Open Issues in Secure Domain Name System Deployment

The Domain Name System's growth has been unprecedented, but protocol vulnerabilities threaten its stability and trustworthiness. The Internet Engineering Task Force's DNS Security Extensions specification aims to protect the system from these attacks.

Ramaswamy Chandramouli and Scott Rose
US National Institute of Standards and Technology

The Domain Name System is the Internet's primary infrastructure component. The DNS translates easy-to-remember Internet destination addresses (URLs) into Internet Protocol addresses. This translation data is distributed hierarchically in domains and housed in authoritative name servers. The topmost level of the hierarchy is the root domain, and the corresponding servers are root servers. The next lower level is the top-level domain (TLD). There's only one root domain, but more than 250 TLDs, which are classified as generic (gTLD)—.com, .org, .edu, and so on—or country code (ccTLD)—.uk, .br, and so on. The third level houses data about an organization's Internet resources in enterprise-level servers and the associated enterprise-level domains, or *zones*, which are the administrative units that manage data pertaining to a portion of the DNS tree (formed by members of a hierarchical chain).

The DNS's growth has been unprecedented, but so have the breadth and sophistication of the abuses and attacks on it. This growth, together with the fact that the Internet has become the foundational technology for information processing, communication, and sharing in a globally interdependent economy, make the DNS a critical universal infrastructure whose protection must be ensured through state-of-the-practice security measures.

DNS Transaction Types

Choosing the protection type and associated security measures requires analyzing past and potential threats to DNS. Before we analyze the threats and how attackers could exploit them, we must look at the overall set of DNS transactions, which belong to two major classes:

- DNS usage transactions (also called DNS query/response) and
- DNS administrative transactions.

DNS query/response transactions give the IP addresses for queried URLs (more precisely, for Internet resources specified using a Fully Qualified Domain Name [FQDN], which is extracted from the URL), and vice versa. As such, they serve the primary purpose behind the DNS's design and make up the majority of DNS protocol traffic. A DNS server stores and serves responses to a DNS query as a set of *resource records*. The most common resource record types in DNS responses are:

- *IPv4 address* (A) and *IPv6 address* (AAAA), which contain the mapping of a URL (the FQDN) to its IP address information, and
- *name server* (NS), which identify the authoritative name server for a zone pertaining to the URL.

DNS administrative transactions are internal to the DNS infrastructure. Zone administrators (or automated maintenance processes) perform these transactions to, for example, keep the data in a set of redundant authoritative name (or secondary) servers in synch with the primary authoritative name servers (through zone transfer). They also use them to dynamically update the URL name to IP address mapping records when an

organization dynamically allocates an IP address from its assigned block to an authorized requesting device. Not surprisingly, DNS query/response transactions account for most DNS traffic. Hence, we focus on open issues in the implementation of security measures proposed for countering threats to these transactions.

DNS Query/Response Transactions

A typical DNS query for a resource such as a Web page (say, www.itl.nist.gov) originates from a DNS client called a *stub resolver*. This query lands on the query originator's local name server (the *resolving name server*). In the basic DNS process, the resolver process resolves this query by processing each segment of the queried URL from right to left. The rightmost segment (implicit in every URL) is the root zone; hence, the local name server will contact one of the 13 root name servers. The root server refers to the domain one level lower in the DNS hierarchy, which is in next rightmost URL segment (.gov in our case), a TLD zone. The referral consists of the name server for the .gov zone—that is, the response consists of one or more name server–type resource records, or *delegation* information. The .gov authoritative name server provides the delegation information for the nist.gov authoritative name server, which in turn provides delegation information for the itl.nist.gov zone. This zone provides the necessary URL to the IP address mapping record (through an A or AAAA resource record). However, in many cases, either the local name server or any server in the hierarchy can provide the IP address information for the queried URL from its cache. Servers that can construct responses using previously learned DNS information are called *caching name servers*. Hence, the source for a response to a DNS query can be the authoritative name server for the URL data in the query or any caching name server. *Name resolution* is the process of providing the IP address for a given URL

As our description of the DNS response-generation process indicates, a DNS server provides two categories of information. First, it provides the information needed to map an Internet resource name (an FQDN) to the IP address. Second, it provides the name of the authoritative name server that either contains the above mapping information or a referral to the name server (the delegation information) for a zone lower in the hierarchy that will eventually provide an FQDN to an IP address mapping.

Threats and Countermeasures to DNS Query/Response Threats

US National Institute for Standards and Technology Special Publication SP 800-81[1] and Internet Engineering Task Force (IETF) RFC 3833[2] identify three main threats to DNS query/response transaction:

- compromise of the authoritative name server for the query's URL data (the target name server) through platform-level and distributed denial-of-service (DDoS) attacks;
- corruption of the cache of any name server (cache poisoning) that has cached the requisite resource record for the queried URL; and
- modification of information over the wire originating from the target name server or a caching name server (a man-in-the-middle attack).

Platform-level attacks are directed against one or more of the elements constituting the host platform for an authoritative name server, such as the operating system, file system, or communication stack. Solutions include secure configuration of operating system parameters, keeping patches up to date, adequate file-level protection through access-control mechanisms, and network-level protections.[1] In a DDoS, several coordinated DNS clients send a huge volume of DNS queries that overwhelm the DNS servers at various levels (root, TLD, and so on), denying legitimate users the name-resolution service. Solutions for DDoS attacks include:

- providing redundancy and fault tolerance by increasing the number of physical servers for a DNS zone;
- housing the servers for a zone in geographically dispersed locations;
- filtering unwanted traffic in and out of the DNS servers; and
- using an anycast routing scheme, which uses a single network destination address to route a packet (containing the DNS query) to one of a set of recipients.[3]

Cache poisoning and man-in-the-middle attacks often result in forged or bogus DNS responses. The impact of this *DNS spoofing* is the redirection (or misdirection) of DNS users to spoofed Web pages, possibly resulting in misinformation or an illegitimate collection of user information such as online banking user IDs and passwords. Because these results threaten the DNS's integrity, the IETF has proposed a set of DNS Security Extensions.[4–6] DNSSEC's main goal is to counter DNS spoofing through authentication. It allows origin authentication and data integrity by offering additional resource records, such as encoded digital signatures, over the regular DNS response resource records.

The DNSSEC Specification

In DNSSEC, a DNSSEC-capable DNS server generates a pair of keys (a private signature key and a public signature verification key) based on a public- or asymmetric-key cryptographic algorithm. The server digitally signs the response resource records for a query (encoding the generated digital signature string in a signature resource record, or RRSIG) using its pri-

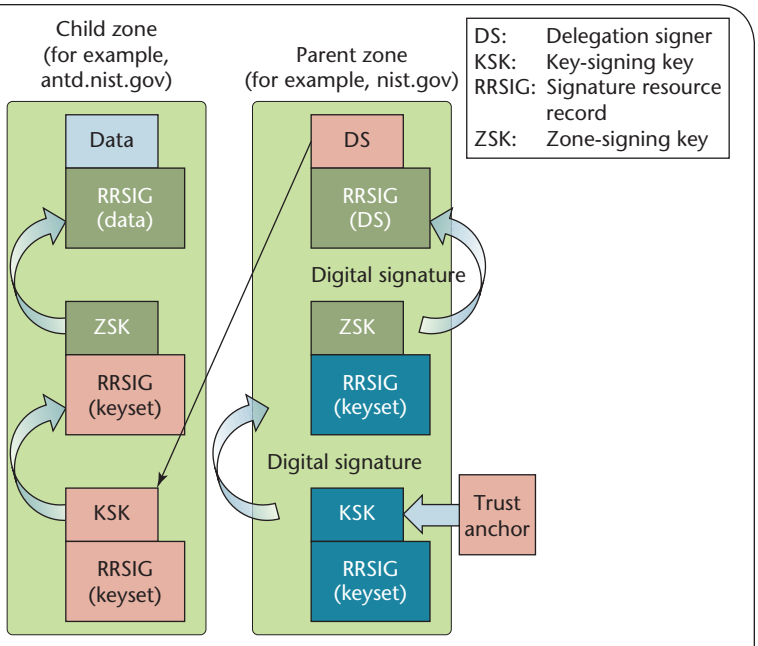

Figure 1. Example of an authentication chain. The arrows indicate which keys are used to validate each signature and show a chain from the trust anchor to a queried data record.

vate signature key. Next, it sends these RRSIGs along with the response resource records (which include encoding of the public signature verification key in a DNSKEY resource record) to the querying DNSSEC-capable DNS clients, or *validating resolvers*. The RRSIG also contains the code for the public-key algorithm, or digital signature algorithm, used to generate them. The validating resolvers then use the public signature verification key sent in the response, together with the algorithm information (found in the RRSIG) to verify the digital signature associated with the response resource records, thus confirming the sender's origin authenticity and verifying the data integrity.

But there's a small problem with the DNSSEC authentication scheme as described so far. The process's legitimacy depends on the validating resolver trusting the public signature verification key sent by the responding zone along with the signed DNS response resource records. A validating resolver has good reason not to trust the public signature verification key sent by a zone with which it has no previous relationship. It can have more trust if it obtains the public signature verification key from the responding zone through a secure, out-of-band protocol (that is, not through the DNS protocol) such as Transport Layer Security (TLS). Alternately, the higher the zone in the DNS hierarchy, the more useful is its public signature verification key for the validating resolver. For example, the validating resolver can use a TLD's public signature verification key to validate any zone under that TLD. Likewise, because the root zone is at the top of the DNS hierarchy, the validating resolver could use the root zone's public signature verification key to validate any zone.

DNSSEC operational experience calls for two pairs of digital signature keys for signing the entire zone contents.[7] The first pair, the *zone-signing key* (ZSK), is for signing the entire zone's resource records. The second pair, the *key-signing key* (KSK), is for signing only the set of resource records containing both key types. Because a key in a DNS zone data file is encoded in a DNSKEY resource record, the KSK effectively generates the RRSIG for the DNSKEY resource record set. Using two pairs of digital signature keys requires a slight expansion of the validation process. A validating resolver receiving signed response resource records from a zone uses the KSK to verify a ZSK authenticity by verifying the RRSIG associated with the DNSKEY resource record set. It then verifies the authenticity of the signed zone data received in the response using the authenticated ZSK. Because the KSK is the entry point into the zone for validation, it's sometimes called the *secure entry point* (SEP). (The currently used KSK DNSKEY resource record has a special bit called a SEP bit set.) Again, when using two pairs of keys, the signature verification process's legitimacy comes from the validating resolver trusting the KSK.

To validate the signed response resource records, the validating resolver must first trust the KSK. If the validating resolver explicitly trusts the SEP key, it becomes the *trust anchor* for the validating resolver. As mentioned earlier, the higher in the DNS hierarchy the KSK comes from, the greater its utility. To facilitate this, the DNSSEC outlines a scheme in which a DNSSEC-capable zone (also called a *signed* zone) can have its parent (one level higher in the hierarchy) vouch for its SEP key's authenticity. The scheme calls for a signed zone to export its SEP key to the parent. The parent encodes the key as a *delegation signer* resource record and digitally signs it using its ZSK private signature key. The parent in turn can export its SEP key to its parent, which can vouch for its authenticity by signing with its ZSK private signature key in a process called *secure delegation*. If this chain of SEP key export goes all the way to a trust anchor, the validating resolver can build an authentication or trust chain (see Figure 1) by following the secure delegation sequence. In this ideal situation (ideal because we assume all zones in all levels of the DNS hierarchy are signed and are willing to perform secure delegation for their children), it's enough that the validating resolver maintains one trust anchor key (that is, the KSK public portion). When a validating resolver receives a signed DNS response from a zone, it can establish trust in that zone's SEP key by starting with the SEP key of the root zone (its trust anchor) and following the delegation chain all the way to the responding zone. It can then use that trusted SEP key to verify the

signature of the DNS response resource records it has received from the responding zone.

However, in the real-world DNS infrastructure, not all zones are signed. The root zone has yet to be signed. Only a handful of zones at the TLD level and a few zones at the enterprise level are signed. In other words, only islands of trust exist, as opposed to chains of trust from the root zone to any given zone at a lower level in the hierarchy. In this practical scenario, the validating resolvers must perform the following security administration operations:

- *Resolver operation 1*: Build a list of trust anchors corresponding to the KSK of many signed zones.
- *Resolver operation 2*: Keep the list of trust anchors updated whenever the KSK pairs are changed in the signed zones.

These two operations occur at the validating resolver end. As a cryptography-based authentication scheme, DNSSEC involves several security administration operations at the signing zones at all levels of the hierarchy. To digitally sign its zone data, a zone requires the following security administration operations:

- *Zone operation 1*: Use state-of-the-practice secure public-key algorithms, hash algorithms, and appropriate key sizes.
- *Zone operation 2*: Schedule, make an emergency change, or roll over cryptographic keys because all cryptographic keys become inherently weak over time.
- *Zone operation 3*: Securely publish the zone's KSK.

In addition to these operations, which are specific to validating resolvers and authoritative name servers (that sign the zone data), DNNSEC requires a fourth security administration operation. Zone operation 4 involves protocol and firewall configuration changes to accommodate the signed response resource records' large size (as opposed to unsigned response resource records).

Zone Operations

The combined integrity of the four security administration operations (Zone operation 1 through Zone operation 4) ensures that DNSSEC's overall security goals are met. However, several open and unresolved issues exist with respect to these operations. A security administration operation has an open/unresolved issue if one of three situations exists. First, the DNSSEC specification pertaining to the operation needs updating to be consistent with current practice. Second, several secure solutions have been proposed but there's no agreement on a standard procedure. Finally, although the organization's IT staff have agreed on a secure procedure, they must address some DNS operational challenges before deploying it.

Table 1. Algorithm combinations and security strengths in the Domain Name System Security Extensions (DNSSEC) specification.

YEAR	SECURITY STRENGTH (IN BITS)	DIGITAL SIGNATURE ALGORITHM	HASH ALGORITHM
Present–2010	80	RSA 1024	SHA-1
2010–2030	112	RSA 2048	SHA-256
2030 onward	128	RSA 3072	SHA-256 or SHA-512

Use State-of-the-Practice Cryptographic Algorithms and Key Sizes

As described previously, a DNSSEC-capable DNS server sends RRSIGs along with response resource records. These records encode the digital signature over the data in the response resource records. To generate these digital signatures, DNSSEC uses two classes of cryptographic algorithms: the hash algorithm and the asymmetric- or public-key algorithm. The DNSSEC signing process uses the hash algorithm to compress the message (the resource record sent in the DNS response) for subsequent digital signature generation by the asymmetric key algorithm (or digital signature algorithm in this usage context). The security strength of cryptographic algorithms is expressed in terms of bits. Federal Information Processing Standards documents give the cryptographic algorithms for different services (encryption, digital signatures, and hashing[8]),and NIST special publication SP 800-57 offers guidance on selecting these algorithms and associated key sizes based on the algorithm strength.[9,10] To account for the increase in attackers' computational power with time, SP 800-57 also provides the minimally required algorithm bit strengths for various time periods into the future. DNSSEC specifies the SHA-*n* family of secure hash algorithms and RSA for digital signatures. Table 1 lists the combination of SHA-*n* and RSA algorithms with their bit strengths and recommended years of use.

As Table 1 shows, DNSSEC-capable DNS servers should sign their resource record sets only with RSA 2048 and SHA-256 from 2010 onward to meet the state-of-practice security strength of 112 bits. However, the IETF has specified SHA-1 as the hash algorithm in the DNSSEC specification. For DNS to provide state-of-practice security, the DNSSEC specification must be updated to specify the sunset dates for existing algorithms and migration dates for the new stronger cryptographic algorithms and key sizes for DNSSEC deployments. Further, we also see the need for new DNNSEC specifications that prescribe the use of more efficient signing algorithms (such as ECDSA) that provide the same cryptographic strengths as RSA versions with much smaller key sizes.

Change or Roll over Cryptographic Keys

To maintain their effectiveness, cryptographic algorithm keys are limited to a certain *cryptoperiod*. Some of the predominant factors determining a key's recommended cryptoperiod are its security function (for DNSSEC, this is a digital signature), the volume of zone data it protects, and the transaction frequency (number of times it's used to generate signatures for a given DNS zone in a given period). The last two factors are important for DNSSEC because digital signatures associated with a large DNS zone and many fresh signatures can provide enough data for cryptanalytic attackers to guess the private signature generation key and compromise the key system. Compromising this key lets the attacker introduce spurious resource records with seemingly valid digital signatures, fooling the DNS client into believing that they came from the legitimate DNS zone.

SP 800-57 calls for a 1- to 3-year cryptoperiod for private signature keys.[9] However, given that one of the determinant factors for a key's cryptoperiod is the volume of data and frequency of transactions, DNSSEC deployments typically choose the recommended cryptoperiod (1 to 2 years) for the KSK pair because it signs only the small-size DNSKEY resource record set containing the ZSK, which changes relatively infrequently compared to the DNS zone data. In addition, they typically choose a relatively shorter cryptoperiod (say 1 to 6 months) for ZSK because it signs a much larger volume of data (virtually all resource record sets), which changes much more frequently.

Key rollovers involving key size changes. Migrating to a larger key length is relatively easy. Because it doesn't require a new algorithm code, the zone administrator can move to a larger key using a normal key rollover procedure. Most cryptographic libraries and implementations are flexible and can handle most reasonable key sizes. Typically, users will never even know that the key size has changed unless they observe DNS responses. The new larger keys will simply appear as a normally scheduled rollover.

Key rollovers involving algorithm and key size changes. Migrating to a new cryptographic suite (either a new digital signature algorithm or new hash algorithm) requires more work. First, even if the new algorithm is implemented in cryptographic libraries, the IETF must develop standards to specify how it is to be used and codified in DNSSEC. Once this is done and the new algorithm is implemented in DNS software (DNSSEC signers, validators, servers, and so on), a zone administrator can start migrating to a new algorithm suite.

This algorithm rollover doesn't need to occur in conjunction with a scheduled rollover because no keys are retired until the end. The process only adds a new, parallel line of signing keys with its own rollover schedule, which need not match the existing keyset using the older, outgoing cryptographic suite. In fact, a staggered rollover might help avoid large responses, especially responses for the DNSKEY resource record set for the zone because the zone contains fewer prepublished keys.

A zone administrator has no explicit way to signal the migration except with the presence of a new DNSKEY resource record in the zone keyset. Likewise, a client and server have no way to negotiate which algorithms to use for DNSSEC signatures. A zone administrator who wishes to provide signed responses for a client who doesn't understand the newly deployed cryptographic suite must maintain both the old and new cryptographic suite deployments for some period of time.

An issue arises when deploying a new cryptographic suite with a signed zone. The zone administrators must now maintain two different key life cycles. A problem can arise when the administrator wishes to prepublish new ZSKs or KSKs in a zone that's signed by two or more cryptographic suites.

One common rollover regime is to prepublish the next ZSK at the same time as the state-of-the-practice ZSK. So, there would normally be three DNSKEY resource records in a zone's keyset: one active KSK, one active ZSK, and one prepublished ZSK. If a zone administrator uses two cryptographic suites, there would be twice that number, or six DNSKEY resource records in the zone's keyset. Once every 1 or 2 years, a zone administrator should roll over the KSK, so there will be a period during which a prepublished KSK will also appear in the zone for a total of four DNSKEYs for each cryptographic suite in use (eight if two suites are used). In an experiment, a full response containing eight 2,048 bit keys (two active RSA/SHA-1 keys [ZSK and KSK], two prepublished keys [ZSK and KSK], and the same for RSA/SHA-256 keys) the size was 6 Kbytes—well beyond the highest suggested UDP packet size advertised using extension mechanisms for DNS (EDNS(0)).[11]

To minimize the number of prepublished keys in the zone, a zone administrator could stagger key rollovers**.** One way to stagger rollovers is to extend each ZSK's active life time to two months, then alternate the rollover of either cryptographic suite's ZSK each month, as Figure 2 shows. Extending the active key life time to two months from the more traditional one month is considered a larger security risk, because the key is in use longer (and therefore subject to a larger window of attack), but the risk is still not severe enough for most zones.

Zone administrators can handle KSK rollovers the same way. However, because KSKs are rolled over less frequently, staggering KSK rollovers doesn't significantly increase zone size compared to ZSK rollover schedules.

Securely Publish KSK Public Signature Keys

Validating resolvers can build a list of trust anchors (trusted KSKs) in its configuration file only if there exists a secure mechanism that lets signed zones publish their KSKs. Publication of KSKs not only lets validating resolvers configure their trust anchor lists but also lets some parent zones perform secure delegation for their child zones by encoding those keys into delegation signer records and signing them.

An in-band mechanism for publishing KSK SEP keys would use the DNS infrastructure to publish those SEPs, whereas an out-of-band mechanism must rely on other protocols in the networking infrastructure. Currently, no in-band mechanism exists. The most common out-of-band mechanism used by some signed zones is to publish their SEP keys through an SSL-secured Web site. This meets the security requirement for these KSKs because they're public and only their origin authenticity and integrity are at issue. Another out-of-band mechanism is the trust anchor repository (TAR), which contains the SEP keys for all signed zones.[12]

Modify Protocol and Network Configurations

In DNSSEC, digital signatures are returned in a response only if requested. The number of signatures returned depends on the query, but DNSSEC specifies that one signature is generated for every resource record set in a DNS zone. As mentioned earlier, these digital signatures are encoded in RRSIGs, which contain the encoded signature as well as additional information about the resource record set they're supposed to cover.

As Table 2 shows, after 2010 with the increase of RSA key lengths to 2,048 bits, each RRSIG in a DNS reply will grow by 128 bytes. By 2030, when RSA keys should be at least 3,072 bits long, each RRSIG will likely grow an additional 128 bytes.

The only issue with larger keys is that new, possibly larger signatures might result in responses that are too large to fit into a UDP packet. If this happens, the DNS server sends a truncated response and the client resends the query using TCP. Zone administrators can conduct a test to see if any resource record set in the zone (with its signature) could cause a response to exceed the size of a UDP packet. When using EDNS(0),[11] which is required for DNSSEC implementations, this won't be a common problem, but might become an issue for zones with large resource record sets and can't restructure their zone contents to reduce the size of these sets.

Resolver Operations

In the context of zone operation 3 (securely publish KSK public signature keys), the following two validating resolver operations become corollary operations.

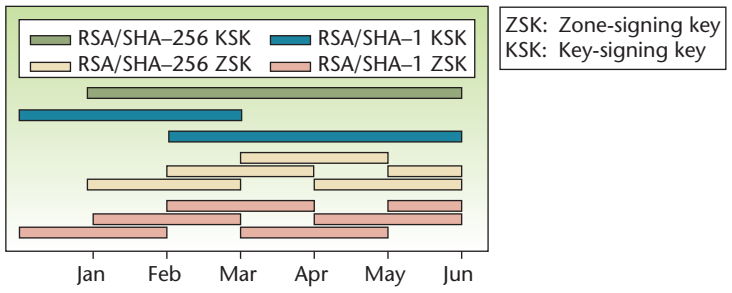

Figure 2. Key life cycles with two algorithm lines. To stagger key rollover, the zone administrator rolls over a different zone-signing key (ZSK) cryptographic suite each month.

Table 2. Key size and RRSIG size by time period.

YEAR	RSA KEY LENGTH (IN BITS)	RRSIG SIZE (IN BYTES)
Present–2010	1,024	215
2010–2030	2,048	343
2030 onward	3,072	471

Securely Build Trust Anchor Lists at Validating Resolvers

Validating resolvers must ensure that the process they use to build trust anchor lists is secure. No standardized secure procedure exists for this, although some proposals exist, including:

- manually obtaining trust anchors from SSL-secured Web sites maintained by signing zones, and
- securely downloading from TARs.

In both of these proposals, the validating resolver needs a home-grown procedure to keep its trust anchors up to date whenever the KSK changes at the signing zones or is updated at the SSL-secured Web sites or at the TARs. In addition, these repositories might not provide secure delegation information (in the form of delegation signer resource records). Hence, the validating resolver must build a chain of trust to the responding zone using alternative trust delegation rather the secure delegation available within DNS.

Update Trust Anchor Lists at Validating Resolvers

Although the IETF doesn't have a guideline for secure publication of SEP keys, IETF RFC 5011 describes how to automatically update trust anchors for the validating resolvers once an initial trust anchor list

is configured.[13] Implementing the scheme articulated in RFC 5011 requires reading the trust anchor lists and looking for a new KSK that's signed with an existing trust anchor key in the signed zone. If such a key exists, the resolver starts accepting the new KSK as a trust anchor after a designated wait period—for example, 30 days. The resolvers also requires a certain lead time under this scheme to drop the old KSK in the signed zone. Because of these two lag times, this scheme isn't useful in situations involving multiple emergency rollovers of signing keys, but only for scheduled key rollovers.

To eliminate the problem of validating resolvers managing multiple trust anchors (that is, building and updating the list), IETF proposed DNSSEC look-aside validation (DLV).[14] Under the DLV scheme, DLV domains are associated with zones in the DNS hierarchy (which serve as target zones for the DLV zone). For example, the DLV domain "trustbroker.example.com" could target the .org zone. In this scheme, it's enough if the validating resolver maintains the trust anchors for a limited set of DLV domains (depending upon the range of zones it accesses). Because a validating resolver obtains trust anchors dynamically, it doesn't need a process to keep its trust anchors up to date. Further, it can build the trust chain using the DNS secure delegation chain because the DLV domain also carries the delegation signer resource records. The scheme's drawback is that it can increase the time and number of queries needed to construct an authentication chain. Likewise, a validating resolver might not know which DLV zone is a given zone's correct authority. The issues we've discussed with respect to building and maintaining trust anchors will take on a new dynamic when the DNS root server is signed and some secure delegations follow suit lower in the DNS hierarchy.

The DNS community has accepted the DNSSEC specifications and underlying mechanisms as having the potential to protect query/response transactions, the core DNS transactions. Although the demonstrated functionality of some pilot DNSSEC-capable zones supports the specifications' use, DNS-wide deployment of DNSSEC calls for certain security administration operations. Best practices, guidelines, and proposals exist for addressing the specification gaps for these operations. To realize ubiquitous DNSSEC deployment, however, the DNS governing bodies and other stakeholders must reach consensus on technical and strategic directions for the DNSSEC security administration operations. □

References

1. R. Chandramouli and S. Rose, *Secure Domain Name System (DNS) Deployment*, US Nat'l Inst. Of Standards and Technology Special Publication (SP 800-81); http://csrc.nist.gov/publications/nistpubs/800-81/SP800-81.pdf.
2. D. Atkins and R. Austein, *Threat Analysis of the Domain Name System (DNS)*, IETF RFC 3833, Aug. 2004; www.ietf.org/rfc/rfc3833.txt.
3. G. Lawton, "Stronger Domain Name System Thwarts Root-Server Attacks," *Computer*, vol. 40, no. 5, May 2007, pp. 14–17.
4. R. Arends et al., *DNS Security Introduction and Requirements*, IETF RFC 4033, Mar. 2005; www.ietf.org/rfc/rfc4033.txt.
5. R. Arends et al., *Resource Records for the DNS Security Extensions*, IETF RFC 4034, Mar. 2005; www.ietf.org/rfc/rfc4034.txt.
6. R. Arends et al., *Protocol Modifications for the DNS Security Extensions*, IETF RFC 4035, Mar. 2005; www.ietf.org/rfc/rfc4035.txt.
7. O. Kolkman et al., *DNSSEC Operational Practices*, IETF RFC 4641, Sept. 2006, www.ietf.org/rfc/rfc4641.txt.
8. *Federal Information Processing Standard 180-3, Secure Hash Standard (SHS)*, FIPS, Oct 2008.
9. E. Barker et al., *Recommendations for Key Management Part 1: General*, NIST Special Publication 800-57, Part 1, Mar. 2007: http://csrc.nist.gov/publications/nistpubs/800-57/SP800-57-Part1-revised2_Mar08-2007.pdf.
10. E. Barker et al., *Recommendations for Key Management Part 3: Application- Specific Key Management Guidance*, NIST Special Publication 800-57, Part 3, Oct. 2008; http://csrc.nist.gov/publications/drafts/800-57-part3/Draft_SP800-57-Part3_Recommendationsforkey-management.pdf.
11. P. Vixie, *Extension Mechanisms for DNS (EDNS0)*, IETF RFC 2671, Aug. 1999; www.ietf.org/rfc/rfc2671.txt.
12. "Trust Anchor Repositories—Statement of Needed Internet Capability," SPARTA, Shinkuro, and NIST, June 2008; www.dnssec-deployment.org/tar/tarpaper.pdf.
13. M. St. Johns, *Automated Updates of DNS Security (DNSSEC) Trust Anchors*, IETF RFC 5011, Sept. 2007; www.ietf.org/rfc/rfc5011.txt.
14. S. Weiler, *DNSSEC Lookaside Validation (DLV)*, IETF RFC 5074 (informational), Nov. 2007; www.ietf.org/rfc/rfc5074.txt.

Ramaswamy Chandramouli *is a supervisory computer scientist at the US National Institute of Standards and Technology. His research interests include role-based access control and model-based test generation. Chandramouli has a PhD in information technology from George Mason University. Contact him at mouli@nist.gov.*

Scott Rose *is a computer scientist at the US National Institute of Standards and Technology. His research interests include core Internet protocols and Internet infrastructure protection. Rose has a MS in computer science from The University of Maryland, Baltimore County. Contact him at scottr@nist.gov.*