Security by Any Other Name: On the Effectiveness of Provider Based Email Security

Ian Foster, Jon Larson, Max Masich, Alex C. Snoeren, Stefan Savage, and Kirill Levchenko
Department of Computer Science and Engineering
University of California, San Diego
{idfoster,mmasich,snoeren,savage,klevchen}@cs.ucsd.edu

ABSTRACT

Email as we use it today makes no guarantees about message integrity, authenticity, or confidentiality. Users must explicitly encrypt and sign message contents using tools like PGP if they wish to protect themselves against message tampering, forgery, or eavesdropping. However, few do, leaving the vast majority of users open to such attacks. Fortunately, transport-layer security mechanisms (available as extensions to SMTP, IMAP, POP3) provide some degree of protection against network-based eavesdropping attacks. At the same time, DKIM and SPF protect against network-based message forgery and tampering.

In this work we evaluate the security provided by these protocols, both in theory and in practice. Using a combination of measurement techniques, we determine whether major providers supports TLS at each point in their email message path, and whether they support SPF and DKIM on incoming and outgoing mail. We found that while more than half of the top 20,000 receiving MTAs supported TLS, and support for TLS is increasing, servers do not check certificates, opening the Internet email system up to man-in-the-middle eavesdropping attacks. At the same time, while use of SPF is common, enforcement is limited. Moreover, few of the senders we examined used DKIM, and fewer still rejected invalid DKIM signatures. Our findings show that the global email system provides some protection against passive eavesdropping, limited protection against unprivileged peer message forgery, and no protection against active network-based attacks. We observe that protection even against the latter is possible using existing protocols with proper enforcement.

1. INTRODUCTION

Few users sign or encrypt email today, despite ready software support for PGP and S/MIME. The vast majority of email users continue to send email in the clear, with no safeguards against eavesdropping, tampering, or forgery. Despite rising public concern about mass surveillance, universal end-to-end email security still remains elusive.

While user adoption of email security tools has been poor, use of lower-layer security mechanisms, namely TLS, SPF, and DKIM, has been on the rise. Google recently reported 59% adoption of TLS by sending and 79% by receiving servers, and our own measurements (reported in this work) echo these statistics. We are led to consider whether some of the security goals of end-to-end tools like PGP can be satisfied using these protocols. Remarkably, when used correctly and enforced, TLS and DKIM with DNSSEC can protect against active network-based man-in-the-middle attacks. These guarantees, however require trusting the provider, something that is explicitly not necessary when end-to-end security mechanisms like PGP are used.

Whether email providers are *trustworthy* is a separate, no less important, question which we do not consider in this work. We recognize that for some readers, any solution that requires trusting an email provider is unacceptable as a matter of principle. For others, trusting an email provider is a fait accompli—users already trust their provider. (In fact, even Google's End-to-End initiative, based on OpenPGP, delegates key verification to centralized providers.[1] Our work, then, should be viewed as answering the question of what guarantees TLS, DKIM and SPF can provide, *if the provider is already trusted*. Of course, use of these provider security mechanisms does not preclude users from using an end-to-end mechanism as well.

If securing email against network attacks is possible using existing protocols, then we must ask whether these protocols are being used in a way that achieves this. The greater portion of this paper is devoted to this question, namely whether the current deployment of TLS, DKIM, and SPF provides these level of security possible under ideal conditions. To answer this question, we measured the level of support for TLS, DKIM, and SPF among top email providers and determined whether they enforce correct protocol use. Because DKIM and SPF depend on DNS for protection against an active (man-in-the-middle) attacker, we also measure the use of DNSSEC among providers.

We relied on a combination of active probing techniques to carry out the measurement study, some new and of independent interest to the Internet measurement community. To determine how much protection TLS provided against eavesdropping, we determined, where possible, whether TLS was supported and used along each hop along a message path between two providers, and whether server certificate checking was done. For hops internal to a provider where we could not interact with servers directly, we relied on information in *Received* email headers. For DKIM and SPF, we measured support on the sending end by examining messages generated by providers (DKIM only), retrieving the necessary DNS records, and noting support for DNSSEC. On the receiving end, we mea-

CCS'15, October 12–16, 2015, Denver, Colorado, USA.

ACM 978-1-4503-3832-5/15/10 ...$15.00.
DOI: http://dx.doi.org/10.1145/2810103.2813607.

[1] End-to-End relies on a mechanism akin to CA Certificate Transparency to mitigate the risk of impersonation, however, the effectiveness of such a mechanism has seen little evaluation.

sured provider verification of DKIM signatures and enforcement of SPF policies by generating mail of varying levels of conformance. Where measurement could be automated, we probed the top million mail servers (ranked by their prevalence among users in the Adobe leaked user database). Whenever manual interaction was necessary, we examined the top 22 email providers (based on Adobe rank) as well as select email-generating services (e.g. `amazon.com`).

In brief, we found that, while many providers support TLS for mail submission, transport, and delivery, all but one do not verify certificates, making them vulnerable to an active attacker. We also found that while DKIM provides message integrity, it is dependent on DNS, which is vulnerable to active attacks.

The final part of this paper briefly addresses the changes necessary to improve the current state of affairs. For some providers, the first step is to deploy these protocols. As we show in the measurement study, of the four protocols, DNSSEC has the lowest deployment, even among the top providers. The second step, enforcement, is less straightforward. Enforcing a security policy necessarily means rejecting non-conforming mail, an action that directly impacts the user experience. In the last part of the paper, we discuss the trade-offs involved in enforcing adherence to each protocol.

Our main contributions are:

- A methodology for determining TLS use along the full message path between two providers. We use both direct measurement (interaction with servers along the path) and information recorded in mail headers to determine whether a server uses TLS, and in the case of the former, whether it does certificate checking.
- An analysis of the current state of TLS, DKIM, SPF, and DNSSEC deployment. We report on the level of TLS support of 96 incoming mail servers and 302938 outgoing mail servers of popular Internet services. We also report on the level of DKIM, SPF, and DNSSEC support among these servers.
- We describe how correct use of the above protocols can provide message confidentiality and integrity in a relaxed threat model. We identify the changes in the current Internet email ecosystem necessary to provide this level of security.

The rest of the paper is organized as follows. Section 2 describes the threat model that this work is mostly concerned with. Section 3 provides the necessary technical background for the rest of the paper. Section 4 describes our methodology used in our analysis. Section 5 provides the final analysis of our collected data. Section 6 lists other work that has been done on the same subject. Section 7 discusses the implications of our results. Section 8 concludes the paper.

2. THREAT MODEL

The subject of this paper is the confidentiality, integrity, and authenticity of email communications achieved when TLS is used along the message path and DKIM, SPF, and DNSSEC by the end providers. In this section we describe the threat model we consider, namely network-based attacks. In particular, we consider three kinds of network attacker:

Active. An active attacker, also called a man-in-the-middle attacker, can observe and modify all packets, and inject arbitrary packets, between a target and the rest of the Internet.

Passive. A passive attacker can observe but not modify the traffic between a target and the rest of the Internet. (We consider passive attacks on confidentiality only.)

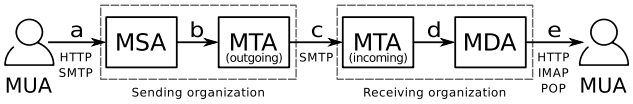

Figure 1: Typical path of a mail message from sender to receiver.

Peer. A network peer is an ordinary host connected to the Internet, capable of sending arbitrary packets and receiving packets for which it is the destination.

Other than the degree of network access above, we assume no other attacker capability. In particular, we assume that the email providers involved are trusted and do not collude with a network attacker. As discussed above, the assumption of a trusted provider is not prescriptive and not meant to suggest that providers are trustworthy or should be trusted. Rather, it should be understood as a logical antecedent of our results: *if* the provider is trusted and the protocols in question are used correctly, *then* a certain level of security against a network attacker can be achieved.

3. BACKGROUND

In this section we describe the level of security achievable when TLS, DKIM, SPF, and DNSSEC are used correctly. Our goal is also to identify those points of the protocol where failure would compromise overall security. In Section 5 we focus on these points to assess the real-world state of affairs.

We expect that most readers are familiar with the protocols in question, however, the Appendix provides the background necessary for the rest of the paper.

3.1 Security Properties

The security properties of concern to us are message confidentiality, message authenticity, and message integrity. Put plainly, these translate into the following:

Confidentiality. Can an attacker *read* a message?

Authenticity. Can an attacker *forge* a message?

Integrity. Can an attacker *modify* a message?

We consider attacks on confidentiality by active (man-in-the-middle) and passive attackers, attacks on authenticity by active and peer attackers, and attacks on integrity by active attackers only. The remaining combinations of attacker and attacked security property are excluded by definition. For example, a passive or peer attacker cannot compromise message integrity because she does not (by definition) have the ability to modify a message.

3.2 Mail Path

A typical path of a mail message is shown in Figure 1. Starting with the sending user's Mail User Agent (MUA), the message is first transmitted to the sender's mail provider (a) using the Simple Mail Transfer Protocol (SMTP), using HTTP via a Web interface, or, in some cases, using a proprietary protocol. After processing inside the provider (b), the message is transmitted to the recipient's provider (c) using SMTP. After receiver processing (d), which may include spam filtering, the message is delivered to the recipient using the Internet Message Access Protocol (IMAP), the Post Office Protocol (POP), HTTP via a Web interface or using a proprietary protocol.

3.3 Confidentiality

A network-based eavesdropper may gain access to a message during submission, provider processing, transfer between providers, and delivery. The standard protocols, namely HTTP, SMTP, IMAP,

and POP, used along the mail path defend against eavesdropping using TLS encryption. While in principle, proprietary protocols used on the submission and delivery hops, labeled (a) and (e), respectively, in Figure 1, may use non-standard cryptographic protocols.

Some providers also use SMTP internally, hops (b) and (d), however there is no requirement to do so. Communication internal to a provider has usually been considered secure, since it is common to collocate a MSA and MTA, or even combine the two roles in a single host. At larger providers, internal hops may be located in geographically separate data centers.[2]

Against a passive attacker, it is sufficient to use TLS on all attacker accessible links, provided a strong cipher suite is used. In particular, use of the STARTTLS option or direct TLS encapsulation with SMTP, IMAP, and POP or HTTPS will protect against a passive attacker. Provider-internal links may be secured physically or using TLS in the same manner. Since a passive attacker cannot modify network traffic, it is possible to exchange keys securely; defense against a passive attacker does not require certificate verification.

An active attacker, on the other hand, can impersonate each side of a connection to the other. Defending against such a man-in-the-middle attack requires the sender to verify the identity of the receiver. If an active attacker can gain control of any hop along the message path and either TLS is not used on that hop, or TLS is used without server certificate verification, the attacker will be able to impersonate the receiver and gain access to the message.

We note that DNS integrity is *not* required for TLS if the common name (`CN`) or the `subjectAltName` is checked against the intended server domain name (rather than IP address). Even if an attacker tricks a client into contacting a server under her control, the client can check that certificate provided by the server matches the host name of the intended server.

3.4 Authenticity

To forge a message, an attacker must be able to inject it at some point in the mail path in a such a way that it is accepted by the receiver. To forge a message at submission (a), the attacker must be able to authenticate as the user to the provider, usually using a password. An active attacker can trick the user into revealing his password by impersonating the provider (at either the submission or delivery end). Secrecy of the password thus requires server certificate checking by the user agent. In addition, failure to verify the server certificate of the MDA would allow a active attacker to impersonate the MDA and, in addition to learning the password, inject a forged message to the user on delivery (e). Message authenticity (and integrity) inside a provider can be achieved by physically securing the links or with the use of *client* certificates.

The hop most vulnerable to forgery, is the hop between providers (c). An incoming MTA, upon receiving a message, must be able to determine if the message originated from a provider authorized to send email with the sender email domain appearing in the message. Recall that our threat model posits a network-based attacker, so it is sufficient that verify that an incoming message originated at the sender's provider; the sender's provider ensures that the submitting MUA is authorized to send email from the user named as the message sender.

Both DKIM and SPF provide protection against forgery. DKIM provides a message signature covering the body and a subset of message headers. The signature is created by the sender's provider and is intended to convince the recipient's provider that the message is authentic and provide protection against tampering. SPF, on the other hand, allows a provider to identify (by domain name or IP address) a set of authorized senders for the domain. An incoming MTA can verify that the outgoing MTA sending the message is authorized to send email for the domain.

In principle, SPF can protect against forgery by a peer attacker by only allowing certain network hosts to send email on behalf of a domain. A peer attack cannot impersonate a host on the allowed list because she cannot complete a TCP connection from an allowed peer. An active attacker, on the other hand, can impersonate any host. SPF, thus, provides no protection against an active attacker.

DKIM, on the other hand, relies on a digital signature to protect the message, including the sender address. As such, DKIM can protect against message forgery and tampering, even against an active attacker. It is worth considering, then, what is required of an attacker to forge a DKIM signature. DKIM relies on DNS for key distribution. The key used to generate a particular signature is provided in a `TXT` record for a name in a special subdomain of the sender's domain. Thus, a message from `example.com` must be verified using a public key published in the `TXT` record of *`selector`*`._domainkey.example.net`, where *`selector`* is a token provided with the signature. An attacker that can forge the `TXT` record retrieved by the verifying server would, thus, be able to generate a valid signature for another domain.

The integrity of DNS responses can be secured using DNSSEC. If the sending provider uses DNSSEC, the receiving provider can obtain the DKIM signing key securely or receive signed confirmation that a record does not exist. The combination of DKIM and DNSSEC can, thus, be used to defend against forgery of signed messages. If an attacker omits the DKIM signature, the receiver must determine whether the message is a forgery, or whether the legitimate sender for the domain does not use DKIM. Two similar mechanisms exist for doing so: Author Domain Signing Practices (ADSP) and Domain-based Message Authentication, Reporting, and Conformance (DMARC). Both allow a provider to publish its message signing policy and requested treatment.

It is also possible to ensure message authenticity using TLS *client* authentication. We are not aware of any providers relying on this mechanism in submission, delivery, or provider-to-provider transport.

3.5 Integrity

In the absence of any protection, an active attacker can modify a mail message along any of the hops in Figure 1. DKIM signatures can be used to protect messages from tampering in transit. As discussed above, this requires DNSSEC if an attacker can tamper with the receiving provider's DNS traffic.

3.6 Enforcement

Both SPF and DKIM were developed primarily as a means to fight spam. By impersonating a user at a large provider, a spammer could improve his delivery rate; DKIM and SPF prevent such forgery. In this role, both SPF and DKIM are regarded as *signals* to a spam filter that the incoming message is legitimate. All other things being equal, a message with a DKIM signature should be less likely to be identified as spam than one without. It is tempting, therefore, for senders to regard DKIM and SPF as optional mechanisms to improve deliverability for some mail, rather than as security mechanisms. For the receiver too, limited adoption of DKIM makes it inadvisable to place too great a weight on the results of DKIM signature verification. ADSP and DMARC provide a means for a provider to advertise its mail authentication policy. However, ADSP and DMARC are newer additions; until they are

[2]Leaked documents indicate that the NSA has been eavesdropping on the (unencrypted) inter-datacenter traffic of both Google and Yahoo [4]. In light of this, many providers have moved to encrypting inter-datacenter traffic [1].

Property	*Active*	*Passive*	*Peer*
Confidentiality	TLS with Cert Verif.	TLS	—
Authenticity	DKIM* and DNSSEC	—	SPF or DKIM*
Integrity	DKIM* and DNSSEC	—	—

Table 1: Minimum protocol requirements for confidentiality, authenticity, and integrity against active, passive, and peer attackers. A "—" entry indicates an inapplicable combination. * *Note*: while DKIM is theoretically sufficient, as used today, it is also necessary to advertise a strict policy using DMARC.

widely adopted, a receiver will have no way of positively determining that a message from a provider should be signed.[3]

An alternative to ADSP and DMARC is for providers to bilaterally disseminate their signing policies to each other. With explicit agreement, DKIM signing policies can be enforced aggressively. We know of at least one case where this happens already: GMail will not accept email from eBay or PayPal if it is not signed [6].

3.7 Implications

We have argued that TLS with server certificate checking and DKIM with DNSSEC can be used to secure email against even an active network attacker. Table 1 summarizes the minimum requirements for each security property and class of attacker. In particular, TLS must be used with server certificate verification and DKIM must be used with DNSSEC to protect against an active adversary.

In the absence of bilateral inter-provider agreements, sending providers should publish a signing policy and receiving providers should not accept unsigned or incorrectly signed mail from providers advertising a policy of signing all outgoing mail.

So far, we have been concerned with the ideal case, what security guarantees can be made in the presence of a network adversary if TLS, DKIM, and DNSSEC are used correctly with an aggressive enforcement policy. In the remainder of this paper, we examine *what actually happens*. The following section describes our measurement methodology, and the section following describes out results. In particular, we report on both the deployment of the above protocols, whether providers use them correctly, and whether they enforce their correct use.

4. METHODOLOGY

In this section, we describe our measurement methodology. The subjects of our study are email providers and major services that generate email (e.g. e-commerce and online social networks). For each subject considered, we determined how each of the security protocols in question were used.

Roughly speaking, our measurement methods can be divided into two kinds: those that could be fully automated and scaled easily, and those that required some manual interaction. For the latter, we used a set of 302938 major email providers and email generators, while for the former, we used a much larger set of a million popular providers occurring in the Adobe leak and the Alexa top million Web sites (as potential email generators).

To determine whether email sent between these services is protected from a network attacker, we experimentally determine if each hop along the message path is properly secured. For hops that are externally accessible, namely MUA to MSA, MTA to MTA, and MDA to MUA, denoted (a), (c), and (e) in Figure 1, we interact with the endpoints directly to determine their TLS behavior. For hops internal to a provider, we rely on information reported in the `Received` mail headers. Our data consists of two measurement

[3] A receiver could infer a signing policy based on previous messages from the provider. However, it cannot be taken as a reliable indication that all mail should be signed.

Domain	*Country*	*Frequency*	*Cumulative*
`hotmail.com`		29.82%	29.82%
`gmail.com`		18.86%	48.68%
`yahoo.com`		14.22%	62.91%
`aol.com`	US	2.83%	65.74%
`gmx.de`	DE	1.06%	66.80%
`mail.ru`	RU	1.05%	67.85%
`yahoo.co.in`	IN	0.99%	68.84%
`comcast.net`	US	0.89%	69.73%
`web.de`	DE	0.88%	70.61%
`qq.com`	CN	0.71%	71.32%
`yahoo.co.jp`	JP	0.71%	72.02%
`naver.com`	KR	0.47%	72.49%
`163.com`	CN	0.46%	72.95%
`twc.com`	US	0.38%	73.33%
`libero.it`	IT	0.34%	73.67%
`yandex.ru`	RU	0.32%	73.99%
`daum.net`	KR	0.27%	74.26%
`cox.net`	US	0.26%	74.52%
`att.net`	US	0.22%	74.73%
`wp.pl`	PL	0.20%	74.93%
`pacbell.net`	US	0.08%	75.01%
`sohu.com`	CN	0.04%	75.05%

Table 2: The top email hosts from the Adobe list. Rows contain all users who use any MTA shared by the domain.

experiments about a year apart (March 2014 and February 2015), giving us a view into the changes in TLS use.

TLS use during a SMTP session requires both client and server support. In particular, a server must offer the STARTTLS option, and the client must issue the `STARTTLS` command. Thus, *we can infer how a client and a server are likely to interact by testing each separately*. If a particular server offers the STARTTLS option to us when we connect to it, and a particular client issues the `STARTTLS` command when it connects to us, we can say that, at least nominally, the two will use TLS with each other. We say *nominally*, because either server, client, or both may be configured to act differently when communicating with each other than when communicating with us, or the two may be using incompatible TLS implementations. To test this premise, we tested how select providers interaction with each other, as described in Section 4.8.

We first describe how we chose the set of services we tested.

4.1 Subject Selection

For our conclusions to be useful, the set of message paths we analyze should be broadly representative of the message paths seen in the global email system. The ideal set case would be a set of paths formed by uniformly sampling message paths on the Internet. Unfortunately, this is impossible in practice. To approximate the ideal sample, we compiled a list of popular email providers (or simply *providers*), email generating services (*generators*). We then used fragments of message paths originating or terminating at these services to piece together a complete picture of possible message paths between them. For verification, a subset of these paths are materialized explicitly, as described in Section 5.3.

4.1.1 Path Uniqueness

The path taken by a message between a given sender and recipient is not unique due to load balancing and email infrastructure evolution. However at a given time, we found message paths to be stable with respect to TLS use characteristics. That is to say, characteristics of TLS use along the path did not change during the measurement period. We note where this was not the case in our analysis.

4.1.2 Provider List

We created the set of popular email providers based on the top 1 million email address domains occurring in the leaked Adobe user data set of September 2013. (The full list consists of 152 million email address spanning 9.2 million distinct domains.) A number of large providers may service more than one domain name; for example, hotmail.com and outlook.com are domains used by the same service, namely Microsoft's Outlook.com. We grouped such domains into a single service based on the incoming MTAs for the domain. Specifically, for each domain, we retrieved its DNS `MX` records. Not all domains had MX records, and some had more than one. (If there was no MX record, we took the domain name itself as the incoming MTA address, per RFC 5321.) We then resolved all host names, to arrive at a set of IP addresses of incoming MTAs servicing a domain. All DNS lookups were done in February 2015. Any domains with at least one common IP address were grouped into a single service. We call the resulting list the *provider list*.[4]

Some of the experiments required manual interaction with a service. For these, we took 22 of the top providers from the Adobe provider list with which we were able to create an account. In particular, experiments where we acted as the receiving provider required us to send a message from a provider to ourselves, a process that required non-trivial manual effort. We call this the *Select provider list*. Table 2 shows these 22 providers on order of their popularity in the leaked Adobe user list. As described above, some of the providers service multiple domains; such providers are identified with their primary domain and their indicated popularity includes the contribution of all the domains they service. For example, `hotmail.com` includes `live.com` and `outlook.com`, and `yahoo.com` includes many regional Yahoo! domains that are serviced by the main Yahoo! mail servers. We note that `yahoo.co.jp` and `yahoo.co.in` are not served by the same MTAs as `yahoo.com`.

4.1.3 Generator List

Much of the email we receive in our inboxes is generated automatically, including e-commerce order confirmations, updates from online social networks, and so on. We created a list of such email generators by attempting to create an account with each service in the Alexa Top 100 list. We succeeded in doing so for 61 of these services. We also created a short list of organizations or services not on the Alexa 100 that we believed might warrant additional email confidentiality and from which we were able to generate an email message. We call these services the *Generator list*, shown in Table 4.

4.2 Incoming MTA Behavior

For TLS to be used on a SMTP hop along the message path, both the client and server must support TLS. We interacted with the incoming MTAs of providers on the provider list to determine whether they supported TLS and with what options. Incoming MTAs were identified by retrieving the `MX` records for each provider's domains. If a domain did not have any `MX` records, which happened with 0.43% of domains, we used the domain name itself as the incoming MTA, as specified in RFC 5321. For each incoming MTA thus identified, the interaction ran as follows:

1. **Connect.** We resolved the SMTP server host name to an IPv4 address and opened a connection on port 25. The initial connection step failed for 7.89% servers.
2. **EHLO.** We issued the `EHLO` command with the fully qualified domain name (FQDN) of our server per RFC 5321. If the server did not acknowledge the `EHLO` then we fell back to the `HELO` command and noted that the server did not support ESMTP. 0.85% of the incoming MTAs we contacted did not support ESMTP, accounting for 0.59% of all domains on the provider list.
3. **ESMTP Options.** Upon successful execution of the `EHLO` command, servers responded with a list of supported ESMTP options. For incoming MTAs, 44.98% of servers in this step did not advertise the STARTTLS extension (44.60% of all servers). Nevertheless, we did not eliminate such servers from consideration and attempted to issue the `STARTTLS` command in the next step regardless of advertised support.
4. **STARTTLS.** We issued the `STARTTLS` command to the server. The `STARTTLS` command failed for 0.51% of incoming MTAs that advertised the STARTTLS option in the previous step (45.31% of all servers). Of the servers that did *not* advertise STARTTLS support, 0.30% *did* respond to the `STARTTLS` by starting a TLS handshake.
5. **TLS handshake.** We carried out the TLS negotiation phase and recorded the options supported by the server and the server certificate. We did not supply a client certificate.
6. **Mail transfer.** With TLS encryption in place, we either proceeded to send an email message (if we had an account with the service as described in Section 4.1.2) or issued the `QUIT` command.

4.3 Outgoing MTA Behavior

A provider's outgoing MTA plays the role of a client when transferring mail to an incoming MTA of another provider. In this role, it must issue the `STARTTLS` command to start the TLS session. To test which outgoing MTAs do so, we generated a message from the provider in question to an incoming MTA server we control. Of course, this requires an account at the provider in question, so the first step in this experiment was creating these account. In all, we created accounts at 22 mail providers representing 75.05% of users according to the provider list ranking. Interaction with the outgoing MTA proceeded as follows:

1. **HELO/EHLO.** The client must first issue a `HELO` or `EHLO` command identifying itself. The latter identifies the client as speaking ESMTP, which was the case for all 22 providers. We accepted both.
2. **ESMTP Options.** If the client used the `EHLO` command, we advertised the STARTTLS extension.
3. **STARTTLS.** A client wishing to use TLS would now issue the `STARTTLS` command, in order to protect the rest of the SMTP session using TLS. 15 of the 22 ESMTP-speaking outgoing MTAs did so. For MSAs, we also attempted to proceed without issuing `STARTTLS` to determine if a provider would accept login credentials and mail over an unsecured connection.
4. **TLS handshake.** We carried out the TLS handshake, offering the client our server certificate. We used different certificates each session to determine the level of certificate checking done by the client. The certificates we used are described in Section 5.3.2. We also requested a client certificate, and recorded it if it was provided.
5. **Mail transfer.** We accepted any mail offered by the client.

4.4 SMTP MSA Behavior

To assess the level of TLS support by SMTP MSAs, we obtained mail submission configuration information from the 22 providers on the select provider list. 15 of the 22 providers instructed the user to configure their mail reader to use TLS. For SMTP with

[4] All merges were validated manually.

```
Received:
   from BLU004-OMC4S27.hotmail.com
       (blu004-omc4s27.hotmail.com. [65.55.111.166])
   by mx.google.com with ESMTPS id ...
   for <...@gmail.com>
       (version=TLSv1.2
        cipher=ECDHE-RSA-AES128-SHA bits=128/128);
   Sun, 15 Feb 2015 16:35:49 -0800 (PST)
```

Figure 2: Example `Received` line added by Google's incoming MTA `mx.google.com` to a message from Hotmail, identifying Hotmail's outgoing MTA `blu004-omc4s27.hotmail.com`, and giving TLS parameters. Omitted information shown as "..."

`STARTTLS`, we performed the same interaction as for SMTP incoming MTAs (Section 4.2), however we also checked that the MSA would proceed without the client issuing `STARTTLS` first. For SMTPS MSAs, we carried out the TLS handshake and captured the server certificate.

4.5 POP and IMAP Behavior

For the 22 providers in the select provider list, we contacted each provider's POP and IMAP server. All 22 offered POP and IMAP support, 15 of the 22 providers instructed the user to configure their mail reader to use TLS. Our interaction ran along similar lines as SMTP MSA. For POP3 and IMAP, we carried out the handshake and captured the certificate. For the POP3 and IMAP with `STARTTLS`, we recorded whether the STARTTLS option was advertised, issued `STARTTLS` and, and captured the certificate. We did not use a client certificate.

4.6 Webmail Behavior

Users may also interact with their email provider using a Web interface. All of the 22 providers in our select provider set supported this option. For each, we recorded whether the Web mail interface supported HTTPS, whether or not it was the default, and whether the certificate was valid.

4.7 Reported TLS Use

The SMTP standard requires mail servers along a message path to prepend a `Received` header line, indicating when, by which server, and from which server, a message was received (Sec. 4.4, RFC 5321). The standard also defines additional information which a server may add to the `Received` line, including protocol information introduced by the `WITH` keyword. RFC 5321 defines two possible values, `SMTP` and `ESMTP`, indicating whether ESMTP was used or not. RFC 3848 extends this list to include others, including `ESMTPS`, which indicates that TLS was used. Figure 2 shows a sample `Received` line.

We used this feature to map TLS use on the internal hops (b) and (d). We sent messages from our account on each provider to our server and from our server to an account at each provider, and collected the `Received` headers from these messages. We then extracted the `WITH` clause, if present, of each line, using it to infer TLS use.

4.8 Cross-Provider Validation

Recall that our message path measurement technique is built on the premise that TLS use between a client and a server can be inferred from their behavior when interacting with us. To determine if this is indeed the case, we sent messages between all pairs of providers on the select provider list (484 messages in all). We then used the `Received` header information described above to determine if providers exhibited different pairwise behavior than might be expected from their interaction with us. Results are shown in Table 6 and discussed in Section 5.3.

4.9 Certificates

When testing server certificates, we checked if the certificate was revoked (via a CRL) or expired. We checked if the certificate common name or any of the `subjectAltName` matched the host name to which we were connecting. We also checked if the certificate was signed by a trusted CA using the Mozilla list as the trusted root.[5] We also noted the signature algorithm used.

4.10 DKIM

To determine DKIM signing by outgoing mail providers, we examined the messages used in the outgoing MTA measurement (Section 4.3) to determine if a DKIM signature is present, and if so, if the signature is correct. Since this measurement required having a DKIM selector from a signed message, we could only perform this measurement for providers on the select provider list, from whom we received email. For each message examined, we extracted the `DKIM-Signature` header from the message, retrieved the DKIM key (if one exists) for the `selector._domainkey.domain.com` `TXT` record where `selector` and `domain.com` are the selector and domain from the DKIM header. The hash from the DKIM Signature was then decrypted with the DKIM key. If the decrypted hash matches the computed hash of the message then the DKIM signature is marked as valid.

To evaluate the effect of DKIM use on incoming mail, we generated mail to providers on the select provider list. We sent three kinds of messages to each: without a DKIM signature, with a valid DKIM signature, and with an invalid DKIM signature. The subject and body of the three messages were identical; however, the date, included in the signature, varied. Each type of test was conducted from a different IP address to avoid IP reputation bias.

We then examined whether the message was rejected, marked as spam, or delivered to the user inbox of our account. In addition to determining message outcome, we also recorded whether the provider queried our DNS server for `TXT` record containing the DKIM signing key. The selector was not previously used to ensure the record would not be cached.

4.11 SPF, ADSP, DMARC, and DNSSEC

To test provider support for SPF and DMARC in outgoing mail, we queried the name server of providers on the provider list for the DNS `TXT` record used by each protocol. In addition, we made note of whether the provider's mail server supported DNSSEC and returned signed records.

DNSSEC was verified by querying each domains `DNSKEY` record. If no `DNSKEY` record was found, then the domain was marked as not supporting DNSSEC. If there was a `DNSKEY`, then we queried two DNS servers, one without DNSSEC support, and one enforcing DNSSEC for the `A` record of the domain. If the DNSSEC enabled server responded without a `SERVFAIL` result then DNSSEC passed, otherwise the domain had invalid DNSSEC. The DNS server without DNSSEC was used as a control to ensure that there exists DNS records for the domain.

SPF validation was tested by setting the SPF `TXT` record for our test domain to "`v=spf1 a -all`" which should fail or reject mail not sent from our domain's `A` record. We then sent messages to the top mail providers from an IP address not in our domain's DNS. We recorded whether the SMTP session to the provider's MTA was successful, and if it was, if the message sent ended up in the recipient's inbox or spam folder.

To test DMARC we set the DMARC `TXT` record for a domain under our control to "`v=DMARC1 p=reject`". We then repeated

[5] https://wiki.mozilla.org/CA:IncludedCAs

	SMTP	POP3	IMAP	HTTP	Out	In
hotmail.com	◐	◐	◐	●	▶	
gmail.com	●	●	●	●	▷▶▷	▷
yahoo.com	●	◐	●	●	· · ·	·
aol.com	●	◐	●	●	▶	
gmx.de	●	●	●	●	▶▷	
mail.ru	●	●	●	●	▷	
yahoo.co.in	◐	●	●	●	· · · · ·	·
comcast.net	●	●	◐	●	▷▶	· ▷
web.de	●	●	●	●	▶	
qq.com	◐	●	◐	●	▶	
yahoo.co.jp	◐	●	●	●	▷ · · · ·	
naver.com	●	○	●	●	▷	▷ ·
163.com	●	●	●	○	▷	
twc.com	◐	◐	◐	●	▷	▷
libero.it	●	●	●	○	▷	▷ ·
yandex.ru	●	●	●	●	▷▷	▷
daum.net	●	●	●	○	▷▷	
cox.net	●	●	●	●	▷▷	▷
att.net	◌	●	●	●	· · ·	·
wp.pl	●	●	●	●	▷	·
pacbell.net	●	●	●	●	· · ·	·
sohu.com	○	○	○	●	▷▷	▷

Table 3: TLS behavior of user-facing SMTP, POP3, IMAP, and HTTP servers of top mail providers (left frame) and the internal structure of each (right frame). Legend: ● valid certificate with matching host name, ◐ valid certificate with different host name, ○ no TLS support; ◌ the provider rejected non-TLS connections. In the right frame, the *Out* column shows MSA to outgoing MTA message path; the *In* column shows the incoming MTA to MDA path. Each symbol indicates an internal hop. Legend: ▶ TLS was used, ▷ no TLS used, · an unknown protocol was used.

sending mail with an invalid DKIM signature as described in Section 4.10. Setting the policy to reject should result in mail being canceled at the SMTP layer if the DKIM signature is not correct. We recorded if the SMTP connection to the provider's MTA was successful, and if so, whether the message sent ended up in the recipient's inbox or spam folder. As a control we also tested with the policy set to "`p=none`".

To test whether providers used SPF, ADSP and DMARC in processing incoming mail, we sent mail to each of the providers on the select provider list. For each, we sent a message both from an IP address not authorized to send email per SPF policy, and one from which email was authorized. As in the DKIM experiment, we then examined whether the message was rejected, marked as spam, or delivered to the inbox. We also recorded whether the receiving provider queried our DNS server for the DKIM, SPF, ADSP, or DMARC `TXT` records.

5. RESULTS

Recall that we set out to determine whether a given message path is secured with TLS along each hop. In Section 4 we described how we can infer TLS use along the path from direct and indirect interaction with the servers. Here we present our findings.

5.1 Submission and Delivery

The first and last hop in a message path, labeled (a) and (e) in Figure 1, involve the user. The security of these two hops depends on the user MUA and on the MSA and MDA of the provider.

5.1.1 SMTP, POP, and IMAP

We tested the SMTP, IMAP and POP servers specified by each provider (of the 22 select providers) in their mail client configuration instructions. Our results are shown in Table 3, where ○ denotes no TLS support, and all other marks indicate TLS was supported. Only one provider, `sohu.com`, did not provide TLS support for submission or delivery. One provider, `naver.com` did not support POP3 with TLS, but did support IMAP with TLS.

One provider, `twc.com`, supported TLS, but the configuration instructions provided to their users did not indicate that TLS should be enabled, leaving it up to the MUA to issue the STARTTLS command.

Some providers required TLS use, and would not serve a client without TLS. These providers are shown in Table 3 with a ◌ mark. The top three providers—Hotmail, GMail, and Yahoo!—all require TLS for mail submission and delivery.

Certificates. Table 3 also indicates the type of TLS certificate presented by the MSA or MDA. The ◐ mark indicates that a certificate was not revoked, expired, or signed by an untrusted CA (Sec. 4.9). All certificates we encountered here met satisfied these conditions. We also checked whether the certificate name matched the name of the host to which we made the connection. Here the situation was less rosy. For MSAs, of 6 of the 22 providers used a server certificate that did not match the server host name to which we were connecting. (Recall that the server name was obtained from the provider's own configuration information.) Hotmail, for example, specifies that `smtp-mail.outlook.com` should be used for mail submission, however the certificate offered by this server has the common name `*.hotmail.com` and no `subjectAltName`. On the delivery side, we found that a number of providers, including Hotmail and Yahoo!, sent mismatched certificates.

5.1.2 Web interface

All of the 22 select providers offered a Web mail interface, however, three (`163.com`, `libero.it`, and `daum.net`) did not offer TLS support. Among the top 10 providers, all except `qq.com` and `comcast.net` *required* SSL/TLS to access Webmail.

Certificates. All certificates used for HTTPS were matching and valid. This is not surprising, given the intimidating warnings issued by modern browsers for mismatched certificates. The failure to check for matching certificates by MUAs likely explains the large number of mismatched certificates used by MSAs and MDAs.

5.2 Inside the Provider

Once submitted to a mail provider, a message may transit a number of internal servers before reaching the outgoing MTA. We do not have visibility into internal message processing, so our measurements are based on information given in the `Received` headers (Section 4.7). Using these, we reconstructed the use of TLS inside the 22 select providers on the outgoing and incoming path (labeled (b) and (d) in Figure 1).

Table 3 shows our results. The *Outgoing* column shows internal hops on the outgoing (MSA to outgoing MTA) path and the *Incoming* column shows internal hops on the incoming (incoming MTA to MDA) path. Each mark represents a hop: ▶ indicates TLS was used, ▷ indicates TLS was not used, and · indicates that a non-standard protocol was used. Yahoo! appears to use a protocol called NNFMP internally. It is not publicly documented, and we do not know if it uses TLS. Some providers had multiple routes a message could take, in this case we favored the route with the most hops.

Overall, TLS use on internal hops is not widespread. (We emphasize that internal hops may be on the same local network, or may be carried on an inter-datacenter VPN.) Incoming message paths are much shorter, and in some cases, record no hops at all. None of the incoming message paths appeared to use TLS.

Providers which report no hops from the MTA to MDA such as `web.de` may be using the same host for both the MTA and MSA, or may not be recording the internal hops to the message headers.

5.3 Provider-to-Provider

The hop between providers, from outgoing to incoming MTA, uses SMTP. In the absence of provider-to-provider peering, messages along this hop will transit the public Internet. It is perhaps here that the risk of mass traffic interception is greatest. As discussed in Sections 4.2 and 4.3, we use the behavior of the outgoing and incoming MTAs when communicating with us to infer how they might behave when communicating with each other. Because of the manual effort required, we evaluated outgoing MTA behavior for select providers only. For incoming MTAs, we used the top 1 million domains representing 245,054 distinct providers in 2015 (266,323 in 2014). We then combine these to form a picture of message paths between select providers and the full set of providers.

5.3.1 TLS Support at Outgoing MTAs

Email providers. The first column in Table 6, labeled CONTROL, shows the use of the `STARTTLS` command by provider outgoing MTAs when contacting our server. The vertical headers are the first character of the sending provider. The ○ mark indicates TLS was not used, ● indicates that TLS was used in both scans (March 2014 and February 2015), and ✪ indicates that TLS was used in the February 2015 scan but not in the March 2014 scan. (There were no cases of TLS being used in 2014 but not 2015 among the select providers.) The top 10 providers all used TLS when offered in 2015.

Other generators. We also examined outgoing MTA TLS support of major Web services (Section 4.1.3) in March 2014. These results are shown in Table 4, grouped by category. Note that some names appear in Table 6 and Table 4 with a different level of TLS support indicated. These are services offer both mail and non-mail services. Table 6 shows TLS support for outgoing MTAs used by the *mail* service, while Table 4 shows TLS support for outgoing MTAs used by the site's user account system. For example, `yandex.ru` is both a mail provider and a popular Web portal. Registering an email account (not necessarily a `@yandex.ru` account) with the site will generate email relayed by an outbound MTA that does not use TLS.

Support for TLS on outgoing MTA links was highest among the financial institutions we examined. All but USBank's outgoing MTA supported TLS. The lowest level of support was among news and dating sites. The latter, in particular, is surprising, given the personal nature of the emails.

Domain	*TLS*	*SPF*	*DM*
Search			
google.com	●	●	●
yahoo.com	●	●	◉
baidu.com	○	◉	○
qq.com	○	●	●
live.com	○	●	○
hao123.com	○	◉	○
sohu.com	○	◉	○
yandex.ru	○	●	●
bing.com	○	○	○
163.com	○	◉	○
mail.ru	●	●	●
Entertainment			
youtube.com	●	◉	●
xvideos.com	○	●	○
imgur.com	●	●	●
xhamster.com	●	○	○
vube.com	○	◉	●
youku.com	○	○	○
pornhub.com	○	◉	○
vimeo.com	○	◉	○
dailymotion.com	○	●	○
netflix.com	○	◉	◉
Government			
healthcare.gov	○	◉	○
whitehouse.gov	○	○	○
Conferences			
easychair.org	●	○	○
hotcrp.com	●	●	●
News			
sina.com.cn	○	◉	○
msn.com	○	◉	○
cnn.com	○	○	○
people.com.cn	○	○	○
gmw.cn	○	○	○
espn.go.com	○	○	○
Dating			
match.com	○	●	●
zoosk.com	○	●	○
okcupid.com	○	●	○
pof.com	○	●	○

Domain	*TLS*	*SPF*	*DM*
Commerce			
amazon.com	●	◉	●
ebay.com	●	●	●
adcash.com	○	●	○
neobux.com	○	◉	◉
godaddy.com	○	◉	●
craigslist.org	○	◉	●
aliexpress.com	○	◉	●
alibaba.com	○	◉	●
alipay.com	●	◉	●
rakuten.co.jp	○	●	●
Misc			
ask.com	○	◉	○
360.cn	●	●	○
microsoft.com	○	◉	●
thepiratebay.se	○	○	○
kickass.to	●	●	○
imdb.com	●	○	○
stackoverflow.com	●	●	○
wikipedia.org	○	○	●
Banks			
bankofamerica.com	●	●	●
paypal.com	○	●	◉
chase.com	●	◉	◉
discover.com	●	○	●
usbank.com	○	◉	●
americanexpress.com	●	◉	◉
Social			
wordpress.org	○	○	○
facebook.com	●	◉	◉
linkedin.com	●	●	◉
twitter.com	●	◉	◉
blogspot.com	●	○	●
weibo.com	○	◉	○
wordpress.com	○	●	●
vk.com	○	●	○
pinterest.com	○	◉	◉
instagram.com	●	◉	◉
tumblr.com	○	●	○
reddit.com	○	◉	○
fc2.com	○	●	○
blogger.com	●	○	●
odnoklassniki.ru	○	●	○

Table 4: TLS, SPF, and DMARC (*DM*) support among outgoing MTAs used by select Web services to send email. ○ indicates no support or protection, ● indicates basic support, and ◉ indicates that SPF or DMARC is configured in a strict manner.

5.3.2 Certificate Checking at Outgoing MTAs

As described in Section 4, we performed the experiment several times, offering different certificates to the outgoing MTA each time it connected. We found that all but three providers, `wp.pl`, `comcast.net`, and `hotmail.com`, did not perform any certificate checking. (For space reasons, results are not presented in tabular form.) In particular, all but those three accepted a revoked, expired, self-signed, mismatched certificate with a weak signature (`sha1WithRSA 512 bit`). The outgoing MTAs for `hotmail.com`, `wp.pl`, and `comcast.net` rejected our certificate only because it was expired. Remedying this, their outgoing MTAs accepted the revoked, self-signed, mismatched, weak certificate.

5.3.3 Outgoing MTA Client Certificates

For each connection from an outgoing MTA, we also recorded the *client* certificate provided by the MTA during TLS negotiation. (Our server was configured to request it.) 7 of the 22 select providers returned a client certificate for our request. Of these, only one, from `comcast.com`, was expired or otherwise invalid.

5.3.4 TLS Support at Incoming MTAs

Select providers. The top row in Table 6, labeled CONTROL, shows support for TLS at the incoming MTA for the 22 select providers. It is surprising to see that more providers support sending with TLS than receiving with TLS. However Google's TLS data discussed in more detail in Section 6 shows that 7 of the providers we observed not sending with TLS do use TLS with Google.

Other providers. As described in Section 4.2, we also tested the incoming MTAs of the providers for the top 1 million domains in the Adobe leak. Among these 302,938 MTAs (covering 245,054 providers), 50.5% supported TLS in March 2014, increasing to 54.6% in February 2015. Among the top 1000 providers, support for TLS increased from 43.7% to 59.2%.

Status	*Freq. 2014*	*Freq. 2015*
Valid	75.86%	79.14%
Self Signed	20.47%	11.39%
Expired	3.41%	2.88%
Revoked	0.17%	0.04%
Non Matched	34.13%	37.26%

Table 5: Certificate status of the top mail receiving MTAs found in the Adobe data set.

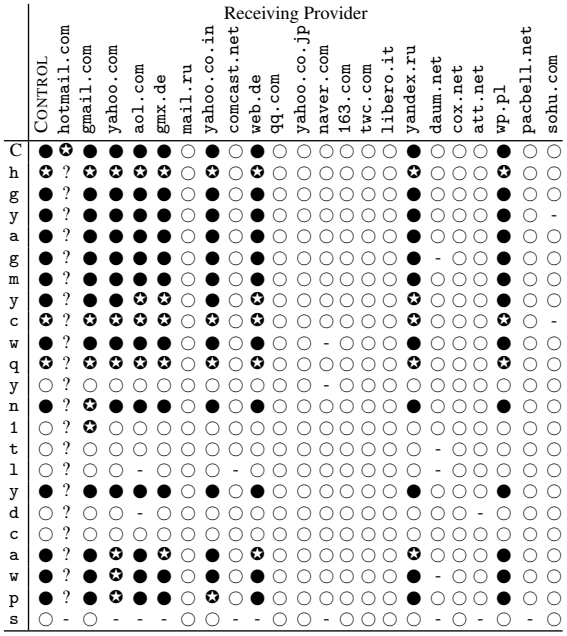

Table 6: Pairwise behavior of top mail providers. Each row is labeled with the first character of the provider's name. ● indicates TLS support, ✪ indicates TLS support was added within the past year, ○ indicates that the message was sent with no TLS whatsoever, ? means that the MTA link did not record any information about the protocol or use of TLS, and - means that the message did not go through.

The top 10 providers cover 70% of the users in the leaked Adobe user list. Given that all of the top 10 providers supported TLS at the incoming MTA, it is worth recasting the above numbers in terms of *users*. When weighted by the number of users, TLS support increased from 52% of users (top 1 million domains only) in 2014 to 89% in 2015, a substantial improvement largely due to Hotmail's adoption TLS at incoming MTAs.

5.3.5 Incoming MTA Server Certificates

Of the 10 select providers that used TLS, the incoming MTAs of 3 used mismatched but otherwise valid certificates. Beyond these providers, however, the certificates produced by the incoming MTAs ranged wildly in quality. Table 5 summarizes our findings. We are pleased to see that in the past year the percentage of valid certificates has risen, and self signed, expired and revoked certificates has fallen. Certificates that did not match the hostname of the incoming MTA did rise, which we attribute to the overall increase in TLS use, and the reduction of otherwise invalid certificates that would have previously been in another category.

5.3.6 Pairwise TLS Support

The premise of our large-scale analysis is that knowing outgoing and incoming MTA behavior allows us to infer how the two would behave talking to each other. To validate this, we sent email between the 22 select providers. Table 6 shows the results of this experiment. Each element of the table shows whether TLS was used on the inter-provider hop, based on `Received` header data. For example, a message from GMail to Hotmail was transferred from GMail to Hotmail over a TLS-secured session, represented by the ✪ mark in the first row labeled `g` (for `gmail.com`) and column labeled `hotmail.com`.

	DNSSEC	*SPF*	*DKIM*	*DMARC*	*SPF*	*DKIM*	*DMARC*	*SPF*	*DKIM*	*DMARC*
Domain	Implementation				DNS Lookup			Enforcement		
`hotmail.com`	○	●	○	●	●	●	●	✪	●	●
`gmail.com`	○	●	●	●	●	○	●	○	○	●
`yahoo.com`	○	○	●	◉	●	●	●	○	○	●
`aol.com`	○	●	○	◉	●	●	●	●	○	○
`gmx.de`	○	◉	○	○	●	○	○	●	○	○
`mail.ru`	○	●	●	●	●	●	○	○	○	○
`yahoo.co.in`	○	○	●	●	●	○	●	●	○	●
`comcast.net`	●	○	●	●	●	●	○	○	○	●
`web.de`	○	◉	○	○	●	●	○	●	●	●
`qq.com`	○	●	●	●	●	●	○	○	●	○
`yahoo.co.jp`	○	●	●	○	●	●	○	●	○	○
`naver.com`	○	●	○	○	●	○	○	✪	○	○
`163.com`	○	◉	●	○	●	●	●	✪	○	●
`twc.com`	○	●	○	○	○	○	○	○	○	○
`libero.it`	○	◉	●	●	●	○	○	○	○	○
`yandex.ru`	○	●	●	●	●	○	●	✪	○	○
`daum.net`	○	●	○	○	○	○	○	○	○	○
`cox.net`	○	○	○	○	●	●	○	○	○	○
`att.net`	○	○	●	○	●	●	●	○	○	○
`wp.pl`	○	○	●	○	●	○	○	✪	○	○
`pacbell.net`	○	○	●	○	●	○	●	○	○	○
`sohu.com`	○	◉	○	○	●	○	○	○	○	○

Table 7: DNS mechanism configuration and behavior of the top mail providers. Implementation indicates whether the provider has the correct DNS records for verifying messages they send. DNS Lookup shows whether the provider queried our DNS server for the selector for each feature when sending mail. Enforcement indicates whether the provider takes any action when receiving a message from a host that SPF forbids, an invalid DKIM signature, or rejecting DMARC records. ○ indicates no support, ● indicates support, ◉ indicates a strict DKIM policy of "reject" and ✪ indicates that the provider took action on mail sent from a host not listed in the SPF record.

A number of entries are absent (shown as "-"). These cases occurred when we could not send a message from one provider to another. We had particular trouble sending getting email from our `sohu.com` account accepted. It turned out that the `sohu.com` SMTP submission servers did not require a user to authenticate, allowing spammers to use their SMTP MSA as an open proxy. Entries marked "?" are cases where `Received` header analysis did not provide a conclusive indication one way or another. This happened with one provider, Hotmail, which used a protocol of `SMTPSVC`, that was used for both TLS and non-TLS connections.

Several abnormalities were observed in the 2014 measurement where both the sending provider and the receiving provider supported TLS but TLS was not used. For example `yahoo.co.in` sending to `aol.com`. We are unable to observe the connection between the two providers to know what caused this behavior, but it is possible that the two providers where using incompatible TLS ciphers, or where misconfigured. However all were resolved when performing the 2015 measurement.

There was only one abnormal result observed in the 2015 study. A message from `163.com` was transferred to `gmail.com` using TLS, even though the `163.com` outgoing MTA did not issue the `STARTTLS` command when send mail to our server.

5.4 SPF and DKIM

A scan of the Alexa and Adobe top million domains shows us that just over 40% have a valid SPF record, which covers 85.02%

Metric	*Alexa Hosts*	*Adobe Hosts*	*Adobe Users*
DNSSEC	3.40%	2.75%	4.92%
Valid	2.96%	2.12%	1.35%
Invalid	0.44%	0.63%	3.57%
DMARC	0.97%	0.90%	67.81%
None	0.73%	0.66%	51.29%
Quarantine	0.08%	0.06%	0.46%
Reject	0.16%	0.18%	16.06%
SPF	42.26%	43.60%	85.02%

Table 8: DNSSEC, DMARC and SPF status of the Alexa and Adobe top million hosts.

of Adobe users (Table 8). In the SPF Implementation column of Table 7, and the SPF column in Table 4, we display a ● if SPF is implemented, and a ◉ if the policy is strict. We define a strict policy for SPF as ending in "`-all`", which instructs the receiver to reject mail not from the correct origin. 15 of the 22 mail providers have SPF, but only 5 implement a strict policy. SPF use is high with most popular web services, except news sites, and is often strict.

The DNS Lookup columns of Table 7 show a ● if the provider made the necessary DNS query to lookup the SPF or DKIM record. The Enforcement columns shows a ● if invalid SPF or DKIM resulted in the message being placed in the users's spam folder. A ✪ indicates that the receiving MTA rejected the message at the SMTP layer. Almost all providers performed the SPF DNS query, but only 10 took action, half at the SMTP layer. 11 of the providers performed the DKIM selector lookup, but only 3 marked the message as spam. Every DKIM message with an invalid signature was allowed to successfully complete SMTP delivery to the MTA.

5.5 DNSSEC, DMARC, and ADSP

Of the top mail providers only `comcast.net` supports DNSSEC. About 3% of the Alexa and Adobe top million have DNSSEC, however 13% of Alexa and 23% of Adobe hosts with DNSSEC fail verification. We note that there are more Adobe users who have invalid DNSSEC that valid, which is caused by popular providers in the Adobe list having improper DNSSEC configuration.

For DMARC entries in the Implementation column of Table 7 and in Table 4, we display a ● if DMARC is implemented, and a ◉ if implementing a strict policy. A DMARC policy is strict if its policy is to reject invalid messages by setting "`p=reject`". DMARC is implemented by about half of the web services and top mail providers, including all the banks, and all but 1 of the commerce sites. About half of the web services with DMARC have a strict policy, most of which are banks or social sites. Only 2 mail providers had a strict policy. When receiving mail, 9 of the top providers performed a DMARC lookup, and 7 took action. 2 providers filtered messages with a strict DMARC policy without doing the DNS lookups which we attribute to the provider taking action on the DKIM signature failing before the DMARC check is done. The only provider to query for an ADSP record was `163.com`.

While SPF, DKIM, and DMARC are widely implemented, we found that SPF offers the strongest authenticity protection and impact on deliverability. We also note that email generated by the paper submission system of this conference was sent to us by a MTA that supports TLS, SPF and DMARC. (A competing conference management system, `easychair.org` only supports TLS.)

6. RELATED WORK

Two prior industry studies have reported on the use of TLS by MTAs. Facebook and Google released reports of observed SMTP TLS deployment as seen by their servers. Their results provide a faithful and valuable picture of server behavior from their vantage point. In this section we compare their results to ours and found agreement on common measurements. Our work provides a more complete view of TLS deployment, covering all parts of the message path. Of course, we also examined DKIM, SPF, DMARC, and DNSSEC, which provide mail authenticity and integrity.

6.1 Facebook Study

In 2014 Facebook reported TLS use when sending notification emails to their users for a day [2]. They reported 76% of incoming MTAs for unique MX records offered `STARTTLS` when sending emails to their users, and about half of the certificates pass validation. 58% of Facebook's outgoing notification email used TLS.

We found 54% of our unique MX records from hosts in the Adobe list allow TLS for receiving mail, which is lower than Facebook's findings. However we find 52% of users in 2014, and in 89% 2015 can receive TLS messages when ranked by the Adobe list. We also observed much higher certificate validation, 75.85% in 2015 and 79.14% in 2015.

6.2 Google Study

Google offers STARTTLS data on an ongoing basis as part of the Google Transparency Report.[6] Google reports 46% of outbound and 40% of inbound messages used TLS at the time of our second measurement (February 2015). Note that the Google report measures number of *messages* transported using TLS on the MTA–MTA link. The closest point of comparison is our estimate of the number of providers weighted by their frequency in the Adoble list (Sec. 5.3.4): 52% in March 2014 and 89% in February 2015.

More recently, Durumeric *et al.* [3] carried out a concurrent study similar to ours, evaluating STARTLS, DKIM, and SPF deployment as seen from Google. Their measurements were carried out in April 2015. During this period they reported TLS use for 80% of incoming connections, while we estimate that 89% of providers in the Adobe list, weighted by number of users, as described above. Durumeric *et al.* report that 74% of incoming mail at GMail does not have a DMARC policy. We found that 32% of providers in the Adobe list, weighted by frequency of occurrence, did not publish a DMARC policy. The difference is significant, and most likely due to the different mix of providers in the two measurement sets. The authors also report that 92% of mail arriving at Google had an SPF record for the sender. We found that 85% of providers in the Adobe list, weighted by frequency of occurrence, had SPF records.

In addition to reporting protocol use statistics, Durumeric *et al.* find evidence of TLS stripping attacks—man-in-the-middle attacks preventing a TLS connection from being established, forcing the mail transfer to take place in the clear. They report that up to 20% of email received by Google's incoming MTAs from certain countries showed evidence of TLS stripping.

7. DISCUSSION

In this section we consider the implications of our results in the presence of each of the three types of attacker we considered.

7.1 Passive Eavesdropping Attacks

A passive attacker can observe network traffic but cannot block or modify it. To protect against such an attacker, it is sufficient to establish a shared secret between sender and receiver. This is effectively what SSL without certificate checking enables, provided other requirements of the protocol are met. We consider a best case and a worst case scenario. By *best case*, we mean a scenario where

[6] https://www.google.com/transparencyreport/

the most secure option is chosen, if possible. In particular, we assume email is submitted and delivered using the most secure means, and that internal links are secure. By *worst case*, we mean that the least secure option is chosen: submission and delivery will use the least secure means.

Best case. All of the 22 select providers we considered provide at least one means of submitting and retrieving email over a secure connection, and 17 of the 22 use TLS on all means of submission and delivery. On the internal hops, we found that few providers use TLS internally. Nonetheless, internal hops may be secured by other means, so for the best case scenario, we assume that this is the case. Thus far, then, user submission, delivery, and provider-internal transport are secure. On the MTA–MTA link, however, TLS is not as widely deployed. Only 135 (about 28%) of the 484 of the pairs of connections between the 22 select providers would use TLS. That amounts to about 50% of the message traffic, when weighted by number of users. If we extend this to messages between select providers and all providers, the proportion of TLS-protected messages 74%.

Worst case. For the worst-case scenario, we assume a user chooses the least secure submission and delivery mechanism available at each provider. In fact, the providers that allow insecure mail submission (Table 3) are the same ones that do not support TLS at the outgoing MTA, and similarly for the delivery side. Thus, while our worst case scenario increases the risk, the number of messages transferred over secured links does not decrease.

If we do not assume that internal links are secure, then the only protection against eavesdropping by a passive attacker is provided by STARTTLS, then any internal hops not using TLS may be targeted by an attack. Of the select providers, only four, `aol.com`, `hotmail.com`, `web.de`, and `qq.com`, use TLS on all internal hops in outgoing mail. These four together cover about 34% of the users in the select provider set. If we consider all internal hops on the receiving and delivery path to be no less secure than the incoming MTA of a provider, then 24% of messages sent from select providers to all providers would travel along a TLS-secured mail path.

Thus, anywhere from 24% to 74% of messages from a select provider to a provider on the full list would be protected against a passive attacker along the entire message path. The former figure is if we allow infiltration of internal links, but no additional exposure introduced on the delivery, the latter, if the only exposure is on the MTA–MTA hop. We consider this an overall success, in view of the fact that it was achieved *at no cost to the user*.

7.2 Peer Forgery Attacks

Our results show that some providers will honor a sender's strict "`-all`" SPF policy, and some of the providers and email generators did have a fail-closed policy. With some exceptions, it is generally possible for a peer attacker to impersonate an email generator or provider to another provider. A domain owner publishing a strict SPF policy can be assured that some providers will not accept forged email from her domain.

DKIM use and enforcement is less widespread. Of the top five providers, only GMail and Yahoo! used DKIM, and only Hotmail marked a message with invalid DKIM signature as spam. However publishing a strict "`p=reject`" DMARC policy resulted in an invalid message being rejected. Only Yahoo! and AOL have such a strict policy, so we can only say with confidence that impersonating those two senders to Hotmail, GMail and Yahoo!, as well as a handful of other providers is not possible. A domain owner publishing a strict DMARC policy can be assured that at least the top three as well as a few others will honor the policy.

7.3 Active Eavesdropping Attacks

An active attacker has full man-in-the-middle capability. To protect against such an attack requires proper certificate checking. Unfortunately, we found that there is no certificate checking on the submission and delivery path except when using a Web mail interface, and no certificate checking at all on the MTA–MTA hop. Even a pair of users accessing mail exclusively via a Web browser would still be vulnerable to an active attacker on the MTA–MTA hop.

A man-in-the-middle attack does not require physically cutting into a link. BGP and DNS hijacking attacks would allow an attacker to redirect traffic to himself during a critical period. BGP security is still in standardization [5]. While DNSSEC is available, of the top 10 providers, only `comcast.net` has a DNSSEC signed `MX` record. (However the `comcast.net` incoming MTA did not support TLS.)

7.4 Active Tampering Attacks

As noted earlier, the only defense against an active attacker is DKIM with DNSSEC and a strict DMARC policy (or a bilateral agreement to verify DKIM signatures). Only one provider, Comcast supports DNSSEC. And of the select providers 14 performed some sort of verification, however only 5 actually enforced the policy. Given the low rates of DNSSEC adoption, the large relative number of invalid DNSSEC records (Table 8), and unenforced SPF and DMARC policies, we conclude that active attacks on message integrity will be unimpeded.

7.5 Recommendations

Our findings show that the Internet mail system is partially vulnerable to passive eavesdropping attacks and peer forgery attacks, and highly vulnerable to active attacks. Fortunately, as discussed earlier, it is possible to protect against even an active network attacker. The following recommendations summarize steps sufficient to achieve this level of security using currently-available protocols.

Recommendation 1: Use TLS. TLS support in SMTP, IMAP, and POP3 is stable and mature. All but one of the 22 provider supports TLS for SMTP mail submission; enabling TLS support at the MTA is the next step.

Recommendation 2: Fix certificates. In the select provider set, 6 of the 21 SMTP MSAs and 3 of the 10 MTAs supporting TLS provided certificates with a name that did not match the DNS name. This should be fixed.

Recommendation 3: Verify certificates. Certificates should be verified, including the host name, by all clients (MUAs and outgoing MTAs). Given the abysmally poor name matching, simply enabling host name verification will break over half of all message paths. An incentive, in the form of delayed mail delivery, may be useful in compelling mail server administrators to deploy TLS support and use matching certificates.

Recommendation 4: Require TLS. Many providers of the 22 we examined already require TLS use. Requiring TLS eliminates the risk posed by misconfigured MUAs. For MSAs and MDAs using STARTTLS, a way to configure the mail reader to require TLS should be provided.

RFC 3207 suggests that outgoing mail servers record the fact that a particular incoming MTA uses TLS and ensure that TLS use in future sessions. While RFC 3207 suggests "generating a warning," we believe a stronger response, perhaps delaying mail or requiring human operator intervention, may be appropriate. On the incoming MTA side, the compatibility requirement articulated in RFC 3207 requires incoming servers to accept mail without TLS. While this may be necessary in the general case, large providers should establish bilateral agreements regarding TLS use, and require that all connections to a provider that supports TLS take place over TLS.

Recommendation 5: Certificate pinning. To protect against rogue CA attacks, the providers should fix the set of each peer's allowed certificates or CAs. Given the overhead of maintaining such certificate information, this option may be limited to a few large providers.
Recommendation 6: Use DKIM and DMARC. Providers should verify sender identity and sign all outgoing mail. Providers should also publish a strict ("`p=reject`") policy. Major providers may also establish bilateral signing policies rather than relying on DMARC.
Recommendation 7: Enforce SPF and DKIM policy. Receiving providers should reject mail from unauthorized sender or mail with a missing or invalid DKIM signature from senders with a "`p=reject`" policy.
Recommendation 8: Use DNSSEC. DNS records should be authenticated to protect against active attacks. DNSSEC is the preferred method for doing so, and most TLDs, including `.com` and `.org`, are signed.

7.6 On Interoperability vs. Security

Security practitioners are often faced with a choice between interoperability and security. The case of email security is no different. Consider certificate verification, the subject of Recommendation 3. On the one hand, Postel's Principle—be conservative in what you send, be liberal in what you accept—advocates transferring mail even if the receiving MTA's certificate is invalid. On the other hand, accepting an invalid certificate leaves the session vulnerable to a man-in-the-middle attack. Making the right trade-off requires weighing interoperability and security on the same scale, something notoriously difficult to do. With two parties involved, the situation becomes even more complicated.

Let us consider Recommendation 3 in Game-Theoretic terms. We have two players, Sender and Receiver, representing the sending MTA and receiving MTA, respectively. Receiver has the option to install and maintain a valid certificate at some additional cost, or the option not to do so. When Sender connects to Receiver, he has the option to verify the certificate (rejecting an invalid one) or to accept all certificates (valid or not). Let us call Receiver strategies "Good certificate" and "Bad certificate" denoted G and B, respectively, and Sender strategies "Check certificate" and "Accept certificate" denoted C and A, respectively. There are four possible outcomes in our simple game. Three out of four outcomes (GC, GA, and BA) result in mail being transferred and one (BC) does not. The latter corresponds to the case where a sending MTA rejects an invalid certificate offered by a receiving MTA.

To evaluate the game, let the cost of maintaining a certificate be C. This is the direct cost to Receiver of playing strategy G and is borne entirely by Receiver. If mail cannot be transferred between Sender and Receiver, both incur a loss, perhaps in the form of brand damage, for failing to deliver or receive messages from each other. Denote the loss to the Sender by L_{S} and loss to the Receiver by L_{R}. Without loss of generality, let the utility of transmitting a message be external (i.e., zero), so that the motivation for transferring a message is to avoid the loss associating with not doing so. Finally, let V_{S} and V_{R} be the expected loss, to Sender and Receiver, respectively, associated with being vulnerable to a man-in-the-middle attack. Note that the expectation is taken over the probability of an attack taking place and the loss incurred from an attack. Alternatively, we can consider V_{S} and V_{R} to be brand damage associated with sending mail unencrypted between Sender and Receiver.

Putting these together, payoff matrix for our game is:

		Sender	
		Check cert	Accept cert
Receiver	Good cert	$(-C, 0)$	$(-C - V_{\mathrm{R}}, -V_{\mathrm{S}})$
	Bad cert	$(-L_{\mathrm{R}}, -L_{\mathrm{S}})$	$(-V_{\mathrm{R}}, -V_{\mathrm{S}})$

Thus, L_{R} and L_{S} are the interoperability penalties, V_{R} and V_{S} are the security penalties, and C is the cost of security, in out case borne by Receiver only.
Sender strategy. From Sender's point of view, let us say that Receiver plays strategy G with probability r_{G} and B with probability r_{B}. Sender should choose strategy C over A if:

$$r_{\mathrm{G}} \cdot 0 - r_{\mathrm{B}} L_{\mathrm{S}} > -r_{\mathrm{G}} V_{\mathrm{S}} - r_{\mathrm{B}} V_{\mathrm{S}} \quad \Rightarrow \quad V_{\mathrm{S}} > r_{\mathrm{B}} L_{\mathrm{S}}.$$

In other words, Sender will prefer to verify certificates if the expected loss from rejecting mail is less than the expected loss associated with being vulnerable to a man-in-the-middle-attack. Sender faces the basic unilateral security-functionality trade-off with security penalty V_{S} and functionality penalty $r_{\mathrm{B}} L_{\mathrm{S}}$. If the former is greater than the latter, Sender will chose the former. If the functionality penalty is greater than the security penalty, she will choose the latter.
Receiver strategy. From the Receiver's point of view, let the probability that Sender plays strategy C be s_{C} and the probability that Sender plays strategy A be s_{A}. Receiver would prefer strategy G over B if

$$-s_{\mathrm{C}} C + s_{\mathrm{A}}(-C - V_{\mathrm{R}}) > -s_{\mathrm{C}} L_{\mathrm{R}} - s_{\mathrm{A}} V_{\mathrm{R}} \;\Rightarrow\; s_{\mathrm{C}} L_{\mathrm{R}} > C.$$

In other words, Receiver will prefer to maintain a valid certificate if the expected loss from not receiving mail is greater than the cost of maintaining a certificate. Note that Receiver's strategy does not depend on V_{R}, the expected loss associated with being vulnerable to a man-in-the-middle attack. No matter how prevalent or severe man-in-the-middle attacks might be, Receiver's strategy depends only on Sender's expected behavior and the cost of maintaining a certificate. Put another way, *Receiver's strategy does not depend on security*.

Our results (Sec. sec:analysis-cert-out-mta) show that among the select providers $s_{\mathrm{C}} < 1\%$.
Equilibria. If $V_{\mathrm{S}} > L_{\mathrm{S}}$ then Sender will always check certificates, and, unless $L_{\mathrm{R}} < C$, Receiver will use a good certificate. In general, the cost of maintaining a certificate is significantly lower than the penalty for not receiving mail, $L_{\mathrm{R}} \ll C$. Thus, if security is more important than deliverability ($V_{\mathrm{S}} > L_{\mathrm{S}}$), the game has one equilibrium: GC.

On the other hand, if $V_{\mathrm{S}} < L_{\mathrm{S}}$, then our game has two equilibria: BA and GC. Note that Sender would always prefer GC; Receiver would prefer GC over BA if $C < V_{\mathrm{R}}$, which may be the case for some providers but not others.
Implications. Stated plainly, the the implications of this analysis is that senders will not enforce certificate validity until nearly all receivers have valid certificates, and receivers won't bother to use valid certificates until senders start refusing to send mail. One way out of this impasse is for senders to credibly threaten to refuse to send mail to receivers with invalid certificates or no TLS at all. The numbers of senders must be large-enough so that $s_{\mathrm{C}} L_{\mathrm{R}} > C$. In other words, the senders together should together generate more than C/L_{R} of all (non-spam) mail. We suspect that C to be much less than L_{R}, so that the market share of providers involved in the threat need not be very large. Certainly, the top three providers that together represent over 60% of the Adobe user list are well-positioned to effect such a change.

8. CONCLUSION

Modern email protocols provide means for achieving confidentiality, authenticity, and integrity during message transfer without any user involvement. TLS use with IMAP, POP, and SMTP provides message confidentiality even in the presence of an active man-in-the-middle adversary, while DKIM with DNSSEC ensures

authenticity and integrity. These guarantees come at the cost of trusting the email provider. While end-to-end mechanisms do not require such trust, user adoption of PGP and S/MIME is poor. Provider-deployed protocols considered in this paper provide a complementary path toward achieving some of the same security goals.

In this work we examined the use of these protocol by major email providers and email generators. We found that TLS support was common, however certificate verification was virtually non-existent, providing protection against a passive adversary only. SPF and DKIM use was also common, however all but a handful of providers used DNSSEC to protect the required DNS records. In addition, few providers enforced SPF policies or rejected messages with invalid DKIM signatures. More aggressive enforcement is required to protect against message forgery or active message tampering.

Acknowledgments

We would like to thank our system administrators Cindy Moore and Brian Kantor and are grateful for the feedback from the anonymous reviewers. This work was supported in part by the National Science Foundation grant CNS-1237264 and by generous research, operational and/or in-kind support from the UCSD Center for Networked Systems (CNS).

REFERENCES

[1] Google encrypts data amid backlash against NSA spying. *The Washington Post*, Sept. 2013.

[2] M. Adkins. The Current State of SMTP STARTTLS Deployment. https://www.facebook.com/notes/1453015901605223, May 2014.

[3] Z. Durumeric, D. Adrian, A. Mirian, J. Kasten, E. Bursztein, N. Lidzborski, K. Thomas, V. Eranti, M. Bailey, and J. A. Halderman. Neither Snow Nor Rain Nor MITM ... An Empirical Analysis of Mail Delivery Security. In *Proceedings of the 2015 Internet Measurement Conference (IMC)*, 2015.

[4] B. Gellman and A. Soltani. NSA infiltrates links to Yahoo, Google data centers worldwide, Snowden documents say. *The Washington Post*, Oct. 2013.

[5] S. Goldberg. Why is it taking so long to secure Internet routing? *Communications of the ACM*, 57(10):56–63, 2014.

[6] B. Taylor. Fighting phishing with eBay and PayPal. http://gmailblog.blogspot.com/2008/07/fighting-phishing-with-ebay-and-paypal.html, July 2008.

APPENDIX

This appendix provides additional information on the mail transport protocols described in the paper and their security extensions.

SMTP

The standard Internet mail transfer protocol is the Simple Mail Transfer Protocol (SMTP), standardized in 1982 by RFC 821. The original SMTP provides little more than a uniform mechanism for transferring messages between two hosts over a TCP connection. A common SMTP interaction consists of a client identifying itself using the `HELO` command and then presenting a message for delivery or forwarding. Only the sender and recipient are identified to the server explicitly (via the `MAIL FROM:` and `RCPT TO:` commands); the remainder of the message, including headers, are transferred as an uninterpreted sequence of bytes. As a simple common language between disparate mail systems, SMTP had no provisions for ensuring message authenticity, integrity, or confidentiality.

ESMTP. SMTP was updated in 1995 with the ability to extend the protocol using extensions. This version of SMTP is called ESMTP (RFC 1869, obsoleted by RFC 5321). An client indicates its support for ESMTP by issuing the `EHLO` instead of `HELO` command at the start of the session. If the server supports ESMTP, it responds with a list of supported extensions. One such extension is STARTTLS. STARTTLS allows a SMTP session to be secured with TLS, and is the preferred mechanism of doing so.

SMTP with STARTTLS

The STARTTLS extension was introduced in 1999 in RFC 2487 (now obsoleted by RFC 3207), allowing a SMTP connection to be secured using TLS. A server indicates support for the STARTTLS extension by including the `STARTTLS` keyword in the list of supported options sent to a client in response to the `EHLO` command. After issuing the `STARTTLS` command, a client can initiate a TLS handshake, after which the remainder of the session is secured.

SMTPS

The STARTTLS mechanism is the preferred way to secure SMTP sessions with TLS. It is also possible, however, to run the entire session over TLS from the start, as with HTTPS, by initiating the handshake immediately on connecting. We call this *direct* use in contrast to STARTTLS. When TLS is used directly for SMTP, the protocol is called SMTPS (less commonly, SSMTP). To avoid the need for servers to auto-detect TLS use, port 465 was reserved for this protocol. SMTPS has since been deprecated;however, many mail providers still use SMTPS for mail submission.

Requiring TLS. Regarding servers requiring the use of the STARTTLS extension, RFC 3207 states:

> A publicly-referenced SMTP server *must not* require use of the STARTTLS extension in order to deliver mail locally. This rule prevents the STARTTLS extension from damaging the interoperability of the Internet's SMTP infrastructure. A publicly-referenced SMTP server is a SMTP server which runs on port 25 of an Internet host listed in the MX record (or A record if a MX record is not present) for the domain name on the right hand side of an Internet mail address. (RFC 3207, Sec. 4)

The standard places a similar requirement on outgoing MTAs. In its discussion of man-in-the-middle attacks, however, RFC 3207 appears to back away from a strict reading of the interoperability requirement:

> In order to defend against such attacks both clients and servers *must* be able to be configured to require successful TLS negotiation of an appropriate cipher suite for selected hosts before messages can be successfully transferred. The additional option of using TLS when possible *should* also be provided. An implementation *may* provide the ability to record that TLS was used in communicating with a given peer and generating a warning if it is not used in a later session. (*ibid.* Sec. 6)

Requiring successful handshake. While a "publicly-referenced" server may not require a client to use the STARTTLS extension, once a client initiates a handshake after a successful STARTTLS command, both client and server may refuse to proceed:

> If the SMTP client decides that the level of authentication or privacy is not high enough for it to continue, it *should* issue an SMTP `QUIT` command immediately after the TLS negotiation is complete. If the SMTP server decides that the level of authentication or privacy is not high enough for it to continue, it *should* reply to every SMTP command from the client (other than a `QUIT` command) with the 554 reply code (with a possible text string such as "Command refused due to lack of security"). (*ibid.* Sec. 4.1)

What requirements clients and servers should enforce are a "local matter," according to the standard, however, two "general rules for the decisions" are offered:

> (1) A SMTP client would probably only want to authenticate a SMTP server whose server certificate has a domain name that is the domain name that the client thought it was connecting to.
>
> (2) A publicly-referenced SMTP server would probably want to accept any verifiable certificate from a SMTP client, and would possibly want to put distinguishing information about the certificate in the Received header of messages that were relayed or submitted from the client. (*ibid.* Sec. 4.1)

In Section 5.3 we test whether outgoing and incoming MTAs follow these recommendations. In it's discussion of security considerations, RFC 3207 warns:

> Both the SMTP client and server must check the result of the TLS negotiation to see whether an acceptable degree of authentication and privacy was achieved. Ignoring this step completely invalidates using TLS for security. The decision about whether acceptable authentication or privacy was achieved is made locally, is implementation-dependent, and is beyond the scope of this document. (*ibid.* Sec. 6)

Determining the degree of "authentication and privacy" achieved by TLS use is the main subject of this work. Section 5.3.2 shows few servers check the result of TLS negotiation.

POP3 and IMAP

The Post Office Protocol (POP), first specified in RFC 918, and the Internet Message Access Protocol (IMAP), first specified in RFC 1064, both allow a client to retrieve mail stored on a server (hop (e) in Figure 1). POP version 3 is often also called POP3.

From our point of view, both POP and IMAP provide the same service, namely message transfer along the last hop between MDA and MUA, identified as (e) in Figure 1. As with other hops, a passive attacker may eavesdrop on the communications between MDA and MUA. Both end-to-end and transport encryption protect against such an attack. An active attacker may also attempt to retrieve a target user's mail by impersonating the user. Protecting against such impersonation means that the MDA must be able to ensure that the MUA is acting on the legitimate user's behalf. Both POP3 and IMAP have traditionally achieved this using plain-text passwords. Although other mechanisms are available, passwords remain the main means of authentication today. Without transport encryption, a passive attacker can recover a user's password and use it to retrieve messages directly. Note that end-to-end encryption does not protect against such an attack; the attacker could still retrieve a user's encrypted mail, although the content of each message would be still be protected. An active attacker can also carry out a traditional man-in-the-middle attack. If a MUA does not properly authenticate a server, an attacker can recover the user's password and intercept messages retrieved by a user.

IMAP and POP with TLS

RFC 2595 introduced the STARTTLS extension to POP and IMAP using the extension mechanisms available in newer versions of each protocol. As in SMTP, the STARTTLS is used by the client to enable TLS. After a successful response, the client initiates the TLS handshake. RFC 2595 is more emphatic than RFC 3207 about certificate checking:

> During the TLS negotiation, the client *must* check its understanding of the server hostname against the server's identity as presented in the server Certificate message, in order to prevent man-in-the-middle attacks. (RFC 2595, Sec. 2.4)

RFC 2595 also unambiguously describes the identity checking process. Like RFC 3207, RFC 2595 warns:

> Both the client and server *must* check the result of the STARTTLS command and subsequent TLS negotiation to see whether acceptable authentication or privacy was achieved. Ignoring this step completely invalidates using TLS for security. The decision about whether acceptable authentication or privacy was achieved is made locally, is implementation-dependent, and is beyond the scope of this document. (*ibid.* Sec. 2.5)

IMAPS and POP3S. As with SMTP and SMTPS, before the introduction of the STARTTLS extension, the common means of securing IMAP and POP with TLS was to start the TLS session immediately. These protocols were called IMAPS and POP3S, and use TCP ports 993 and 995. Both are still used by email providers.

DKIM

DKIM (RFC 4861, updated by RFC 6387) introduces provider-generated message signatures intended to prevent forgery and tampering with email from a provider. A DKIM signature, covering the message body and a subset of the headers, is appended as a "`DKIM-Signature`" header to the message. In addition to the cryptographic signature, a DKIM signature also contains a *selector* identifying the key used to generate the signature. The corresponding public key is published in a DNS `TXT` record of

`selector``._domainkey.example.net`

where *`selector`* is the selector token provided with the signature. DKIM thus relies on DNS for key distribution. It is also worth noting that DKIM does not provide a means of advertising a signing policy. Thus, a message without a signature may be the result of the sender not using DKIM or an attacker stripping the signature.

SPF

SPF (RFC 4408, updated by RFC 7208) introduces a means for a sender to publish a policy identifying the network hosts which may originate mail from the sender. The policy is disseminated via a DNS `TXT` record for the domain. Like DKIM, SPF thus relies on the integrity of DNS. The SPF standard states that receivers "should" reject the message immediately during the SMTP session and communicate this explicitly to the sending MTA. If the message is not rejected in this manner by the incoming MTA, the MTA should add a "`Received-SPF`" or "`Authentication-Results`" header to record SPF failure. We know of no mail readers that interpret this header, however, built-in spam filters may use this as a signal.

ADSP and DMARC

From a security point of view, a major limitation of DKIM is that the absence of a signature is ambiguous: either the sender does not implement DKIM or an attacker is forging a message or tampering with a message. ADSP (RFC 5617) and DMARC (RFC 7489), the former now historical, provide a way for a sender to indicate that a DKIM signature should be expected. Both ADSP and DMARC publish this information via DNS `TXT` records in a special subdomain of the domain in question. As with the DNS-based mechanisms above, record integrity in the presence of an active network attacker requires use of DNSSEC.

DNSSEC

DNSSEC (RFC 2535 updated by RFC 4034 and others) provides a mechanism for authenticating DNS records using digital signatures. DNSSEC also introduces a hierarchical public key infrastructure mirroring the DNS hierarchy. The root key is used to sign each TLD, each TLD signs the next level, and so on. Verifying a DNSSEC signatures thus requires knowing the root public key fingerprint/digest. In addition to authenticating records, DNSSEC also authenticates negative results, so that an active attacker cannot convince a victim that a DNS record does not exist by forging a negative reply or blocking the legitimate reply.