam contacts RegistrarCo himself, and removes Bernice's name servers from the set for example.tld. RegistrarCo then updates the TLD registry.

Under the above series of steps, Bernice can actually stop responding to DNS requests shortly after Charlie's name servers make it into the TLD registry, because there is for the most part enough data in the system to cause queries to be directed towards Charlie if Bernice does not respond, or if she responds with a non-authoritative answer. (In fact, for some clients such action may result in a transient outage, but the outage may well be short enough that nobody notices.) Indeed, former DNS providers sometimes do not stop responding with authoritative answers in a timely way, and continue to provide such answers for some time after the customer has ended their relationship. This can be a problem because of the way some resolvers behave. See III-D.

Note that none of the above has acknowledged the possibility that the service that depends on the DNS has to move too – as, for instance, when Web service and DNS service are offered as part of a single "package deal," so that the Website Bernice was operating for Adam will stop working when the DNS moves, or the DNS Bernice was operating will stop working when the Website moves.

D. Child-centric and parent-centric resolvers

Another complication to the process of transferring DNS operation between operators is the variation in behavior among recursive resolvers widely deployed in the Internet. The variation relates to different strategies for renewing a record after

[5]This is sometimes spelled "RRSet", as in [6], which also clarified the notion of the RRset.

[6]Strictly, the TTL is defined by each resource record. [6] requires that all TTLs in an RRset be the same, and also requires any client to treat the TTL on an RRset as the lowest value of those TTLs.

its TTL has expired. Caching name servers have two strategies for getting data from the DNS. Recall that the NS RRset for a domain is present in two places: on the parent side of the zone cut, at the delegation point; and on the child side, at the apex of the zone in the authoritative server. When resolving names inside a zone, some resolvers prefer to use the first NS RRset they get for that zone, and always ask a name server in that RRset when resolving requests in the zone (for as long as the NS RRset remains in cache). Since the first name server that will be encountered is usually from the delegating name server – the parent – we call these resolvers "parent-centric," to distinguish them from the other cases. (We are unaware of any resolvers that actually follow this approach consistently.) Other resolvers, upon seeing the NS RRset that came from the authoritative servers for the domain – the child – will replace the NS set from the parent in their cache, and prefer the child-provided NS RRset. We call these resolvers "child-centric"; Unbound is an example of this strategy [13]. Note an important effect here: in the event the NS RRset is different at the child and parent, the different kinds of resolvers will decide to query different sets of name servers.

A subclass of child-centric resolvers are those that opportunistically extend the TTL of an RRset whenever they see the NS RRset for the zone in authoritative responses to other queries. In the case of high-volume recursive resolvers (such as resolvers at ISPs serving many customers), this "TTL stretching" might continue indefinitely, thereby preventing a TTL expiration indefinitely. We call such resolvers "sticky child-centric" resolvers. BIND 4 exhibits this behavior (but Unbound does not). This strategy works well in the usual case where the NS RRset is indeed the same and the TTL just needs to be refreshed, but it does not work well if the name server RRset has changed. In fact, if the parent is now pointing to another set of name servers but the old name servers are continuing to answer, the recursive resolver will never learn the name of the new name server. However, if the old name server stops answering queries, the child-centric resolver will eventually go back to the parent (albeit after a bit of pause before it resolves the query).

IV. Transferring Zone Operation in the Presence of DNSSEC

A. Why DNSSEC matters for the transfer

RRSIG records are generated with a particular set of keys, and those keys need to be available for validation. The keys also need to be available for RRSIG generation, and in that case it is the private key that is needed. Moreover, in order to validate the delegation from the parent, a DS [3] record in the parent needs to match one of the DNSKEY [3] records in the child. These two facts make DNS operator transfers in the presence of DNSSEC more delicate, because transfers that used to work in the face of a sloppy procedure will suddenly fail; either signatures will be provided that cannot be checked with any of the available keys, or else the apex DNSKEY RRset will not contain a key that corresponds to the DS record from the parent. In either case, the domain will validate as bogus and validating resolvers will stop processing the name.

In the pre-DNSSEC case, when conflicting data from different sources are received, resolvers can try to make the best of the situation and return whatever result seems most likely to allow resolution to happen. Under DNSSEC, this operational expedient is indistinguishable from an attack on the DNS messages, the messages validate as bogus, and the result is rejected. This suggests that the gaining and losing operator of the DNS for the target zone need to cooperate, but the losing operator has little incentive to undertake the expense of contributing to the solution when the only reward is a lost customer.

B. Why not go insecure?

One answer to the complications above is to say that the registrant should simply take the domain through an insecure step, removing the DS record from the parent for a time and thereby stopping validation at the parent in a provably insecure way. But that approach implies that the purpose of DNSSEC is only to secure the traditional uses of the DNS. We can already tell that this will not be the case, as efforts to leverage DNSSEC to build new security models for the online world are well under way. For instance, the IETF DANE working group is developing a mechanism to use DNSSEC to get certificates for use with TLS [5], but that protocol depends on DNSSEC [7]. A domain that relies on DANE will not be able to go through an insecure step without giving up TLS via DANE as well. If we believe that DNSSEC is a path to new security applications on the Internet, then “go insecure” is not good advice.

C. How DNSSEC affects the required behavior of the actors

In III-C, we outlined which actors must perform which steps in what order. An important feature in that series is that change control over the zone is required to move from one actor to another. But under DNSSEC, the new operator (Charlie in our example) also needs to provide signatures over the data in the zone.

We work under the assumption that private keys cannot (or at least in practice will not) be transferred from one operator to another. This means that the data in the zone operated by Bernice is signed by Bernice’s private key, and Charlie cannot get that key. So Charlie has to introduce his own key to the DNSKEY set. In effect, a key rollover must be performed during the transition period. Charlie adds his key to the DNSKEY set at the same time his name servers are added to the NS RRset. This ensures that the additions are signed in such a way that they are validatable at the moment they are added.

Once Bernice gives up change control on the zone, her DNSKEY record remains in the zone. In addition, a DS record in the parent corresponding to her key remains in the parent. Charlie now signs the DNSKEY RRset with his key, and adds a DS record corresponding to his key to the parent. At this point, no matter whether a querying client queries Charlie or Bernice, the answer is still validatable.

Once Charlie removes Bernice’s name servers from the NS RRset, however, things begin to change. As long as the DS

record in the parent corresponding to Bernice's key remains, then answers from Bernice will still validate. Once the DS record is removed, then answers from Bernice will be treated as bogus. This means that when the DS record corresponding to Bernice's key is removed at the parent, Bernice must stop answering queries for example.tld. Otherwise, child-centric resolvers will fail to use those answers (because they fail validation).

V. Conclusion: Next Steps

Having determined who each of the actors are and having provided a rough outline of what must happen, the next tasks are to outline in fine detail what each actor must do and when exactly he or she must do it, and to describe the minimal cooperation that must be expected from any DNS operator at handover time. That description can serve as a basis for future operator contracts that define the responsibilities of the various parties.

References

[1] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. DNS Security Introduction and Requirements. RFC 4033 (Proposed Standard), March 2005. Updated by RFC 6014.

[2] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Protocol Modifications for the DNS Security Extensions. RFC 4035 (Proposed Standard), March 2005. Updated by RFCs 4470, 6014.

[3] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Resource Records for the DNS Security Extensions. RFC 4034 (Proposed Standard), March 2005. Updated by RFCs 4470, 6014.

[4] D. Conrad. Indicating Resolver Support of DNSSEC. RFC 3225 (Proposed Standard), December 2001. Updated by RFCs 4033, 4034, 4035.

[5] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), August 2008. Updated by RFCs 5746, 5878.

[6] R. Elz and R. Bush. Clarifications to the DNS Specification. RFC 2181 (Proposed Standard), July 1997. Updated by RFCs 4035, 2535, 4343, 4033, 4034, 5452.

[7] P. Hoffman and J. Schlyter. Using secure dns to associate certificates with domain names for tls. Internet Draft, Work in Progress, March 2001.

[8] B. Laurie, G. Sisson, R. Arends, and D. Blacka. DNS Security (DNSSEC) Hashed Authenticated Denial of Existence. RFC 5155 (Proposed Standard), March 2008.

[9] E. Lewis and A. Hoenes. DNS Zone Transfer Protocol (AXFR). RFC 5936 (Proposed Standard), June 2010.

[10] P.V. Mockapetris. Domain names - concepts and facilities. RFC 1034 (Standard), November 1987. Updated by RFCs 1101, 1183, 1348, 1876, 1982, 2065, 2181, 2308, 2535, 4033, 4034, 4035, 4343, 4035, 4592, 5936.

[11] P.V. Mockapetris. Domain names - implementation and specification. RFC 1035 (Standard), November 1987. Updated by RFCs 1101, 1183, 1348, 1876, 1982, 1995, 1996, 2065, 2136, 2181, 2137, 2308, 2535, 2845, 3425, 3658, 4033, 4034, 4035, 4343, 5936, 5966.

[12] S. Morris, J. Ihren, and J. Dickinson. Dnssec key timing considerations. draft-morris-dnsop-dnssec-key-timing-02.txt, March 2010: work in progress.

[13] NLnet Labs. Unbound.

[14] M. Ohta. Incremental Zone Transfer in DNS. RFC 1995 (Proposed Standard), August 1996.

[15] P. Vixie. Extension Mechanisms for DNS (EDNS0). RFC 2671 (Proposed Standard), August 1999.

[16] P. Vixie, S. Thomson, Y. Rekhter, and J. Bound. Dynamic Updates in the Domain Name System (DNS UPDATE). RFC 2136 (Proposed Standard), April 1997. Updated by RFCs 3007, 4035, 4033, 4034.

This is an informal glossary intended to convey the sense of some of the important terms used to a reader unfamiliar with the DNS. For more rigorous definitions, the reader is urged to consult the RFCs mentioned in the text.

DNSKEY	A resource record type containing the key used to generate DNSSEC data for the named zone. DNSKEY records are authoritative data only on the child side of a zone cut.
DO	An EDNS0 header bit that signals a DNS resolver's ability to understand DNSSEC.
DS	A resource record type that corresponds to a DNSKEY of the same name on the child side of the zone cut. It is authoritative data only on the parent side.
EDNS0	A mechanism to permit DNS messages larger than the original protocol permitted. Defined in RFC 2671 [15].
Glue	Resource records (A or AAAA) that exist in a zone just to permit resolution. Glue is necessary to start chains of resolution when the name servers for a node are below that nodename.
NS record	A resource record that specifies the name server for the named node. NS records exist at both the parent and child side of the zone cut.
RRset	A group of resource records at the same node with the same RRTYPE and CLASS. Sometimes spelled RRSet.
RRSIG	An RR that delivers a cryptographic signature over an RRset.