A Scalable Key Distribution Hierarchy

Patrick McDaniel   Sugih Jamin
Electrical Engineering and Computer Science Department
University of Michigan
Ann Arbor, MI 48109-2122
{*pdmcdan,jamin*}*@eecs.umich.edu*

April 13, 1998

Abstract

As the use of the Internet for electronic commerce, audio and video conferencing, and other applications with sensitive content grows, the need for secure services becomes critical. Central to the success of these services is the support for secure public key distribution. Although there are several existing services available for this purpose, they are not very scalable, either because they depend on a centralized server or rely on *ad hoc* trust relationships.

In this paper, we present and examine a flexible approach to certificate distribution scalable to arbitrarily large networks. We propose a two level hierarchy where certificates can be independently authenticated by one or more peer authorities, called *keyservers*. Certificates for end-user and host entities are managed within local domains, called *enterprises*. By administering certificates close to the source, we reduce the load on the key servers and the effects of network topology changes. We describe the design of our system and present a preliminary performance analysis based on traces of present-day DNS requests.

1 Introduction

Over the past several years, the use of distributed applications has grown immensely. These applications allow geographically distant users to communicate, leading to social, educational, and commercial interactions that were previously impossible. Unfortunately, because of the openness of the Internet, the form and content of these interactions is vulnerable to attack. Limiting these vulnerabilities is essential to the future success of these applications.

Central to securing communication over an insecure medium is the distribution of cryptographic keys. These keys allow the communication participants the ability to ensure the authenticity, privacy, and integrity of the content. These protections are fundamentally predicated on the security of the key distribution mechanism. Although the existing body of work on this problem is extensive, no single solution has adequately addressed the flexibility and scalability problems inherent to key distribution in networks the size of the Internet.

In this paper we propose a global certification hierarchy used to distribute public key certificates over the Internet. The architecture of the system is driven by several design goals. First, the design must be scalable. The target domain (Internet) contains millions of users and services, which will require the distribution of a like number of certificates. Any solution that limits accessibility based on the size of the user base or distance between users will be of limited use. Second, certificate retrieval must be timely. Effects on user-perceived performance aside, certificate retrieval that incurs long delays may invalidate the content of the information exchange. Finally and most importantly, certificate distribution must be secure.

The focus of our investigation reported in this paper is on the *scalability of the architecture and protocols of our public key certificate distribution hierarchy*. The two defining features of our architecture are:

1. A set of *keyservers* form a fully-connected group of mutually authenticated entities which, in concert, serve as our certificate authentication authority. By mandating full-connectivity of authentication between the keyservers, we bound certificate authentication time to at most four requests.

2. The window of vulnerability that a revoked certificate may be used is controlled by user-specified certificate hold time. Certificate hold time can be set to less than or equal to the Certification Revocation List (CRL) publication period of systems that depend on CRLs to ensure certificate integrity.

In evaluating this architecture we attempt to determine the viability of the system as a key distribution mechanism by estimating its performance characteristics. Further, we attempt to verify the scalability and timeliness of certificate retrieval. While we also identify several important issues related to the security of the proposed architecture itself, a rigorous analysis of the security provided by the architecture is outside the scope of this paper.

In the Privacy Enhance Mail system (PEM) [Ken93], the authors define a certification authority whose requirements are similar to that defined in this paper. They describe a rooted tree, where the security of all certificates are ultimately guaranteed by the root authority. In a global network, finding one authority which all users trust is problematic. This shortcoming is common to many existing key distribution systems. In this paper, we address this design limitation by defining a set of root nodes called *keyservers*. End users indirectly subscribe only to those keyservers in which they have some level of trust. Verification of certificates may be performed by polling a subset of the subscribed keyservers, from which a majority result is used.

The DNSSEC [EK99] specification outlines a key distribution hierarchy that uses the existing DNS infrastructure to support authenticated retrieval of DNS data. When retrieving a DNS record an application will traverse the DNS hierarchy, recursively authenticating each zone. This scheme introduces arbitrarily long chains of trust, which reduce the requestor's confidence in the retrieved content. We avoid these long chains of authentication by proposing a design that ensures that each keyserver is able to authenticate any other. In this way, we reduce the maximum length of the authentication chain to four.

A Certification Revocation List (CRL) is a periodically published list of certificates that have been revoked. Revocation may be stated either explicitly (by unique indentifier) or implicitly (by latest valid certificate timestamp or serial number). Systems that use Certification Revocation Lists (CRL) to announce the revocation of certificates force users to accept a window of vulnerability in which a revoked certificate can be used. This window is equal to the CRL publication period, which is controlled by the administrators of the authorities. Moreover, an attacker may interfere with the delivery or modify these lists in transit. In our architecture, we address these limitations by allowing the user to specify the length of time a certificate will be cached, and thus the window of vulnerability.

A possible limitation of this architecture is that keyserver resources may become the bottleneck. With millions of end users, there is a potential for the keyserver and associated network resources to become overloaded. We attempt to limit the load on the keyservers by certificate caching and the aggregation of end user requests. We define an *enterprise* as a collection of end users served by one distinct member called the enterprise root. The enterprise root proxies requests for certificates from internal hosts by communicating with keyservers and other enterprise roots, caching the returned certificates. As warranted, an enterprise can be further divided into smaller groups to decrease the load on the enterprise root.

A certificate contains the public key of some entity and a series of other fields that indicate the identity, contact information, expiration time, and other features of the certificate. The X.509 [Rec93] specification describes a popular format for public key certificates. In this paper, we assume all certificates contain a public key generated using the RSA [RSA92, RSA78] cryptographic algorithm. We do not limit the type of entity that can own a certificate. End-users, applications and hosts are examples of potential certificate owners. Throughout this paper and without loss of generality, we assume that certificates are owned by hosts. Each enterprise host (non-root node in an enterprise), enterprise root, and keyserver creates a certificate. The primary function of the system is the distribution and authentication of these certificates. Distribution of certificates is performed by the protocol described in Section 2.4. A retrieved certificate is authenticated by the verification of an attached digital signature.

The length of time a certificate is held in a cache, called the certificate hold time, is an important parameter of this architecture. If certificate locality can be observed, the performance of the system will be effected by this value. A long hold time will increase the cache hit rate, resulting in less aggregate traffic. This parameter also bounds the time that a revoked key can be used. A short hold time reduces the possibility of obtaining a revoked key from the cache. When selecting a value for this parameter, administrators must balance this tradeoff between performance and security.

In determining the viability of the architecture, we need a characterization of traffic the system will see. From an analysis of expected usage, we can generate heuristics for determining preferred values for the protocol parameters. Primarily, we seek to find the best choices for caching policies and certificate hold times. Unfortunately, there are limited existing systems of this type from which we can collect data.

We argue that DNS (Domain Name Service) [Moc87a, Moc87b] likely has the usage characteristics that our system will encounter. Similar to certificate requests, DNS requests are most often used as a precursor to a session [DOK92]. From an analysis of DNS traffic we can determine when, how often, and to whom connections are made. Assuming that the secure communication supported by our key distribution hierarchy follows the same traffic pattern as DNS, we propose to estimate the performance of our system based on DNS traffic.

The structure of the paper is as follows. We outline the design of our key distribution hierarchy in Section 2. In Section 3 we use traces of DNS traffic from existing networks to analyze the performance of the design. Section 4 describes some of the existing and proposed designs for certificate distribution systems. Section 5 concludes with a summary of our findings.

2 Design

In this section we describe the design of our proposed key distribution hierarchy.

2.1 Goals

The purpose of this work is to define a key distribution hierarchy that is scalable to the Internet, while avoiding some of the problems found in existing solutions. We see four goals that are central to the design of an Inter-

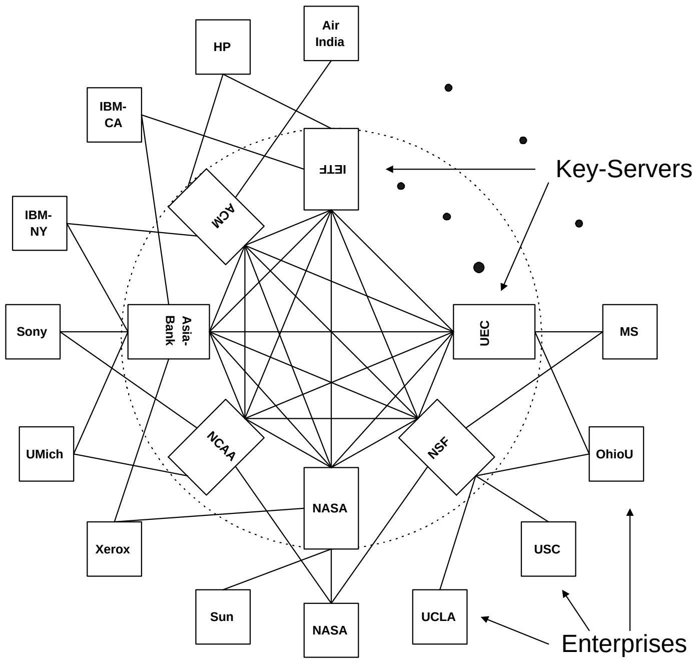

Figure 1: Internet Level Architecture

net certificate distribution service: scalability, availability, robustness, and flexibility.

The architecture must be scalable to the Internet. In our design, we attempt to manage certificates close to the certificate owner, thus reducing the administrative load on the upper levels of the hierarchy.

Certificate retrieval must be highly available and robust. We define an *authentication channel* as the request-reply process of an enterprise root obtaining a certificate from a keyserver. In our architecture, we attempt to distribute critical authenticating information through multiple independent channels. Should confidence in or connectivity to some node be lost, an enterprise root can use an alternate authentication channel. By making these channels independent, an enterprise root can validate received certificates. Users may also opt to validate certificates through several independent channels, resulting in increased confidence in their authenticity. This approach is similar to mechanisms for high-availability proposed in [GS91].

The architecture must be flexible. The topology of the Internet is constantly changing, so the architecture and underlying protocols must not be dependent on the physical connectivity or location of any singular authority. Mobility of users is of equal importance. As users travel from one domain to another, it is necessary for the certificate and associated authenticity information to travel with them. By limiting the amount of time and expense of this transition, we provide for the dynamic nature of users in the Internet.

2.2 The Architecture

In our design, we introduce a two level hierarchy consisting of the *keyserver* level and the *enterprise* level. The keyserver level contains a set of servers from which enterprise and keyserver certificates can be retrieved. The enterprise level contains independent hierarchies of end users. Figure 1 describes an Internet-centric view of one possible configuration of the architecture. In the figure, a link between two entities represents an exchange of digital signatures, where each end-point signs and permanently caches the others' certificates. Keyservers form a fully-connected graph of peers, where all keyservers have exchanged certificates with all others. By mandating a fully-connected graph, we limit the number of certificates needed to be retrieved during authentication of other certificates. An authenticated certificate of any keyserver can be retrieved from any other keyserver. In limiting the number of retrieved certificates, we reduce the points of failure or attack.[1] Using an out-of-band channel, enterprises exchange signatures with keyservers in which they trust. It is from these keyservers that they later retrieve authenticating certificates. In essence, the exchange of signatures between a keyserver and an enterprise states that the enterprise trusts the keyserver to advertise cor-

[1]The requirement that all keyservers exchange signatures is used to bound the transitivity of trust during certificate authentication. The effect of relaxing this requirement would be the introduction of additional intermediate keyservers into the authentication process, which may lower confidence in the process. In the degenerate case, the retrieval process would become similar to authentication in the PGP [Zim94] system (see Section 2.3).

rect certificates. However, this trust need not be absolute. Later, during authentication, multiple keyservers may be consulted.

Keyservers are intended to be administered independently by regional, national, or global organizations. In terms of hardware and administrative practices, these servers should have many of the same characteristics of those defined for the PCA services of RFC 1422 [Ken93]. These practices define procedures used for mutual authentication before the exchange of signatures. For example, a keyserver may choose not to exchange signatures with some enterprise unless the administrators can provide sufficient proof of the security of their own internal signing practices. An enterprise provides its certificate to each keyserver with which it wishes to register. After appropriate mutual validation of credentials, the keyserver signs and caches the enterprise certificate and the enterprise root signs and caches the keyserver certificate. A thorough description of the use of digital signatures can be found in [DH76].

Each enterprise encompasses some organization of end users. The enterprise is intended to represent a group of geographically close local area networks under control of a single administrative authority. A distinct host, called the enterprise root, is logically the single point of contact for requests for certificates of the enterprise.

We propose two models for communication internal to the enterprise. The first (or *enterprise root based*) model uses the enterprise root node as a single point of contact for certificate retrieval. Similar to DNS, enterprise hosts directly contact the local service (enterprise root) to make requests for internal or external certificates. Retrieved certificates are cached at the enterprise root node and each end user host. In the second (*group based*) model the enterprise is a hierarchy of multicast groups which form the enterprise tree. Each group in the tree contains a node called the *parent node*, which is also a member of the next higher group in the tree. The role of the parent node is to propagate requests and responses to the vertically adjacent groups in the enterprise tree. These requests are only propagated when the certificate is not held in any cache within the local group. In response to a certificate request, an enterprise internal protocol is used to search the caches of all hosts within the enterprise. Using these groups, the enterprise forms a logical enterprise-global cache. In Section 3, we analyze the performance tradeoffs between these two approaches.

All certificates for entities within the enterprise are permanently stored at the enterprise root. When a local host registers its public key with the enterprise, they mutually authenticate and sign each other's certificates. When an external entity requests a certificate for one of these hosts, the enterprise root will immediately respond with the cached certificate. Similar to DNS, if the root is properly placed (i.e. at a network border), very little traffic should be generated by external requests on the enterprise network.

The purpose of the hierarchy is the secure retrieval of public key certificates for arbitrary end points. This is done by the collection and verification of signed certificates. The requester logically traverses a graph representing signature exchanges between the enterprises and keyservers, collecting certificates at each hop. After assembling the certificates, the signatures are authenticated. If all certificates are authenticated, the user is free to use the retrieved end point certificate.

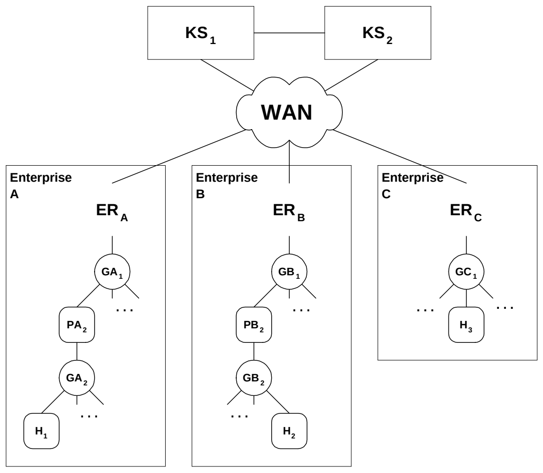

Figure 2: Protocol Example - group oriented enterprise, where H_i represents a host, GX_i represents group i in enterprise X, ER_X represents the root node for enterprise X, and KS_i represents a keyserver.

This scheme is best illustrated through an example. In Figure 2, we show a possible configuration of the hierarchy. In this example host H_1 is attempting to retrieve a certificate for host H_3. Through the protocol described in Section 2.4.1, H_1 will retrieve three certificates; the certificate for keyserver KS_1 signed by enterprise root ER_A ($\{KS_1\}_{ER_A}$), the certificate enterprise root ER_C signed by keyserver KS_1 ($\{ER_C\}_{KS_1}$), and the certificate for the host H_3 signed by enterprise root ER_C ($\{H_3\}_{ER_C}$). H_1 authenticates the signature in $\{KS_1\}_{ER_A}$ using the public key in locally stored certificate $\{ER_A\}_{H_1}$, the signature in $\{ER_C\}_{KS_1}$ using the certificate $\{KS_1\}_{ER_A}$, and so on. H_1 may choose to validate $\{ER_C\}_{KS_1}$ by contacting additional keyservers. If the enterprises do not share a keyserver (in the sense of signature exchanges), only *one* additional certificate is retrieved, and validated (see Section 2.4.1 for an example), because our architecture mandates that keyservers be fully connected in their trust relationship, as described earlier in this section.

2.3 Trust Model

In this design, an end user implicitly trusts the local administration of the enterprises for both end-points, and to a lesser degree the keyservers. Specifically, the end-user trusts both the local and remote enterprise root nodes to advertise correct certificates. The keyservers are also trusted in this way, but alternate channels for validating certificates are available. Keyservers are logically the entities furthest removed from the end-points, and are likely to be least trusted. This model is simlar to other public key and trusted third party systems.

Kerberos [SNS88, NT94] is a trusted third party key distribution system based on symmetric key cryptography. Users appeal to *Ticket Granting Servers* (TGS) for tickets to services and other users. These tickets are then used to establish a symmetric session key under which secure communication can be supported. Larger networks (such as the Internet) are divided into *Realms*. Realms securely register secrets with other realms, which are later used establish inter-realm tickets. Users communicating between realms are then required to trust both end-points; the source and target realms.

As with the single rooted hierarchy described in the PEM specification [Ken93], the quality of retrieved certificates is only as good as the local administration of both end points. In fact, when both enterprises share a keyserver, the trust model is very similar. With respect to trust, our proposed system has two distinct advantages over PEM. First, our system does not rely on a single root authority for guarantees of authenticity. Certificates from several keyservers may be retrieved, the results being used to increase confidence in authenticity. This increased confidence is predicated on the independent operation of the keyservers and the correctness of their signing practices. A second advantage is that the selection of authorities is decided by the end-user, not dictated by the architecture.

The web-of-trust systems such as PGP offer a more relaxed trust model. Certificates are signed and distributed in an *ad hoc* manner. A user is required to trust all the intermediate signers. The advantage of the PGP system is that the intermediaries are fluid, allowing anyone to sign. Unfortunately, finding a certificate for an arbitrary end point in which one can trust the signers is difficult. The certificate location problem is avoided in our architecture through the use of the keyserver and enterprise services. By allowing the requester to determine the intermediate signers, control of the trust relationship is left to certificate users.

2.4 The Protocol

In this section we describe the certificate retrieval protocol used in the hierarchy. We divide the protocol into two parts, the enterprise external and enterprise internal protocols. The enterprise external protocol describes the process used to retrieve certificates that are not local to the enterprise. The enterprise internal protocol describes the communication of request forwarding and caching between the enterprise root and hosts. For simplicity, we describe the aquisition of each end-user, enterprise, and keyserver certificate independently. Protocol optimizations, such as request aggregation and negative caching, will reduce the retreival costs.

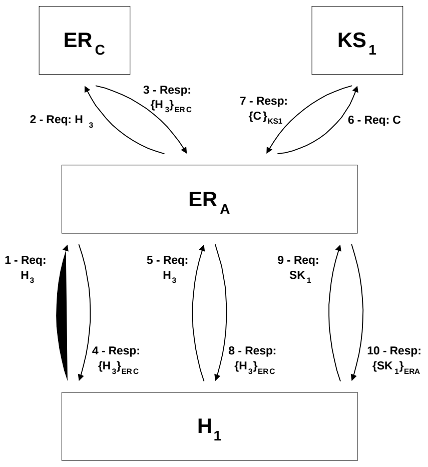

Figure 3: The certificate request process.

2.4.1 The Enterprise External Protocol

The enterprise external protocol describes the retrieval of certificates of hosts local to the enterprise by external hosts and the retrieval of external certificates by local hosts. Each enterprise root node begins operation by warming its cache with permanent entries for the certificates of entities within the enterprise, the enterprise certificate, and the certificates of each keyserver with which it has exchanged signatures.

When the root node receives an external request for a certificate belonging to an entity within the enterprise, it returns the certificate and appends the list of keyservers with which it has exchanged signatures. The list of keyservers associated with the enterprise is always cached with the certificate.

As dictated by the communication model used in the enterprise, requests for certificates that do not currently reside in a host cache are ultimately forwarded to the enterprise root node. If the certificate is found in the enterprise root cache, the certificate and associated keyserver information is returned. Otherwise, the request is forwarded to the external enterprise. The response is cached and returned to the requesting host via the enterprise internal protocol. A similar process is used for keyserver certificates, with the originating requester specifying from which keyserver it wishes to retrieve the certificate.

It is worth noting that we do not specify a mechanism for locating the enterprise root node of an external enterprise. There are several existing designs for scalable network directory services, such as DNS [Moc87a, Moc87b]. These services are readily available within today's Internet infrastructure, and as such are beyond the scope of this paper.

We illustrate this protocol through an example. We return to our example hierarchy description in Figure 2. Assume all hosts initially have empty caches, save the permanent entries. We state that both the enterprise root nodes ER_A and ER_C have only exchanged signatures with keyserver KS_1 and that enterprise root node ER_B has exchanged signatures with keyserver KS_2. In Figure 3, we illustrate the request process used by enterprise host H_1 in enterprise A to obtain and authenticate the certificate of host H_3 in enterprise C. H_1 begins by requesting the certificate of H_3. Via the internal protocol, the request ultimately arrives at the enterprise root node ER_A (step 1 in Figure 3). ER_A forwards the request to ER_C, returning the results to H_1 (steps 2, 3 and 4). H_1 then determines that the certificate of ER_C is needed, and repeats the request process, specifying that the certificate be retrieved from the keyserver KS_1 (steps 5-8). Based on the keyserver information returned in the H_3 request, H_1 notes that both enterprises shared the keyserver. H_1 determines that this is an acceptable trust relationship because they share a common keyserver, which it trusts. Finally, H_1 requests and receives the certificate for keyserver KS_1 (steps 9 and 10 in Figure 3). Having assembled all the certificates, H_1 recursively authenticates the digital signatures. Based on the results of the authentication, H_1 may initiate some secure action using the certificate.

When the enterprise of a requester host and the enterprise of the requested certificate do not share a common keyserver (in terms of signature exchange) an additional step is necessary. An example of this scenario (from Figure 2) would be H_1 requesting the certificate of H_2. The certificate retrieval would proceed as in the previous example, with the addition of the retrieval of the certificate for keyserver KS_2. During authentication of the certificates, H_1 would authenticate KS_2's certificate using the certificate of KS_1. Recall that we require each keyserver to exchange signatures with all the other keyservers, so any trusted keyserver (KS_1) may be used to retrieve and authenticated the certificate of another keyserver (KS_2) with a single additional request.

An advantage of this architecture is that any host can retrieve an arbitrary certificate with a maximum of four requests. This is an advantage over previous systems where the retrieval of potentially many certificates may be required. Additionally, the effects of network partitioning can be reduced. When some keyserver is unavailable, others can be consulted. However, a certificate cannot be retrieved from an unavailable enterprise. This may not be detrimental to the usefulness of our proposed architecture, as the target enterprise host is likely to also be unavailable.

When more than one certificate for a single target is received with valid signatures, the enterprise and end user nodes must make a policy decision. The most secure policy would be to disallow the use of any of the certificates until the conflict can be resolved. An alternative approach is to bias towards those keyservers with whom the local enterprise has exchanged keys. The administration of these keyservers is known to the local enterprise, and thus should have more trust associated with them. Another policy is to allow the end-user or application to make the decision on a case by case basis. The requesting entity knows the importance of the operation about to take place, so is in a better position to make a decision about the risk involved in using a questionable certificate. Our architecture does not dictate the policy, as any single policy will not be appropriate to all environments.

2.4.2 The Enterprise Internal Protocol

Recall that we have two models for communication internal to the enterprise. In this section, we describe the operation of both the *enterprise root* and *group based* models.

In the enterprise root based communication model enterprise hosts communicate with the enterprise root via unicast. Enterprise hosts make requests for certificates directly to the enterprise root. The protocol for retrieving certificates from external entities proceeds as described above, the results of which are cached and returned to the requesting host. We study several cache replacement policies for this model in Section 3.5.

In the group based communication model, each group in the enterprise tree represents a multicast group to which all members' requests are sent. A group has a distinct node, called the parent node, which operates as the link to the higher groups in the tree. The enterprise root acts as the parent node of the highest group in the tree. The request process begins by a member host multicasting the request to the local group. Each member of the group looks in their local cache for the requested certificate. If the certificate is found, the member starts a timer initialized to a random value between 0 and some fixed time α. If the timer expires and no response has been seen in the group, the member multicasts its response to the group. When the request is received by a parent node, it sets its timer to $\alpha + \delta$. If the parent node timer expires before a response is seen in the group, it multicasts the request to the next higher group in the tree. This random wait response scheme at each level is similar to those used in the SRM [Flo95] reliable multicast protocol. This process is repeated until a response is received or the request timer expires at the root node in the tree. In the latter case, the enterprise root retrieves the certificate via the enterprise external protocol and returns the response certificate. The response is then recursively multicast to the enterprise groups, where it is eventually received by the requester.

2.4.3 Key Revocation

Key revocation is a primary concern of any key distribution system. If the private key associated with a certificate becomes exposed, the owner will wish to immediately invalidate all cached copies. Due to the large number of hosts, finding all the cached copies in large networks can be problematic. In extant systems, Certificate Revocation Lists (CRLs) are used for this purpose. Unfortunately, CRLs create a different set of problems. The mechanics of disbursing this potentially large set of revoked certificates to an enumerable set of interested parties presents a daunting task.

Similar to DNSsec [Eas98], we avoid many of the problems associated with key revocation by requiring short hold times on cache entries. When a certificate is revoked, the enterprise or keyservers that advertise the certificate immediately flush the certificate from its permanent cache. In this way, the time a compromised certificate can be used is bounded by the cache hold time. An enterprise can ensure that no revoked certificates are used by setting the cache hold time in the enterprise hosts to zero.

A byproduct of our key revocation mechanism is user mobility. Users who wish to move from one enterprise to another can do so without revoking thier key. When moving between enterprises, the only requirement is that the certificate is registered (digitally signed and advertised) by the new enterprise. An enterprise never has access to the user's private key, so their are no security risks associated with using the old key.

Implicit to this approach is the need for freshness assurances in the certificate retrieval process. In the absence of such assurances, an attacker may obtain certificates before revocation and later impersonate one or more keyservers. In this way, the attacker could induce a user into using a revoked key. We stipulate that all requests are retrieved using a protocol that ensures freshness, but cite no specific method. There are several approaches for achieving freshness described in [NS78] and [Sch96].

Our approach to certificate revocation that relies on certificate hold time offers several advantages over the use of CRLs. First, the control over the window of vulnerability is left to the user. With CRLs, the user is forced to use the last published CRL, which is broadcast at a rate controlled by the certification authority. Moreover, an attacker may delete or intercept the CRL publication. Secondly, revocation in our system is a simple, one step process. Revocation is not predicated on the support of external entities.

Note that we assume an out-of-band method for the revocation itself. Social processes are needed for this purpose. One possible solution would be to assign secret revocation codes during the signature exchange that would be used as credentials during revocation.

3 Performance Evaluation

In this section we evaluate the performance of our architecture by using traces of Domain Name Service (DNS) traffic as a model of expected usage. We investigate the performance characteristics of the three central components of our architecture: the (end-user) enterprise hosts, the enterprise root node, and the keyserver. We attempt to quantify the following factors:

1. First and foremost, the locality of DNS requests. The scalability of our proposed architecture hinges crucially on the locality of certificate requests.

2. The effect of cache replacement policies on enterprise root node's certificate cache hit rate.

3. The cost benefit trade offs of the two enterprise internal communication models described in Section 2.4.2.

4. The effect of specific enterprise characteristics, mainly, the number of hosts within an enterprise, on the validity of our results.

5. The load on keyservers as measured by the number of requests received; and the effect of enterprise-level certificate caching on keyserver load.

The main results of our evaluation of the proposed key distribution hierarchy based on DNS traces can be summarized as follow:

The traces we collected on DNS requests show a high degree of locality.

Enterprise hosts tend to be idle, in terms of initiating DNS requests, for long periods of time. During these idle periods host caches remain empty, such that enterprise host caching has minimal effect on the overall performance of the architecture.

The group based communication model provides us a practically infinite cache size. Hence enterprise-level hit rate is best under the group based communication model. Under the enterprise root communication model, we find the LFU (Least Frequently Used) cache replacement policy to provide marginally better performance than the LRU (Least Recently Used) cache replacement polices. The performance of the LFU cache replacement policy under the enterprise-root communication model approaches that of the group based communication model. In all cases, the enterprise root is not overwhelmed by the requests seen in our DNS traces, even during the period of highest load.

The observation that enterprise host caches are often empty explains why caching at the enterprise root alone provides cache hit rate comparable to that of the group based model. This, coupled with the observation that enterprise-root sees an acceptable load, leads us to conclude that the management overhead of and the additional retrieval latency introduced by the group-based model are not warranted by the marginal performance gain achieved.

Using traces from enterprises of varying sizes, we determine that these findings appear to be independent of enterprise size; we assert that our Enterprise External and Internal Protocols are scalable to existing domain sizes.

DNS traces from a top-level domain (*us*) shows that in the worst case the top-level DNS server processes a little over 100,000 requests per hour. Arguing that traffic seen by a top-level DNS server models the traffic our keyservers will see, we stipulate that the load placed on keyserver hosts will be low. Our simulation results also show that enterprise-level certificate caching can further reduce keyserver load by as much as 65%.

Based on these results, we claim that our key distribution hierarchy is scalable to networks the size of the Internet.

In the remainder of this section we first present the collection and characteristics of our traces. After describing our cache simulation setup, we present data supporting our performance claims on the proposed key distribution hierarchy summarized above.

Table 1: Collected DNS traces

Trace	Hosts	Start		End		Requests		
		Date	Time	Date	Time	Local	Remote	Total
AT&T	360	1/7/98	4:08pm	1/8/98	4:08pm	27,744	29,315	57,059
(trace 2)*EECS 2*	1,104	2/6/98	11:57pm	2/7/98	11:47pm	204,410	272,656	477,066
CAEN	7,080	1/8/98	1:55pm	1/9/98	3:02pm	175,177	286,375	461,552
umich.edu	31,113	1/6/98	11:55pm	1/7/98	9:59pm	1,027,967	543,273	1,571,240
us	N/A	1/7/98	11:38pm	1/9/98	9:14pm	N/A	N/A	2,365,122

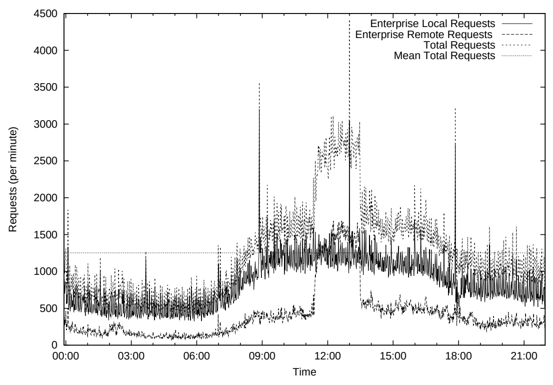

Figure 4: Requests per minute for *umich.edu* trace

3.1 Trace Collection

The architecture of DNS as implemented on the current Internet parallels our proposed key distribution hierarchy in that DNS recognizes a set of top-level root servers that resolve top-level domain names (e.g. *edu, com, gov, mil, org,* etc.) and domain-level name servers that resolve domain specific names. We collected DNS request traces from both top-level and domain-level name servers. We use traces from top-level name servers to model expected traffic our keyservers will see. We use domain-level traces to model expected enterprise root traffic. To study the effect of domain size on the expected performance characteristics of our enterprise root nodes, we collected DNS traces from large, medium, and small DNS domains. DNS request traces were collected using the logging and debug options built-in to the `named` name server. A summary of the collected traces is given in Table 1.[2]

We choose the University of Michigan (*umich.edu*) as a representative large-size DNS domain. The University is served by four primary name servers. We collected traces from the name server that is set up to be the main name server for the University by the network administrators. We consider the College of Engineering of the University a representative medium-size DNS domain. The *engin.umich.edu* domain is a subdomain of the *umich.edu* domain, but operates its own two DNS servers independently. In our study, we treat *engin.umich.edu* and *umich.edu* as two separate enterprises. We use partial data for the *engin.umich.edu* and *AT&T Labs Research*. In the *engin.umich.edu* trace, we collected from one of two servers, and one of an unknownnumber of servers for the *AT&T Labs Research* trace. To simulate small enterprises, we use two sets of traces: one collected from the Electrical Engineering and Computer Science (EECS) Department of the University (*eecs.umich.edu*), which, again, though a subdomain of *umich.edu*, operates its own name server, and the other from *AT&T Labs Research*.

[2]We are attempting to collect DNS traces from a wider range of domains with varying sizes.

Top-level DNS servers have many of the same properties of our proposed keyservers. Most importantly, the top level DNS servers are used as directory service for addresses that are not local to a requestor's domain. This is precisely the same function that a keyserver fulfills, with the exception that the keyserver distributes certificates instead of address information. We collected top-level DNS request traces from the name server *excalibur.usc.edu* which serves as one of nine authoritative domain name servers for the *us* domain.

3.2 Trace Reduction and Analysis

The rate of requests seen in the *umich.edu* trace follows the diurnal distribution cited in many network traffic studies. The overall peak rate is seen during the normal 8am to 5pm periods, while the remaining periods see less traffic. We found DNS request arrival rates to be Poisson distributed. Since DNS requests usually precede TCP session arrivals, this observation is in agreement with the finding reported in [PF95] that TCP session arrivals are Poisson. In Figure 4, we show the number of request per minute, as seen over the entire *umich.edu* trace. Enterprise local request arrivals vary between .5 and 4 of the average, measured over one minute periods. From this data we can assume the high-water mark periods, in terms of requests over some period, are infrequent and within the same order of magnitude as the average. The rate of external requests have essentially the same characteristics. This leads us to believe that a solution which performs well for the average case will perform well in practice. Figure 4 indicates a surge of remote requests between 12:00pm and 2:00pm. A cursory visual inspection of the data for those hours shows a high degree of requests for Internet web-sites. From this, we attribute this surge to lunch hour

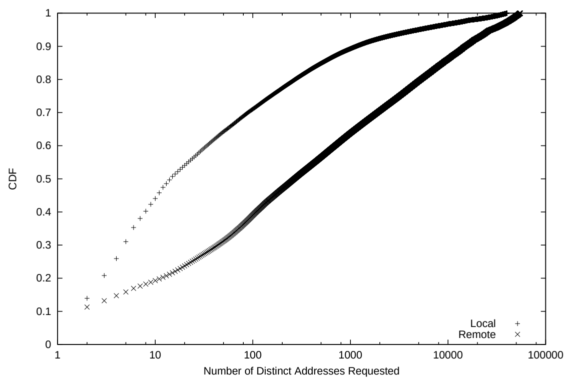

Figure 5: Cumulative distribution function for IP addresses requests

Internet surfing.

Due to errors in some DNS implementations, the DNS resolver may send several requests for the same address to the server rapidly [DOK92]. We consider a request for a single address within a single second from the same source as a suspect request. We cannot state that a suspect request is invalid, as a host that does not cache DNS data will correctly generate them. Of the requests in the data collected from the *umich.edu* domain, 17% were suspect. In the event that some or all of these requests are erroneous, our results are still valid. The existence of these spurious requests would sightly increase the locality of requests, resulting in higher cache hit rates. We make no definitive statements about absolute cache hit rates. Some applications append an additional local domain name onto a request which contains a fully resolvable name. These requests will never be resolved, and were thus removed. We found in and removed from our traces 5% of these unresolvable requests.

In our simulation, we interpret each DNS request as a corresponding certificate request. Using the name server configuration information, we map all hostnames for which the server acts as an authority into a simulated enterprise. As defined by the cache replacement policy, caches are maintained for each simulated host. In the simulation, information about the cache hit rate and size are recorded over the simulated time.

In the remainder of this section, unless otherwise noted, all reported results are from the *umich.edu* traces.

3.3 Request Locality

The performance of our system is chiefly dependent on the locality of certificate requests. In a system where no locality can be observed, a caching mechanism is of no use. In Figure 5, we show the distributions of enterprise local and external requests for the *umich.edu* domain trace. The figure shows a high degree of locality for enterprise local requests. More than 76% of the requests where made for the same 100 addresses, while over 93% where for the same 1000. We can attribute this locality to several factors. All hosts within the domain access services provided by domain specific servers. Moreover, users tend to have a set of hosts on which they perform their daily tasks. Over time, end users access the same machines (e.g. fileserver), while never utilizing resources on the vast majority of available machines.

Enterprise external traffic shows a lower, but still significant, degree of locality. Figure 5 shows that over the measured period, over 40% of the requests where for the same 100 addresses. Similar to the enterprise local data, the vast majority of addresses are only requested a few times. Given this observed locality, we conclude that caching external certificates will provide performance improvements.

3.4 Enterprise Host Performance

In our simulation of enterprise hosts. We find that certificate caching at these hosts has little effect on the overall performance of the system. As depicted in Figure 6, the average cache occupancy per minute of the average host remains small during the entire simulation period. We note that host caches are empty the majority of time. During short periods of activity, the caches grow larger, but never exceeding 1 entry on the average, and empty when the entries time out. For longer timeouts, we see overlapping periods of activity for some machines, but the majority of machines remain idle with respect to certificate requests.

We next investigate the maximum resource usage at the enterprise hosts. We again find that even at the highest observed loads resource requirements at the hosts are within an acceptable range. In Figure 7, we show the maximum cache occupancy over all hosts within the enterprise for the duration of our trace. Under extreme load conditions, we see that the maximum cache size remains under 1000 entries, which for most systems is acceptable load.[3] Therefore, significant effort to optimize enterprise host caches will result in very little performance gain.

3.5 Enterprise Root Performance

To determine the expected load on the enterprise root, we analyze traces from the *umich.edu* domain. We also simulate small, medium, and large enterprises to determine that our results are independent of enterprise size. We find the load on the enterprise root to be acceptable. Reviewing the traces for the domain, we find that the DNS server processed an average of 78,420 requests/hour, with a high of 151,115 and a low of 33,395 requests per hour. The domain contains over 30,000 hosts. These request arrival rates can easily be processed by a dedicated high-end workstation. Should an enterprise root become overwhelmed by requests, the server may be replicated over a cluster of workstations.

[3]A typical size of a public-key certificate is 1KB.

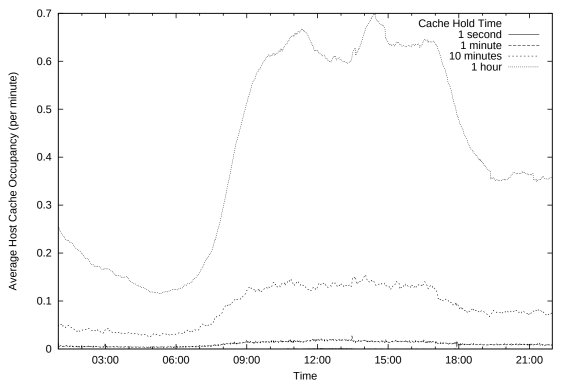

Figure 6: Average Enterprise Host Cache Occupancy (per minute)

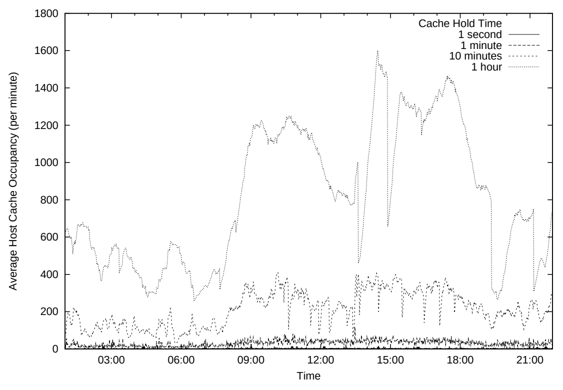

Figure 7: Maximum Enterprise Host Cache Occupancy (per minute)

In implementing the enterprise root cache, we selected the LRU (least recently used) and LFU (least frequently used) cache replacement policies to study. We use the standard definition for the LRU policy, but use a modified version of LFU. In our definition of LFU, we keep a permanent hit count for every certificate seen, whether it is present in the cache or not. The cache uses this counter in its entry replacement computation. This way, a certificate's access frequency is not reset when its hold time expires. We simulate the system using the DNS trace with a cache size of 1000 certificates. The results of this simulation show that certificate hold times of less than 10 minutes never cause the cache to become full. Hence for certificate hold times less than 10 minutes, all the alternative caching policies perform the same. In Figure 8, we show the performance of the different caching policies for certificate hold times between 10 minutes and 1 hour. In Figure 8 we include the performance of the group-based communication model. The group based model represents the best case scenario, where the sharing of caches among the enterprise hosts represents a cache of practically infinite size. A cache replacement policy that achieves hit rates at or near those of the group based communication model represents a viable alternative. In this case, we find the LFU cache replacement policy to perform marginally better than the LRU policy and to approach the performance of the group-based model. Note that the performance reported in Figure 8 is the hit rate of external certificate requests. Our key distribution architecture dictates that certificates of local hosts and trusted keyservers be permanently held in the enterprise root cache, and as such will always be readily available.

Given the marginal performance gain of the group based communication model over the enterprise-root cache with LFU replacement policy, the observation in the previous section that host caches are often empty, and the moderate number of requests per hour the root node can expect to see, we do not believe that the extra overhead of certificate sharing via the group communication model, with the attendant additional retrieval latency, is warranted.

We repeated the above simulations using all the traces available to us and found the performance to be comparable. Figure 9 shows the expected enterprise root cache hit rate as a function of cache hold time from 1 second to 1 hour. From this data, we state that the scalability of our proposed key distribution architecture is independent of enterprise size. We make no claim about the exact expected quantitative performance of our architecture, only that we expect to see a high degree of locality in certificate requests and acceptable request arrival rates.

3.6 Keyserver Performance

The dominant role keyservers play in our proposed architecture could potentially make them the bottleneck that limit the scalability of the architecture. We use traces from an authoritative server for the *us* domain to estimate the load on a typical keyserver. In this trace, we see arrival rates of between 19,787 and 111,038 requests/hour, with an average of 52,558 requests. Given the state of high end workstations, a dedicated host will be able to serve loads considerably larger than those indicated by this trace. Should keyservers become overwhelmed by requests, it could be replicated over a cluster of workstations to alleviate the problem.

In a series of tests, we seek to determine the effects of enterprise caching on keyserver performance. Our simulations shows that certificate hold time at each enterprise has a significant effect on request arrivals at the keyserver. Certificate hold times over 5 minutes at each enterprise reduce the load on the keyserver by as much as 60%. Using the *us* domain trace, we simulate a keyserver and the associated requesting enterprises. Because the CIDR blocks [FLYV93] of the requesters in our trace were unavailable, we execute two simulations. In the first simulation, we treat all addresses as Class-C addresses, using the first 24 bits of their address. If two addresses have the same first 24 bits, they are deemed to be in the same enterprise. In the second simulation, we repeat the simulation using only the first 16 bits: a Class-B addresses. Because real networks encompass more than one Class C, but less than

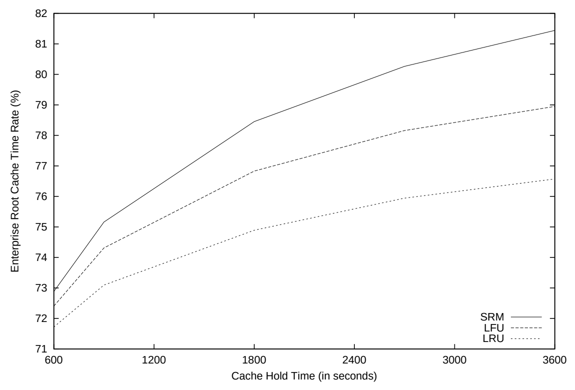

Figure 8: Enterprise Root Caching Policy Performance

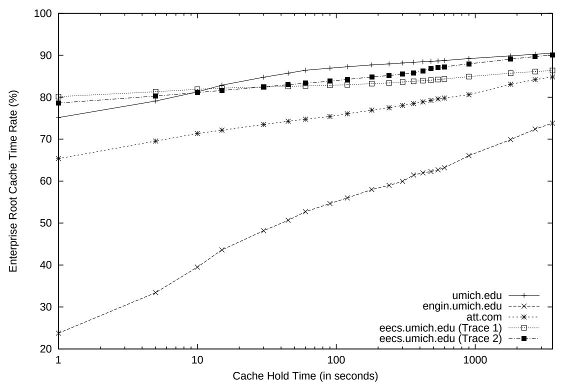

Figure 9: Enterprise Cache Hit Rate - hit rate for requests for certificates external to each simulated enterprise. The tests were performed under the enterprise root communication model using a 100 certificate LFU cache.

one Class B, addresses, the first simulation reports worse performance than reality and the second better. The results from these two simulation, shown in Figure 10, are very similar, and give us a tight bound on expected real performance. The figure shows keyserver residual load defined as a percentage of total requests in the original trace that are not served by enterprise root cache. For very short cache hold times (1 second), 20% of the requests will be serviced by a cache internal to the enterprise. These dramatic improvements continue until cache hold times reach about 5 minutes, where only 40% of the original requests are forwarded to the keyserver. Certificate hold times longer than 5 minutes do little to improve this performance, consistent with the enterprise root cache performance as a function of certificate hold times presented in Figure 9.

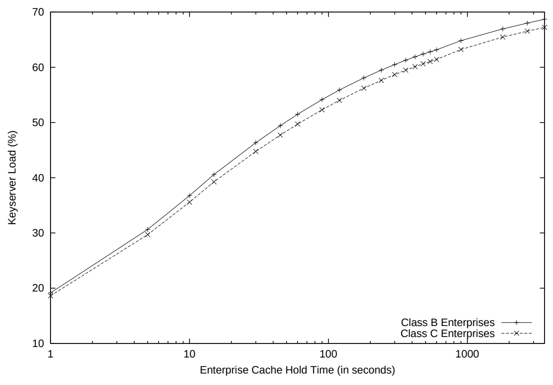

Figure 10: Keyserver load - simulated with enterprises modeled from class B and class C addresses in the DNS trace.

4 Related Work

There has been significant investigation of key distribution both in the standards bodies and in industry. We classify the resultant key distribution systems into three classes: trusted third party, hierarchical, and *ad-hoc*.

Trusted third party systems such as Kerberos [NT94, SNS88] rely on secrets shared with local trusted athorities. In larger networks, the authorities collaborate or form hierarchies to allow intra-domain secure communication. The Kerberos system is a symmetric (private key) system used to provide authentication in a network environment. Before accessing network services, a user provides a password known only to the user and the Kerberos system. After being authenticated, the user retreives tickets which are presented to services. These tickets are used to bootstrap secure communication between the service and the user. The Kerberos services act as a third party in which both end-points (entities) trust.

In strict hierarchies, such as the Privacy Enhanced Mail [Ken93], X.500 Directory [], Public Key Infrastructure [Cho94], and DNSsec [Eas98] services form a rooted hierarchy under which all secruity is ultimately gaurenteed by the root authority. The Privacy Enhanced Mail (PEM) Certificate Hierarchy defines a design for certificate (public key) distribution to be used by all entities on the Internet. As described in RFC 1422, the authors propose a hierarchy which contains a tree of certification authorities rooted at the Internet Policy Registration Authority (IPRA). Each node (Certification Authority) in the tree is classified by policies implemented by the administrators. Intended to increase end user confidence in the authorities, each classification has a set of social and physical requirements. These requirements define the administrative practices and hardware environment in which the authority operates. The signers for the certificates are defined by their position in the tree, whose path within the tree is required to be unique.

Extentions to the existing DNS service that provide secure distribution of public key certificates are specified in [Eas98]. The service, called DNSsec, introduces new resource records containing public key certificates. In retrieving certificates, DNSsec uses the same distribution hierarchy and communication channel as DNS. Revocation of certificates is similar to those found in KDH. Each certificate is distributed with a TTL that specifies how long the certificate should be cached. Note that the TTL is controlled by the certificate provider, and not the requestor, as in KDH. Revocation is handled by the removal of the revoked certificate from the authoritative DNS server.

Ad-hoc distribution services do not specify an architecture or service model for distribution. Users are free to exchange certificates in any way they wish. The public key certificate based Pretty Good Privacy [Zim94] (PGP) system defines a key signing scheme based on "webs of trust". In PGP, public key certificates are distributed insecurely. Certificates are digitally signed serially resulting in an ordered list of signatures attached to the original certificate. By convention, a user will only sign a certificate if they trust the source from which it was received. Therefore, when a certificate is received from a trusted source (such as a network administrator), their signature should provide assurance that the certificate is valid.

The X.509 [Rec93] specification describes both the format of public key certificates and Certificate Revocation Lists (CRL). The certificate revocation scheme used in X.509 describes the periodic publication of CRLs. When a user wishes to verify the authenticity of the certificate, she checks a recently published list for the identities of the certificate. If it is not on the list, the certificates are assumed to be valid.

The IETF Internet Public-Key Infrastructure working group (PKIX) outlines version 3 of the X.509 specification in [HFPS98]. The specification adds several new fields in the certificates, as well as propose a new mechanism for limiting the length of the certification path of hierarchical systems. Directed by strictly defined policies, certificates may be retrieved from peer authorities without consulting the hierarchy. Unfortuneatly this requires the peer authories establish a level of trust prior to certificate retreival.

5 Conclusions

In this paper we have defined a certificate distribution hierarchy that attempts to avoid the limitations found in existing designs. We propose a solution where users are free to select some set of authorities that they trust. By polling a subset of selected authorities, the end user may increase their confidence that received certificates are authentic. We also avoid the problems inherent to loosely connected or "web of trust" systems by stipulating that a received certificate has only been signed by authorities trusted by the end user.

The design of our system attempts to de-centralize the signing of certificates from a singular authority. In doing this, we enable the scalability and flexibility needed for large networks. By managing the certificates close to the source and by aggregating requests within the local environment, we attempt to limit the burden placed on network resources. Finally, users need only trust the practices of the administrators of their local environment.

In the security community, key revocation is widely accepted as a difficult problem. Existing design use revocation lists that are distributed to all interested parties. This solution does not scale to millions of users, and is subject to many denial of service attacks. We address this problem by limiting the hold time of certificates. We follow the example used in the DNS protocol, by advertising only correct keying material. End users bound the time in which they are vulnerable to compromised certificates by setting the hold time for received certificates. Our architecture can accommodate much shorter bounds on the window of vulnerability than systems described in the X.509 specification [Rec93], which use certificate revocation lists to advertise explicitly that keys have been revoked.

We evaluate the scalability and performance of our proposed key distribution hierarchy by using traces of Domain Name Service traffic. We argue that characteristics of DNS traffic mirror those of future certificate hierarchies. Our analysis find that the proposed architecture perform well under the DNS traffic patterns. The locality of requests found in the DNS traces show that our caching mechanisms will be effective for various domain sizes.

Acknowledgements

We wish to thank the following people for assistance in collecting DNS traces. Paul Kiley and Andrew Inman helped in the collection of DNS traces from the *engin.umich.edu* and *umich.edu* domains, respectively. Ramón Cáceres assisted in the collection of DNS traces from the *AT&T* domain. We also wish to thank Deborah Estrin and Peter Danzig for allowing us to collect DNS traces from the *us* domain.

References

[BAN90] M. Burrows, M. Abadi, and R.M. Needham. A Logic of Authentication. *ACM Transactions on Computer Systems*, 8, February 1990.

[Cho94] S. Chokhani. Towards a National Public Key Infrastructure. *IEEE Communications Magazine*, pages 70–74, September 1994.

[DH76] W. Diffie and M.E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976.

[DOK92] Peter B. Danzig, Katia Obraczka, and Anant Kumar. An Analysis of Wide-Area Name Server Traffic. In *SIGCOMM 92'*, 1992.

[Eas98] D. Eastlake. Internet, Draft, Domain Name System Security Extensions. *DNS Security Working Group*, March 1998.

[EK99] D. Eastlake and C. Kaufman. RFC 2065, Domain Name System Security Extensions. *RFC 2065, Internet Network Working Group*, January 1999.

[Flo95] S. Floyd. A Reliable Multicast Framework for Light-Weight Sessions and Application Level Framing. In *Proceedings of SIGCOMM '95*, pages 342–356, August 1995.

[FLYV93] V. Fuller, T. Li, J.I. Yu, and K. Varadhan. Classless Inter-Domain Routing (CIDR): An Address Assignment and Aggregation Strategy. *RFC 1519, Internet Network Working Group*, September 1993.

[GS91] Jim Grey and Daniel P. Siewiorek. High-Availability Computer Systems. *IEEE Computer*, 24(9):39–48, September 1991.

[HFPS98] R. Housley, W. Ford, W. Polk, and D. Solo. Internet, Draft, Internet Public Key Infrastructure. *PKIX Working Group*, March 1998.

[Ken93] S. Kent. RFC 1422, Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management. *RFC 1422, Internet Network Working Group*, February 1993.

[KPS95] Charlie Kaufman, Radia Perlman, and Mike Speciner. *Network Security, Private Communication in a Public World*. Prentice Hall, Englewood Cliffs, New Jersey, 1995.

[Moc87a] P. Mockapetris. Domain Names - Concepts and Facilities. *RFC 1034, Internet Network Working Group*, November 1987.

[Moc87b] P. Mockapetris. Domain Names - Implementation and Specification. *RFC 1035, Internet Network Working Group*, November 1987.

[NS78] R.M. Needham and M.D. Schroeder. Using Encryption for Authentication in Large Networks of Computers. *Communications of the ACM*, 21(12):993–999, December 1978.

[NT94] B. C. Neuman and T. Ts'o. Kerberos: An authentication service for computer networks. *IEEE Communications*, pages 33–38, September 1994.

[PF95] V. Paxson and S. Floyd. Wide-Area Traffic: The Failure of Poisson Modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, June 1995.

[Rec93] Recommendation X.509. The Directory: Authentication framework. *Information technology - Open Systems Interconnection*, November 1993.

[RSA78] R.L. Rivest, A. Shamir, and L.M. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[RSA92] RSA Laboratories. *RSAREF(TM): A Cryptographic Toolkit for Privacy-Enhanced Mail*. RSA Laboratories, Redwood City, CA, 1992.

[Sch96] Bruce Schneier. *Applied Cryptography*. John Wiley & Sons, Inc., New York, Chichester, Brisbane, Toronto, Singapore, second edition, 1996.

[SNS88] J. G. Steiner, B. C. Neuman, and J. J. Schiller. Kerberos: An authentication service for open network systems. In *Proceedings of the Usenix Conference*, pages 191–202, 1988.

[Zim94] P. Zimmermann. PGP user's guide. Distributed by the Massachusetts Institute of Technology, May 1994.