Accredited DomainKeys: A Service Architecture for Improved Email Validation[*]

Michael T. Goodrich
Department of Computer Science
University of California
Irvine, CA 92697
goodrich(at)acm.org

Roberto Tamassia
Department of Computer Science
Brown University
Providence, RI 02912
rt(at)cs.brown.edu

Danfeng Yao
Department of Computer Science
Brown University
Providence, RI 02912
dyao(at)cs.brown.edu

Abstract

We present an architecture called *Accredited DomainKeys*, which builds on the DomainKeys email authentication infrastructure to address the following questions:

- "Did the sender actually send this email?"
- "Is the sender of this email trustworthy?"

The proposed DomainKeys architecture already addresses the first question but not the second. Accredited DomainKeys strengthens the reliability of a positive answer to the first question and provides a mechanism to answer the second. In terms of infrastructure requirements, Accredited DomainKeys involves a modest additional use of DNS over the existing DomainKeys proposal. In addition, the specification of Accredited DomainKeys provides a mechanism for historical non-repudiation of email messages sent from a given domain, which is useful for the enforcement of acceptable usage policies. Several compliant implementations of Accredited DomainKeys are possible. This paper describes two implementations, one based on time-stamped signatures, and the other based on authenticated dictionaries and the secure transaction management system (STMS) architecture.

1 Introduction

As domain spoofing (a major contributor to spam, spim, phishing and malware) becomes a billion-dollar problem, several sender verification frameworks have been proposed for email [2, 3, 5, 8, 10, 16]. All such schemes rely on the existing DNS infrastructure to store authentication information on authorized email sending servers.

These schemes can be classified primarily as being either path-based or cryptographic. In path-based schemes, such as SPF [9, 15] and Sender ID [12], a domain administrator places into DNS records the IP addresses of authorized email sending servers for the domain. Thus, receiving email servers can check whether incoming email messages are being sent by an authorized server. Path-based schemes are very simple to implement and require minimal computing and communication overhead. However, they do not work well with forwarded email messages, since they are vulnerable to "email laundering" attacks (see, e.g., [13]).

In cryptographic schemes, such as DomainKeys [3], outgoing email messages are digitally signed by the sending server using public-key cryptography. In order to allow for signature verification by the receiving email server, the domain administrator places into DNS TXT records the public keys used to sign email messages sent by authorized email sending servers of the domain. Cryptographic schemes provide strong assurances of the sender domain and message integrity and are resilient to email laundering attacks. However, they require a small computational overhead for signing messages at the senders and verifying signatures of messages at the receivers.

Both the path-based and the cryptographic schemes, however, are only as strong as DNS itself, which has some known weaknesses. For example, DNS spoofing (malicious cache poisoning) is an attack where forged data is placed in the cache of the name servers. As a consequence, the security of DomainKeys can compromised. For instance, an attacker can generate a public/private key pair, and launch a DNS spoofing attack against a certain name server. The attack places the public-key under namespace example.net in the cache of the name server. The attacker then signs spam mes-

[*]This work was principally supported by a gift to Brown University from IAM Registry, Inc. The work of the first author was done primarily as a consultant to Brown University supported by this gift.

sages with his private key, and uses example.net as the sender domain. Unfortunately, the domain with the spoofed name server cannot identify the spams because the email signatures are verified against the wrong public key obtained from the poisoned DNS cache.

Furthermore, existing sender verification schemes are directed at answering the following question:

"Did the sender actually send this email?"

where "the sender" is the email domain that a message claims to be originated from. But answering this sender-domain-authentication question only provides the first part of an email assurance solution.

1.1 Sender Accreditation

An effective email assurance infrastructure should also answer the following question:

"Is the sender trustworthy?"

Identities in the digital world have no explicit correlations with behaviors. Knowing with certainty that an email comes from a domain that is otherwise unfamiliar does not provide much information about the reputation of the sender domain, or whether the sender domain can be trusted not to send spam or malware. Indeed, many spammers have been early and consistent adopters of path-based domain authentication mechanisms, such as SPF and Sender ID, as a way to cloak their emails in legitimacy [14]. Thus, if DomainKeys is widely-deployed only as an authentication technology, as it is currently proposed, we would naturally expect that spammers would be early adopters of DomainKeys as well. That is, performing domain authentication alone seems to invite spammer adoption, which does not address the trustworthiness question and may even discourage trustworthy senders from adopting domain authentication.

Thus, being able to answer the trustworthiness question for a sender domain provides recipients with crucial information about the credibility of message content. This is especially crucial to corporate and government communications and legitimate email senders from countries with a reputation for being spam origins.

Therefore, given the risks of decoupling authentication and accreditation, there is a strong need for the establishments of *accreditation bureaus* that maintain trusted *domain registries*, which are directories of domain names whose administrators have agreed to acceptable usage policies (AUPs) prohibiting spam, phishing, and other email abuses. This approach of coupling accreditation with sender domain authentication is advocated, for example, in the Lumos whitepaper [11], which details the kinds of statements that belong in an email AUP, such as the one accepted by the affiliated Email Service Provider Coalition. The Lumos whitepaper also describes in general terms the kinds of infrastructure requirements needed to enforce an email AUP. In a nutshell, the main idea is that domains adhering to such an AUP (allowing for probation for first offenses and minor infractions) would be allowed to keep their domain names in the registry, which is dynamically updated by the accreditation bureau. The challenge in establishing an infrastructure for domain accreditation, then, is to design a validation system that can quickly validate recipient emails and utilize low-overhead solutions for email senders and the accreditation bureaus.

Although the DNS spoofing attack described earlier can be prevented by widely deploying DNSSEC on the Internet, existing email authentication frameworks still cannot provide the trust assurance needed for email. Our proposed Accredited DomainKeys framework is resilient to DNS spoofing attacks and provides irrefutable trust assurance for mail recipients.

1.2 Summary of Contributions

In this paper, we present the *Accredited DomainKeys* framework that addresses both of the above authentication and accreditation questions. Accredited DomainKeys is a low-overhead service architecture for domain accreditation, which achieves its efficiency by extending the DomainKeys framework [3] to add an *accreditation seal* to emails. This seal is essentially a time-stamped micro-certificate, which provides a proof-of-membership of the sender's domain in an accreditation bureau's registry at a certain time and date. This seal provides additional evidence that a public key used to sign a given domain's outbound email was correct and valid, and therefore eliminates the security vulnerabilities caused by DNS spoofing attacks. More importantly, the seal shows that the administrator of this domain has agreed to an acceptable usage policy (AUP) that specifically prohibits spam and phishing. Thus, such an email carries a high level of trust; and if it violates that trust, the email and its accreditation seal can be used as nonrepudiable evidence of an AUP violation.

We also describe several possible implementations of accreditation seals and the architectures that support them, including an implementation based on authenticated dictionaries [1, 6, 7], where accreditation seals are cached at a collection of responders deployed in the network. Even when the responders are located in insecure, untrusted locations, a client can easily identify a forged or tampered seal so that the integrity of a trusted-domain registry is maintained.

Accredited DomainKeys extends the existing DomainKeys framework [3] to easily accommodate accreditation registries, so that not only the domain name of a sender can be authenticated but also the trustworthiness of that domain. Specifically, it involves inserting one additional key-value pair to outgoing email messages authenticated with DomainKeys, and one additional DNS lookup and (usually lightweight) signature verification for the receiving email server. Thus, the design of Accredited DomainKeys focuses on the usability, performance, and scalability for both the sender and receiver mail systems.

The Accredited DomainKeys protocol is lightweight and easy to adopt. It uses DNS records for storing both public keys and their accreditation seals. It is compatible with existing DomainKeys-enabled email systems, and there are no extra computation/communication costs if Accredited DomainKeys is not supported by a sender or recipient mail system. Thus, legacy systems will inter-operate smoothly while accreditation bureaus (and their respective registries) are established. Querying and verifying accreditation seals are simple and fast to perform in Accredited DomainKeys, using just one additional DNS lookup (for the accreditation seal itself).

2 Preliminaries

In this section, we briefly summarize the DomainKeys framework [3], including its architecture, syntax, and related cryptography and DNS background.

DomainKeys is a framework for email sender authentication recently-proposed by Yahoo! [3]. It combines public-key cryptography and DNS to provide domain-level authentication for email. Given an email message that claims to be generated from a certain domain, DomainKeys describes a mechanism for the recipient's *mail transfer agent* (MTA) to verify whether or not the email originated from an MTA that is authorized to send emails for that domain. Several open source implementations of DomainKeys are available, including the *dk-milter* plugin [4] for the popular Sendmail MTA.

Under DomainKeys, a domain owner generates one or more private/public key-pairs. The public-key is placed in a DNS TXT record associated with that domain, and the private-key is used by the sender MTA to sign messages originating from authorized users of that domain. The signature is added as a header to the email, and the message is transferred to its recipients in the usual way. Upon receiving an email with a DomainKeys signature header, the receiving MTA authenticates the sender domain. It first extracts the signature and claimed sending domain from the email. Then the public-key is fetched from the claimed sending domain namespace through a DNS lookup, and is used to verify the signature.

Any type of signature schemes can be used to implement the DomainKeys framework. The most widely-used signature scheme on the Internet is the RSA signature scheme, whose current minimum public-key size is 1024 bits. Public-keys are stored in DNS TXT resource record (RR). The design of DomainKeys even allows a 2048-bit RSA public-key to be transmitted in a single DNS UDP packet with a 512-byte payload. The limit on the payload size of the UDP packet is due to the cache limits of some legacy DNS servers.

DomainKeys reserves the DNS namespace _domainkey within the sending domain for storing public-keys. In addition, to support multiple public keys per sending domain, it specifies that the DNS namespace can be further subdivided with *selectors*, which are arbitrary names below the _domainkey. namespace. An example of namespace using the selector brisbane is: brisbane._domainkey.example.net. This approach of storing data in DNS is used in this paper to store accreditation seals.

3 Accredited DomainKeys

Accredited DomainKeys supports both domain authentication and accreditation registries. The domain authentication is handled the same way as in the DomainKeys framework [3] using a signature and a DNS query. In addition, Accredited DomainKeys accommodates accreditation bureaus that issue accreditation seals for their respective members. Next, we give the definitions of the concepts used in the Accredited DomainKeys framework and we then describe the operations and data storage in the framework.

- **Acceptable usage policy (AUP):** A policy that prohibit spam, phishing, and other email abuses, established from the best practices of like-minded organizations acting in federation.
- **Accreditation bureau and trusted domain registries:** An accreditation bureau maintains trusted domain registries, which are directories of domain names whose administrators have agreed to an AUP.
- **Accreditation seal:** An accreditation seal provides a proof of membership of the sender's domain in an accreditation bureau's registry. The accreditation seal of a domain is generated and periodically refreshed by the accreditation bureau. This seal provides additional evidence that a public key used to sign a given domain's outbound email is correct and valid. More importantly, the seal shows that the managers of this domain have agreed to an AUP. The accreditation seal of a do-

main is stored in a DNS TXT record under the domain's namespace, and a recipient MTA obtains the seal through a DNS query for the TXT record under that namespace.

Under Accredited DomainKeys, a qualified domain registers its public-key with the Accreditation Bureau, which then sends an accreditation seal to the registered domain. The seal is stored as a DNS TXT record on the authoritative name server of the domain. Outbound emails from the domain are signed with the domain's private-key. The recipient MTA first queries the sender's name server for its public-key to verify the signature, and then queries for its accreditation seal to verify the sender's membership in the trusted domain registry. The long-term public-key of the Accreditation Bureau is needed to authenticate the accreditation seal.

In the rest of this section we describe the Accredited DomainKeys framework and discuss in detail the following topics: key setup and registration, storage of accreditation seals, sending an email message, domain authentication, and seal verification. The registration and processing of outgoing email messages are shown in Figure 1. The processing of incoming email messages is shown in Figure 2.

3.1 Key Setup and Registration

As in DomainKeys, a domain owner or commercial certificate authority (CCA) generates one or more public/private key pairs for the domain, where each public key is stored in a DNS TXT record associated with that domain, under a certain selector. The syntax for the public-key in the TXT record is the same as in DomainKeys [3]. The following is an example.

```
brisbane._domainkey IN TXT "k=rsa; p=MHww ...
IDAQAB;"
```

In the above, brisbane is the selector, _domainkeys is the namespace for the domain public-key, k represents the key type, and p represents the public-key data.

To register in an accreditation bureau's registry, a domain has to agree to an acceptable usage policy (AUP). Once joined, the accreditation seal of the domain is issued by the bureau. The accreditation seal is stored in a DNS TXT record under the namespace of the domain. The accreditation seal is periodically refreshed by the accreditation bureau, and the TXT record on the name server is updated. The storage of accreditation seals in DNS is described next.

3.2 Storage of an Accreditation Seal

As with the domain public-key, the accreditation seal of a domain is stored in the namespace _domainkey within that domain. To distinguish the TXT record storing the accreditation seal from the one storing the domain public-key, Accredited DomainKeys uses a *seal-selector*. For example, the seal-selector for example.net may be adelaide, so the accreditation seal would be stored in the namespace adelaide._domainkey.example.net.

The seal-selector also allows to distinguish among multiple accreditation seals, each of them corresponding to a registered public-key of the domain. The seal-selector must be different from the selector for the public key.

The DNS TXT record stores the accreditation seal and its attributes as tag=value pairs separated by semicolons. Accredited DomainKeys defines tags f and i for accreditation seal data.

- f: the attribute field that stores the accreditation seal data of a domain.
- i: a time-stamp indicating when the accreditation seal is generated.

For example,

```
adelaide._domainkey IN TXT "a=rsa;
i=2005.03.21.1700;f=wMDw ... IQDAAH;"
```

In the above example, a represents the signing algorithm (which should be used to validate this seal), f represents Base64-encoded seal data, and i is the time when the seal is generated. This TXT record should be transmitted in one DNS UDP packet with a 512-byte payload. In Section 4, several implementations of Accredited DomainKeys are given that satisfy this size requirement.

3.3 Sending an Email Message

In order to authenticate a sender domain and verify its accreditation registry membership, the email has to provide information for a recipient MTA to locate the sender's domain public-key and accreditation seal. This information is given by the sender MTA in the signature header of an email as follows.

The sender MTA signs an outbound email with its domain private-key, as in DomainKeys: the signature and its attribute data are stored in the header DomainKey-Signature of an email. Accredited Domain inherits the header DomainKey-Signature, and all the attribute fields of DomainKeys [3], such as field b for the signature data. In addition, Accredited DomainKeys introduces a new header Accredited-DomainKeys and an attribute field v for locating the accreditation seal:

- Accredited-DomainKeys: the email header that stores the information of an accreditation seal.

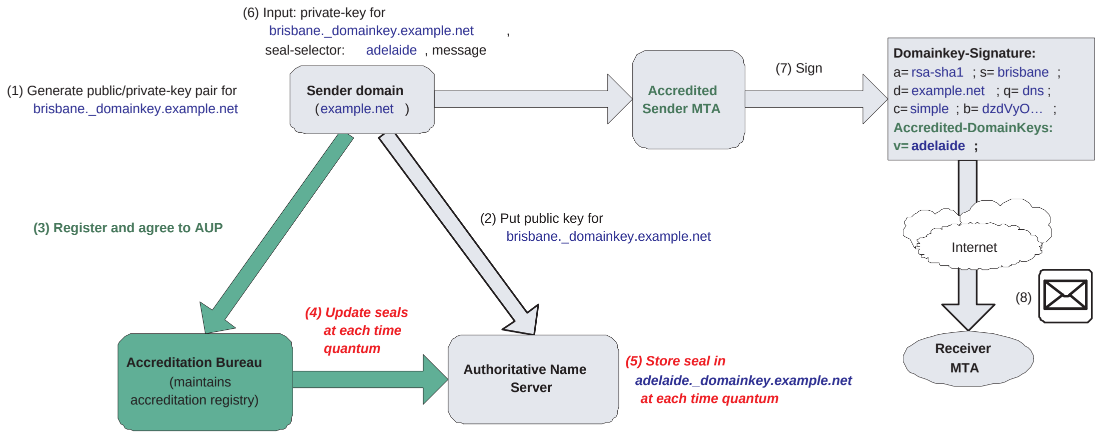
(6) Input: private-key for
brisbane._domainkey.example.net ,
seal-selector: adelaide , message
(1) Generate public/private-key pair for
brisbane._domainkey.example.net
Sender domain
(example.net)
Accredited Sender MTA
(7) Sign
Domainkey-Signature:
a= rsa-sha1 ; s= brisbane ;
d= example.net ; q= dns ;
c= simple ; b= dzdVyO... ;
Accredited-DomainKeys:
v= adelaide ;
(3) Register and agree to AUP
(2) Put public key for
brisbane._domainkey.example.net
Internet
(8)
(4) Update seals at each time quantum
Accreditation Bureau
(maintains accreditation registry)
Authoritative Name Server
(5) Store seal in
adelaide._domainkey.example.net
at each time quantum
Receiver MTA

Figure 1: A schematic drawing of the registration and signature generation process of the sender domain and related organizations. Components that Accredited DomainKeys adds to the DomainKeys framework are shown in green and red, where red indicates operations performed at each time quantum.

- v: attribute field for a seal-selector in the Accredited-DomainKeys header. The seal-selector is used to identify the namespace that stores the accreditation seal of the domain.

Headers contain attribute data in tag=value pairs separated by semicolons. For example:

DomainKey-Signature:	a=rsa-sha1; s=brisbane; d=example.net; q=dns; b=dzdVyO. . .;
Accredited-DomainKeys:	v=adelaide;

In the above, the header Accredited-DomainKeys and tag v are introduced by Accredited DomainKeys. The header DomainKey-Signature and tags a, s, d, q, and b are inherited from DomainKeys [3]. These tags represent the signing algorithm, selector, sending domain name, query method for retrieving public-key, and the signature data, respectively. A picture that illustrates the sending process is shown in Figure 1.

3.4 Domain Authentication

Under the Accredited DomainKeys framework, when an MTA receives an email, it has two tasks to accomplish before accepting the mail. The first task is to authenticate the sender domain, and the second task is to verify the sender's membership in the accreditation registry. A schematic drawing of the operations by the recipient MTA is shown in Figure 2.

Domain authentication is done as in DomainKeys, which is briefly described as follows. The recipient MTA extracts the domain name from the From header of the email, and extracts the selector from the field s in the DomainKey-Signature header. The namespace for the domain public-key is the concatenation of the selector, string ._domainkey., and the domain name. For example, brisbane._domainkey.example.net. Then, the recipient MTA queries for the DNS TXT record stored in the namespace brisbane._domainkey.example.net. The public-key data in the TXT record is then extracted, and the signature extracted from the DomainKey-Signature header is verified as in DomainKeys.

If the signature verification does not succeed, the email is not accepted. DomainKeys specifies how the recipient MTA may contact the sender DNS server for more information about its sending policies [3]. This is not repeated here. If the signature is successfully verified, the recipient MTA proceeds to membership verification, which is described next.

3.5 Seal Verification

If the sender domain has been successfully authenticated, the recipient MTA extracts the attribute value from field v in the Accredited-DomainKeys header of the email. If the header or the field does not exist or is null, this means the sender does not support Accredited DomainKeys. Then the membership verification is not performed by the recipient MTA. It is up to the recipient domain's policies to decide how to process the email in this case. Here, we consider the case where field v exists and is not null.

The recipient MTA extracts the seal-selector from field v in the Accredited-DomainKeys header of an email. The namespace that stores the accreditation seal is obtained by concatenating the seal-selector, string ._domainkey., and the sender domain. For example, adelaide._domainkey.example.net. The MTA queries for the DNS TXT record stored in that namespace, and the accreditation seal data is extracted.

Once the recipient MTA has obtained the accreditation seal, the membership of the sender domain

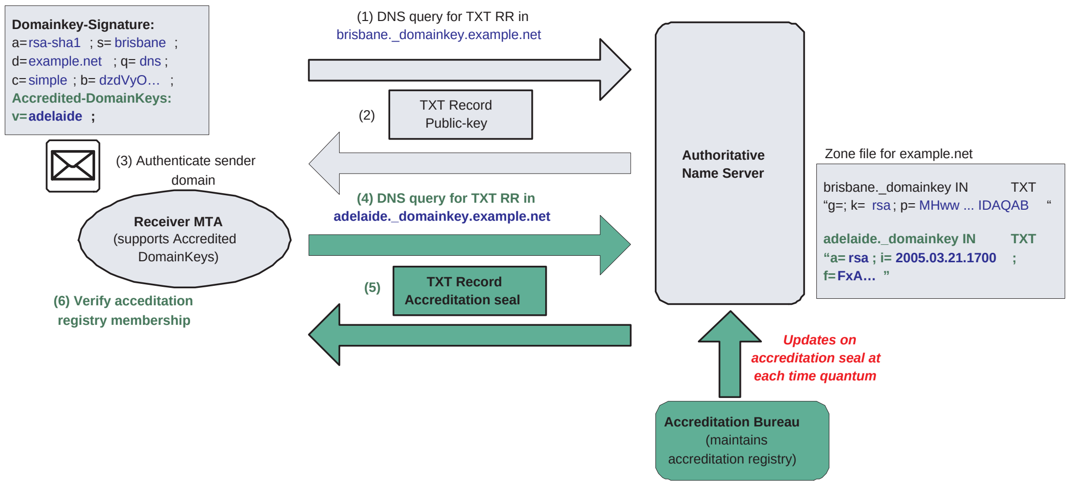
Domainkey-Signature:
a=rsa-sha1 ; s= brisbane ;
d=example.net ; q= dns ;
c=simple ; b= dzdVyO… ;
Accredited-DomainKeys:
v=adelaide ;
(1) DNS query for TXT RR in brisbane._domainkey.example.net
(2)
TXT Record Public-key
(3) Authenticate sender domain
Receiver MTA (supports Accredited DomainKeys)
(4) DNS query for TXT RR in adelaide._domainkey.example.net
(5)
TXT Record Accreditation seal
(6) Verify accreditation registry membership
Authoritative Name Server
Zone file for example.net
brisbane._domainkey IN TXT "g=; k= rsa ; p= MHww ... IDAQAB "
adelaide._domainkey IN TXT "a= rsa ; i= 2005.03.21.1700 ; f=FxA… "
Updates on accreditation seal at each time quantum
Accreditation Bureau (maintains accreditation registry)

Figure 2: Schematic illustration of the operations performed by a recipient MTA to authenticate the sender domain and verify the sender's registry membership.

in the accreditation registry can be verified. Several cryptographic implementations of accreditation seals for Accredited DomainKeys are presented in the next section. Accreditation seals in these implementations typically contain a special time-stamped signature signed by the accreditation bureau on registry information. A recipient MTA needs to obtain the public-key of the accreditation bureau to verify this signature. We assume that existing public-key infrastructure can be employed for this task. The recipient MTA obtains a long-term public-key certificate of the accreditation bureau. Once this key is obtained and accepted, it can be locally cached. The operations of a recipient MTA in Accredited DomainKeys are illustrated in Figure 2.

4 Implementation of Accredited DomainKeys

In this section, we describe two possible architectures for implementing Accredited DomainKeys. These architectures are specified simply by using different keywords in the a field of an accreditation seal record to indicate the algorithm to use to validate the given seal. Both architectures can be deployed in two different ways—either the domain can manage its own _domainkeys subdomain or it can delegate this task to authoritative name servers managed by the accreditation bureau (or an affiliated entity of the accreditation bureau).

4.1 Simple Time-Stamped Signatures

One possible solution for accreditation seals is for an accreditation bureau to individually sign every possible pair of a domain name and an affiliated DomainKeys public key for that domain, and publish these signatures in DNS using the associated accreditation seal selectors. These signatures can be very short—they only need to store the actual signature data, an identifier for the accreditation bureau (since we assume the bureau's public key is obtained out-of-band), and a time stamp. The publication of this signature can either be done at the domain's name server (e.g. seal._domainkey.example.net), or on the accreditation bureau's DNS server network (e.g. seal.example_net._domainkey.bureau.org), depending on the management structure chosen for the seals.

This solution requires the accreditation bureau to sign all of its client's public keys at every time quantum dictated by its AUP validity guarantee, e.g., every day or every several hours. Thus, this solution is only realistic for accreditation bureaus with a small number of clients (e.g., during the initial deployment phase of Accredited DomainKeys) or accreditation bureaus with significant computational resources. The implementation we describe next is more scalable.

4.2 Secure Transaction Management System (STMS)

In Accredited DomainKeys, we can use STMS to implement a scalable trusted-domain registry. The computing abstraction underlying STMS is a data structure called an authenticated dictionary (see, e.g., [1, 7, 17]), which is a system for publishing data and supporting authenticated responses to queries about the data. In an authenticated dictionary, the data originates at a secure central site *the STMS source* and is distributed to servers scattered across the network *STMS responders*. The responders answer queries about the data made by clients on behalf of the source. It is desirable to delegate query answering to the responders for two reasons: (1) The source is subject to risks such as denial-of-service attacks if it provides services directly on the network (2) The large volume and diverse geographic origination of the queries require distributed servers to provide responses efficiently.

The main feature of STMS is that it maintains trust even when responders are located in insecure, untrusted locations. That is, when a client makes a query to an STMS responder, it gets back not only an answer but also a proof of the answer. The client can easily validate the answer and determine that the responder has not been tampered with, while relying solely on trusted statements signed by the source. The design of STMS allows untrusted responders, which do not store private keys, to provide verifiable authentication services on behalf of a trusted source. This nonintuitive yet mathematically provable fact is the key to achieve cost effectiveness.

Figure 3 shows a high-level description of the STMS parties and protocol. The source sends periodic updates to the responders together with a special signed time-stamped fingerprint of the database called the *basis*. A responder replies to a query with an authenticated response, consisting of the answer to the query, the proof of the answer and the basis. Informally speaking, the proof is a *partial fingerprint* of the database that, combined with the subject of the query, should yield the fingerprint of the entire database. A proof consists of a very small amount of data (less than 300 bytes for most applications) and can be validated quickly. The client finally evaluates the risk associated with trusting the answer using the freshness of the time-stamp.

In applying the STMS framework to Accredited DomainKeys, we would publish the proof-basis pair to DNS, either using the domain's DNS server or the DNS responder network of the accreditation bureau. In either case, the overhead for the accreditation bureau is much reduced. In addition, recipient email servers can also have reduced costs, by caching verified bases. This caching allows recipient email servers to verify accreditation seals using a small number of cryptographic hashes.

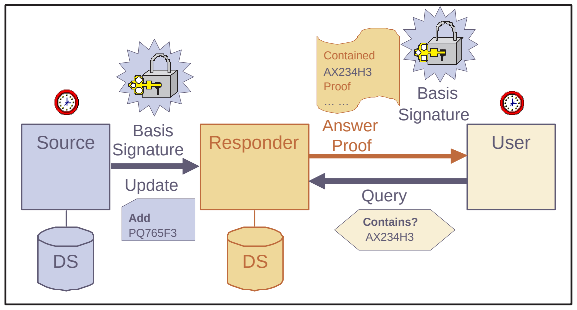

Figure 3: An overview of the protocol for STMS. The source pushes updates containing a signed basis to the responder. The responder then answers user queries with a proof of the answer, and a copy of the signed basis from the source.

If the number of public-key/domain pairs associated with a single basis is less than 1,000, the proof-basis pair can fit in a single UDP 512-bit packet, which satisfies the caching requirements for DNS. Thus, a centralized accreditation bureau will be able to scale to millions of proof-basis pairs simply by subdividing its managed data across multiple bases. Therefore, there is no technical limit to the number of public-key/domain pairs a registry can manage under Accredited DomainKeys. Thus, the STMS solution is efficient for modest size domain communities and it is extensible for larger communities as well.

4.3 Auditability and Nonrepudiation

The accreditation seal used in the Accredited DomainKeys framework can further be used a historic "audit stamp" that assures non-repudiation of messages and their historical verification. This property adds a critical element for global regulatory compliance. The STMS implementation uses an optimal compression technique for storing historical information [1], which allows responders to efficiently answer queries about seals at arbitrary times in the past.

5 Related and Alternative Solutions

The market is currently served by a handful of accreditation registries, which reactively query a list, as in IronPort's Bonded Sender Program, with accreditation provided by Truste and the Habeas Registry. In addition, the Email Service Provider Coalition (ESPC) has proposed a registry framework in their Lumos whitepaper [11].

One alternative approach to domain accreditation one could naturally imagine is to use PKI certificates, such as SSL X.509 certificates, issued by trusted third-party certification authorities (CAs), as accreditation seals. In this case, a domain agreeing to an AUP could obtain a trusted-domain certificate from a CA, and the certificate would then sent along with emails or could be retrieved from DNS. The recipient mail system verifies the certificate against the CA's public-key.

There are several practical concerns with this approach, however. The first concern is that PKI certificates are much larger than standard DNS TXT records, which, for caching purposes, are preferred to be less than 512 bytes. Thus, delivering these certificates in DNS poses problems with fragmentation and truncation. More importantly, however, PKI certificates are meant to have long-term validity, whereas accreditation seals are to be used for time-specific validation of an email AUP. Thus, with PKI certificates used to validate adherence to an AUP, a trusted-domain certificate may be revoked by CA before it expires. This could happen if there are complaints against the domain or the domain changes it owner. Therefore, the recipient MTA has to query the issuer CA to ensure the validity of the certificate. Using traditional CRLs for validity-checking in this case poses serious scalability issues and performing OCSP queries to test validity adds extra unnecessary overheads to the email validation. Alternatively, attaching a digital certificate to each outbound email significantly increases Internet traffic and email caching overhead. For those recipient mail systems that do not support trusted-domain certificates, this increased traffic and storage costs have no benefit.

6 Conclusions and Future Directions

In this paper, we propose the use of a trusted-domain registry for managing the trustworthiness of a sender domain in email. We describe a service architecture for accrediting domains through an extension of Yahoo! DomainKeys. We also describe two possible compliant implementations of Accredited DomainKeys, one based on simple signatures and one based on the existing Secure Transaction Management System (STMS), which gives rise to a scalable trusted-domain registry system. An interesting direction for future work would be to design a similar service architecture to integrate accreditation with path-based domain authentication, such as Sender ID [12] or SPF [9]. Another interesting research topic is to investigate how to integrate different accreditation results from multiple accreditation bureaus in evaulating the trustworthiness of a domain.

Acknowledgments

We are grateful to David Croston, of IAM Registry, Inc., for his helpful suggestions and support regarding the topics of this paper. We would also like to thank Eric Allman, Jon Callas, and Jim Fenton for several useful conversations concerning the topics of this paper. This work was principally supported by IAM Registry, Inc. Additional partial support was provided by NSF grants CCR–0225642, CCR–0311720, CCF–0311510, CCR–0312760, and IIS–0324846.

References

[1] A. Anagnostopoulos, M. T. Goodrich, and R. Tamassia. Persistent authenticated dictionaries and their applications. In *Proc. Information Security Conference (ISC 2001)*, volume 2200 of *LNCS*, pages 379–393. Springer-Verlag, 2001.

[2] H. Danisch. The RMX DNS RR and method for lightweight SMTP sender authorization, May 2004. `http://www.watersprings.org/pub/id/draft-danisch-dns-rr-smtp-04.txt`.

[3] M. Delany, Yahoo!, Inc. Domain-based email authentication using public-keys advertised in the DNS (DomainKeys), February 2005. `http://www.watersprings.org/pub/id/draft-delany-domainkeys-base-01.txt`.

[4] dk-milter. A DomainKeys implementation. `http://sendmail.net/dk-milter/`.

[5] J. Fenton and M. Thomas. Identified internet mail, October 2004. `http://www.watersprings.org/pub/id/draft-fenton-identified-mail-01.txt`.

[6] M. T. Goodrich and R. Tamassia. Efficient authenticated dictionaries with skip lists and commutative hashing. Technical report, Johns Hopkins Information Security Institute, 2000. Available from `http://www.cs.brown.edu/cgc/stms/papers/hashskip.pdf`.

[7] M. T. Goodrich, R. Tamassia, and A. Schwerin. Implementation of an authenticated dictionary with skip lists and commutative hashing. In *Proc. 2001 DARPA Information Survivability Conference and Exposition*, volume 2, pages 68–82, 2001.

[8] Identified Internet Mail. `http://www.identifiedmail.com/IIM%20WP_v3.pdf`.

[9] M. Lentczner and M. W. Wong. Sender policy framework (SPF) a convention to describe hosts authorized to send SMTP traffic, May 2004. `http://www.watersprings.org/pub/id/draft-mengwong-spf-01.txt`.

[10] J. R. Levine. A flexible method to validate SMTP senders in DNS, April 2004. `http://www.watersprings.org/pub/id/draft-levine-fsv-01.txt`.

[11] Project Lumos white paper. `http://www.espcoalition.org/lumos_white_paper.php`.

[12] Microsoft Sender ID Framework. `http://www.microsoft.com/mscorp/safety/technologies/senderid/default.mspx`.

[13] E-Mail Authentication Via Sender ID. `http://www.networkmagazine.com/shared/article/showArticle.jhtml?articleId=47903211&classroom=`.

[14] Spammers use sender authentication too, study says. `http://www.computerworld.com/printthis/2004/0,4814,95617,00.html`.

[15] SPF: Sender Policy Framework . `http://spf.pobox.com/`.

[16] M. Stumpf and S. Hoehne. Marking mail transfer agents in reverse DNS with TXT RRs, October 2004. `http://www.watersprings.org/pub/id/draft-stumpf-dns-mtamark-03.txt`.

[17] R. Tamassia. Authenticated data structures. In *Proc. European Symp. on Algorithms*, volume 2832 of *Lecture Notes in Computer Science*, pages 2–5. Springer-Verlag, 2003.