Layering Public Key Distribution Over Secure DNS using Authenticated Delegation [*]

John P. Jones, Daniel F. Berger, Chinya V. Ravishankar
Department of Computer Science & Engineering
University of California, Riverside
Riverside, CA, 92521
{jjones,dberger,ravi}@cs.ucr.edu

Abstract

We present the Internet Key Service (IKS), a distributed architecture for authenticated distribution of public keys, layered on Secure DNS (DNSSEC). Clients use DNSSEC to securely discover the identities of the relevant IKS servers, and send key lookup or management requests directly to these servers using a special-purpose protocol. Clients authenticate keys retrieved from IKS servers using key commitments published in DNSSEC. IKS derives its authentication authority from the authority DNS domains have over Internet names. The IKS architecture is loosely coupled with DNS to minimize overhead on DNS servers. We also present RIKS, a prototype IKS implementation.

1. Introduction

Digital communication has become pervasive, but there are few guarantees that such communications are secure and private. Indeed, security and privacy threats, long seen as hypothetical, are already real; in 2004, for the first time ever, an arrest was publicly acknowledged as having resulted from passive email monitoring [2].

Though cryptographic techniques exist that can address these concerns, no infrastructure is available to facilitate their use by a variety of applications, and across the Internet. Cryptography has been most successfully deployed in protocols where a clear client-server relationship exists, such as Secure Socket Layer/Transport Layer Security (SSL/TLS) [22, 13], and Secure Shell (SSH) [50]. While proposals exist for securing less hierarchical applications, such as the Privacy Enhanced Mail (PEM) [32], and Secure Multimedia Internet Mail Extensions (S/MIME) [29] specifications for securing email, they have not been widely adopted.

1.1. The Internet Key Service

We focus on a capability crucial to pervasive adoption of cryptography: *simple, scalable, authenticated public key distribution.* We present the Internet Key Service (IKS), a practical and deployable architecture for providing application-independent public key distribution layered on top of Secure DNS (DNSSEC). Our approach is flexible and extensible, and can store and serve a variety of key types in support of a variety of applications.

The Internet Key Service bases its key-authentication authority on DNS's authority to manage Internet names. This is a significant feature, since all global names encountered on the Internet, regardless of their syntax or structure, ultimately represent resources or entities that are only addressible over the network as a part of the DNS namespace. A DNS domain has naming authority over the objects that belong to it so, in this sense at least, all names are ultimately DNS names.

The DNS namespace is the Internet-wide standard for defining who has control over which names. IKS is loosely coupled to DNS, so that it can provide specialized key distribution protocols without requiring changes to or imposing significant overhead on DNS.

The remainder of this document is organized as follows: Section 2 provides necessary background. Section 3 briefly surveys related work. Section 4 gives a high-level view of our proposed solution. Section 5 delves deeper into the protocol design. Section 6 presents the Riverside Internet Key Server (RIKS) our proof-of-concept implementation of the IKS. Finally, Section 7 summarizes our findings, and points out avenues for future exploration.

[*]This work was supported in part by the Defense Advanced Projects Research Agency under contract F30602-01-2-0536.

2. Background

We assume familiarity with asymmetric (public key) cryptography, digital signatures and one-way hash functions. Readers unfamiliar with these topics are encouraged to consult a cryptography text, such as [42].

2.1. Key Authentication

Key authentication is the process of validating the binding of a cryptographic key to a named entity. Public key cryptosystems simplify, but do not solve, the problem of key distribution, since public keys must be authenticated to prevent impersonation and man-in-the-middle attacks. The most widely used approaches for solving the key authentication problem are the *certifying authority* model, exemplified by SSL/TLS, and the *web-of-trust* model, exemplified by Pretty Good Privacy (PGP).

Certifying Authorities. The certifying authority (CA) model assumes a small number of highly trusted individuals or organizations. Each key-identity binding must be certified by one of these trusted entities. Certificate verification requires the certifier's public key to first be authenticated. In practice, a small set of *root certificates*, which are public keys for various recognized certifying authorities, are typically preloaded into the cryptographic application.

Webs Of Trust. The web-of-trust model, relies on peers to vouch for the validity and trustworthiness of other peers. An unfamiliar key is accompanied by affirmations (digital signatures) from a set of community members who assert that the provided key is associated with the claimed identity. A recipient accepts the key only upon receiving enough verifiable affirmations from individuals that they trust.

IKS follows the certifying authority model; the IKS server for a domain acts as a CA for that domain and its public key can be authenticated by its key commitment published via DNSSEC.

2.2. The Domain Name System (DNS)

The Domain Name System (DNS) [36] is the most effective and widely-used mechanism for name registration and resolution on the Internet, and is a critical component of the Internet infrastructure. DNS names are assigned from a hierarchical namespace, and organizations are granted control over a sub-tree rooted at the domain they have registered. The DNS top-level domains (e.g. *.com*, *.org*, *.edu*, *.us*, *.uk*) are administered by ICANN. Domain administrators manage DNS servers to provide authoritative answers to queries regarding the domain and to participate in resolving DNS queries for clients belonging to the domain.

Security was not a primary consideration during the design and implementation of DNS. Its security shortcomings were first discussed in [7, 47]. The Internet Engineering Task Force (IETF) launched the DNSSEC effort in 1993 to secure DNS. Presently, the DNSSEC working group proposal is nearing operational readiness, bringing with it the promise of a trustworthy name service.

2.3. DNSSEC Overview

DNSSEC is a collection of proposals for securing the data stored in DNS. Using cryptographic techniques, responses can be strongly authenticated, greatly reducing the potential for abuse present in the current DNS. An IETF draft [5] enumerates the threats DNSSEC is intended to guard against. We focus here on the portions of DNSSEC relevant to our work. A detailed overview appears in [3].

Zone Signing. A DNSSEC-enabled DNS server responsible for a given domain (called a *zone*) signs the resource records comprising the zone with a public/private key pair bound to that zone, and delivers those signatures to querying clients. These Resource Record SIGnatures are stored in a new DNS record type, *RRSIG*, which contains a signature that authenticates a specific named set of resource records (*RRSet*). Each named resource in a secured DNS zone will have at least one associated *RRSIG* record.

DNSSEC responds to a query from a DNSSEC-enabled client with the DNS record for the name specified, along with the associated *RRSIG* record. The client obtains the public key associated with the zone and verifies the provided signature. If the signature is valid, the client can trust that the response was provided by the authoritative source.

Key Distribution in DNSSEC. To verify signatures, the client must be either statically configured with the public key for the queried zone (the zone key), or be able to obtain and authenticate it. To facilitate distribution of zone keys, DNSSEC defines a *DNSKEY* resource record type.

A DNS client queries for a zone key in the same way it queries for any other DNS record type. To authenticate the retrieved key, the *DNSKEY* record must be signed by a key which the client has previously authenticated, typically the key of the parent domain. By recursively requesting keys and moving up the DNS hierarchy, the client will either find a trusted key, or exhaust the name space without doing so, causing the key authentication attempt to fail. (This description is conceptually sufficient, but not technically precise. Full details are in [27].)

DNSSEC Implementation Status. DNSSEC has recently matured into an implementable system. An IETF draft exists that updates RFC 2535 and details the DNS protocol changes required to support DNSSEC [4]. A DNSSEC deployment working group has been formed with support of

NIST and ICANN. Consensus is growing that DNSSEC is largely ready for deployment, and that 2006 may see the beginnings of wide-spread adoption.

2.4. Barriers to Distributing Keys in DNS

Unfortunately, DNSSEC does not generally solve authenticated key distribution. The *KEY* record was originally intended to store various key types, including application keys [14]. This decision was explicitly reversed due to scalability concerns, query interface limitations, and administrative authority mismatches [33].

Scalability. Proposals to house per-user information in DNS did not anticipate that the growth in Internet user population would far surpass the growth in DNS-registered host systems. Estimates for 2004 suggest about 945 million users [11], compared with 230 million hosts [30].

Adding DNSSEC signature records to a zone increases the size of the zone data by a factor of 8 or 9 [24], and adding per-user keys and their signatures would further increase the size of the zone data. Finally, DNS has been designed and optimized for very small (∼300 B) query/response exchanges. Returning key data and associated signatures (∼1.2 KB) in DNS responses would significantly increase network load.

Query Interface. The DNS query interface does not match the requirements of an application seeking authenticated key distribution. Different types of keys stored in *KEY* records were to be differentiated by subtype, so that a single named entity may have multiple key records, each storing a different type of key. Unfortunately, the DNS resolver interface does not support query by subtype, so the client was forced to retrieve all key records present for the named entity before sifting through the results for the "right one." Since DNSSEC internally requires keys retrieved from foreign servers, this affected not only applications but the efficiency of the name service itself.

Administrative Authority. DNS data tends to change slowly and is under the control of a domain administrator. Allowing users some level of direct control over their keys would violate the existing administrative model. Supporting dynamic DNS update in the context of DNSSEC is difficult in general; RFC 3007 discusses it in detail and several researchers have contributed solutions [20, 48].

3. Related Work

Here we briefly survey previous approaches to key distribution, from application-specific to general approaches.

3.1. In-Band Key Transmission

A common approach to key distribution is to relegate it to the communication protocol. The SSH and SSL/TLS protocols both transmit the necessary keying information during connection setup, but use different authentication methods.

Secure Shell (SSH). SSH performs initial key authentication by asking the user to certify the key-host association. A hash of the public key (a *key fingerprint*) is then stored locally. Subsequent connections use this stored fingerprint to authenticate known hosts without further user intervention.

This approach assumes that the end-user will know the appropriate key fingerprint during initial connection setup. While it limits the window for a successful attack to the initial connection, it does not eliminate the threat. This is generally an acceptable level of risk mitigation when trust relationships are fairly static (users tend to repeatedly connect to the same small set of hosts). However, this sort of manual, out-of-band, process is not viable when the trust relationships are more dynamic (i.e. end-user to end-user communication).

Secure Socket Layer (SSL/TLS). SSL/TLS uses the certifying authority model; the connecting client is provided with the server's certificate, signed by one or more CAs. Clients (such as web browsers) are preconfigured with a number of "root certificates," which are public keys of trusted CAs. If the certificate provided by the server has been signed by a statically known certifying authority, the connection is established without user intervention.

3.2. Dedicated Key Distribution Services

Another approach to key distribution is to deploy a dedicated distributed service to handle the registration and query of public keys. Several proposals have been made, mainly differing on how keys are named and bound to individuals, how clients verify responses, and how servers distribute the responsibility of key distribution.

PGP/GPG. The MIT Pretty Good Privacy (PGP) key server is perhaps the best known dedicated key distribution service. PGP and Gnu Privacy Guard (GPG) [25] support locating and publishing keys via the PGP key-server.

SDSI/SPKI. The Simple Distributed Security Infrastructure (SDSI) [38] was intended as an integrated solution to authentication and authorization based on capabilities. This proposal has subsequently been incorporated into the IETF's Simple Public Key Infrastructure (SPKI) working group's proposal [19]. In SPKI certificates bind specific authorizations to keys. Names in SPKI can be assigned to keys and can exist in a local namespace or rooted in a global namespace such as DNS.

SPKI effectively switches from a model that separates authentication from authorization to a model that performs these functions jointly. This pushes the burden of key distribution onto applications making authorization choices about the resources they control. These requirements have significantly hindered the further development and deployment of SPKI.

Scalable Key Distribution Hierarchy. McDaniel and Jamin [34] describe a scheme for a hierarchical set of certificate servers similar in capabilities to the certification authority requirements outlined in the Privacy Enhanced Email (PEM) specification [32]. The authors describe their design, which is based on a well-meshed trust graph and is not directly related to the DNS namespace, and examine its behavior under hypothesized load. They do not discuss operational issues such as off-line signing keys and heterogeneous keys.

3.3. Distribution by Directory Service

Many proposals have chosen to incorporate key distribution into existing directory services.

X.500, LDAP. ISO and CCITT maintain recommendations for distributed replicable directory services under the name X.500 [31]. Clients typically access these directories using the LDAP protocol defined in RFC 1487. Configuration and maintenance of X.500/LDAP directories is perceived as difficult and complex. Standard schemas exist for various object types, including X.509 certificates [9], but are not always used and LDAP is often forbidden across network boundaries, creating disconnected islands of information.

Most damagingly, X.500 complexity is exposed beyond implementor and administrator. Users searching an X.500/LDAP directory must specify values for unfamiliar terms such as "Search Base" and "Search Scope." Correct values are required to obtain useful search results, and most tools provide little guidance. X.500/LDAP has not emerged as a practical Internet-wide key distribution tool.

DNS. Efforts have been made to standardize storing keys of various types [15, 18, 16, 17, 41] and X.509 certificates [18] within DNS. Yahoo! has submitted an IETF draft [12] that describes using DNS to distribute public keys for authenticating email delivery.

The FreeS/WAN Project [21], an open source IPSec implementation, supports "opportunistic encryption;" by automatically retrieving host keys from DNS, end-to-end IPSec encryption can be setup without user intervention. While the FreeS/WAN solution made retrieving keys from DNS invisible, it did not address key publication.

In [23], Galvin presented an overview of DNSSEC and briefly discussed the potential for using DNSSEC to distribute end-user public keys. In a subsequent RFC [6] the author describes a DNS key exchange record type with semantics similar to DNS mail exchange records. Though focus was on IPSec, the author briefly describes the potential for this mechanism to delegate authority to a more general key distribution center.

3.4. Identity Based Encryption (IBE)

Identity-based cryptography addresses the key distribution problem by allowing a sender to directly derive a public key from a recipient's name. Each recipient obtains its secret key from a trusted key generator, which generates this private key from the receiver's name, public system parameters, and a system secret. Since the key generator knows all private keys, this system implies key escrow. The work in [44] describes a domain-level key-distribution scheme using the identity based encryption scheme of Boneh and Franklin [10]. Requiring key escrow is not generally acceptable and even careful implementations carry significant risks [1].

4. The Internet Key Service

Instead of placing cryptographic keys directly in DNS, as previously proposed, we use DNSSEC for *authenticated delegation*. This approach imposes minimal overhead on DNS. As we argued in Section 1.1, DNS's role in naming is a fundamental aspect of the Internet, so any mechanism to bind keys to named entities on the Internet must derive its authority from DNS. All names addressible over the Internet are ultimately reducible to DNS names, and all authority to bind names to objects must ultimately derive from the authority DNS has over names. This basic observation drives our design of IKS.

Previous proposals for authenticated key distribution in the Internet have failed either because they were unable to root their authentication mechanisms in DNS, or because they used DNS directly to manage keys, leading to the problems discussed in Section 2.4.

The Internet Key Service addresses these problems by using DNSSEC to securely delegate a part of DNS's authority over name resolution to a specialized service designed to meet the requirements of authenticated key distribution. No natural secure delegation mechanism existed for the Internet prior to DNSSEC. The imminent deployment of DNSSEC has made the key distribution problem tractable.

4.1. IKS Overview

IKS allows public keys to be registered to any entity that can be assigned a DNS name, such as a host, user, or service port. These keys are stored in and managed by IKS servers, which may be discovered securely through DNSSEC. In

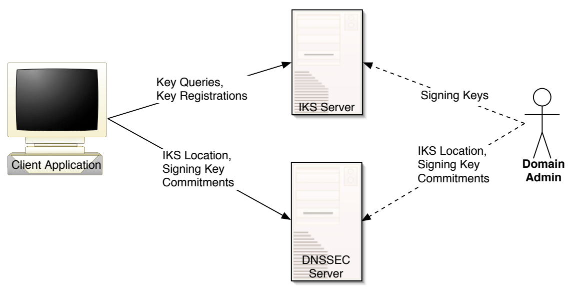

Figure 1. IKS Architecture: Naming and authentication authority is placed in DNS and IKS, which clients use to resolve names securely.

turn, IKS servers handle key registration and lookup requests for names belonging to a specific DNS domain in an authenticated manner.

A client wishing to register or retrieve a public key for a DNS named entity first uses DNSSEC to discover and verify the identity of an IKS server responsible for that domain. The client then sends the key-registration or lookup request to this server using the IKS protocol, and authenticates the server's responses using key commitments placed in DNS. One can now guarantee that the requests have been processed by a server to which key management authority for the domain was properly delegated. These keys may now be used for establishing secure communications channels or validating digital signatures with confidence. Figure 1 shows high-level information flows in the Internet Key Service architecture.

Since each domain administers its own IKS server (or delegates this task to a trusted organization) there are no communications between IKS servers for different domains. IKS operation is thus structured differently from that of DNS. With authenticated delegation, we avoid the bottlenecks that can arise in DNS-like hierarchies.

4.2. Design Requirements/Constraints

We have designed IKS to meet the following requirements, which we feel are essential for a succesful secure key distribution service for the Internet.

Authority: The service's authority to bind DNS names to keys must derive from DNS's naming authority.

Scalability: The service must not result in a substantial increase in load on DNS.

Compatibility: Changes to DNS must be avoided. The service must not create new resource record types.

Flexibility: Domain- and application- specific mechanisms for authenticating users during key registration must be supported.

Efficiency: The number of required messages must be small for performance and reliability reasons.

Generality: Service mechanisms must be application-independent. The key service must honor the end-to-end principle, providing service to any application.

Security: To protect the private signing keys, key registration and query servers should have limited contact with the system's key-signing keys.

Consistency: The key-authentication guarantees expected by an end-user (the "user-model") must be consistent with the guarantees actually provided by the system (the "system model").

4.3. IKS Architecture

Our use of DNSSEC for *authenticated delegation* provides both a secure hand-off between DNS and IKS servers, and a mechanism for authenticating server responses. Each participating DNS domain delegates to one or more IKS servers the responsibility for handling IKS requests. This delegation is accomplished by adding resource records to the DNS zone, and is under the domain administrator's direct control. There is no implicit delegation in IKS; a domain that does not explicitly publish delegation records is choosing not to participate in IKS.

To allow clients to validate responses, IKS servers sign all keys returned with a named key-signing key (KSK), the public half of which is committed in DNSSEC. This commitment can be securely retrieved and used in verification.

Registration acknowledgments and query failure responses are signed on-line with a response-signing key (RSK), the public half of which can be retrieved from the IKS. Using the KSK for this purpose would require on-line use of its secret half, placing the KSK itself at risk.

Our scheme is conceptually independent of DNSSEC, relying only on the presence of some trustworthy name service; the impact of changes to DNSSEC standards on our proposal is likely to be minimal. Unlike previous attempts to distribute keys via DNS(SEC), our proposal avoids the three pitfalls enumerated in Section 2.4: poor scalability, poor query interface, and mismatch of administrative authority.

Scalability. By layering IKS over DNS, we decentralize key distribution without impeding scalability. The additional data placed in DNS (delegation records and KSK commitments) is negligible, and does not increase with the number of keys active in the system. The DNS overhead for discovering delegations and retrieving KSK commitments is comparable to that for resolving other services (say, the IP address of a website). IKS permits domain administrators wide latitude in distributing key management workloads, say, using a simple weighted load balancing across a group of IKS servers. Our design thus supports system scalability.

Query Interface. IKS uses a specialized query and registration protocol which provides the appropriate level of expressiveness for key registration and distribution. Clients can perform narrow searches, based on attributes such as key length, cryptographic algorithm, and key-container format, to discover keys suitable for their purposes.

Administrative Authority. Unlike proposals to distribute keys in DNS, IKS places minimal burden on DNS administrators, and does not cause rapid changes to DNS zone data. The delegation records for a given domain are static, and publication and revocation of key-signing keys are infrequent events, not driven by end-user behavior.

5. Protocol Overview

We introduce IKS by showing how to fetch and register keys. We briefly discuss key revocation in IKS.

5.1. Key Lookup

A name `N = (E,D)` consists of two components: the *entity* part `E` that designates the user, host, or communication endpoint, and the *domain* part `D`, which is a DNS domain name. A query specifies a name `N` and a set of key *selection criteria* `C`, and is processed as follows.

1. Query DNSSEC for the IKS server for domain `D`.
2. Send a key-query message to the IKS server for `N`.
3. IKS responds with metadata for all public keys registered for `N`, signed with a key-signing key (KSK).
4. Validate the IKS response as follows:
   (a) Fetch the KSK from the IKS server.
   (b) Request DNS for the commitment for this KSK. Validate the response using DNSSEC zone signatures.
   (c) Validate the KSK using this commitment.
   (d) Verify the IKS signature on the metadata for `N`'s keys using the KSK.
5. At the client end, the key metadata are processed, and keys matching the criteria `C` are identified. These keys are fetched from IKS and verified.

If `N` has no keys published in IKS the query server generates a negative response signed with a response-signing key (RSK) registered in IKS.

Trust Guarantees for IKS Responses. In accordance with the consistency constraint (Section 4.2) we explicitly state the guarantee made by IKS. A valid signature on a query response indicates that

1. The IKS server for domain `D` asserts that the key provided is bound to name `N` in domain `D`.
2. The administrators of domain `D` have verified, to the extent **they** see fit, that the key in question was registered in the IKS for domain `D` by the user or agent in control of name `N`.

IKS **offers no guarantees** about how verification was performed during key registration. IKS is decoupled from any standards or protocols for identity verification within domains. Clients receiving validated query responses from domain `D` decide for themselves what trust to place in `D`. IKS insulates the end-user from the complexities of key distribution, but does not pretend to make guarantees about the trustworthiness of domains. In all contexts, including IKS, one must not confuse authentication with trustworthiness.

5.2. Key Registration

Given a target name `N = (E,D)` and a public key `K`, registration proceeds as follows:

1. Query DNSSEC for the IKS server for domain `D`.
2. Send a key-registration message to the IKS server specifying entity `N`, the key `K`, relevant metadata (permitted uses, expiration date, and so on), and authentication information.
3. If authentication succeeds, and the registration is authorized, the server returns a success message signed by a named response-signing key (RSK). The client can authenticate this RSK and the response as it does for other key lookups.

If authentication fails, the registration server generates a signed response indicating acceptable methods of authentication to guide the client in completing this transaction. As with negative query responses these authentication failure responses are signed with a response-signing key (RSK).

Authentication During Registration. The flexibility constraint (Section 4.2) grants the registration servers latitude in performing authentication when keys are registered.

To ensure correct operation in domains using different authentication standards, we have recognized three different authentication mechanisms as appropriate for IKS. Each domain supports a subset of these mechanisms, as determined by its local security policies.

Username/Password: A registrant may authenticate to an IKS server using a shared secret, such as a username and password. An RSA encryption key is published in IKS, which registrants may use to securely send the secret to the server.

User Key Management: A client acting on behalf of a registrant `U` may authenticate its request by signing it with a special purpose key, which has been registered to `U` using IKS.

Third-Party Authentication: IKS supports other, possibly domain-specific, authentication methods by allowing the authentication of requests based on signatures using other certifying keys, again published via IKS. The private half of these keys can be distributed to a number of third-party authentication servers, which can implement arbitrary authentication protocols and use signatures with these keys to validate their authentication with the IKS server. These external authentication protocols are outside of the scope of IKS.

In practice, it is likely that the Username/Password and User Key Management authentication methods will be sufficient for the majority of IKS installations.

5.3. Key Revocation

When a key is registered in IKS an optional expiration time may be provided to allow for the graceful expiration of old keys. IKS also supports a simple key revocation mechanism for the exceptional case where a compromised key must be revoked before its expiration time. In such an event, the key's holder must detect the key compromise and submit an authenticated key revocation request to the IKS server. We note that mechanisms for detecting key compromises and timely revocation are application specific, and a generally open problem outside the scope of IKS. Once a key revocation request is authenticated and accepted by the IKS server, the key's IKS entry is updated and a signed key revocation response is published in IKS.

To reduce the dangers presented by compromised keys, IKS implementations must seek to minimize the delay between the acceptance of a key revocation request and the publication of the signed key revocation. Caching of IKS responses by clients reduces reduce IKS server loads, but complictes the challenge of key revocation, since clients are guaranteed to refresh their caches only upon key certificate expiration. IKS implementations may choose short key certificate lifetimes to mitigate this problem.

5.4. Locating an IKS Server

A fundamental issue is how a client identifies the IKS servers responsible for a domain. We use the existing *SRV* record type, intended for service location [28]. DNS *SRV* records are intended to allow clients to perform service discovery using DNS. As defined in RFC 1700, a client locates a server for service `S` running a protocol `P` in domain `D`, through a DNS query for `_S._P.D`. The response includes a list of `(host, port)` pairs, along with a priority for each matching record and a weight used to distribute load across servers of the same priority. To locate a server using DNS *SRV* records, a client must first know the service and protocol names. Since key lookups may be handled by a different set of servers than registration requests, we use two distinct service names; `ikqs`, for "Internet Key-Query Service", and `ikrs`, for "Internet Key-Registration Service,". These IKS services both utilize the TCP protocol.

The IKS for a domain need not actually be hosted in or by the owner of the domain. The domain administrator may delegate this function by adding the required *SRV* records. We see the potential for organizations, possibly existing certifying authorities who already have a good understanding of the operational security issues involved in key management, to offer IKS services to domain administrators.

5.5. Message Marshalling and Transport

We have chosen XML as the format for IKS messages to ensure compatibility with the dominant message format protocol and the dominant class of applications on the Internet. It is relatively straightforward for a client to parse and to generate the simple XML messages used in IKS.

Marshalling. The World Wide Web Consortium (W3C) has published a multi-part recommendation called SOAP (Simple Object Access Protocol) specifying an interoperable means of using XML to exchange structured and typed information in a distributed environment or application [26]. SOAP specifies message formatting, including the overall structure of the message as an XML document. A more complete overview of SOAP, with supporting examples, can be found in [35].

Transport. With the increasing popularity of the Web Services model of remote service invocation, HTTP is fast becoming the de facto standard transport protocols for remote procedure call. One of the primary reasons for this adoption is that HTTP is typically permitted through firewalls and across different administrative domains within an organization. This is reinforced by the relative simplicity of the protocol, as well as the availability of implementations.

Query Optimization. Due to the relative simplicity of query operations, and the need to optimize this common operation, we provide an optimized interface to lookup. Query requests are mapped, by the IKS client, into HTTP requests for static XML documents using a URL-safe encoding [39] of the queried object's name. The response XML format is similar to the SOAP formats used by the registration server sans the SOAP envelope.

5.6. Authenticating Key-Signing Keys

As mentioned in Section 4.3, query response messages are signed by one of the domain's key-signing keys (KSK). To verify this signature, the client must fetch the KSK from IKS as well as its commitment from DNSSEC.

A named KSK `K` for domain `D` must be a DSA key published in IKS. The hash of the key is stored in a DNS text record with the name `sha1_K.D`. This record contains a hexadecimal representation of the SHA-1 hash. (Recent cryptanalytic results against SHA-1 mandate re-evaluating the use of SHA-1 as a secure hash function [8, 49].)

To verify the results of a query, the client first obtains the KSK by requesting the key named in the query response Subsequently, the client retrieves the commitment of that KSK from DNSSEC and confirms that the retrieved key matches the commitment. Finally, the KSK is used to verify the query results.

6. The Riverside Internet Key Server

We have built a prototype implementation of IKS, the Riverside Internet Key Server (RIKS). In this section, we will describe the issues, the design choices, and our preliminary experience with this system.

The RIKS server is composed of three components, one to handle query requests, one to handle registration and revocation requests, and a separate update process to generate the KSK-signed query responses. The components of the server communicate through a relational database.

We have identified three distinct signature generation strategies, which differ in the time at which the KSK is needed and which processes have access to it.

On-line: The key-registration handler signs keys with the KSK immediately upon their acceptance by the system.

On-demand: The key-lookup handler checks the database for a response object. If it exists and is signed, it is returned. Otherwise, it is immediately signed and returned to the requesting client.

Off-line: All signatures are generated by an off-line process that runs periodically. This method has the advantag that the key-signing key can be kept offline during operation.

RIKS currently supports only the off-line method of signature generation. However, it would be very easy to add the other signature methods, and for RIKS deployments to select one as a configuration option. RIKS is designed to make key lookups efficient; all valid keys are stored in a database as pre-signed XML responses. A lookup is simply a retrieval from the database. The update process periodically ensures that these responses are current. Figure 2 shows the current RIKS Architecture.

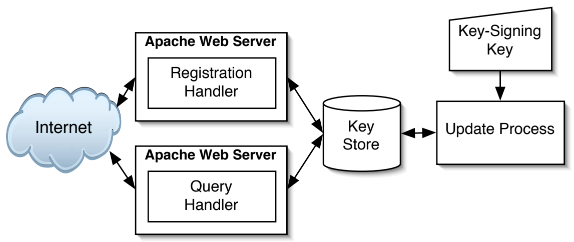

Figure 2. RIKS Architecture using off-line signature generation.

RIKS is implemented in Python, using the Zolera SOAP Infrastructure (ZSI) [40], and *mod_python* [46] to allow the Apache web server to host the server processes. The M2Crypto [43] wrapper provides Python access to the cryptographic functionality in the OpenSSL library. The server uses SQLite [45], an embeddable SQL'92 compliant RDBMS engine, and *pysqlite* [37], a Python interface layer, for underlying data storage.

6.1. Registration Handler

Key-registration and key-revocation requests are sent to the Registration handler. These requests must be authenticated and authorized before execution. The server's response is signed by a response-signing key (RSK), as confirmation to the client that its request was received.

6.2. Update Process

Before the effects of registration and revocation operations performed by the registration handler are made visible to querying clients, the corresponding signed key-query response messages must be generated by the update process, which is granted access to one of the domain's KSKs. Additionally, as query response messages expire, replacement signatures must be generated.

6.3. Query Handler

When a request for keys registered under a given name arrives, the query handler simply looks in the database for a pre-signed message, with this information already placed there by the update process. If no such object is found the query handler returns a failure response signed with the RSK.

6.4. Performance

Our current, proof-of-concept, RIKS implementation is not yet properly optimized, but we ran a series of tests to measure the registration, update, and query performance of our RIKS prototype. These tests were run on a single CPU (1.5 GHz Pentium 4M) laptop machine with 512 MB of RAM. The tests were run with a moderate-sized (∼300 MB) database, containing about 50,000 entries (10 keys registered to each user). Table 1 summarizes RIKS performance.

Operation	Registration	Query	Update
Transactions/sec	6.1	295	68

Table 1. RIKS Performance Summary

Our query handler handled 295 lookup requests/second. Key lookup performance will likely be similar to that of serving static web content, since IKS simply returns pre-generated query responses. Our design also allows RIKS to be parallelized to improve performance.

The registration handler was able to complete 6.1 registration requests per second. The bulk of its time was spent parsing incoming requests, serializing responses and sending them to the client. Approximately 12% of the registration handler's time was spent authenticating requests, storing the new keys in the database, and signing responses.

The update activity identifies keys that must be registered, re-signed, expired, or purged, and then process them. The update process took 70 seconds to identify entries requiring processing, with the database on disk. Once the database was loaded into memory this same operation took 2.5 seconds. After constructing this worklist, the update process completed generating and signing query responses at a rate of 68 per second.

A successful registration of a 1024-bit DSA key requires about 4 KB of SOAP messages to be sent between the client and server. XML query responses were about 1.8 KB each.

6.5. RIKS Client Library

Currently, the only complete client library available for RIKS is a Python module. While this client library is functional, it is not appropriate for inclusion in most client applications. We are re-implementing our client library in ANSI C, and expect this effort to be completed shortly.

We have taken measures to limit the complexity of the client library so that we can reasonably expect any cryptographically aware application to include it in order to publish and lookup keys in IKS. Aside from the cryptographic operations provided by the OpenSSL toolkit most IKS operations are handled by libraries included with languages such as Python, Java, C, and C++.

7. Conclusions & Future Work

Powerful cryptographic tools exist to address security and privacy concerns, but have not been widely used since no convenient infrastructure is available for authenticated key distribution. IKS is intended to accelerate the adoption of cryptographically-enabled applications. IKS is a simple, scalable public key distribution service, and its protocols have been designed specifically to meet the requirements of this domain, conforming to current industry best practices and standards for remote service location and invocation.

We rely on DNSSEC to provide authenticated delegation, while keeping the functional overhead of key distribution outside the critical DNS infrastructure. This strategy allows us to use the name service infrastructure to guarantee authenticity, while avoiding the scalability, efficiency, and administrative pitfalls of earlier DNS-based mechanisms. Furthermore, we use DNS names directly, and not a namespace orthogonal to it, facilitating its integration into the existing Internet infrastructure.

We have presented RIKS, the Riverside Internet Key Server, a prototype implementation of IKS. RIKS consists of approximately 4000 lines of Python code, and demonstrates performance adequate to justify confidence in our approach. The RIKS client library API provides a simple interface to IKS, making it easier to incorporate key authentication into existing collaborative tools.

Future Work. We hope to develop an IKS standard specification, to incorporate input from the community, and motivate deployment in tandem with DNSSEC. We will continue to improve RIKS performance, security, and manageability, and make it suitable for use in large ISPs.

To verify the ease with which existing applications can be extended to use IKS, we are planning the deployment of a secure application. While distributed applications, such as email and VoIP, will benefit most from IKS in the longer term, it should be straightforward to deploy IKS within a single domain, even with the current deployment status of DNSSEC. Centralized applications, including certain Instant Messaging applications, could easily be secured using IKS today.

As DNSSEC gains adoption and penetration, we believe IKS will facilitate authenticated public key distribution, improving the security of existing network applications and protocols, and enabling new developments.

In the future, when Alice must locate Bob's key, she can turn to IKS.

References

[1] H. Abelson, R. Anderson, S. Bellovin, J. Benaloh, M. Blaze, W. Diffie, J. Gilmore, P. Neumann, R. Rivest, J. Schiller, and

B. Schneier. The Risks of Key Recovery, Key Escrow, and Trusted Third-Party Encryption, 1998.

[2] D. Akin. Arrests key win for NSA hackers. The Globe And Mail, April 2004.

[3] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. DNS Security Introduction and Requirements. IETF draft: dnsext-dnssec-intro-09, February 2004.

[4] R. Arends, M. Larson, R. Austein, D. Massey, and S. Rose. Protocol Modifications for the DNS Security Extensions. IETF draft: dnsext-dnssec-protocol-06, May 2004. Updates RFC 2535.

[5] D. Atkins and R. Austein. Threat Analysis of the Domain Name System. IETF draft: dnsext-dns-threats-07, April 2004.

[6] R. Atkinson. Key Exchange Delegation Records for the DNS. RFC 2230, November 1997.

[7] S. Bellovin. Using the Domain Name System for System Break-ins. In *Proc. Fifth USENIX Security Symposium*, June 1995.

[8] E. Biham and R. Chen. Near-Collisions of SHA-0. In *Proc. of CRYPTO 04*, 2004.

[9] S. Boeyen, T. Howes, and P. Richard. Internet X.509 Public Key Infrastructure LDAPv2 Schema. RFC 2587, June 1999.

[10] D. Boneh and M. Franklin. Identity-based Encryption from the Weil Paring. In *Proc. of CRYPTO 01*, pages 213–229, 2001.

[11] ClickZ Stats. Population Explosion!, April 2004.

[12] M. Delany. Domain-based Email Authentication Using Public-Keys Advertised in the DNS (DomainKeys). IETF draft: delany-domainkeys-base-00, May 2004.

[13] T. Dierks and C. Allen. The TLS Protocol Version 1.0. RFC 2246, January 1999.

[14] D. Eastlake. Domain Name System Security Extensions. RFC 2535, March 1999.

[15] D. Eastlake. DSA KEYs and SIGs in the Domain Name System (DNS). RFC 2536, March 1999.

[16] D. Eastlake. Storage of Diffie-Hellman Keys in the Domain Name System (DNS). RFC 2539, March 1999.

[17] D. Eastlake. RSA/SHA-1 SIGs and RSA KEYs in the Domain Name System (DNS). RFC 3110, May 2001.

[18] D. Eastlake and O. Gudmunsdsson. Storing Certificates in the Domain Name System (DNS). RFC 2538, March 1999.

[19] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Yolen. SPKI Certificate Theory. RFC 2693, September 1999.

[20] P. Eronen. Applying Decentralized Trust Management to DNS Dynamic Updates. In *Proc. Third NordU/USENIX Conference*, 2001.

[21] FreeS/WAN Project. FreeS/WAN. `www.freeswan.org`.

[22] A. Freier, P. Karlton, and P. Kocher. The SSL Protocol Version 3.0, November 1996.

[23] J. Galvin. Public Key Distribution with Secure DNS. In *Proc. Sixth USENIX Security Symposium*, 1996.

[24] R. Gieben. DNSSEC in NL, January 2004.

[25] Gnu Privacy Guard Project. `www.gnupg.org`.

[26] M. Gudgin, M. Hadley, N. Mendelsohn, J. Moreau, and H. Nielsen. SOAP Version 1.2 Part 1: Messaging Framework, June 23.

[27] O. Gudmundsson. Delegation Signer (DS) Resource Record (RR). RFC 3658, November 2003.

[28] A. Gulbrandsen, P. Vixie, and L. Esibov. A DNS RR For Specifying the Location of Services (DNS SRV). RFC 2782, March 1999.

[29] IETF S/MIME Working Group. S/MIME Mail Security Charter.

[30] Internet Software Consortium. Internet Domain Survey Host Count, May 2004.

[31] ISO and CCITT, editors. *Recommendation X.500: The Directory: Overview of Concepts, Models and Services*. ITU, 1993.

[32] J. Linn. Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures. RFC 1421, February 1993.

[33] D. Massey and S. Rose. Limiting the Scope of the KEY Resource Record (RR). RFC 3445, December 2002.

[34] P. McDaniel and S. Jamin. A Scalable Key Distribution Hierarchy. Technical Report CSE-TR-366-98, E.E. & C.S. Department, University of Michigan, March 1998.

[35] N. Mitra. SOAP Version 1.2 Part 0: Primer, June 23.

[36] P. Mockapetris. Domain Names – Concepts and Facilities. RFC 882, November 1983. Superceded by RFC 1034.

[37] M. Owens and G. Häring. PySQLite. `pysqlite.sourceforge.net`.

[38] R. L. Rivest and B. Lampson. SDSI – A Simple Distributed Security Infrastructure, April 1996.

[39] E. S. Joseffson. The Base16, Base32, and Base64 Data Encodings. RFC 3548, July 2003.

[40] R. Salz. Zolera Soap Infrastructure. `pywebsvcs.sourceforge.net`.

[41] J. Schlyter and W. Griffin. Using DNS to Securely Publish SSH Key Fingerprints. IETF draft: secsh-dns-05, September 2003.

[42] B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, second edition, October 1995.

[43] N. P. Siong. M2Crypto. `sandbox.rulemaker.net/ngps/m2`.

[44] D. K. Smetters and G. Durfee. Domain-Based Administration of Identity-Based Cryptosystems for Secure Email and IPSEC. In *Proc. Twelfth USENIX Security Symposium*, pages 215–230, August 2003.

[45] SQLite.org. SQLite. `www.sqlite.org`.

[46] G. Trubetskoy. `mod_python`: Apache/Python Integration. `www.modpython.org`.

[47] P. Vixie. DNS and BIND Security Issues. In *Proc. Fifth USENIX Security Symposium*, June 1995.

[48] X. Wang, Y. Huang, Y. Desmedt, and D. Rine. Enabling Secure On-line DNS Dynamic Update. In *Proc. Annual Computer Security Applications Conference*, December 2000.

[49] X. Wang, Y. Yin, and H. Yu. Collision Search Attacks on SHA1, February 2005.

[50] T. Ylonen and D. Moffat. SSH Protocol Architecture. IETF draft: secsh-architecture-15, October 2003.