
# Extended Hell(o): A Comprehensive Large-Scale Study on Email Confidentiality and Integrity Mechanisms in the Wild

Birk Blechschmidt, *Saarland University;*
Ben Stock, *CISPA Helmholtz Center for Information Security*

https://www.usenix.org/conference/usenixsecurity23/presentation/blechschmidt

This paper is included in the Proceedings of the 32nd USENIX Security Symposium.

August 9–11, 2023 • Anaheim, CA, USA

978-1-939133-37-3

Open access to the Proceedings of the 32nd USENIX Security Symposium is sponsored by USENIX.

# Extended Hell(o): A Comprehensive Large-Scale Study on Email Confidentiality and Integrity Mechanisms in the Wild

Birk Blechschmidt† and Ben Stock‡

† *Saarland University* ‡ *CISPA Helmholtz Center for Information Security*

*birk@blechschmidt.io, stock@cispa.de*

## Abstract

The core specifications of electronic mail as used today date back as early as the 1970s. At that time, security did not play a significant role in developing communication protocols. These shortcomings still manifest themselves today in the prevalence of phishing and the reliance on opportunistic encryption. Besides STARTTLS, various mechanisms such as SPF, DKIM, DMARC, DANE, and MTA-STS have been proposed. However, related work has shown that not all providers support them and that misconfigurations are common. In this work, we provide a comprehensive overview of the current state of email confidentiality and integrity measures, as well as the effectiveness of their deployment. On a positive note, support for incoming TLS connections has significantly increased over the years, with over 96% of reachable MXs in the top 10 million domains allowing for explicit TLS. Notably, 30% of presented certificates are invalid, though, with the majority of issues related to the presented hostnames. In light of this, all 47 providers we tested connect to hosts with expired, self-signed, non-matching certificates, making it trivial for attackers to intercept their connections. Our analysis also shows that still only around 40% of sites specify SPF, and even high-ranked providers like t-online.de do not enforce it. Similarly, while DNS lookups are performed for both DKIM and DANE, neither mechanism is validated or enforced by all providers. In addition, we show that MTA-STS is only slowly getting traction (six providers support it) and provide the first large-scale analysis into OPENPGPKEY and SMIMEA records. All in all, this still paints a grim yet slightly improving picture for the state of email security by late 2022.

## 1 Introduction

Even in days of instant messaging or Slack, email still is a cornerstone of digital communication. We first look at its historical evolution to understand why email comprises such a patchwork of protocols and multiple competing security mechanisms today. Our journey begins with the first standardization of SMTP in 1982 [39], as it serves as the base of SMTP today. Lacking cryptographic mechanisms, it does not protect the integrity or confidentiality of transmitted messages. This insecurity motivated the introduction of the STARTTLS extension, adding support for opportunistic encryption [20], i.e., to enable protection against a passive MitM attacker. To combat attacks like STARTTLS stripping, two competing standards have been proposed: DANE-TLSA [21] and MTA-STS [35]. DANE-TLSA leverages the security guarantees of DNSSEC to publish a certificate description of the receiving mail transfer agent (*MTA*). The sending MTA then ensures that the certificate presented by the receiving MTA matches this description. Although this is a stripping-resistant solution, the complexity required to implement DNSSEC has been a motivation for the proposal of MTA-STS, which instead relies on trust upon first use. The recipient publishes a DNS record instructing the sending MTA to fetch a security policy via HTTPS, authenticated by a trusted certificate authority (CA). The policy can then specify that STARTTLS must be used and that only certificates which can be validated against trusted roots are accepted.

The aforementioned mechanisms mainly aim to protect the confidentiality between MTAs rather than between end-users. They do not protect against the snooping of curious administrators or lawful interception at the provider level. Confidentiality and integrity on an individual level can be guaranteed by OpenPGP and S/MIME. Both support encrypting and signing email messages. Similar to DANE-TLSA, there are DANE bindings for OpenPGP keys and S/MIME certificates that allow for automated key/certificate distribution.

The fight against impersonation requires a notion of authenticity at the domain level. Without additional security mechanisms, it is unclear who may transmit email on behalf of `example.com`. At this point, the *Sender Policy Framework* (SPF), *DomainKeys Identified Mail* (DKIM), and *Domain-based Message Authentication, Reporting and Conformance* (DMARC) come into play. SPF dates back to 2006 [42] and is a DNS-based mechanism that enables sender domains to specify who may send emails on their behalf. The administrator of

| Prior Work | Year | STARTTLS | SPF | DKIM | DMARC | DANE-TLSA | MTA-STS | SMIMEA/OPENPGPKEY |
|---|---|---|---|---|---|---|---|---|
| Foster et al. [17] | 2015 | ⇔ | ⇔ | ⇔ | ⇔ | ✗ | ✗ | ✗ |
| Durumeric et al. [16] | 2015 | ⇐ | ⇒ | ⇒ | ⇒ | ✗ | ✗ | ✗ |
| Kambourakis et al. [25] | 2020 | ⇔ | ⇒ | ⇒ | ⇒ | ⇐ | ⇐ | ✗ |
| Lee et al. [31] | 2020 | ⇔ | ✗ | ✗ | ✗ | ⇔ | ✗ | ✗ |
| Maroofi et al. [36] | 2021 | ✗ | ⇒ | ✗ | ⇒ | ✗ | ✗ | ✗ |
| Tatang et al. [47] | 2021 | ⇐ | ✗ | ✗ | ✗ | ✗ | ⇐ | ✗ |
| Deccio et al. [12] | 2021 | ✗ | ⇔ | ⇔ | ⇔ | ✗ | ✗ | ✗ |
| Tatang et al. [48] | 2021 | ✗ | ⇐ | ⇐ | ⇐ | ✗ | ✗ | ✗ |
| Wang et al. [50] | 2022 | ✗ | ✗ | ⇔ | ✗ | ✗ | ✗ | ✗ |
| Our work | | ⇔ | ⇔ | ⇔ | ⇔ | ⇔ | ⇔ | ⇔ |

Table 1: Related Work Overview (⇐ and ⇒ denote inbound and outbound support investigations, respectively; ⇔ implies both, and ✗ means the mechanism was not considered.)

the sender domain publishes a DNS record that, e.g., contains or refers to the IP addresses of the email servers authorized to send emails from the domain. As a competing mechanism for SPF, DKIM was published as RFC 4870 in 2007 [13]. With DKIM, the administrator of the sender domain publishes a cryptographic public key through DNS. When sending an email, the MTA signs each message with the corresponding private key, and the mail server adds a header containing the message's signature. Importantly, the header also indicates *which* DKIM key should be selected for validation. This then instructs the receiving mail server to query the corresponding DNS entry (`selector._dkim.domain.com`) and retrieve the public key to validate the incoming mail. Unfortunately, SPF and DKIM both have shortcomings, which motivated yet another security mechanism: DMARC [29]. While SPF does not authenticate the sender domain displayed by most user agents, DKIM per se allows emails to be signed from entirely unrelated domains. Given the opt-in nature of DKIM, when a mail server receives an email without a DKIM signature, this can be because of two reasons: the sender does not support DKIM or the email was spoofed. DMARC provides a remedy to this problem. A sender domain that supports DMARC publishes a DNS record containing instructions on handling emails that do not pass SPF and DKIM validation. More specifically, DMARC requires one mechanism to successfully authenticate the email domain, which is in alignment with the domain in the `From` header, which most user agents display. Hence, compliant mail servers can check not only the SPF record (which is published under a fixed DNS entry, i.e., as a `TXT` record for the sender domain), but also require a valid DKIM signature when SPF cannot authenticate the domain. This way, DMARC closes the loophole for attackers who could otherwise simply omit any DKIM headers.

The security of the email ecosystem has been the focus of several studies before, as shown in Table 1. Prior work has already looked into STARTTLS [16, 17, 25]. SPF, DKIM, and DMARC have been studied by [12, 16, 17], and [36]. Wang et al. [50] were the first to perform a large-scale analysis on DKIM. Lee et al. [31] have looked at DANE-TLSA in isolation, and a first analysis on MTA-STS has been performed by Tatang et al. [47], which the authors followed up with an analysis of deployment and misconfigurations of SPF, DMARC, and DKIM. Kambourakis et al. [25] have published a tool to test the security mechanisms of single providers online.

We build upon prior work by replicating and extending previous experiments. In doing so, we present a holistic measurement of the status quo of email security as of 2022. In doing so, we make the following contributions:

- We provide a comprehensive overview of security mechanisms supported by popular email providers and their behavior under failure conditions. To this end, we replicate the findings from [31], [10], and [17]. We find that the state of email security has barely improved, with SPF not gaining more traction than in 2015 and major providers not enforcing even this simple standard. Moreover, not a single mail provider refused to connect to a host presenting invalid certificates. Our results indicate that misconfigurations for TLS certificates are far too widespread for mailservers to take a more aggressive stance for connection security, given that a third of the discovered certificates fail validation.
- We are the first to investigate *outbound* support for MTA-STS by popular email providers and evaluate the effectiveness of its deployment.
- We perform the first large-scale analysis on `OPENPGPKEY` and `SMIMEA` mechanisms and identify their users. In addition, we find, report, and remedy implementation errors allowing for encryption stripping attacks.

## 2 Background

This section provides an overview of the fundamentals of electronic mail and the security measures that have been added retrospectively to provide confidentiality and integrity.

**SPF** The Sender Policy Framework (SPF) is a simple means to authenticate the use of email domains. To use SPF, a sender publishes a DNS `TXT` record, providing mechanisms to resolve IP addresses allowed to send emails from the domain. These mechanisms include `ip4` and `ip6` to specify IP address ranges, `mx`, `a`, `aaaa` and `ptr`, which are resolved to IP addresses by DNS lookups, an `all` mechanism matching every IP, as well as an `include` mechanism to include other SPF records. Qualifiers such as `+`, which is the optional default qualifier, or `-`

declare how a receiver should classify emails matching a mechanism. For example, `v=spf1 ip4:1.2.3.0/24 +a:mx.example.org +mx:example.com -all` indicates that email from `1.2.3.0/24`, from the IPv4 addresses of `mx.example.org`, and those coming from the MX (Mail eXchange) responsible for `example.com` passes the SPF test while email from other sources fails it. SPF also supports an `exists` mechanism, which in conjunction with macros, enables dynamic DNSBL-like policies. It is noteworthy that SPF authenticates the domain from the SMTP `MAIL FROM` command or the domain provided with the SMTP `HELO` or `EHLO` commands. It does not authenticate the `From` header defined in RFC 5322 [41], which is displayed by most email clients.

**DKIM** DKIM, which is short for DomainKeys Identified Mail [30], is another approach to email authentication. With DKIM, emails are cryptographically signed with a key whose public part is published in a DNS `TXT` record, and signature information is added to the email as `DKIM-Signature` header. Such a header comprises a `tag=value` list that contains, among others, the signature computed over a hash of the message body and the headers defined in the signature header, the domain for which a message is signed, and a selector. Selector and domain define where to find the key used for signing. If the selector is `hello` and the domain is `example.org`, the verifier will look for the key at `hello._domainkey.example.org`. Note that the domain may be entirely unrelated to the domains from the `EHLO` command, the envelope sender, or message originator headers, such as `From`. DKIM also supports the `l=` tag, which allows message signing to be constrained to the first N number of bytes as specified by the tag. Chen et al. [10] present body spoofing attacks allowing to *overlay* a second body if the `Content-Type` header has not been signed by DKIM.

**DMARC** The lack of relationship between the email author as indicated by the `From` header and the DKIM signature domain is one motivation for DMARC (Domain-based Message Authentification, Reporting and Conformance) [29]. Essentially, DMARC participants publish a policy as a DNS `TXT` record for the `_dmarc` subdomain that can be used by receivers to determine whether emails are authentic. An email passes DMARC if it passes SPF or DKIM validation (or both), requiring that the SPF or DKIM domain matches the domain of the address in the `From` header of the email. In DMARC terminology, this additional requirement is referred to as identifier alignment. A DMARC policy can indicate to either reject or quarantine email upon authentication failure. Subdomains are handled by first querying the DMARC record of the subdomain. If such a record does not exist, the subdomain policy defined by the record of the eTLD+1 domain must be consulted.

**DANE** The shortcomings of STARTTLS motivated the introduction of DANE, an acronym for DNS-based Authentication of Named Entities [21]. DANE-TLSA leverages the integrity guarantees of DNSSEC by publishing signed `TLSA` resource records that contain four fields: certificate usage, selector, matching type, and certificate association data. The certificate usage field can be one of four values: PKIX-TA, PKIX-EE, DANE-TA, DANE-EE [18]. In the case of PKIX-TA, the client validates the certificate against a PKI trust anchor, such as Let's Encrypt. In the case of PKIX-EE, the certificate must match the pin *and* needs to be signed by a trusted CA. DANE-TA and DANE-EE work analogously, except that the trust anchor or certificate may be self-signed. The selector specifies whether the full certificate or only the public key is used for validation. The matching type specifies whether a hash, the full certificate, or the public key is used for comparison. Finally, the certificate association data contains the data the presented certificate is matched against, either as a hash or the full hex-encoded bytes. In other words, a `TLSA` record provides instructions on how to validate a certificate that is presented upon a TLS connection. RFC 7672 [15] suggests leveraging `TLSA` records as a downgrade-resistant mechanism for SMTP transport security. If an MTA implements opportunistic encryption according to this RFC, a `TLSA` record for SMTP indicates that STARTTLS must be used. As an example, if `mx.receiver.com` were the mail server for `receiver.com`, it would publish a `TLSA` record at `_25._tcp.mx.receiver.com`, where 25 is the SMTP port. A sending MTA supporting DANE-TLSA would then use DNSSEC to query the `TLSA` record for `_25._tcp.mx.receiver.com` and compare the certificate served by `mx.receiver.com` to the description as obtained through the `TLSA` record.

**SMIMEA and OPENPGPKEY** TLS only protects the confidentiality between sender-MTA and receiver-MTA. It does not protect confidentiality against the provider of the email infrastructure. Therefore, OpenPGP and S/MIME exist as competing mechanisms. They both provide encryption and signing capabilities using asymmetric cryptography. To solve the problem of key distribution, `SMIMEA` [22] and `OPENPGPKEY` [52] records allow the publishing of S/MIME certificates and OpenPGP keys in DNS, leveraging the security guarantees of DNSSEC. They both work similarly by constructing a DNS name for which the records are queried. The local part of the destination email address is hashed and prepended as the left-most DNS label to either the `_smimecert` or the `_openpgpkey` subdomain of the email domain. While `OPENPGPKEY` records only comprise an OpenPGP key, `SMIMEA` records additionally contain certificate usage, selector, and matching type fields analogous to DANE-TLSA.

**MTA-STS** The complexity of DNSSEC has caused DANE deployment to be very slow. For this reason, SMTP MTA

Strict Transport Security (MTA-STS) has been introduced as an alternative security measure for environments in which "deploying DNSSEC is undesirable or impractical" [35]. MTA-STS works similarly to HTTP Strict Transport Security and relies on trust upon first use. To implement MTA-STS, a participating policy domain, such as `example.org`, publishes a DNS `TXT` record containing an ID for `_mta-sts.example.org` indicating that it makes use of MTA-STS. When a supporting SMTP client transmits an email to `example.org` for the first time, it will check for the presence of the DNS record and, if present, fetch a policy from `https://mta-sts.example.org/.well-known/mta-sts.txt`. The policy states for how long it may be cached and the MX servers for which it is valid. The client then caches the policy accordingly. The cache can be updated prematurely by changing the ID inside the DNS record, upon which the policy is fetched anew. When the client has cached a policy for a domain that is in enforced mode, email transmission to that domain is limited to the MX servers included in the policy, and transmission only takes place when a valid PKI certificate is used for STARTTLS. Assuming the perfect implementation of MTA-STS, this mechanism is still vulnerable to active man-in-the-middle (MITM) attacks: an attacker can either suppress the DNS response or disallow the connection to the HTTPS source for the policy. In both cases, an MTA cannot determine whether a policy is in place and, therefore, cannot enforce connection security. In fact, as we show in Section 5.2, the vast majority of domains with a DNS entry do not serve an HTTPS policy file.

## 3 Methodology

This section describes the methodology of our study.

### 3.1 Provider-Scoped Tests

For the providers, we first looked at closely related work to identify their used lists [17, 31]. We then augmented this with a set of known security-aware providers [5] as well as those popular globally and in Germany [3, 4]. In total, this provides us with a list of 47 providers for which we can register accounts, of which we share 11/20[1] with Foster et al. [17] and 24/29 with Lee et al. [31]. For a full comparison of the providers, we refer the reader to Table 12 in the appendix. Given the manual nature of registration, sending emails, verifying delivery status, etc., we ran our experiments from September to November 2022.

#### 3.1.1 Outbound Scenarios

To test whether email providers support SPF, DKIM, and DMARC validation for incoming email, we send emails from our server, i.e., outbound email, to accounts we registered with them. We send each email with a different server configuration, including misconfigurations for which we expect the delivery to fail. The parameters of scenarios for outbound email include DNS records for *the-email-study.com*, for which we operate an authoritative DNS server, and the configuration of the outgoing SMTP session, including the email message itself. This allows us to observe whether email providers request SPF, DKIM, DMARC, or TLSA records and whether they adhere to our SPF and DMARC policies. We also introduce two scenarios to find UI mismatch attacks from Chen et al. [10] for additional email providers. For each scenario run, we dynamically create a dedicated DNS zone below *the-email-study.com*. For each case, an MTA delivering emails, even though it should not, represents a security issue.

**Standard**: The email server is correctly configured for sending emails with the goal of maximum deliverability. Outgoing email is DKIM-signed with a 2048-bit key, which is published in the DNS. The DMARC policy is `p=reject; sp=reject`, i.e., both non-passing emails from the domain itself (**p**olicy) and subdomains (**s**ubdomain **p**olicy) should be rejected. The SPF policy explicitly allows our server IP address (`+`) and disallows other senders (`-all`). The scenario makes use of the `include` mechanism. If the receiver queries the TXT record for the subdomain specified by the mechanism, we can infer that the provider may consider SPF when delivering an email. Similarly, we can detect general support for DKIM and DMARC. If STARTTLS is available, it is used for email transmission.

- **No TLS**: This scenario is equivalent to the standard scenario but without STARTTLS. This is to ensure the delivery of emails to providers who present certificates that are too weak. All following scenarios also do not use STARTTLS for maximum compatibility.
- **SPF only**: The SPF policy is configured to produce a `pass` result for our server IP address. No other security measures are configured.
- **DKIM only**: Emails are signed with a 2048-bit DKIM key, which is correctly published in DNS. No other security measures are configured.
- **SPF and DKIM**: This scenario combines both previous scenarios to check their combined effect on the deliverability of our emails.
- **Insecure**: The email server is configured without any security mechanisms. No SPF and DMARC records are set up for the test domain, and emails are not DKIM-signed. This scenario is used for comparison with the SPF and DKIM scenarios. It enables us to assess the impact of SPF and DKIM on the deliverability of email.
- **SPF ?, DMARC -**: The DMARC policy is `p=reject; sp=reject`, while SPF and DKIM are not used. This

[1] We deduplicate providers with different eTLDs, e.g., gmx.de and gmx.net

scenario tests general DMARC support. Email providers honoring DMARC should reject this message.

- **SPF -, DMARC -**: The SPF policy is configured to explicitly disallow our server IP address, and the DKIM key used to sign the email is not published in the DNS. A provider that does not support DMARC should reject this message if it takes action on SPF failures.
- **SPF -, DMARC ~**: This scenario is equivalent to the previous scenario, except that the DMARC policy is `p=quarantine; sp=quarantine`. It tests support for less rigid DMARC policies.
- **Parent reject**: A DMARC record with `sp=reject` is published for the parent domain. Emails are not DKIM-signed and there is no SPF record. This is to check whether providers correctly implement subdomain policies.
- **Double From 1**: The mail server is configured as in the standard scenario. A second `From` header with an email address from another domain with a DMARC policy of `p=reject` and no SPF record is inserted before the DMARC-aligned `From` header to detect possible UI mismatch attacks [10]. Note that since DMARC is only meant to protect a single `From` header, attacks using the `Sender` header [37] are out of scope for our analysis. Specifically, MTAs should reject emails with a second `From` header, given that the DMARC RFC explicitly states this "renders the message invalid" [29].
- **Double From 2**: This scenario is equivalent to *Double From 1*, except that a non-matching `From` header is added after the DMARC-aligned `From` header.

#### 3.1.2 Inbound TLS Scenarios

Similar to the scenarios for outbound emails, we also introduce scenarios for inbound emails to detect the correctness of DANE-TLSA and MTA-STS implementations, as well as STARTTLS and DNSSEC support. The parameters of these scenarios additionally include the settings of our receiving MTA, such as STARTTLS support and the certificate to be served. For each scenario, an MTA connecting to us when it should instead close the connection is a security problem, as it may allow a MitM attacker to steal email traffic.

- **Normal**: The email server uses a self-signed and expired certificate, which matches neither the name of the recipient domain nor its `MX` record. No security measures are implemented. If the email is delivered nonetheless, the sending MTA accepts self-signed certificates, causing the delivery to be trivially vulnerable to man-in-the-middle attacks. In contrast to Lee et al. [31], we do not have separate scenarios for CA-signed certificates with non-matching names or CA-signed certificates that have expired, as all providers already accepted self-signed certificates (see Section 4.2), rending additional certificate configurations redundant.
- **Wrong MX record sig.**: The `RSIG` record of the `MX` record is invalid. DNSSEC-aware MTAs should not deliver emails in this scenario.
- **TLSA no cert.**: The email server is configured to not support STARTTLS at all while a `TLSA` record is served. This scenario allows uncovering implementations that implement `TLSA` but do not use it for opportunistic encryption as suggested by [15]. A mail server that delivers emails in such a scenario is still vulnerable to network attackers performing STARTTLS stripping [38].
- **TLSA wrong data**: Our email server supports STARTTLS and we serve a `TLSA` record with association data that does not match the presented certificate. A provider that implements DANE-TLSA correctly should reject delivering mail in this scenario. This and the following scenarios are equivalent to those from Lee et al. [31].
- **TLSA wrong usage**: Our email server supports STARTTLS and we serve a `TLSA` record with association data matching the presented certificate. However, the `TLSA` record indicates that the certificate is a `PKIX-EE` certificate, while we serve a self-signed certificate.
- **TLSA wrong name**: Our email server supports STARTTLS and we serve a `TLSA` record with association data matching the presented self-signed certificate. However, the certificate does not match the name of the `MX` record.
- **STS connect**: A valid certificate signed by *Let's Encrypt* is served during STARTTLS. To detect whether an MTA generally supports MTA-STS, we also serve an MTA-STS DNS record along with an MTA-STS policy that is not enforced, using a web server with a certificate from *Let's Encrypt*.
- **STS enforce 1st.**: We serve a self-signed certificate during STARTTLS while the MTA-STS policy we serve is enforced with a short `max_age`. When MTA-STS supporting providers connect to our server, they fetch the policy for the first time. They should not deliver the email in this case.
- **STS enforce long**: We serve a valid certificate during STARTTLS while the MTA-STS policy we serve is enforced with a very long `max_age`. This is a helper scenario for the following six misconfiguration scenarios in which email delivery should be thwarted by the policy being cached. That is, we first run this scenario such that the MTA-STS policy gets cached by the destination provider. Afterward, we run the following six scenarios.

In contrast to non-MTA-STS scenarios, these scenarios always get the same provider-specific DNS zone name assigned, as policies are cached by domain.

- **STS serving no TLS**: We rely on the cached policy from the *MTA-STS enforce long* scenario and keep serving the MTA-STS DNS record and policy like in the *MTA-STS enforce long* scenario. We misconfigure our email server to not support STARTTLS.
- **STS serving self-sig.**: In this scenario, we analogously misconfigure our email server to serve a self-signed certificate during STARTTLS.
- **STS serving expired**: An expired but otherwise valid certificate is served during STARTTLS.
- **STS cached no TLS**: We rely on the cached policy from the *MTA-STS enforce long* scenario but do not keep serving the MTA-STS DNS record or policy. We misconfigure our email server to not support STARTTLS. Thus, we can detect whether a provider only processes MTA-STS policies that are currently served.
- **STS cached self-sig.**: A self-signed certificate is served during STARTTLS with a cached policy.
- **STS cached expired**: A valid but expired certificate is served during STARTTLS, again with a cached MTA-STS policy.

## 3.2 Domain-Scoped Tests

As the basis for our large-scale analyses, we rely on the DomCop Top 10M list [14]. This is the largest publicly available list of domain names, albeit based on web traffic rather than email traffic. Even though prior work suffers from similar limitations (e.g., Durumeric et al. [16] relies on Alexa's top million), we nevertheless checked the overlap with email domains in the Adobe leak, a list of emails leaked in a 2013 hack [49]. This showed that of the 153M emails in that dataset, 88% have a domain suffix that is contained in the DomCop list. Therefore, we conclude that DomCop is applicable to measure the security of the vast majority of email traffic.

### 3.2.1 SPF and DMARC

For SPF, we call the `check_host` function from [28] for each record and supply with the target domain and an IP address that is under our control. If the function evaluates to `pass`, we have found a misconfiguration. For DMARC, duplicate records or records with syntax errors can thwart the original intention of the records' author. While the author believes to have configured DMARC correctly, RFC-compliant software will, in fact, reject those records. Therefore, we fetch the `TXT` records for the `_dmarc` subdomain and filter those for which there is at least one record starting with `v=DMARC`, as there may be unrelated `TXT` records for the wildcard domain if DMARC is not configured. We then parse the records for these domains using a suitable software library and count syntax errors and duplicates that were correctly parsed.

### 3.2.2 MTA Strict Transport Security

For MTA-STS, domains first need to specify a TXT record under `_mta-sts.domain.com` to indicate that the MTA should look up the policy through HTTPS. Therefore, we first resolve these DNS first and check for their syntactical validity. For those domains, we then collect the policy and compare the specified MX in the policy with the ones set through DNS. Finally, to validate if a domain could receive emails with MTA-STS enabled, we collect the certificate from the allowed MXs to see if they pass validation.

### 3.2.3 OPENPGPKEY and SMIMEA

To investigate the use of `OPENPGPKEY` and `SMIMEA` in the wild, we follow two approaches. First, we use a simple DNS-based enumeration approach. As the local part of an email address is hashed separately from the domain part to obtain the DNS name for the corresponding DNS record, we can precompute the hashes of the 119 most common email address local parts (which we obtain from the well-known Adobe leak and from [7]). We prepend these hashes to the dedicated subdomains for OpenPGP keys and S/MIME certificates and perform the respective DNS lookups for each domain.

As a second approach, we leverage that empty non-terminal nodes in the DNS graph return a `NOERROR` response in contrast to non-existing nodes returning an `NXDOMAIN` status. As an example, to detect whether the *example.org* domain supports `SMIMEA`, we query the `SOA` record for a random subdomain, as well as the `_smimecert` subdomain. If the `SOA` query for the random subdomain returns `NXDOMAIN` while it returns `NOERROR` for the `_smimecert` subdomain, we infer that the mechanism is supported. Similarly, we assume that the mechanism is supported when a `SOA` record for the `_smimecert` subdomain indicates zone delegation.

After we have inferred that a domain supports either mechanism, we attempt to enumerate the zone that is authoritative for the mechanism subdomain in three ways. First, we try to perform a zone transfer of the zone. Second, if the zone makes use of NSEC, we perform NSEC walking. Both approaches yield DNS records below the mechanism subdomain in plain text. Third, if the zone uses NSEC3 instead, we perform a dictionary and a combinatorial attack on the NSEC3 hashes using a custom hashcat [2] module. To this end, we use the email local parts from the well-known Adobe leak, CrackStation's Password Cracking Dictionary [1], as well as combinations of first and last names that occurred at least 50 times in a Facebook leak [6, 40].

| | gmail.com | yahoo.com | yandex.ru | zoho.com | aol.com | icloud.com | t-online.de | protonmail.com | web.de | outlook.com | naver.com | fastmail.com | laposte.fr | tutanota.com | vodafone.de | 163.com | mail.com | seznam.cz | mail.ru | rambler.ru | mynet.com | posteo.de | mailfence.com | 1und1.de | sapo.pt | daum.net | thexyz.com | hushmail.com | freenet.de | sina.com | mailbox.org | mail.de | interia.pl | hosteurope.de | inbox.lv | runbox.com | strato.de | startmail.com | gmx.com | privatemail.com | df.eu | juno.com | rediffmail.com | kolabnow.com | bol.com.br | freemail.hu | eclipso.de |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SPF Pub. | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| SPF Req. | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ |
| DKIM Sig. | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ |
| DKIM Req. | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| DMARC Pub. | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✗ | ✗ | ✓ | ✓ | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | ✗ | ✗ | ✓ | ✓ | ✗ | ✗ | ✓ |
| DMARC Req. | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| MTA-STS Pub. | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | ✓ | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| MTA-STS Req. | ✓ | ✓ | ✗ | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |

Table 2: SPF, DKIM, DMARC, and MTA-STS analysis per provider, showing where providers publish the corresponding entry for their own domain (DNS) and whether they request them for incoming email

### 3.3 MX-Scoped Tests: STARTTLS and DANE

In addition to tests that directly relate to the email domain, security mechanisms such as TLS connections are bound to the configured MX. Hence, for all tests related to STARTTLS and DANE, we first resolve the MX for each of the 10 million domains and apply our measurements to them. We conducted an updated measurement on June 6, 2023.

Here, we connect to each resolved MX through port 25 and ask it for *explicit* TLS through STARTTLS. Preliminary scans showed that only 19% of MXs support implicit TLS through port 465. In cases where both implicit and explicit TLS were available, the presented certificates were shared for 96.9%. This highlights that relying on port 25 alone gives a comprehensive and accurate picture of TLS. By collecting the certificates, we could then evaluate whether these match the given hostname, are not expired, and are properly signed by a CA. In addition, we also investigated the usage of DANE and whether the presented certificates match the ones pinned through DANE. For this, we retrieved the TLSA record for the corresponding DNS name (i.e., `_25._tcp.MX`) and compared the DANE pin to the presented certificate.

## 4 Provider-Based Results

Before investigating the exact behavior of providers for each of the tested cases, we analyzed the awareness and support of the security mechanisms we study. We conduct this analysis for SPF, DKIM, DMARC, and MTA-STS. Here, we analyzed both their publication for a security mechanism, e.g., whether they publish an SPF record for the email domain as well as the requests for the mechanism, i.e., whether they would even request the DNS record of SPF for incoming email. The overview of the results is shown in Table 2. The SPF rows show whether a provider publishes or requests SPF records, while the DKIM rows indicate whether a provider signed emails sent to our server and whether the provider performed a DNS request for our DKIM selector when processing incoming emails. The semantics of the DMARC and MTA-STS rows follow the semantics of the SPF rows.

For SPF, we notice that except for *t-online.de* all providers either publish or request an SPF record. *freemail.hu* is the only additional provider that does not request our SPF record. In the case of *strato.de*, which is used with a custom domain, SPF publication is not enabled by default, but it can be enabled through the customer menu. Comparing the results to those of Foster et al. from 2015 [17], we notice that since then, *yahoo.com* has added an SPF record. Interestingly, all providers performed a DNS lookup for the DKIM key, whereas only half performed a DNS lookup for DKIM keys in [17]. Of the intersecting providers, *gmail.com*, *Yandex*, *naver.com*, and *daum.net* have added DKIM since. We also notice discrepancies between providers signing messages with DKIM and providers fetching the DKIM selector. As a reason, we suspect that providers prioritize protecting their users from spam, while the lack of a DKIM signature for outbound mail has no immediately apparent effect as SPF is sufficient to pass DMARC. Also, when providers implement DMARC validation for incoming mail, they must also implement DKIM validation. As indicated by the ✓ symbol, *mailfence.com*, *juno.com*, and *freemail.hu* make use of the `l=` tag. As they sign the `Content-Type` header, however, they are not vulnerable to the body spoofing attacks described by Chen et al. [10].

Only 13 out of 47 providers do not publish a DMARC record, while 17 do not request one. This shows that DMARC has gained traction since the publication of Foster et al. [17], which found that only nine out of 22 providers performed a DMARC lookup at that time.

As MTA-STS is a very recent protocol, only 17 providers publish a policy, while six request the policies for the senders of inbound emails. This discrepancy is easily explained by the low effort it requires to set up a DNS TXT record and an HTTPS server serving a policy when the receiving MTA already serves a valid Web PKI certificate. In comparison, adding MTA-STS support to a sending MTA is error-prone and complex as MTA-STS was often not supported by MTA software natively [47]. (Since that publication, *sendmail* has added plugin support and *MDaemon* has added native support.) Further, for effective deployment, MTA-STS requires a shared cache for all sending MTAs of a provider, which needs to be configured and secured.

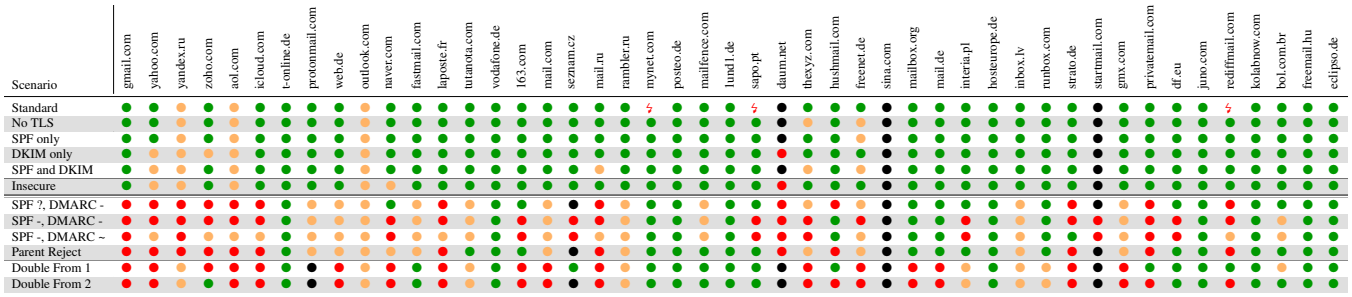


Table 3: Email delivery for outgoing mail under different configuration scenarios. ● indicates delivery to the inbox, ● indicates delivery to the spam/junk folder, ● indicates rejection during the SMTP session and ● indicates blackholing, i.e. acceptance during the SMTP session without delivery. ϟ indicates too weak TLS parameters.

## 4.1 Outbound Scenario Results

The results for the different scenarios outlined in Section 3.1.1 are shown in Table 3. We first note that our email is *not* always delivered in the standard scenario. In the cases of *daum.net*, *sina.com*, and *startmail.com*, our emails are blackholed, i.e. the receiving MTA accepts the emails via SMTP without delivering them to our inbox, spam, or junk folder. Nevertheless, for *daum.net* we can still infer that a missing or failing SPF record serves as an SMTP-level knock-out criterium. Similarly, *startmail.com* is less strict but rejects emails from senders with a failing SPF policy. *sina.com* blackholes all our emails. As emails from Gmail are also not received by our account at *daum.net*, we identify this to be an issue with the account. In contrast, the reason is likely an IP address or domain reputation issue in the case of *startmail.com* and *sina.com*.

For *mynet.com*, *sapo.pt*, and *rediffmail.com*, the standard scenario failed, as indicated by the ϟ symbol, because our implementation used OpenSSL's default security level for TLS connections. As their DH key was too small, we configured the other scenarios to not use STARTTLS. Obviously, these providers should upgrade their TLS configuration.

Some providers deliver emails to our inbox in all scenarios. In those cases, senders can be trivially spoofed. This even includes Posteo, which is otherwise known as a secure email provider, although headers of emails in our inbox show that Posteo validates SPF, DKIM, and DMARC. While this is not in violation of the SPF, DKIM, and DMARC specifications that leave final message handling up to local policies, it is not immediately apparent why a provider chooses not to reject email based on SPF or DMARC failures.

Moreover, setting corresponding policies for one's own domain through DMARC is not sufficient to protect against phishing by third parties. This is demonstrated by *163.com*, *sapo.pt*, *interia.pl*, *df.eu*, and *bol.com.br*. These providers reject or quarantine emails in the SPF-failing scenarios but deliver them to the inbox in the SPF-neutral but DMARC-rejecting scenarios because they do not implement DMARC or do not honor its result.

The last two rows of the table disclose that multiple providers are vulnerable to software composition attacks through double `From` headers. Any provider that delivers emails in one of the *Double From* scenarios is potentially vulnerable to UI mismatch attacks. We confirm UI mismatch attacks by sending emails with double `From` headers, one of them being `From: root@nsa.gov`, as `nsa.gov` publishes a DMARC record with a reject policy for the organizational domain and subdomains. For *privatemail.com*, *hushmail.com*, and *seznam.cz*, we can produce a UI mismatch in their respective web interfaces and the email is displayed as coming from the National Security Agency. *fastmail.com* was already found to be vulnerable to this type of attack in [10] and remains vulnerable. The web interfaces of *bol.com.br* and *rediffmail.com* display the address for which the DMARC validation succeeded, i.e., the address of the test domain instead of the NSA address, in both scenarios. In these cases, UI mismatch attacks are likely to be possible for some third-party POP3 clients.

Our experiments also show that the DMARC implementations of *tutanota.com* and *mailfence.com* are not RFC-compliant as they deliver email to the inbox in the *Parent Reject* scenario but do not do so in the other DMARC rejecting scenarios. In particular, they do not take the DMARC record of the organizational domain into account when producing a DMARC policy result. As a consequence, email is displayed as if it came from a source that does not make use of DMARC. In the case of *mailfence.com*, this means that the email is delivered to the inbox normally, whereas *tutanota.com* displays a warning for both unauthenticated as well as forged messages: "We could not prove that the content or sender of this message is valid."

To summarize, we find that sender integrity can be attacked for users of up to 28 out of 47 services, even if the sender domain implements one security mechanism. This paints a grim picture of the state of sender integrity in 2022.

| | gmail.com | yahoo.com | yandex.ru | zoho.com | aol.com | icloud.com | t-online.de | protonmail.com | web.de | outlook.com | navercom | fastmail.com | laposte.fr | tutanota.com | vodafone.de | 163.com | mail.com | seznam.cz | mail.ru | rambler.ru | mynet.com | posteo.de | mailfence.com | lund1.de | sapo.pt | thexyz.com | hushmail.com | freenet.de | sina.com | mailbox.org | mail.de | interia.pl | hosteurope.de | inbox.lv | runbox.com | strato.de | startmail.com | gmx.com | privatemail.com | df.eu | juno.com | rediffmail.com | kolabnow.com | bol.com.br | freemail.hu | eclipso.de |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| STARTTLS out | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ |
| DO bit | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ |
| DNSKEY requested | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ | ✓ | ✓ | ✓ | ✗ | ✗ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✗ | ✗ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ | ✓ |
| DS requested | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ | ✓ | ✓ | ✓ | ✗ | ✗ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✗ | ✗ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ | ✓ |
| DNSKEY req. & CD unset | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ |
| DNSSEC Validation | - | - | - | ✗ | - | ✓ | - | ✓ | ✓ | ✓ | - | - | ✓ | ✗ | ✓ | - | ✓ | ✓ | - | - | ✓ | ✓ | ✗ | ✓ | - | - | - | ✓ | ✗ | ✓ | ✓ | - | - | - | ✓ | ✓ | ✓ | ✓ | ✗ | - | - | - | ✓ | - | ✗ | ✓ |
| DNS same operator | ✓ | ✓ | ✗ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ |
| MX record DNSSEC | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✓ | ✓ | ✗ | ✗ | ✗ | ✓ | ✓ | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ | ✓ | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| TLSA Published | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ | ✓ | ✓ | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✓ |
| TLSA Protected | - | - | - | - | - | - | - | ✓ | ✓ | - | - | - | ✓ | - | - | ✓ | - | - | - | - | - | ✓ | ✓ | - | - | - | - | ✓ | - | ✓ | ✓ | - | - | - | ✓ | - | ✓ | ✓ | - | - | - | - | ✓ | - | - | ✗ |
| TLSA Requested | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ | ✗ | ✗ | ✓ | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| TLSA implies TLS | - | - | - | - | - | - | - | ✓ | ✓ | ✓ | - | - | ✗ | ✓ | - | - | ✓ | - | - | - | - | ✓ | - | - | - | - | - | ✓ | - | ✓ | ✓ | - | - | - | ✓ | - | ✓ | ✓ | - | - | - | - | - | - | - | - |
| Rej. TLSA wrong data | - | - | - | - | - | - | - | ✓ | ✓ | ✓ | - | - | ✓ | ✓ | - | - | ✓ | - | - | - | - | ✓ | - | - | - | - | - | ✓ | - | ✓ | ✓ | - | - | - | - | ✓ | ✓ | ✓ | - | - | - | - | - | - | - | - |
| Rej. DANE-EE wrong usage | - | - | - | - | - | - | - | ✗ | ✓ | ✗ | - | - | ✗ | ✗ | - | - | ✓ | - | - | - | - | ✗ | - | - | - | - | - | ✗ | - | ✗ | ✗ | - | - | - | - | ✓ | ✗ | ✓ | - | - | - | - | - | - | - | - |
| Rej. TLSA wrong name | - | - | - | - | - | - | - | ✗ | ✗ | ✗ | - | - | ✗ | ✗ | - | - | ✗ | - | - | - | - | ✗ | - | - | - | - | - | ✗ | - | ✗ | ✗ | - | - | - | - | ✗ | ✗ | ✗ | - | - | - | - | - | - | - | - |

Table 4: DNSSEC and DANE-TLSA support by the providers we tested

## 4.2 Inbound TLS Measurements

Table 4 shows the evaluation of the incoming scenarios described in section 3.1.2. The table structure equals the structure of Table 3 of Lee et al. [31] to a large extent. First of all, we note that email from all providers is successfully delivered in the *Normal* scenario, even though our receiving MTA presents a self-signed, non-matching, and expired certificate. Thus, the scenario is not visualized in the table. (The only exception is *daum.net*, which has been removed from this analysis since we were banned from sending emails at some point in time during our experiments.) The first row of the table indicates whether the provider's sending MTA transmits messages through STARTTLS, the second row indicates whether the DNSSEC OK bit was set for queries arriving at our authoritative name server, the third and fourth rows indicate whether the `DNSKEY` and `DS` records were requested for our scenario zone, which is a prerequisite for DNSSEC validation. The *DNSKEY requested & CD unset* row corresponds to the *DNSSEC Validation* row of Table 3 in Lee et al. [31], while the *DNSSEC Validation* row in our table specifies whether email transmission was correctly rejected for the *wrong MX record signature* scenario. In contrast to [31], we do not use the *CD* bit as an indicator for DNSSEC validation and explain the reasoning below. The *DNS same operator* row, as in Lee et al. [31], displays whether the mail server and the DNS resolver belong to the same Autonomous System. Furthermore, the table shows whether a provider's published `MX` record is DNSSEC-protected, whether the provider publishes a `TLSA` record, whether that record is DNSSEC-protected, whether the provider requests a `TLSA` record, and whether the presence of a `TLSA` record enforces the use of STARTTLS. In the last row group, the behavior for the different TLSA scenarios as described in section 3.1.2 is displayed.

**DNSSEC Validation** While most providers properly support STARTTLS, DNSSEC validation is only performed by 20 providers for outgoing mail. Although providers like *zoho.com* fetch DNSSEC-related records, they do not necessarily perform validation. Moreover, DANE is not used by the top seven providers, for both their own email servers or when connecting to others. For all providers which do check TLSA records, they all reject the connection in the *wrong data* case. However, all of these ignore the mismatch in the name of the certificate (last row).

Comparing the results to those of Lee et al. [31] 's study on DANE-TLSA from 2020, we notice that *sapo.pt* no longer queries `DNSKEY` and `DS` records and does not have the `DO` bit set. Lee et al. specifically mention *sapo.pt* in their paper as one of the providers using the `CD` bit. In this context, they claim that providers who "explicitly disable DNSSEC validation by setting the `CD` bit [...] do not bother to validate the results" and criticize an alleged communication overhead. The remediation of a potentially undesirable overhead may be a reason why *sapo.pt* no longer performs DNSSEC-specific lookups. However, a communication overhead does not necessarily follow from the `CD` bit being set. The DNS server for the test domain is an authoritative server and the queries for names in our zone are not recursive queries, i.e., the `RD` bit is not set. In contrast to the relationship between a stub implementation and a recursor, there is no pre-established trust between a recursor and the authoritative nameservers it queries. Since validating the chain of trust is the task of a well-behaving recursor, the `CD` bit is irrelevant. If a recursor relied on the fact that authenticity is simply reported by a third party, it would be vulnerable to MITM attacks. We, therefore, reject the concept of the `CD` bit serving as an indicator of DNSSEC validation. To properly detect DNSSEC validation, we introduce a new scenario. In this scenario, we do not serve an `RSIG` record for the `MX` record of the scenario test domain. A well-behaving DNSSEC-validating resolver should thus reject resolving the IP address of the email server, and emails should not be delivered in this scenario. Although *protonmail.com* makes use of the `CD` bit, the delivery of emails is correctly refused. The web interface even issues a warning and asks for confirmation before the actual email transmission is attempted. Likely, the software component handling DNS resolution just receives a failure without indication of reason, such as a `SERVFAIL` response. Similarly, *mynet.com* does perform DNS validation. In the case of *sina.com*, Lee

et al. are correct when they claim that they perform superfluous DNSSEC lookups because `DNSKEY` and `DS` records are requested, while the signature of the `MX` record is not validated. Nonetheless, the `CD` bit is not a suitable indicator thereof.

**DANE Support** Apart from that, our findings show that *protonmail.com* and *outlook.com* have added DANE-TLSA support since the publication of [31]. In addition, *gmx.com* now differentiates between `PKIX-EE` and `DANE-EE` certificates. Regarding DANE-TLSA support, we can classify three main types of behaviors: providers that do not implement it, providers that do not use it for opportunistic encryption, and providers that do and implement it in a way that is secure against downgrading. Most providers are not aware of DANE-TLSA at all. Out of 46 providers, only 13 requested the `TLSA` record for the `MX` record of our mail server. *laposte.fr* is the only provider to request the `TLS` record while not implementing [15]. The third group of 12 providers implements DANE-TLSA in a downgrade-resistant way. While some providers may treat `PKIX-EE` like `DANE-EE`, this is not security-relevant, and performing name matching after having validated that the certificate matches the `TLSA` record brings no security benefit either.

**MTA-STS** During our tests, only *gmail.com*, *yahoo.com*, *aol.com*, *protonmail.com*, *outlook.com*, and *mail.ru* request the MTA-STS record of our email server, showing that the vast majority of providers do not support the mechanism. To potentially identify the software used by the providers, we use the non-enforcing MTA-STS scenario as described in section 3.1.2 and capture the user agent for requests to `/.well-known/mta-sts.txt`.

Only *protonmail.com* uses an implementation that we can identify as *postfix-mta-sts-resolver*, an open-source plugin for the Postfix email server. The user agent of *gmail.com*, *Google-SMTP-STS*, hints at a proprietary implementation, while *AHC/2.1* used by *aol.com* and *yahoo.com* appears to be the user agent of the *AsyncHttpClient* library for Java. *mail.ru* identifies as *Go-http-client/1.1* and *outlook.com* does not announce its user agent.

Table 5 shows the results from our MTA-STS analysis. Our experiments show that *yahoo.com* and *aol.com* only request the MTA-STS record or policy when the server indicates support for STARTTLS. This is just a side note since they do not perform MTA-STS validation anyway. *gmail.com* and *mail.ru* are marked with an asterisk (*) since their behavior changed during our experiments. At first, they did not reject self-signed certificates and connections without STARTTLS support, even when an MTA-STS record and a policy were served. When these providers fetch a fresh policy that they have not yet cached, it is not immediately enforced in contrast to the behavior of *protonmail.com* and *outlook.com*. This exposes a major challenge of MTA-STS setups in practice. To reliably implement MTA-STS, large providers with multiple sending MTAs have to either use a global cache or ensure that cache synchronization occurs reliably.

| Provider | Reject new self-signed | Reject serving self-signed | Reject serving no TLS | Reject serving wrong MX | Reject serving expired | Reject cached self-signed | Reject cached no TLS | Reject cached wrong MX | Reject cached expired | Universal policy caching |
|---|---|---|---|---|---|---|---|---|---|---|
| gmail.com* | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| yahoo.com | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ? |
| aol.com | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ? |
| protonmail.com | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ |
| outlook.com | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ |
| mail.ru* | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

Table 5: MTA-STS delivery by scenario

It is not readily apparent why *protonmail.com* accepted an expired certificate in the case when an enforcing policy was actively served and (should have been) cached vs. in a case when such a policy was only cached. Nevertheless, it again shows the fragility of the MTA-STS mechanism. *outlook.com* also showed caching issues, as it did not reject transmission in those scenarios when the policy that should have been cached was no longer served, while correctly rejecting transmission when the policy was being actively served.

## 5 Domain-Scoped Measurements

In this section, we outline our domain-scoped measurements, i.e., SPF and DMARC deployment as well our analysis of OPENPGPKEY and SMIMEA records.

### 5.1 SPF and DMARC

We first investigate the deployment of SPF and DMARC as a protection mechanism, i.e., if servers in the wild deploy the corresponding records.

**SPF Results** Our analysis shows that only 4,167,633 domains in the top ten million implement SPF. The policy results of our scan are displayed in Table 6. For the largest fraction, our server is classified as `softfail`, leaving the choice of failing to the checking mail server. Only 38 % are configured to flag the (obvious) bogus emails from us as `fail`. Notably, 4,183 domains explicitly allow our servers to deliver mails for them, mostly due to the usage of the `all` mechanism. Notably, 371,968 domains are also misconfigured but produce a `permerror`. If MTAs are strict, they should then not accept any incoming email for that domain. The majority (199,514) of the domains contained too many inclusions (at most 10 DNS queries are allowed per SPF check), followed by 86,434 which had multiple DNS entries for SPF.

| Policy Result | Occurrences | Fraction |
|---|---|---|
| Softfail | 1,939,796 | 46.54 % |
| Fail | 1,583,269 | 37.99 % |
| Permerror | 371,968 | 8.93 % |
| Neutral | 242,020 | 5.81 % |
| Temperror | 26,397 | 0.63 % |
| Pass | 4,183 | 0.10 % |
| Total | 4,167,633 | - |

Table 6: SPF policies in the wild

| Classified Reason | Occurrences |
|---|---|
| Misconception | 6 |
| Simple Solution | 3 |
| Simple Slip | 3 |
| Migration | 2 |
| Coordination | 1 |
| Default Configuration | 1 |

Table 7: Self-reported reasons for SPF misconfigurations

**DMARC Results** Of the top 10 million domains, 882,183 deployed a DMARC policy at the time of our final scan in June 2023. We parsed each policy according to the strict standard, e.g., enforcing that notification endpoints start with `mailto://` or `https://` [29]. The vast majority of domains has a syntactically valid DMARC policy (867,506 or 98.33%). However, 9,873 domains have syntax errors (mostly because they miss a `p` to specify the policy) and 4,804 have multiple DNS records configuring DMARC, which also renders the records ineffective.

**Operator Notification** To find out why the SPF records are misconfigured, in October 2022 we notified the postmasters for affected domains that allow impersonation. In that analysis run, we had uncovered 4,484 domains with obvious misconfigurations. We followed best practices identified by prior work [33, 44, 45] and used functional addresses and did not store any personal data beyond those from operators' answers. Out of 4,484 emails, at least 2,700 were not delivered and bounced. We only received 26 non-automated responses. Of those, 16 answered the question meaningfully enough for us to classify them (classification shown in Table 7). Here, **Misconception** refers to cases operators admitted (or it became apparent to them) in their response that they do not correctly understand SPF. Operators classified as **Simple Solution** indicated that they had emails rejected because of missing SPF records and wanted to fix the issues. Three more operators noted it was a **simple slip**, i.e., they claimed to know how SPF works and their misconfiguration was merely a result of a typing error, e.g., a forgotten `-` before `all`. More, two claimed the misconfiguration originated from a relaxation in light of a server **migration**. **Coordination** refers to a case where a respondent noted the coordination effort for SPF was too high. Specifically, they claimed a non-technical customer with self-hosted email uses alternating IP addresses for sending email. Last, but not least, one answer indicated that the record was a provider **default configuration**.

Overall, while the response size is too small to draw a definite conclusion, only 3/16 respondents indicated deeper knowledge of SPF. One user, who referred to himself as a student developer, reported that he was also using DMARC and DKIM. He thought that a DMARC check would fail if DKIM is not used. In his mental model, DKIM was a successor to SPF and therefore his conception was that DMARC would mandate the presence of DKIM. As a result, he did not consider a too-permissive SPF record a problem. Another user explained that the coordination effort with a customer was simply too high. Others named temporary reasons, such as an ongoing project or a server migration. A few admitted that they simply were not knowledgeable on the topic.

We find that our notifications had been effective, although replies were rare. Out of the 2,700 domains that produced a `pass` result in October and could not be notified, 2,141 are still misconfigured one month later, a decrease of 20.7%. In contrast, out of the 1,784 domains that produced a `pass` result in October and were potentially notified (i.e. the misconfiguration notification did not bounce), only 1,076 were still misconfigured in November, which is a decrease of 39.7%, indicating that most likely several operators fixed their setup but did not reply to our inquiry. Overall, comparing the fixed domains for both scenarios, we find that using Fisher's exact test the null hypothesis of the difference being due to random chance is rejected with a $p$-value of $7.8 * 10^{-43}$. Nevertheless, we cannot conclude a causal relation since the lack of deliverable emails may be a hidden variable which influences fix rates.

In June 2023, we also notified the operators which were now affected by SPF misconfigurations, yet did not ask for any feedback on their reasoning.

## 5.2 MTA Strict Transport Security

Regarding MTA-STS, we found that 6,948 domains with an MX have an MTA-STS record to indicate that delivering MTAs should further check for the policy specified under `https://mta-sts.<domain>/.well-known/mta-sts.txt`. However, only 569 of these domains actually had a policy file hosted in that location. Of those, 326 set this to the enforcement mode. Of these, three domains specified MXs in the policy file which were not part of the MXs specified through DNS, i.e., emails to those domains could not be delivered by MTAs adhering to MTA-STS. Another three domains specified an MTA-STS policy and had a matching MX, yet that MX did not have a valid certificate, i.e., MTAs should also refuse connection. Notably, all three implemented TLSRPT [34] to report such errors; however, they all specified recipient addresses hosted on their own domain, which would also not be delivered given the MTA-STS misconfiguration. For the

remaining 320 domains, at least one MX specified in the MTA-STS policy could be successfully connected to with STARTTLS, and for 223 domains, all MXs supported a proper TLS connection. Overall, this shows that MTA-STS is not prevalent, confirming results from Tatang et al. [47] which showed a slightly higher adoption rate within the top million domains.

## 5.3 OPENPGPKEY and SMIMEA

Relatively new additions to email security are OPENPGPKEY and SMIMEA DNS records. These can be used to publish public keys for email encryption and are primarily meant for mailserver's milters to encrypt email towards the receiver. Due to the nature of DNS, these records can be used to enumerate the email addresses. We are the first to leverage a subdomain enumeration technique which is often overlooked [26] to perform a large-scale analysis on these mechanisms. We were able to reverse engineer a high number of email addresses per zone and obtain their OpenPGP keys or S/MIME certificates.

Combining the results from both DNS scans, we identify 115 unique domains with 126 different zones combined. Of those, 100 zones support `OPENPGPKEY` and 26 support `SMIMEA` records. 80 zones support zone walking, with 29 zones supporting `NSEC` records, allowing us to obtain these zones in plain text. The remaining 51 zones only enable DNSSEC walking through `NSEC3` records. *_smimecert.unitymedia.de* and *_openpgpkey.unitymedia.de* are not signed and thus do not support zone walking but allow zone transfers by third parties. The zones with the most records for OpenPGP keys and S/MIME certificates are displayed in Table 8. The *Method* column indicates the best method available to fetch as many records as possible. In some cases not displayed here, we retrieved the records through zone transfers. When NSEC3 was the best method available, we performed hash cracking, and in case of unsigned zones or zones preventing zone walking, the active DNS enumeration approach was left as the only method to fetch records. The *Recs.* column indicates the maximum number of records that we know exist in the zone. The *Id.* column specifies how many of those records we were able to obtain or reverse engineer, while the *Exp.* column specifies the number of expired keys or certificates. In the case of Debian, we were only able to reverse engineer that many record names despite NSEC3 because they publish

| Type | Zone | Method | Recs. | Id. | Exp. |
|---|---|---|---|---|---|
| OPENPGPKEY | _openpgpkey.fedoraproject.org. | NSEC | 3430 | 3430 | 1041 |
| OPENPGPKEY | _openpgpkey.debian.org. | NSEC3 | 853 | 852 | 27 |
| OPENPGPKEY | mail.de. | NSEC3 | 726 | 557 | 127 |
| OPENPGPKEY | samba.org. | NSEC3 | 38 | 37 | 8 |
| OPENPGPKEY | cert.ee. | NSEC3 | 26 | 25 | 4 |
| SMIME | mail.de. | NSEC3 | 124 | 107 | 78 |
| SMIME | _smimecert.secure64.com. | NSEC3 | 41 | 36 | 36 |
| SMIME | _smimecert.nist.gov. | NSEC | 6 | 6 | 1 |

Table 8: Top zones with `OPENPGPKEY` and `SMIMEA` records

| Category | Number of entities |
|---|---|
| Individual | 54 |
| Business | 22 |
| Non-commercial tech | 19 |
| Educational institutions | 7 |
| Email providers | 4 |
| Unidentified | 4 |
| Government institutions | 3 |
| Other | 1 |

Table 9: Types of entities supporting `OPENPGPKEY` or `SMIMEA`

a list of developers at `https://db.debian.org`, which we used as another word list for hash cracking.

To understand the types of users who rely on `OPENPGPKEY` and `SMIMEA`, we categorize the domains making use of these mechanisms. For that purpose, we manually visit the web pages served on these domains, rely on the Internet Archive's Wayback Machine, or perform a Google search. Table 9 shows the results of our categorization.

To perform the categorization, we count entities instead of domains, i.e., *posteo.net* and *posteo.de* belong to the same single entity. We identify 54 individuals who are mostly developers. They mostly support one of the mechanisms as part of their non-commercial personal site or are sometimes advertising small one-person businesses. The individuals are followed by 22 businesses that are mostly active in the space of digital technology. We only find four instances of email providers supporting the mechanisms, with indications of active support by *posteo.de* and *mail.de* only. Both providers do not support automatic outbound encryption, e.g., through milters. The email providers *directbox.com* and *kabelmail.de* only publish what appear to be test records. We also introduce a category unifying non-commercial tech-related sites which do not fit into the other categories. This includes *icann.org* or *debian.org*, for example. All in all, this categorization shows that these mechanisms are mostly used by a small community of technically skilled users and specialized businesses.

We also conducted an analysis of the key material used in the wild. In total, we could obtain 5,168 `OPENPGPKEY` records. Of those, only 6 could not be decoded by `gpg`. All decodable records contain 12,076 keys, including PGP subkeys. Table 10 shows the ten most occurring cryptographic algorithms for the OpenPGP keys we discovered.

While the 4096-bit RSA keys are most prominent, we find

| Algorithm | Incl. revoked/expired | Excl. revoked/expired |
|---|---|---|
| RSA 4096 | 3,977 | 2,816 |
| DSA 1024 | 2,757 | 1,877 |
| Elgamal 2048 | 1,773 | 1,161 |
| RSA 2048 | 1,747 | 1,192 |
| Elgamal 1024 | 810 | 531 |
| RSA 3072 | 257 | 224 |
| Elgaml 4096 | 246 | 155 |
| Ed25519 | 127 | 116 |
| RSA1024 | 84 | 47 |
| Curve25519 | 63 | 58 |

Table 10: Cryptographic algorithms for OpenPGP keys

that 1024-bit DSA keys are the second-most published keys. NIST already deprecated these keys in 2011 and forbid their use starting from 2013 [8, 27]. In contrast, except for one certificate, public keys of S/MIME certificates retrieved via SMIMEA are safe, as 2048-bit RSA keys are still the standard for certificates on the Web. In addition, the table also highlights that significant fractions of the discovered keys are expired or revoked. This shows that not only are SMIMEA and OPENGPGKEY rare in practice, but they also frequently lead to outdated or even plain dangerous key material.

**Error Case Study: Authenticated NXDOMAIN Replay in posteo.de** As a result of being unable to walk the zone of *posteo.de*, we identified one misconfiguration and one security vulnerability in their DNSSEC configuration. *posteo.de* allows its users to publish S/MIME certificates and OpenPGP keys in dedicated zones, including *_openpgpkey* and *_smimecert.posteo.de*. Both zones are secured using DNSSEC.

First, we notice that the zones publish NSEC3 records with 300 additional hash iterations. Since this places a computational burden on verifying resolvers, many implementations discourage the use of such a high number of iterations and treat zones with such a high number of iterations as insecure or return SERVFAIL. RFC 9276 [19] considers zero additional iterations to be the best current practice.

Second, our wordlist DNS scan identifies an OPENPGPKEY record for *jobs@posteo.de*. However, even though the zone *_openpgpkey.posteo.de* makes use of DNSSEC, enumeration using nsec3map fails because one NSEC3 record indicates that no name inside the zone exists by specifying a next hashed owner that equals the hashed owner. This is a faulty record since a zone has to contain at least a SOA record. Nevertheless, the record can serve as proof that the zone is empty, while it is actually not. An active network attacker can therefore replay the signature that proves the non-existence of *nx._openpgpkey.posteo.de* to authenticate the non-existence of the OPENPGPKEY record for *jobs@posteo.de*. Software that relies on DNSSEC to automatically fetch, authenticate and use OpenPGP keys for encryption, such as *openpgpkey-milter* [51], could therefore be tricked into not encrypting messages.

An online search suggests that this issue had persisted since at least 2016 [43]. After we notified posteo.de, they remedied both issues. Yet, similar to our disclosure of misconfigured SPF policies in the wild, we did not receive any response.

**Software Support** In the context of DNSSEC validation and to assess the impact of the security vulnerability at Posteo, we investigate the behavior of software that supports DANE, such as *openpgpkey-milter* [51]. Reviewing its source code, we notice that a SERVFAIL response leaves messages unencrypted. In contrast, a bogus DNSSEC response causes the message to be queued for later delivery attempts instead. This behavior is inconsistent because it protects against manipulation on the DNSSEC level, which requires the capabilities of a network attacker. On the other hand, inducing a SERVFAIL response is as simple as performing a denial of service attack on the authoritative name servers for the _openpgpkey zone of the recipient domain. This leaves the implementation vulnerable to downgrade attacks on end-to-end encryption.

With its source code being based on *openpgpkey-milter*, the SMIMEA milter *smilla* [46] analogously suffers from the same problem. We implement patches to remedy this vulnerability [9]. Our changes will not affect recipients that do not support DNS bindings for OpenPGP keys or S/MIME certificates, as they do not delegate the respective subdomains.
Since *gnupg* is not directly used to encrypt emails without user interaction, it is not vulnerable to downgrade attacks without further context. In case of failure, *gnupg* aborts the encryption and returns an exit code indicating an error.

## 6 MX-Scoped Results

Not a single email provider rejected delivery to receiving MTAs with self-signed certificates. This motivates us to investigate whether a behavior change would have a significant impact. Out of the top ten million domains we investigated on June 6, 2023, 5,363,128 serve at least one *valid* MX record (i.e., not merely an empty string or a dot). In total, these records point to 2,491,404 unique MXs. For each of those MXs, we then queried for the corresponding TLSA record since, e.g., using DANE-EE relaxes the validation rules (e.g., DANE-EE allows even expired certificates to be accepted if the pin matches). This yielded 9,480 MXs which had a DANE record. However, only 8,398 of these records were signed with DNSSEC, which is required for DANE to be effective. Hence, for those 8,398 MX, we considered the DANE records in our further analysis.

In the next step, we attempted to connect to each discovered MX. However, only 2,194,910 (88.1%) were connectable within a timeout of 60 seconds. Of these, 82,228 (3.7%) did not support STARTTLS, i.e., we could, therefore, not collect any certificates. Since `openssl s_client` stops on the first encountered error during validation and does not consider mismatching hostnames as an issue, we instead implemented our own validation in Python. We note that the hostname mismatch is critical: if emails are delivered to any mailserver that presents *some* valid certificate, there is no protection over sending emails in plain text, as an MitM attacker could trivially present their own valid certificate.

Of the remaining 2,112,682 MXs, only 1,478,060 (70.0%) passed proper certificate validation. Table 11 shows the overview of validation errors. We note that hostname mismatching occurs most frequently, with over 600k MXs failing validation because of that. The table also shows that the vast majority (467,818/600,047 cases) of these MXs fail *only* because of the incorrect hostname. Hence, fixing these issues

| Validation Status | MX | Fraction | Affected Domains |
|---|---|---|---|
| Hostname mismatch | 600,047 | 28.4 % | 803,303 |
| *Only hostname mismatch* | 467,818 | 22.1 % | 621,540 |
| Missing certificate in the chain | 90,127 | 4.3 % | 111,745 |
| Expired | 86,352 | 4.1 % | 116,544 |
| Self signed | 79,576 | 3.8 % | 187,065 |
| Weak hash algorithm | 19,955 | 0.9 % | 28,990 |
| Missing chain | 18,308 | 0.9 % | 25,425 |
| EE certificate too weak | 13,643 | 0.6 % | 16,768 |
| Self-signed in chain | 1,977 | 0.1 % | 2,403 |
| Incorrect type | 618 | 0.0 % | 698 |
| Malformed certificate in chain | 247 | 0.0 % | 446 |
| DANE: signature mismatch | 79 | 0.0 % | 135 |
| Not yet valid | 11 | 0.0 % | 12 |
| DANE: expired | 2 | 0.0 % | 3 |
| Signature validation failed in chain | 2 | 0.0 % | 2 |
| DANE: hostname mismatch | 2 | 0.0 % | 2 |
| *Any error* | *634,622* | *30.0%* | *924,174* |

Table 11: STARTTLS certificate validation of top 10M

appears straightforward since operators already have everything in place to obtain valid certificates. The second-most prevalent issue stems from self-signed certificates, which — without DANE-EE — are both invalid and undermine security if MTAs need to connect to servers using them. Similarly, 4.1% of certificates are expired, which could also be overcome through DANE-EE. Other sources of misconfigurations stem from not providing the intermediate certificates required to validate the trust chain, usage of weak hash algorithms such as MD5 or SHA1, or end-entity certificates with cryptographic keys that are too weak (e.g., RSA 1024 bit). Additionally, a small fraction of presented certificates had incorrect purposes (e.g., client authentication), contained malformed certificates in the chain (e.g., using the non-existent version 4 of X.509), or were not valid yet at the time of our scan.

In summary, 30.0% of mailservers — responsible for 924,174/5,363,128 (17.2%) domains with an MX — fail validation in one way or another. These results show that the choice of email providers to *not* reject delivery even in cases where validation fails is a necessary evil to allow utility. Naturally, this comes with the significant downside that any email communication can be intercepted if certificates are not properly validated.

**Security Impact of DANE** For the 8,393 MXs that have a properly signed TLSA record, 129 (responsible for 314 domains) are not connectable in the first place. Another 10 MXs (for 38 domains) could be connected to but did not present a certificate. Finally, 83 MXs (for 140 domains) failed DANE validation, e.g., through a lack of matching DANE pins. Overall, this leaves 8,176 MXs (responsible for 117,126 domains) that are connectable and fulfil the DANE pins. However, to ensure proper encryption even in light of an active attacker who can forge DNS responses, both the MX's TLSA record *and* the domain's MX record need to be DNSSEC-signed. Notably, only 71,176 domains (serviced by 7,433 MXs) fulfill both of these requirements. Hence, this points to many domain operators not being fully aware of how to properly protect their domains even though the MX actually supports DANE.

## 7 Discussion

**Limitations** When deciding whether to accept, reject, or quarantine emails, providers use various signals, some of which are black boxes to external observers. These signals can include IP and domain reputation, which may not be directly under the sender's control. Although we took care of correctly configuring our email setup and used IP addresses that should have a good reputation and were not listed on public DNSBLs, we were unable to deliver email to some email providers. In some cases, when email was rejected due to IP address policies, we contacted the providers to allowlist them. Nonetheless, the reproducibility of our results investigating the deliverability of email under certain misconfigurations is limited to some extent as it is unclear whether a misconfiguration serves as a knock-out criterion or only causes the adjustment of an internal score. Furthermore, we have conducted our scenario-based experiments from subdomains of a single eTLD+1. The number and characteristics of received emails for this domain could have an impact on long-lived internal spam or deliverability scores of recipient providers.

**Ethical Considerations** Some email providers restrict account creation to IP addresses in certain geographic areas while at the same time banning IP addresses from VPN providers. To overcome this hurdle, we use ethically sourced residential proxies when necessary. When performing DNS bulk lookups, we leverage Cloudflare's DNS infrastructure. This is in line with the methodology employed by other research, such as Izhikevich et al. [24]. With Cloudflare handling 1,706 billion DNS queries per day [11], our additional load is negligible. Regarding the SPF misconfiguration notification, we only send one non-intrusive email per domain. While for some recipients a lax SPF configuration is deliberate, we provide most recipients with the benefit of notifying them that their SPF configuration allows malicious third parties to send email on behalf of their domain. Furthermore, the STARTTLS scan is performed in a non-intrusive manner, as we only send the `STARTTLS` command without sending `MAIL` commands. We notified the operators allowing for zone transfers and those that rely on NSEC, suggesting they upgrade to NSEC3. The zone transfer issue was addressed, while for the NSEC notification, some operators explicitly stated no sensitive information was contained in their zones, hence the computational overhead of NSEC3 was intentionally avoided by them.

**Future Directions** Unfortunately, the state of email security has not improved significantly over the years. In practice, in light of various misconfigurations, operators are often faced with being unable to enforce TLS-secured connections be-

tween mailservers, as evidenced by our analysis of invalid certificates. Moreover, even a rather simple mechanism such as SPF causes frustration by operators, who end up (accidentally) allowing anyone to send emails on their domain's behalf. Future work should, therefore, better understand the mental models of administrators tasked with mailserver configuration, to improve usability and therefore reduce frustration and misunderstanding. Moreover, privacy-aware users should be given the option to configure how their mail provider enforces connection security, such as disallowing unencrypted outbound delivery for their emails.

## 8 Related Work

In 2015, Durumeric et al. presented the first report on global adoption rates of STARTTLS, SPF, DKIM, and DMARC [16]. In the same year, a study by Foster et al. was published, measuring the adoption of SPF, DKIM, and DMARC, among others [17]. Both studies are almost a decade old at the time of this writing, necessitating a comprehensive analysis and comparison as done by our work. We find that TLS support has significantly increased since these studies, yet SPF publication and enforcement is at a similarly low number. In [31], the authors performed a large-scale analysis on DANE-TLSA. They also crafted misconfiguration scenarios to detect the support for DANE-TLSA. Our findings show that DANE support is only slowly increasing, with several security-aware providers requesting the DNS records yet failing to properly validate them. Lee et al. [32] further investigated the reasons for misconfigured DANE entries and discovered issues related to key rollover and DNS caching. Maroofi et al. also performed a large-scale analysis on SPF and DMARC [36]. They discover misconfigured SPF records by emulating the `check_host` function described in RFC 7208 [28] and using their own IP address as the sender. Deccio et al. studied the prevalence of email sender validation by leveraging a vulnerability notification message sent to over 26,000 domains in 2021 [12]. The absence or presence of triggered queries for SPF, DKIM, and DMARC to their DNS server allowed them to measure the deployment of these mechanisms on the receiver side. They found that 50% of all mail servers employed all three mechanisms. In 2021, Tatang et al. presented the first large-scale analysis on the usage of MTA-STS [47]. They crawled and validated MTA-STS DNS records and policies over a period of roughly six months and found that it is supported by 219 domains from the Alexa and Tranco top million domains lists. Notably, they did not test for the support of major providers, which we analyzed in more detail, highlighting that the records are barely checked in practice. The first large-scale and longitudinal measurement study of DKIM deployment was published by Wang et al. [50]. Since evaluating DKIM deployment requires knowing the selectors of the keys used to sign messages, the authors extracted public DKIM keys from passive DNS datasets and combined those with a dataset of email headers from a major email provider. They found that 28% of the Alexa top million domains have enabled DKIM, of which 2.9% are misconfigured. They further found that DKIM keys often have a long lifetime. Chen et al. [10] identified component-based software design as a major weakness regarding email sender authentication using DKIM and DMARC. Orthogonally to their work, Müller et al. [37] highlighted that UI mismatch attacks are also feasible given insecure cryptographic implementation in OpenPGP and SMIME. Kambourakis et al. [25] analyzed STARTTLS, SPF, DKIM, DMARC, DANE-TLSA, and MTA-STS support for emails that they receive through their MECSA tool. However, they do not provide insights into the support of particular security mechanisms by popular email providers. Due to limitations of their tool, which only receives one email, they also do not perform outbound checks for MTA-STS and DANE-TLSA and are unable to investigate failure conditions using differential tests. More recently, Tatang et al. [48] conducted an analysis on the adoption of SPF, DMARC, and DKIM, focusing explicitly on misconfigurations and insecurities of weak key material. In 2022, Holzbauer et al. [23] studied the email ecosystem as a whole and noted that it lacks behind in the adoption of more modern standards such as DNSSEC and IPv6.

We unify and partially replicate all of these research works (see also Table 1) into a comprehensive analysis of the state of email security by late 2022 to mid-2023. For full comparison in studied providers, we refer the reader to Table 12. Even though our analysis does not allow us to study DKIM in as much detail as Wang et al. [50], we find that DKIM support is far from perfect. While all providers request the DKIM signature key, even major providers like t-online.de do not follow DMARC rejections in case of incorrect validation. This highlights that even by late 2022, email confidentiality and integrity are still threatened for a large fraction of users.

## 9 Conclusion

We provide a holistic overview of the support of email integrity and confidentiality security mechanisms as of late 2022. Our findings show that the state of email security has not significantly improved over the past years. 96.3% of mail exchanges support explicit TLS (up from 35% in 2015 [16]), yet 30.0% fail certificate validation. Moreover, even though DANE configurations are mostly functional, only about half of the domains which could benefit from DANE protection can still be attacked in light of missing DNSSEC usage for the domains' MX records. Overall, DANE support by providers has increased since 2020 [31], with security-aware providers like Protonmail or Zoho retrieving DNSSEC keys yet still not validating records. This is counterintuitive, considering our findings indicate that DANE is functional for the majority of servers. Even the 10-year-old SPF standard is still only deployed by 40% of domains (exactly as in 2015 [17]),

but a policy is specified for 42 out of 47 high-traffic mail providers. However, enforcing SPF is much less frequent, with 13 providers even delivering SPF `fail` emails to the inbox. For DKIM, 38 providers provide signatures, while all tested ones request the selector when processing inbound emails, which is a significant improvement from 2015. However, 14 servers deliver emails with invalid DKIM signatures into the inbox. DMARC support is increasing (37 out of 47 published a record), but enforcement lacks behind (25 out of 47), yet has increased since 2015 (from 7 out of 20). Notably, though, two providers do not implement subdomain policies correctly. Finally, MTA-STS is only implemented by six providers, and we found evidence for incorrect caching, undermining MTA-STS even further. Considering the low adoption on just 326 out of 10 million domains, this mechanism does not appear to add much to the overall security of the email ecosystem.

All in all, retrofitting protection mechanisms to the SMTP protocol has drastically increased the complexity for email implementers. Ensuring email confidentiality or integrity requires knowledge of various protocols and frictionless collaboration of software components, which, as our SPF misconfiguration notifications showed, cannot be tacitly assumed at scale. Moreover, possibly one of the most surprising insights is that a third of TLS certificates would fail strict validation, showing that enforcing TLS within the email ecosystem would lead to breakage at a large scale.

**Acknowledgements** The authors would like to thank the anonymous shepherd for their guidance in improving the structure and highlighted insights of our paper.

## References

[1] Crackstation's password cracking dictionary. URL https://crackstation.net/crackstation-wordlist-password-cracking-dictionary.htm.

[2] Hashcat advanced password recovery. URL https://hashcat.net/hashcat/.

[3] List of most popular email domains (by number of live emails). https://email-verify.my-addr.com/list-of-most-popular-email-domains.php, .

[4] E-Mail Dienste im Vergleich. https://trusted.de/email, .

[5] Die Top 14 der sicheren E-Mail-Anbieter im Jahr 2023. https://kinsta.com/de/blog/sicheren-email-anbieter/.

[6] Facebook data leak: Details from 533 million users found on website for hackers, 2021. URL https://www.theguardian.com/technology/2021/apr/03/500-million-facebook-users-website-hackers.

[7] The most common email aliases backed by data, 2022. URL https://blog.101domain.com/google-workspace/most-common-email-aliases.

[8] Elaine Barker, Allen Roginsky, et al. Transitions: Recommendation for transitioning the use of cryptographic algorithms and key lengths. *NIST Special Publication*, 800:131A, 2011.

[9] Birk Blechschmidt. Return TEMPFAIL upon SERVFAIL. https://github.com/letoams/openpgpkey-milter/pull/8, 2022.

[10] Jianjun Chen, Vern Paxson, and Jian Jiang. Composition Kills: A Case Study of Email Sender Authentication. In *USENIX Security Symposium*, 2020. URL https://www.usenix.org/conference/usenixsecurity20/presentation/chen-jianjun.

[11] Cloudflare. Our reliability products. URL https://www.cloudflare.com/reliability/.

[12] Casey Deccio, Tarun Yadav, Nathaniel Bennett, Alden Hilton, Michael Howe, Tanner Norton, Jacob Rohde, Eunice Tan, and Bradley Taylor. Measuring Email Sender Validation in the Wild. In *Conference on Emerging Networking EXperiments and Technologies*, 2021. DOI: 10.1145/3485983.3494868.

[13] Mark Delany. Domain-Based Email Authentication Using Public Keys Advertised in the DNS (DomainKeys). RFC 4870, 2007.

[14] DomCop. Download list of top 10 million domains based on open data from common crawl & common search. URL https://www.domcop.com/top-10-million-domains.

[15] Viktor Dukhovni and Wes Hardaker. SMTP Security via Opportunistic DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS). RFC 7672, 2015.

[16] Zakir Durumeric, David Adrian, Ariana Mirian, James Kasten, Elie Bursztein, Nicolas Lidzborski, Kurt Thomas, Vijay Eranti, Michael Bailey, and J. Alex Halderman. Neither Snow Nor Rain Nor MITM...: An Empirical Analysis of Email Delivery Security. In *ACM IMC*, 2015. DOI: 10.1145/2815675.2815695.

[17] Ian D. Foster, Jon Larson, Max Masich, Alex C. Snoeren, Stefan Savage, and Kirill Levchenko. Security by Any Other Name: On the Effectiveness of Provider Based Email Security. In *ACM CCS*, 2015. DOI: 10.1145/2810103.2813607.

[18] Ólafur Guðmundsson. Adding Acronyms to Simplify Conversations about DNS-Based Authentication of Named Entities (DANE). RFC 7218, 2014.

[19] Wes Hardaker and Viktor Dukhovni. Guidance for NSEC3 Parameter Settings. RFC 9276, 2022.

[20] Paul E. Hoffman. SMTP Service Extension for Secure SMTP over TLS. RFC 2487, 1999.

[21] Paul E. Hoffman and Jakob Schlyter. The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA. RFC 6698, 2012.

[22] Paul E. Hoffman and Jakob Schlyter. Using Secure DNS to Associate Certificates with Domain Names for S/MIME. RFC 8162, 2017.

[23] Florian Holzbauer, Johanna Ullrich, Martina Lindorfer, and Tobias Fiebig. Not that simple: Email delivery in the 21st century. In *USENIX Annual Technical Conference*, 2022. URL https://www.usenix.org/conference/atc22/presentation/holzbauer.

[24] Liz Izhikevich, Gautam Akiwate, Briana Berger, Spencer Drakontaidis, Anna Ascheman, Paul Pearce, David Adrian, and Zakir Durumeric. ZDNS: a fast DNS toolkit for internet measurement. In *ACM IMC*, 2022. DOI: 10.1145/3517745.3561434.

[25] Georgios Kambourakis, Gerard Draper Gil, and Ignacio Sanchez. What Email Servers Can Tell to Johnny: An Empirical Study of Provider-to-Provider Email Security. *IEEE Access*, 8, 2020. DOI: 10.1109/access.2020.3009122.

[26] Bastian Kanbach. Enhancing Subdomain Enumeration - ENTs and NOERROR, 2022. URL https://www.securesystems.de/blog/enhancing-subdomain-enumeration-ents-and-noerror/.

[27] Cameron F. Kerry and Patrick D. Gallagher. FIPS PUB 186-4 Federal Information Processing Standards Publication: Digital Signature Standard (DSS). 2013.

[28] Scott Kitterman. Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1. RFC 7208, 2014.

[29] Murray Kucherawy and Elizabeth Zwicky. Domain-based Message Authentication, Reporting, and Conformance (DMARC). RFC 7489, 2015.

[30] Murray Kucherawy, Dave Crocker, and Tony Hansen. DomainKeys Identified Mail (DKIM) Signatures. RFC 6376, 2011.

[31] Hyeonmin Lee, Aniketh Gireesh, Roland van Rijswijk-Deij, Taekyoung "Ted" Kwon, and Taejoong Chung. A Longitudinal and Comprehensive Study of the DANE Ecosystem in Email. In *USENIX Security Symposium*, 2020. URL https://www.usenix.org/conference/usenixsecurity20/presentation/lee-hyeonmin.

[32] Hyeonmin Lee, Md Ishtiaq Ashiq, Moritz Müller, Roland van Rijswijk-Deij, Taekyoung "Ted" Kwon, and Taejoong Chung. Under the Hood of DANE Mismanagement in SMTP. pages 1–16, 2022. ISBN 978-1-939133-31-1. URL https://www.usenix.org/conference/usenixsecurity22/presentation/lee.

[33] Frank Li, Zakir Durumeric, Jakub Czyz, Mohammad Karami, Michael Bailey, Damon McCoy, Stefan Savage, and Vern Paxson. You've got vulnerability: Exploring effective vulnerability notifications. In *USENIX Security Symposium*, volume 16, 2016. URL https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/li.

[34] D. Margolis, A. Brotman, B. Ramakrishnan, J. Jones, and M. Risher. SMTP TLS Reporting. RFC 8460, 2018.

[35] Daniel Margolis, Mark Risher, Binu Ramakrishnan, Alex Brotman, and Janet Jones. SMTP MTA Strict Transport Security (MTA-STS). RFC 8461, 2018.

[36] Sourena Maroofi, Maciej Korczyński, Arnold Hölzel, and Andrzej Duda. Adoption of Email Anti-Spoofing Schemes: A Large Scale Analysis. *IEEE Transactions on Network and Service Management*, 2021. DOI: 10.1109/tnsm.2021.3065422.

[37] Jens Müller, Marcus Brinkmann, Damian Poddebniak, Hanno Böck, Sebastian Schinzel, Juraj Somorovsky, and Jörg Schwenk. "Johnny, you are fired!"-Spoofing OpenPGP and S/MIME Signatures in Emails. In *USENIX Security Symposium*, 2019. URL https://www.usenix.org/conference/usenixsecurity19/presentation/muller.

[38] Damian Poddebniak, Fabian Ising, Hanno Böck, and Sebastian Schinzel. Why TLS is better without STARTTLS: A Security Analysis of STARTTLS in the Email Context. In *USENIX Security Symposium*, 2021. URL https://www.usenix.org/conference/usenixsecurity21/presentation/poddebniak.

[39] Jonathan B. Postel. Simple Mail Transfer Protocol. RFC 821, 1982.

[40] Philippe Rémy. Philipperemy/name-dataset: The python library for names. URL https://github.com/philipperemy/name-dataset.

[41] Pete Resnick. Internet Message Format. RFC 5322, 2008.

[42] Wayne Schlitt and Meng Weng Wong. Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail, Version 1. RFC 4408, 2006.

[43] Daniel Stirnimann. Is BIND9 DNSSEC validation too strict?, 2016. URL https://lists.isc.org/pipermail/bind-users/2016-October/097825.html.

[44] Ben Stock, Giancarlo Pellegrino, Christian Rossow, Martin Johns, and Michael Backes. Hey, you have a problem: On the feasibility of large-scale web vulnerability notification. In *USENIX Security Symposium*, 2016. URL https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/stock.

[45] Ben Stock, Giancarlo Pellegrino, Frank Li, Michael Backes, and Christian Rossow. Didn't you hear me? - towards more successful web vulnerability notifications. In *NDSS*, 2018. DOI: 10.14722/ndss.2018.23171.

[46] sys4.de. SMIMEA Milter. URL https://github.com/sys4/smilla.

[47] Dennis Tatang, Robin Flume, and Thorsten Holz. Extended Abstract: A First Large-Scale Analysis on Usage of MTA-STS. In *DIMVA*, 2021. DOI: 10.1007/978-3-030-80825-9_18.

[48] Dennis Tatang, Florian Zettl, and Thorsten Holz. The evolution of dns-based email authentication: Measuring adoption and finding flaws. In *RAID*, 2021. DOI: 10.1145/3471621.3471842.

[49] The Verge. Over 150 million breached records from adobe hack have surfaced online. https://www.theverge.com/2013/11/7/5078560/over-150-million-breached-records-from-adobe-hack-surface-online, 2013.

[50] Chuhan Wang, Kaiwen Shen, Minglei Guo, Yuxuan Zhao, Mingming Zhang, Jianjun Chen, Baojun Liu, Xiaofeng Zheng, Haixin Duan, Yanzhong Lin, and Qingfeng Pan. A large-scale and longitudinal measurement study of DKIM deployment. In *USENIX Security Symposium*, 2022. URL https://www.usenix.org/conference/usenixsecurity22/presentation/wang-chuhan.

[51] Paul Wouters. OpenPGP milter service to automatically PGP encrypt plaintext emails when possible. URL https://github.com/letoams/openpgpkey-milter.

[52] Paul Wouters. DNS-Based Authentication of Named Entities (DANE) Bindings for OpenPGP. RFC 7929, 2016.

# A Provider Comparison

| | Our work | Lee [31] | Foster [17] |
|---|---|---|---|
| aol.com | ✓ | ✓ | ✓ |
| daum.net | ✓ | ✓ | ✓ |
| gmail.com | ✓ | ✓ | ✓ |
| gmx.com/.de | ✓ | ✓ | ✓ |
| hotmail.com/outlook.com | ✓ | ✓ | ✓ |
| mail.ru | ✓ | ✓ | ✓ |
| naver.com | ✓ | ✓ | ✓ |
| yahoo.* | ✓ | ✓ | ✓ |
| fastmail.com | ✓ | ✓ | ✗ |
| freemail.hu | ✓ | ✓ | ✗ |
| icloud.com | ✓ | ✓ | ✗ |
| inbox.lv | ✓ | ✓ | ✗ |
| interia.pl | ✓ | ✓ | ✗ |
| mail.com | ✓ | ✓ | ✗ |
| mynet.com | ✓ | ✓ | ✗ |
| protonmail.com | ✓ | ✓ | ✗ |
| rediffmail.com | ✓ | ✓ | ✗ |
| runbox.com | ✓ | ✓ | ✗ |
| sapo.pt | ✓ | ✓ | ✗ |
| seznam.cz | ✓ | ✓ | ✗ |
| sina.com | ✓ | ✓ | ✗ |
| t-online.de | ✓ | ✓ | ✗ |
| tutanota.com | ✓ | ✓ | ✗ |
| zoho.com/.in | ✓ | ✓ | ✗ |
| 163.com | ✓ | ✗ | ✓ |
| web.de | ✓ | ✗ | ✓ |
| yandex.ru | ✓ | ✗ | ✓ |
| 1und1.de | ✓ | ✗ | ✗ |
| bol.com.br | ✓ | ✗ | ✗ |
| df.eu | ✓ | ✗ | ✗ |
| eclipso.de | ✓ | ✗ | ✗ |
| freenet.de | ✓ | ✗ | ✗ |
| hosteurope.de | ✓ | ✗ | ✗ |
| hushmail.com | ✓ | ✗ | ✗ |
| juno.com | ✓ | ✗ | ✗ |
| kolabnow.com | ✓ | ✗ | ✗ |
| laposte.fr | ✓ | ✗ | ✗ |
| mail.de | ✓ | ✗ | ✗ |
| mailbox.org | ✓ | ✗ | ✗ |
| mailfence.com | ✓ | ✗ | ✗ |
| posteo.de | ✓ | ✗ | ✗ |
| privatemail.com | ✓ | ✗ | ✗ |
| rambler.ru | ✓ | ✗ | ✗ |
| startmail.com | ✓ | ✗ | ✗ |
| strato.de | ✓ | ✗ | ✗ |
| thexyz.com | ✓ | ✗ | ✗ |
| vodafone.de | ✓ | ✗ | ✗ |
| comcast.net | ✗ | ✓ | ✓ |
| sohu.com | ✗ | ✓ | ✓ |
| wp.pl | ✗ | ✓ | ✓ |
| att.net | ✗ | ✗ | ✓ |
| cox.net | ✗ | ✗ | ✓ |
| libero.it | ✗ | ✗ | ✓ |
| pacbell.net | ✗ | ✗ | ✓ |
| qq.com | ✗ | ✗ | ✓ |
| twc.com | ✗ | ✗ | ✓ |
| excite.com | ✗ | ✓ | ✗ |
| o2.pl | ✗ | ✓ | ✗ |
| Total | 47 | 29 | 20 |

Table 12: Comparison of tested providers between our work and closely related ones