Public Key Validation for the DNS Security Extensions *

Daniel Massey
USC/ISI
masseyd@isi.edu

Ed Lewis
Network Associates, Inc.
lewis@tislabs.com

Olafur Gudmundsson
Ogud.com
ogud@ogud.com

Russ Mundy
Network Associates, Inc.
mundy@tislabs.com

Allison Mankin
USC/ISI
mankin@isi.edu

Abstract

The deployment of DNS Security (DNSSEC) can only succeed if there is an effective mechanism for DNS public key validation. This paper compares three potential approaches to DNS key validation. A tree based approach utilizes the existing structure of the DNS tree to form highly structured key signing rules. This makes following chains of trust simple, but it allows no flexibility for individual zones and makes incremental deployment impossible. A pure web of trust based approach imposes no structure what so ever on the key signing process. This lack of structure provides a high degree of local control, but also makes it difficult to find trusted chains or specify security policies. The third approach is a new proposal based on a the concept of a fault-tolerant mesh of trust. The mesh approach utilizes some structured elements from the tree-based approach while maintaining the local flexibility found in the web of trust. Our analysis shows the hybrid mesh approach has the best chance of succeeding in the Internet.

1 Introduction

In this paper we consider the problem of public key validation in the Domain Name System (DNS)[6, 7]. The DNS is a fundamental part of the Internet infrastructure. Despite this fundamental importance, the DNS includes very little security and is vulnerable to both faults and attacks.[2, 8]. The DNS Security Extensions (DNSSEC), proposed in [3], use public key authentication to reduce several of the vulnerabilities. Every system that uses public key authentica-

tion requires some mechanism for learning and validating public keys. This paper compares three approaches that could be used to learn and validate the public keys associated with DNS zones.[1]

The public key validation problem is particularly problematic for the DNS. Many existing public key authentication systems assume that some external database or some universally trusted certification authority can validate public keys. When applied to the DNS, this approach creates a sort of chicken and egg problem. The role of the DNS is to provide fundamental information about the Internet infrastructure and any DNS public keys would become a fundamental part of the infrastructure. An external public key database or certification authority would either recreate the functionality of the DNS or would itself rely on the current DNS for other fundamental information. Because of its role as the fundamental information provider, the framework for validating DNS public keys must be contained within the DNS structure.

Before designing a system for validating DNS public keys, it is important to understand how these keys will be used and what structures are already defined within the DNS. Section 2 provides background material on the DNS and the DNS Security Extensions. Section 3 describes our fundamental assumptions about DNS key validation and presents a common framework for describing key validation approaches.

Sections 4, 5, and 6 present the three different approaches for validating DNS public keys. Section 4 presents a tree-based model used for validating DNS public keys. This is the approach currently proposed in the DNSSEC documentation[3]. We will show it has a number of potentially fatal disadvantages that could prevent it from being

---

[1]DNSSEC actually defines several types of public keys, including zone keys, host keys, and user keys. This paper is only concerned with the process for validating DNS zone keys. Typically, the other key types can be treated simply as DNS data and can be easily validated once the DNS zone keys are known.
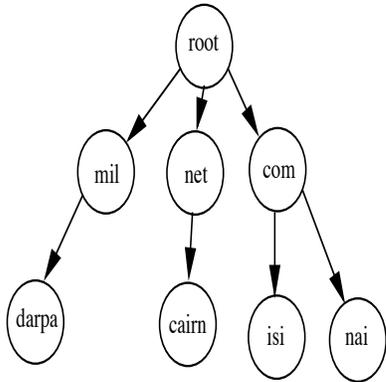
Figure 1. The DNS Tree Structure

deployed in the Internet. Section 5 considers how a web of trust approach, similar to the approach used by PGP[1], might be applied to DNS. The web of trust model eliminates many deployment problems faced by the tree-based model, but its lack of structure makes finding legitimate chains of trust difficult. The third model, presented in Section 6, is a new approach that attempts to combine some of the structure from the a tree-based approach with the flexibility of a web of trust. The result is a mesh of trust which could be incrementally deployed in the Internet.

2 DNS and DNSSEC

The Domain Name System (DNS) is a distributed database used to map host names to IP addresses, map email addresses to mail servers, and store a wide range of other infrastructure data. Virtually every Internet application relies on some form of DNS data and current estimates place the size of DNS namespace at nearly 1G of names in 40M zones.

The DNS namespace is divided into a series of zones. One or more **nameservers** store the **authoritative** data for a particular zone. A zone may also delegate authority for portions of its namespace and these delegations result in the DNS tree structure shown in Figure 1. At the root of the DNS tree, the root nameservers store the authoritative data for the root zone. The root has delegated authority for .com, .mil, .net, and so forth. The zones .com, .mil, and, .net operate their own nameservers and have further delegated authority for nai.com, darpa.mil, and cairn.net (respectively). These zones operate their own nameservers and can further delegate their namespace.

DNS **glue records** provide hints on how to move from a parent zone to a child zone. The child zone provides the parent with a list of the child's nameservers and the parent stores this glue data. By using the glue records, a **resolver** can start at the root nameserver and recurse down the DNS

tree to obtain any data.

For example, suppose a resolver wants to obtain the IP address (in DNS terms, the "A record") for www.cairn.net. Assume the resolver is configured with a list of root nameservers. The resolver first queries one of the root nameservers and receives the glue records listing the .net nameservers. Once this list of .net nameservers is known, the resolver queries one of the .net nameservers and receives the glue records listing the cairn.net nameservers. Once the cairn.net nameservers are known, the resolver queries one of the cairn.net nameservers and receives the desired www.cairn.net A record.

In practice, DNS resolution is slightly more complicated. An end host is typically configured with the IP addresses of one or more local nameservers. The initial query is sent from the end host to the local nameserver. The local nameserver may already have the data needed to answer the query, but often the local nameserver will not have the necessary data. The local nameserver then becomes the resolver and looks up the DNS data on behalf of the end host (as described in the example above). Once the local nameserver obtains the data, it returns the result to the end host and also stores the data in its cache. If another host requests the same information, the nameserver can reply using the cached data instead of having to repeat the look-up process. As a result of caching, most nameservers will have both authoritative data (for the nameserver's zones) and a changing set of (non-authoritative) cached data.

2.1 DNS Fault Tolerance and Security

The DNS was designed to be robust against **fail-stop** faults. In other words, DNS assumes that some nameservers may fail and stop operating. To counter fail-stop faults, each DNS zone uses several nameservers. If a nameserver fails, data can still be retrieved from any of the zone's remaining nameservers. To protect against fail-stop faults caused by link failures or network partitions, the zone's nameservers should be deployed in disjoint parts of the network. The use of multiple nameservers for each zone helps ensure that the loss of a small number of nameservers will not break the DNS.

For example, the cairn.net zone uses nameservers at lila.east.isi.edu, flag.ep.net, or ns.isi.edu. The cairn.net zone data is administered and stored on the **primary** nameserver, lila.east.isi.edu. The **secondary** nameservers, flag.ep.net and ns.isi.edu, periodically poll the primary nameserver and update their copy of the cairn.net data accordingly. Assuming the secondary nameservers maintain an up to date copy of the zone, a resolver can obtain authoritative data from the cairn.net zone as long as one of the nameservers is available. Note also that the cairn.net nameservers are located in disjoint parts of the network in order to protect against link

failures and network partitions.[2]

While fail-stop faults are common in the Internet, other types of faults can also occur. A nameserver might cache invalid data or fail to properly expire cached data. Some event might corrupt the DNS data stored in a nameserver. In general, a nameserver might simply behave incorrectly. An attacker might intentionally cause problems, either by gaining access to a nameserver or sending messages that appear to come from a valid nameserver. The existing DNS was not designed to address these types of faults and attacks.

For example, a query for the IP address of www.cairn.net should return an A record (recall an A record stores an IP address). This A record may come directly from one of the cairn.net nameservers. However, the A record may also come from some faulty nameserver's cache or may even come from an attacker who is imitating a nameserver along the path to www.cairn.net. At various points in the process, the A record might have been corrupted or intentionally modified. But with the current DNS, the resolver has no choice other than to simply believe whatever A record is returned in the response.

2.2 Security Extensions (DNSSEC)

The DNS Security Extensions[3] (DNSSEC) respond to more complex faults and attacks by adding public key authentication to the DNS. With DNSSEC, a zone creates a public/private key pair. The zone uses the private key to sign its authoritative data and makes the public key available to all resolvers. A DNSSEC query will return both the requested DNS data and a digital signature (SIG record). A resolver that knows the zone's public key can verify the signature and authenticate the DNS response.

For example, the cairn.net administrators would create a public/private key pair and use the private key to create digital signatures (SIG records) for all data in the cairn.net zone. A query for the IP address of www.cairn.net will return both the A record and a SIG record, containing the corresponding digital signature. If the resolver trusted the cairn.net public key, the resolver could verify the digital signature and assert that the A record was created by someone with access to the private cairn.net zone key.

Note the example failed to mention how the resolver obtained the cairn.net public key. DNSSEC defines a KEY record that can be used to store a zone's public key and the resolver could simply query the cairn.net zone for this KEY record. However, DNSSEC introduced digital signatures precisely because the resolver does not trust a DNS response. Our problem is how to determine whether the

KEY record itself is correct. Sections 4, 5, and 6 present differing approaches for obtaining and validating the KEY record.

2.2.1 Threats and Counter-Measures

Before proceeding with our solutions to the public key validation problem, it is useful to review some of the threats faced by the DNS and the counter-measures DNSSEC public keys might provide. Toward this end, we conclude this section by briefly reviewing what counter-measures DNSSEC public key authentication would (and would not) provide.

Assuming an effective public key validation approach is available, DNSSEC would respond to the following type of threats:

- **Forged DNS Responses:**

  A DNS query is identified by a 16 bit query identification number. A response with a matching query id number will be accepted as a valid answer. To forge a response, an attacker might guess the query id number, might observe the query, or might have obtained control of an nameserver involved in the response. DNSSEC would prevent forged responses. The data in DNS response would need to be signed and the attacker could not generate these signatures without first compromising the zone's private key.

- **Cache Poisoning:**

  Instead of forging individual replies, an attacker can achieve a greater benefit by poisoning the cache of a valid (non-compromised) nameserver. There are a number of techniques an attacker might use to poison a nameserver's cache. Some unintentional faults, such as failing to properly expire data, can also result in cache poisoning. DNSSEC helps to prevent these problems since the cached data must now have a valid signature and the SIG contains an unalterable lifetime. Both the data and the corresponding signatures need to be present in order to poison the cache.

- **General Data Authentication:**

  In general, the addition of authentication gives resolvers and nameservers a better chance to detect faults and attacks. Without DNSSEC, it can be very difficult to trace the true origin of invalid data. A digital signature ties invalid data to the holder of a particular private key. Even if a private key has been compromised, the key validation approach may help identify the point at which the fault or attack originated. Authentication adds the potential for a number of new counter-measures that previously would have been extremely difficult, if not impossible. In addition, authentication

---

[2]While the names lila.east.isi.edu and ns.isi.edu may sound topologically similar, lila.east.isi.edu is located near Washington D.C. and ns.isi.edu is located near Los Angeles. Different ISPs provide connectivity to the two locations.

enables a number of new DNS applications, such as using the DNS to store Secure Shell[10] host keys or using the DNS a type of public key infrastructure.

But DNSSEC is not intended as the counter-measure for every DNS threat. It is important to note that DNSSEC does not address the following threats:

- **Data Mis-configuration:**

  Authenticated data is not equivalent to correct data. Verifying a signature only proves that the data was produced by someone who had access to the corresponding private key. An administrator can still create incorrect data, perhaps by making a typo in a name or address. In addition, an attacker who has compromised a private key can sign whatever she chooses. Finally, the authentication process assumes the underlying public key is valid, but by convincing a resolver to trust a forged public key, the attacker can create what appears to be valid signed data. A signature does not prove the corresponding data is correct.

- **Denial of Service Attacks:**

  In general, DNSSEC only helps a resolver after a response has been received. DNSSEC only prevents one type of denial of service attack and DNSSEC may actually increase the vulnerability to most other denial service attacks. Without DNSSEC, an attacker might try to deny service by falsely reporting that a particular record does not exist. DNSSEC prevents this attack since even a report of non-existence must be signed.[3] But in all other cases, the added overhead of DNSSEC may actually increases the vulnerability to denial of service.

  If all the nameservers for zone are made unavailable, then a resolver will be unable to retrieve any information from the zone. In addition, if the glue stored at any parent zone or the parent zone itself is unavailable, a resolver will again be denied service. For example, a successful distributed denial of server attack against the root nameservers would completely cripple the entire DNS. DNSSEC does not prevent this type of attack and, by adding computational intensive authentication to the DNS process, DNSSEC may make it easier to overload a nameserver or introduce an unacceptable amount of delay.

  DNSSEC also adds a number of new denial of service attacks. A resolver should not accept any data with an invalid SIG. If a fault or attack causes a problem with the signature or the validation process, the result

is essentially a denial of service. For example, all the authoritative data for zone will be made unavailable if the administrator forgets to update SIG records when they expire. In another example, an attacker could add delay and consume resources by introducing purposely false SIG records.

The public key validation approach also plays a large part in determining how susceptible the DNS is to new denial of service attacks. If the resolver can be convinced to believe an invalid key, the resolver will not be able to verify legitimate data and will disregard the legitimate data. An attacker might also insert false information designed to delay the key validation process.

3 The DNS Key Validation Problem

The DNS key validation problem considers how a resolver might obtain and validate the public key(s) for a remote DNS zone. A resolver might use some out of band mechanism to obtain and validate a small set of initial public keys. It would be unreasonable to assume a resolver can use some out of band mechanism to learn and maintain a list of all the public keys for all DNS zones. In addition to large number of DNS zone keys, DNS zone keys periodically expire (one month has been recommended as a DNS key lifetime). Instead of using an out of band mechanism to obtain public keys, a viable solution to key validation problem must assume that a resolver starts with some small initial set of trusted keys and then uses this set to validate new public keys.

For example, assume a resolver has received a response that includes the www.cairn.net A record and a corresponding SIG record. If the resolver knows the cairn.net public key, then the resolver can verify the SIG and authenticate the response. However, it is very likely that the resolver has not been configured with the current cairn.net public key. The DNS key validation problem considers how the resolver can use its set of initial trusted keys to obtain and validate the public key for the cairn.net zone.

The three solutions to the DNS key validation problem will use very different approaches. Despite these differences, all of the approaches share some common assumptions and a common framework. This remainder of this section presents these assumptions and the common framework. The following sections then fill in the details and discuss the merits of each individual approach.

3.1 Data Validation

Regardless of the approach used for DNS key validation, the same rules will apply to DNS data validation. Before a DNS record is considered valid, it must be signed. The signatures are stored in SIG records. A SIG record indicates

---

[3]Currently, the NXT record is used to indicate there is no record between X and Y[3]. However, the NXT record is controversial and may be replaced by a NO record that relies on a hashing technique[4].

what data has been signed, provides a validity period for the signature, and name the public key needed to verify the signature. After receiving a signed response, the resolver first checks the SIG(s). The resolver confirms the SIG applies to the data found in the response, confirms the SIG record is valid for the current time, and checks to see which public key is needed to verify the SIG.

We will restrict which keys may be used to sign DNS data. Only a signature by the data's authoritative zone key should be considered valid. We specifically prohibit using a zone key to sign data from another zone. For example, only the darpa.mil zone key is authorized sign the A record for www.darpa.mil. Of course, we can not prevent yahoo.com or hacker.org from signing the www.darpa.mil A record. However, we can require that resolvers disregard these unauthorized SIGs. According to our rules, records in the darpa.mil zone can only be signed by the darpa.mil key. Similarly, the darpa.mil can not be used to sign data from any zone other than darpa.mil. If unauthorized signatures do appear, we assume the resolvers will simply ignore them.

Having created a rule, we immediately introduce two exceptions. First, the SIG records in a zone should not be signed. A SIG record contains the digital signature used to prove that other data is valid and it is not useful to sign a signature. Second, a KEY record[4] can be signed by a key from another zone. The KEY record contains the zone's public key and is part of a zone's authoritative data. However in order to build chains of trust, we allow the KEY record to be signed by another zone. For example, it may be valid for yahoo.com or hacker.org to sign the darpa.mil KEY record. Each key validation approach will define its own rules for who can sign a KEY record and the KEY records are always the only type of zone data that can be signed by another zone.

3.2 Key Chaining

Regardless of the approach used for DNS key validation, some form of **key chaining** must be used to validate public keys. The public key for a zone is stored in KEY record and is retrieved in the same manner as any other DNS data. Like any other DNS record, a KEY record must be signed before it can be validated. A KEY record may be considered valid if it is signed by some other trusted key. This approach of using a signature from key A to validate key B is known as

key chaining. Key chaining allows a resolver to begin with a small initial set of public keys and then use these initial keys to validate other public keys.

For example, a resolver might begin by trusting the public key for .mil. If the KEY record for .darpa.mil has been signed by the .mil private key, then the resolver can use the SIG from .mil to validate the darpa.mil KEY. If the KEY record for isi.edu has been signed by the darpa.mil key, the resolver can use the SIG from darpa.mil to validate the isi.edu KEY. This creates a chain of trust, starting a .mil and ending at isi.edu. The resolvers trusts the darpa.mil key because it is signed by .mil and the resolver trusts the isi.edu key because it is signed by darpa.mil.

Each solution to the key validation problem will use key chaining, but each solution establishes different rules for which chains of trust can be considered valid.

3.3 The Key Signing Mechanism

Key chaining assumes that the public key for zone A can be signed by private key of zone B. Our solutions to the key validation problem do not require a specific mechanism for exchanging and signing a public key. We only assume that some trusted out of band mechanism is available.

Note that the out of band mechanism used for exchanging and signing public keys will play an important role in the overall security of the DNS. Signing an invalid public key gives authority to the invalid key. Severe problems can occur if fault or attack results in the signing of invalid public keys. In the example above, if an attacker convinces .mil to sign a forged darpa.mil public key, then the attacker can use this forged key to sign invalid darpa.mil data. The attacker can also use the forged key to create and sign forged keys for other zones such as isi.edu.

While a precise mechanism for signing keys is not important, any approach to key validation should not impose an unreasonable burden on the key signing mechanism. Given the size of the the DNS, it is unreasonable to propose a key validation approach that assumes single zone could securely obtain and sign all the other public keys. In addition, the key validation approach should try to avoid creating single points of failure in the key signing process. In practice, we must expect that faults or attacks will result in some forged keys being signed and the key validation approach should be designed so that a single forgery will not break the entire DNS.

3.4 Components of a Key Validation Solution

A solution to the DNS key validation problem will identify two common components: the initial set of trusted keys and the key validation rules.

---

[4]A KEY record can actually hold a number of different types of keys. A zone key holds the public key for DNS zone. A host key might hold the Secure Shell key for a host. A user key might hold a key used in PGP. Other types of keys may be defined. A flag in the KEY record identifies the type of key in the KEY record. For ease of presentation, we assume a KEY record always holds a DNS zone key. To be technically precise, replace the term "KEY record" with the term "KEY record with the type flag set to zone key".

- **The Initial Set of Trusted Keys:** A solution must begin with some initial set of trusted keys. We will assume that some out of band mechanism is used to obtain the initial set of trusted keys. A solution must specify which public keys make up the initial trusted set. To have a reasonable chance of success, the set of initial trusted keys should be small and easy to update when a trusted key expires.

- **The Key Validation Rules:** A solution must define the rules for validating new public keys. These rules should specify the number and type of signatures needed to validate public key. Essentially, these rules define what constitutes a valid chain of trust.

In general, any solution to the key validation problem can be summarized by describing how the initial trusted keys are selected and how new public keys can be validated.

4 DNS Tree-Based Key Validation

The tree-based approach to key validation utilizes the existing DNS tree structure to create global key signing rules and define valid chains of trust. Every zone has a unique parent in the DNS tree and each zone will have its public key signed by its parent. For example, the darpa.mil public key will be signed by its parent, .mil. The .mil public key will be signed by its parent, the root zone. The root zone is the only zone with no parent and the root public key will be signed by a master key. All resolvers will be configured to trust the master public key.

Using the key validation framework, the tree-based approach can be summarized as follows:

- **The Initial Set of Trusted Keys:** All resolvers are initially configured to trust the master public key. Since changing the master key would require updating every resolver in t he Internet, this public key should change infrequently, if ever.

- **The Key Validation Rules:** A zone's public key is considered valid if and only if the key is signed by its parent and the parent's key is considered valid. This results in a chain of trust that starts from the master key and proceeds down the DNS tree. Any signatures that deviate from the DNS tree structure should be ignored by a resolver.

Note that the tree-based approach to key validation is the approach currently proposed in the DNSSEC specification[3, 9], but this approach has not been deployed in the production Internet.

4.1 Advantages of Tree-Based Validation

The advantages of tree-based validation come primarily from its well defined structure. This tree structure provides a universal key signing rule and allows resolvers to efficiently find chains of trust.

4.1.1 A Universal Key Signing Rule

The tree-based approach provides a universal rule for determining whether a signature is authorized. Only the parent is authorized to sign a zone's public key. Any other signature is unauthorized and should be ignored by a resolver. This universal rule allows a resolver to distinguish valid signatures from attempted faulty or intentionally misleading signatures. It also clearly defines the potential damage that could result if key is compromised.

For example, suppose an attacker has compromised the darpa.mil private key. The attacker can forge data in darpa.mil and can sign forged keys for any zone below darpa.mil. However, the attacker can not sign forged keys for zones in other parts of the tree, such as pentagon.mil or yahoo.com. The universal key signing rule clearly indicates which signatures can (and can not) be trusted.

Note also that the tree-based key signing rule does not introduce any new structures into the DNS hierarchy. In the current DNS tree, there is already a relationship between a zone and its parent. The tree-based approach simply adds a key signing component to this existing relationship.

4.1.2 Efficient Key Chaining

With the tree-based approach, a resolver can efficiently find a chain of trust. To obtain DNS data, the resolver starts at the root and moves down the tree. To validate DNS keys, the resolver also starts at the root and moves down the tree. There is no ambiguity about how to find a chain of trust and key validation can proceed in parallel with the data discovery process. This reduces the overhead and delay associated with validating keys.

4.2 Disadvantages of Tree-Based Validation

The disadvantages of tree-based validation also come from its structure. The rigid structure of the tree-based approach makes it difficult, if not impossible, to incrementally deploy DNSSEC in the general Internet. In addition to the deployment problems, the tree-based approach fails to achieve many theoretical goals. In particular, the tree-based model introduces a single point of failure, imposes undesirable trust relationships, and places a heavy burden on the external key signing mechanism.

4.2.1 Incremental Deployment

Perhaps the most fundamental flaw in the tree-based approach is that it dictates how DNSSEC must be deployed and makes it impossible to incrementally deploy DNSSEC. The tree-based approach requires a parent to deploy DNSSEC before any of its children can consider deploying DNSSEC. In the Internet, there is no central authority that can dictate when technology is deployed. Typically, some early adaptors experiment with the new technology and more sites are motivated to deploy the technology when the early adaptors see benefits. Eventually a successful technology becomes widely deployed, but history also shows that even the very best new technologies are never fully adopted by all sites. By contrast, the tree-based approach strictly dictates the order in which DNSSEC must be deployed, offers no benefits for early adaptors, and the resulting key validation approach only begins work when DNSSEC is fully deployed. Furthermore, if some zone chooses not to deploy DNSSEC, all its descendants in tree will also be prevented from deploying DNSSEC.

For example, the cairn.net zone "deployed" DNSSEC over a year ago[5] but this deployment is experimental at best. DNSSEC has not been deployed in the .net or root zones and there is no DNSSEC-capable parent to sign the cairn.net public key. Since there is no chain of trust, the cairn.net zone receives no tangible benefit from its DNSSEC deployment. Furthermore, if the .net zone chooses never to deploy DNSSEC, the cairn.net zone can never benefit from DNSSEC. A zone does not have the option of "switching" parents if the parent does not provide DNSSEC.

4.2.2 A Single Point of Failure

In the tree-based approach, each key relies on a predefined single chain of trust. This chain of trust must follow the DNS tree, there are no secondary paths and there is no ability for additional verification. A failure at some point in the tree breaks key validation for every zone below that point. In the worst case, a compromise of the master key allows an attacker to forge **any** DNS key.

For example, the darpa.mil key must be signed by .mil, the .mil key must be signed by the root, and the root must be signed by the master key. There is no other chain of trust that leads to the darpa.mil key. A failure at any point in this chain breaks key validation for darpa.mil and may allow an attacker to forge the darpa.mil key. Using a compromised master key, the attacker could create and sign a false root key. With the false root key, the attacker could create and sign a false .mil. Finally, the attacker could use the false .mil key to create and sign a false darpa.mil key.

The master key creates a particularly troubling single point of failure. A public key should have a limited lifetime that is based partly on the key's size (larger keys are harder to break and can have longer lifetimes). Due to its importance, the ideal master key would have a large size and a short lifetime. However, the size of the master key is limited by restrictions on the size of DNS queries and the lifetime must be long since changing the master key would require changing all resolvers in the Internet. The result is that all DNS security depends on a relatively small public key with an extremely long lifetime.

4.2.3 Undesirable Trust Relationships

The tree structure imposes trust relationships on organizations and offers no flexibility for individual zones. A zone can not decide that its parent offers insufficient security and can not enhance its security with additional chains of trust. A zone has a fixed place in the tree and can not switch to a more secure parent. A zone simply has to trust its parent and accept the security policies of the parent, as well as the security policies of every zone above the parent.

This tree structure results in questionable trust relationships that may not be politically or commercially viable. For example, the tree structure implies that DNS authentication between the U.S. State Department and the Pentagon relies on the same root key used for DNS authentication between Iraq and Russia. The storage, security, and maintenance of this root key could present a problem. By forcing organizations into predefined trust relationships, DNSSEC may create a type of authentication that is simply not useful in most cases.

But one might argue this simply reflects the reality of the DNS. A DNS zone is at least partly dependent on its parent. Currently, an attack against a parent can result in forged data or denial of service for all lower level zones (see Section 2). The tree-based approach to key validation simply further entrenches this existing vulnerability. Even with an alternate approach to key validation, the compromise of a parent could still lead to denial of service for lower layer zones.

However, there is an important distinction between denial of service and a failure of the authentication system. Regardless of the denial service issues, an attacker should not be given the ability forge data. By compromising a parent, an attacker will gain the ability to deny service to lower level zones, but an alternate key validation approach could still prevent the attacker from forging authenticated data. In practice, this means a compromised parent can prevent a user from reaching a service, but an alternate key validation approach could still prevent the attacker from imitating that service and prevent the user from revealing confidential data, such as a password or credit card number. The inability to forge data limits the power of the attacker in meaningful way.

For example, the tree-based approach to key validation allows an attacker who has compromised .mil to forge any data from darpa.mil. An alternate key validation system might have two chains of trust that lead to the darpa.mil key. At best, the attacker breaks only one of the chains by compromising the parent. This would not give the attacker the ability to forge data from darpa.mil. The attacker may still be able to deny service to darpa.mil by preventing resolvers from learning the darpa.mil nameservers, but the attacker can not provide false information about darpa.mil.

4.2.4 Burdens on the Key Signing Mechanism

In Section 3, we claimed that it would be unreasonable for one zone to securely obtain and sign the DNS keys for all zones. Due to the realities of DNS deployment, the tree-validation comes dangerously close to creating a single signer. The tree-approach creates a very heavy burden on large zones such as .com. The size of zones such as .com, .de, and so forth will present major administrative challenges for secure key exchange and reliable key signing. Note that these problems apply only to the zone administrators. Once the keys have been securely signed, a resolver will not care whether the zone has 2 children or 200 million children. Unfortunately, key signing is not a one time event since keys should expire on a monthly basis and new keys will need to be resigned. The key signing burdens placed on large zones may be unworkable or may degrade the level of trust in the DNS.

4.3 Recommendations for the Tree-Based Approach

The incremental deployment problems make the tree-based approach an unattractive candidate for use in the general Internet. Even if the tree-based approach was deployed, the limitations caused by single points of failure and undesirable trust relationships would severely limit its overall effectiveness. Problems with key exchange at sites such as .com w ould also present serious challenges that would grow worse as the Internet grows in size.

However, the tree-based approach may be well-suited for some portions of the DNS tree. In particular, the tree-based approach may work well in regions where there is a common authority and structured hierarchy that closely matches the DNS tree structure. The .mil region may meet these requirements, but most other DNS zones do not.

5 Web of Trust Key Validation

The web of trust approach to key validation is intentionally unstructured. The objective is to allow a maximum degree of flexibility for a resolver. Zones are free to select their own trust relationships and there are no restrictions on key signing. Instead of forming a fixed tree or other structure, the collection of signatures simply forms a random "web of trust" and a resolver can choose to follow any chain in this web.

For example, the .darpa.mil public key may be signed by the yahoo.com private key and the yahoo.com public key may be signed by the cairn.net private key. If a resolver trusts the cairn.net public key, the resolver can follow this chain of signatures and validate the darpa.mil key. Alternately, the resolver can decide that yahoo.com is not trusted to sign darpa.mil and the resolver may search for some other chain that leads to darpa.mil.

Using the key validation framework, the web-based approach can be summarized as follows:

- **The Initial Set of Trusted Keys:** A resolver is free to select any set of initial trusted keys. Typically a resolver will be configured to trust the public key for its own local zone. The local zone will establish some external trust relationships with a small number of other zones and sign the public keys belonging to this small set of other zones. These signatures form the first links in the resolver's potential chains of trust.

- **The Key Validation Rules:** In order to be considered valid, a key needs to be signed by at least one other valid key. Any chain of signatures is allowed and a resolver decides which signatures it will (or will not) believe. Note that the web of trust does not specify how to find a chain of trust or even guarantee that a chain of trust will exist for every key.

The web of trust approach is used in PGP[1], a popular system for signing and encrypting email.

5.1 Advantages of Web of Trust Key Validation

The strength of the web of trust lies in its robustness and the flexibility it offers to individual resolvers. This flexibility allows the web of trust to excel in precisely the areas where the tree-based approach failed. The web of trust advantages include ease of deployment, no single point of failure, and scalable key signing requirements.

5.1.1 Ease of Deployment

Incremental deployment of DNSSEC is trivial in the web of trust. To join the web of trust, a zone simply creates a public/private key pair and arranges to have its public key signed by some existing member of the web. Any zone can join at any time, provided it establishes a trust relationship with at least one existing member of the web.

There is no guarantee that all zones will be connected by the web. But if any two zone detect that a partition exisits,

then these zones can agree to sign each others keys and connect their respective partitions. For example, the DNSSEC testbed at cairn.net and sigz.net could connect their respective partitions by signing each others keys.

5.1.2 No Single Point of Failure

In the web of trust, there is no notion of a universal master key on which all security depends. If a resolver suspects that a key is compromised, the resolver can simply choose to ignore signatures from that key and search for alternate chains that do not involve that key. Ideally, the web of trust is richly connected and there are several chains of trust that lead to any given key.

5.1.3 Scalable Key Signing

Key signing is a local decision, negotiated between pairs of zones. Since there is no central authority, the signing process need not be concentrated in any one site. If a particular zone receives too many signing requests, it can simply ask some zones to obtain signatures elsewhere. As long as a the number of zones willing to sign public keys increases along with the size of the DNS, there is no limit on how many zones can be added to the web of trust and no single zone will presents a bottleneck in the signing process.

5.2 Disadvantages of Web of Trust Key Validation

The web of trusts disadvantages are also a result of its local flexibility. Without the structure found in the tree-based model, the web of trust fails in areas where the tree-based model excelled. In the web of trust, it can be extremely difficult to identify false signatures that have been intentionally created by an attacker or mistakenly generated due to an administrative error. It can also be extremely difficult to find chains of trust in the web.

5.2.1 Unauthorized Signatures

In the web of trust, a zone can choose which public keys it will sign. However, a zone can not control who signs its own public key. The web of trust does not define signing rules and any zone may sign the public key of any other zone. For example, the zone hacker.com can obtain[5] and sign the darpa.mil public key. The darpa.mil zone may not approve of this signature, but it has no way to prevent this signature. Furthermore, the web of trust does not provide any (in-band) way for darpa.mil to indicate whether a resolver should trust this signature.

---

[5]Note that the point of having a public key is so that anyone can obtain it so it is reasonable to assume hacker.com can obtain the darpa.mil public key.

In the worst case, a compromised or malicious zone might intentionally create and sign false public keys. Nothing prevents hacker.com from making up a false darpa.mil key and signing this false key. Nothing tells a resolver whether the signature from hacker.com should be trusted. The web of trust is likely to contain both valid and invalid chains and there is no pre-defined rule for distinguishing a valid signature from an invalid one.

Note that the web of trust responds to this problem by placing responsibility in the resolver. In systems such as PGP[1], the resolver is typically a human who can approve or reject a signature. In this environment, human intuition can be used to resolve conflicts and detect potential attacks. In DNS, a resolver works without assistance from a human and it is notoriously difficult to build human intuition into an automated program.

5.2.2 Finding a Chain of Trust

The other fatal flaw in the web of trust involves finding a chain of trust. First, there is no guarantee that a particular chain of trust even exists. For example, a resolver may trust only the cairn.net key but there may not be a chain of trust that starts with the cairn.net key and ends at the darpa.mil key. There is no key signing rule to guarantee the web of trust will be fully connected.

Second, there are no guidelines for finding a chain of trust. An exhaustive search of all potential paths could would eventually find a chain or prove it does not exist, but this approach would not scale to the size of the DNS. The delay required to blindly search for a a trusted chain would be unreasonably long. In addition, an attacker may insert false chains in an attempt to delay the discovery chain.

For example, a resolver may trust the public keys for cairn.net and nai.com. In order to validate the darpa.mil key, the resolver must find a chain of trust that leads from either cairn.net or nai.com to darpa.mil. A query asking for the darpa.mil key might return the KEY record along with SIG records from hacker.com, yahoo.com, and isi.edu. From this, the resolver knows it must find a chain of trust that leads from cairn.net or nai.com to any one of hacker.com, yahoo.com, or isi.edu. The search for a single darpa.mil key has now turned into an equally challenging search for any of three zone keys that have signed darpa.mil. Only a tedious exploration of all potential chains is guaranteed to find the necessary chain of trust (or prove such a chain does not exist). The problem is complicated further if an attacker tries to lead the resolver down potentially never ending false paths.

5.3 Recommendations for the Web of Trust Approach

As presented here, the web of trust is primarily a theoretical concept. The lack of structure makes it difficult to find chains of trust and an attacker who has compromised one zone key can sign false keys for any other zone. The attacker can also create false chains of trust designed to delay the key validation process. This makes a pure web of trust impractical for use in DNSSEC, but in the final section we will repair these problems to create a mesh approach to DNS key validation.

6 Mesh of Trust Key Validation

A mesh of trust is very similar to the web of trust, but the mesh of trust adds two important enhancements. First, the mesh includes **routing records** that can be used to find chains of trust. Second, the mesh allows a zone to define a **default route** that does not include any unauthorized signatures. The underlying idea is to think of the web of trust as a network topology. In this network topology, there is a link from zone A to zone B if and only if zone A has signed the key of zone B. The problem of finding a chain of trust is equivalent to the problem of finding a route in this network. The mesh will provide routing information in the form of routing records. In addition, a zone can specify a default route that can be used to verify the zone key. A resolver can use the route records and/or default route to find a chain of trust. A resolver can also gain added confidence in a key by finding two or more disjoint routes[6] in the mesh.

Using the key validation framework, the mesh of trust approach can be summarized as follows:

- **The Initial Set of Trusted Keys:**

  The set of initial trusted keys is a local resolver decision. It is recommended that a resolver trusts only the local zone key.

  The local zone should try to provide signatures such that there are multiple disjoint chains of trust that lead from the local zone key to the public keys of some well known authorities. A signed root zone would be an example of one such as authority, but any zone could potentially become a well known authority. If such a key is needed by the resolver, it would use the disjoint chains of trust to validate the current public key for the desired authority.

  Alternately, a resolver may simply be configured with public keys for some well known authorities or any

---

[6]Two routes are disjoint if have no common edges. In the mesh of trust, two disjoint routes are equivalent to two chains of trust that share no common signatures and hence the compromise of a single key could not have compromised both chains.

other set of public keys. The choice of initial trusted keys is a local resolver decision.

- **The Key Validation Rules:**

  Any zone may sign the public key of any other zone and a resolver may decide which signatures are considered valid.

  Associated with each zone key is routing record. The routing record includes entries that lists the last link in a chain of trust leading from some well known authority to the zone itself.

6.1 Routing Records

The mesh introduces a new type of DNS record, the routing record (RTE record). A routing record is associated with a DNS zone key and must be signed by that DNS zone key. The routing record is the DNS equivalent of a routing table. The routing record contains one or more entries. Each entry contains a **destination**, a **next hop**, and a distance. The destination names some zone in the DNS. The next hop contains the name of the last key in the chain of trust leading from the destination to the owner of the route record. The distance lists the number of keys in this chain. By following a sequence of route records, a resolver can discover a chain of trust that leads from the destination to the owner of the route record.

For example, suppose that the zone certauth.org has signed the public key for darpa.mil, darpa.mil has signed the public key for isi.edu, and isi.edu has signed the public key for cairn.net. In other words, there is a chain of trust that looks like "certauth.org - darpa.mil - isi.edu - cairn.net". The cairn.net zone could advertise this chain of trust by adding an entry to cairn.net route record. The entry in the cairn.net route record would list certauth.org as the destination, isi.edu as the next hop, and 3 as the distance. Similarly, an entry in the isi.edu route record would list certauth.org as the destination, darpa.mil as the next hop, and 2 as the distance. Finally, an entry in the route record at darpa.mil would list certauth.org as the destination, certauth.org as the next hop, and 1 as the distance. By following the sequence of route records, a resolver can discover a chain of trust leading from certauth.org to cairn.net.

The size of the DNS is too large to have every zone compute and store route record entry for every other zone. To address scaling problems, we envision a two tier hierarchy (shown in Figure 2). At the top of the hierarchy is a collection of zones that act as a type of routing backbone. These zones should have a route record entries that indicate how to reach every other zone in the top tier. In the second tier, zones may group themselves into regions. The zones in a region should have route record entries that indicate how to
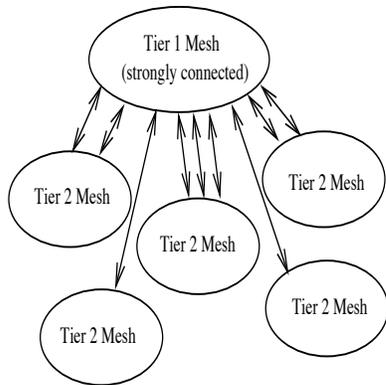
Figure 2. Routing in the Mesh

reach some subset of top tier zones. No zone is ever required to maintain a RTE record with an entry for every DNS zone.

Zones in either tier can also add other entries to their route record as they see fit. If the root zone is ever signed, the root zone should be placed in the top tier and every zone should be required to have a route record entry that indicates how to reach the root zone. Provided it has at least one route to a top tier zone, a second tier zone can select the scale of route entries that best suites its security needs.

6.2 Default Route Records

The route record solves the problem of finding chains of trust in the web, but it does not solve the problem of unauthorized signatures. The hacker.com zone can still create and sign a false key for darpa.mil. The hacker.com zone can also create a false route record and sign this false route record with the false zone key. A resolver that uses a single chain of trust to validate a key may still rely on the hacker.com signature to validate the false key[7]

To counter this threat, a zone can define a default route. The default route lists a chain of zones that starts at some well known authority and ends at the zone itself. This indicates who the zone has authorized to validate its key and a resolver following the default route record should avoid unauthorized signatures. The default route record must be signed by the zones parent and is included as part of the glue records stored at the parent. In the event the parent is unsigned, the zone walks up the tree to find the first signed ancestor. Note we can not simply allow anyone to sign the default route record since then hacker.com would forge and sign such a record.

---

[7]One signature from hacker.com would not be enough to convince a resolver that used multiple disjoint chains of trust to validate keys. To fool such a resolver, hacker.com would need some other zones to also sign the false darpa.mil key.

Note also that the default route lists zone names, not keys which change periodically. The chain of zone names, and hence the default route record, should rarely change. The default route record should be roughly equivalent to an NS record. A zone can construct these records and gives them to the parent when the zone is first created. The zone rarely, if ever, needs to contact the parent and update this record. There is no new burden added to the parent/child relationship.

For example, suppose the darpa.mil zone wants other zones to use the DNS tree for validating the darpa.mil key. The darpa.mil zone can create a default route record and list the zones root, .mil, and darpa.net in the default route record. The default route record would be sent to the .mil zone, signed by .mil, and stored as part of the glue for darpa.mil.

6.3 Advantages of Mesh Key Validation

The mesh has the same strengths as the web of trust, but in addition the mesh of trust includes route records used to find chains of trust and default route records to detect unauthorized signatures.

6.3.1 Ease of Deployment

Incremental deployment of DNSSEC is trivial in the mesh of trust. To join the mesh of trust, a zone simply creates a public/private key pair and arranges to have its public key signed by some existing member of the mesh. By construction, there will be a chain of trust that starts at the well known authority and ends at the existing member. By having its key signed by this existing member, the new zone also gains a chain of trust that starts at some well known authority and ends at the new zone. The existing member is the last hop in this chain of trust so the new zone constructs a route record entry that lists the well known authority as the destination and the existing member as the last hop.

Ideally, a new zone arranges to have its key signed by several existing members and may offer the keys of other zones. This helps insure the mesh of trust is richly connected.

6.3.2 No Single Point of Failure

In the mesh of trust, there should be several disjoint chains of trust between any pair of zones. In addition, each zone should have disjoint chains of trust leading from well known authorities to the zone. A resolver can pursue any of the chains of trust and may validate the key using multiple disjoint chains of trust if the application requires a strong form of authentication.

6.3.3   Scalable Key Signing

Key signing is a local decision, negotiated between pairs of zones. The signing process is not concentrated in any one site. If a particular zone receives too many signing request, it can simply ask some zones to obtain signatures elsewhere. As long as a the number of zones willing to sign public keys and act as well known authorities increases along with the size of the DNS, there is no limit on how many zones can be added to the mesh of trust.

6.4   Overall Recommendations

The mesh of trust is a promising approach for DNS key validation. At this point, the mesh of trust is still primarily a concept. But this approach will be further explored and implemented as part of on-going research on DNSSEC and DNS key validation. We believe the both the web of trust and the tree-based approach will not allow DNSSEC to succeed and the mesh of trust offers a potential way forward.

References

[1] D. Atkins, W. Stallings, and P. Zimmerman. PGP Message Exchange Formats. RFC 1991, Internet Engineering Task Force, Aug. 1996.

[2] S. Bellovin. Using the Domain Name System for System Break-ins. In *Proceeding of the Fifth Usenix UNIX Security Symposium*, June 1995.

[3] D. Eastlake. Domain Name System Security Extensions. RFC 2535, Internet Engineering Task Force, Mar. 1999.

[4] S. Josefsson. Authenticating Denial of Existence in DNS with Minimum Disclosure. Internet Draft draft-jas-dnsext-no-00.txt, Jan. 2001.

[5] D. Massey, T. Lehman, and E. Lewis. DNSSEC Implementation in the CAIRN Testbed. Internet Draft draft-ietf-dnsop-dnsseccairn-00.txt, Nov. 1999.

[6] P. Mockapetris. Domain Names - Concepts and Facilities. RFC 1034, Internet Engineering Task Force, Nov. 1987.

[7] P. Mockapetris. Domain Names - Implementation and Specifications. RFC 1035, Internet Engineering Task Force, Nov. 1987.

[8] P. Vixie. DNS and BIND Security Issues. In *Proceeding of the Fifth Usenix UNIX Security Symposium*, June 1995.

[9] B. Wellington. Domain Name System Security (DNSSEC) Signing Authority. RFC 3008, Internet Engineering Task Force, Nov. 2000.

[10] T. Ylonen, T. Kivinen, M.-J. O. Saarinen, T. J. Rinne, and S. Lehtinen. SSH Protocol Architecture. Internet Draft draft-ietf-secsh-architecture-06.txt, Nov. 2000.