SoK: Securing Email

***Abstract*—Email is the most ubiquitous and interoperable form of online communication today. However, email was not conceived with strong security guarantees, and the ensuing security enhancements are, by contrast, lacking in both ubiquity and interoperability. This state of affairs motivates our work. Our contributions include: (1) a systematization of secure email approaches taken by industry, academia, and independent developers; (2) an evaluation framework for proposed or deployed email security enhancements and a measurement of their security, deployability, and usability; (3) an informed discussion of the current state of secure email; and (4) a research agenda highlighting open problems judged to be useful in pursuing secure email.**

I. Introduction

For its over 50-year history, email has been an open medium—if you know someone's email address, you can send email to them [103]. It provides seamless interoperability among users with a diverse range of desktop, mobile, and web client software. As an indication of its near-universal acceptance, an email address is often required to create online accounts and to make online purchases. As of 2017, there were 6.3 billion email accounts, owned by 3.7 billion users, sending over 269 billion email messages per day.[1]

Despite its ubiquity, email was created without security in mind and remains largely insecure today [31], [39], [90], [60]. Email is only sometimes transmitted over an encrypted connection, with limited protection from passive network eavesdropping and active network attacks. There is only limited protection against message forgery. Email archives are typically extensive, stored in plaintext, and vulnerable to hacking. Spam and malware are prevalent, though filtered by many email providers. Phishing and spear phishing [109] remain problems.

Secure email products based on S/MIME, such as Microsoft Outlook and IBM Notes, have been adopted when there is significant motivation to protect intellectual property (enterprises) or conform with regulatory burdens (government). However, non-enterprise users have fewer choices. PGP has long been the champion of secure email for the masses but has been plagued by lack of adoption and severe usability issues surrounding key management. A variety of experts are abandoning it, including Phil Zimmermann [**?**], the creator of PGP; Moxie Marlinspike [88], who called PGP a "*a glorious experiment that has run its course*," and Filippo Valsorda [130], who bemoans the challenges of maintaining long-term PGP keys. Academic research has invested significant effort in improving email PKI for non-enterprise users (enterprise users are often the beneficiaries of IT staff), but has made little impact on deployed software after nearly 20 years.

The relatively low level of adoption of secure email is often contrasted with the wider success of secure messaging applications. WhatsApp and Facebook Messenger have over a billion users, while iMessage, Signal, Telegram, Line, and Viber have millions. The best of these provide forward secrecy and message deniability [14], [104] in addition to end-to-end encryption. Yet, despite some calls to abandon secure email in favor of Signal [130], there are important reasons to not give up on email. Email is an open system, in contrast to messaging's walled gardens, giving it fundamentally different uses, often involving longer messages, archival, search, and attachments. Email is likely to continue to be a primary form of communication on the Internet for years to come, and users of the service could benefit from added privacy. Our work is an important addition to the literature, which already includes an excellent systemization of knowledge of secure messaging [129].

Security architectures for email are complicated by the expectation of universal interoperability. Acquiring public keys in a single administrative domain can be done almost transparently to the user, but many complications arise with countless unrelated and uncoordinated administrative domains worldwide. The expectation that anyone can email anyone else directly leads to the need for spam and malware filtering, along with protection from phishing. Secure messaging largely avoids the need for these features by restricting who can contact you. Finally, the expectation of archive and search functionality makes it challenging to provide encryption for webmail or for mobile clients, since in both cases email is typically stored on a service provider's infrastructure, where access to plaintext is currently needed for these features.

Securing email is a thorny problem. Not only is ubiquitous deployment of secure email elusive, none seems evident on the horizon. There are many stakeholders involved and a variety of use cases for secure email, making it hard for a single solution to emerge. Our primary goals are to examine the history of failures and successes in secure email, including what the term 'secure email' even means to different stakeholders; to identify roadblocks and to distinguish the ones that might be removed; to identify tradeoffs among existing systems and clarify how they address different needs; and to use our acquired knowledge to nudge future efforts in fruitful directions.

II. Background

To provide some context for our discussion of efforts to secure email, we first describe how email works, explain why it is insecure, and illustrate how criminals leverage this insecurity.

A. How Email Works

A series of protocols are used to send email, transfer it from the sender's email provider to the recipient's provider, and then retrieve it. Figure 1 shows the most basic steps involved, in

[1] https://www.radicati.com/wp/wp-content/uploads/2017/01/Email-Statistics-Report-2017-2021-Executive-Summary.pdf

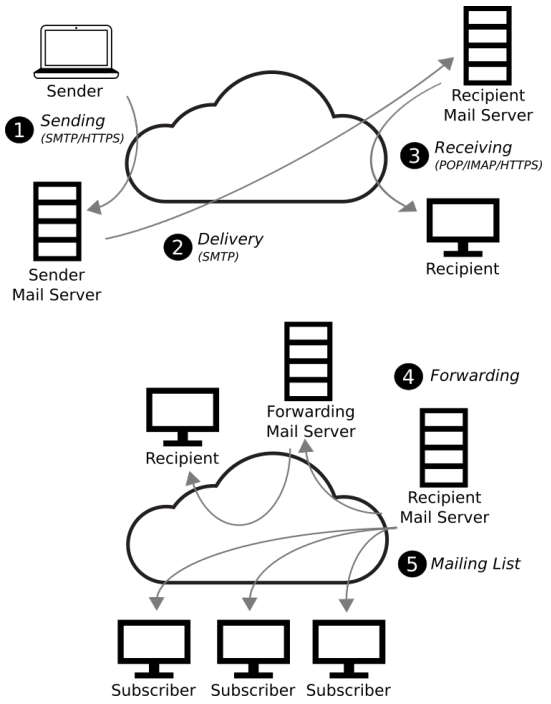

Fig. 1. Overview of email operation and protocols. (1) Sending email generally uses SMTP or HTTPS between a client and its mail server. (2) Delivery of email between mail servers uses SMTP. (3) Receiving email generally uses POP, IMAP, or HTTPS. (4) A recipient mail server may forward email to another server. (5) A recipient mail server may forward an incoming email to a set of mailing list subscribers.

steps marked (1) through (3). When a user initiates sending an email, her client may use SMTP [74] to submit the message to her organization's mail server. The sender's mail server uses DNS to locate the mail server for the recipient's domain, then uses SMTP to transfer the message. Finally, the recipient retrieves the message from her own organization's mail server, possibly using POP or IMAP. If either the sender or receiver is using webmail, then step 1 or step 3 may use HTTPS instead. Note also that the version of SMTP used to submit a message in step 1 is somewhat modified from the version of SMTP used to transfer messages [49].

This sequence of events is complicated somewhat by additional features supported by email, shown by steps (4) and (5) in Figure 1. First, a receiving mail server can be configured to forward email for a recipient on to another mail server; *e.g.,* forwarding email from `bob@company.org` to `bob@gmail.com`. This can repeat an arbitrary number of times. Second, a destination email address may correspond to a mailing list server which forwards the email to all subscribers on the list (a potentially large number). This adds numerous other recipient email servers to the process.

B. Why Email is Insecure

Every aspect of email was initially specified, designed, and developed without consideration for security. As a result, these initial designs led to numerous problems, many of which persist today despite decades of work to fix them. Consider one example: as originally designed, there is no authenticity for email messages. This means anyone could send email to anyone else, using a fake sender email address.

To see how this is possible, it is useful to distinguish between the two parts of an email message: the envelope and the body. The envelope contains SMTP commands that direct mail servers regarding how the message should be delivered. These consist of connection negotiation (`HELO` or `EHLO`), the sender's email address (`MAIL FROM`), one or more recipient addresses (`RCPT TO`), and potentially additional options. Of particular note is that the original specifications of SMTP do not include any validation of the sender's email address. In addition, many email servers were initially set up to accept email from any client, known as open relays.

Combined, these features meant that anyone could forge an email to anyone else, leading to the emergence of what we call unsolicited email or spam [73]. Moreover, the message body has a separate format [25], including the familiar *From*, *To*, *CC*, and *Subject* header fields. Email clients generally display the sender's email address shown in the *From* header, rather than the one in the SMTP envelope. Again, the original specifications contain nothing that prevents forgery of the *From* header. The features of email that make it open to spam also make it vulnerable to phishing and delivery of malware.

In addition to lacking authenticity, the original designs of protocols used to send, receive, and deliver email among clients and servers contained no protections for integrity or confidentiality. All messages were transmitted in the clear and could be intercepted and modified by anyone able to act as a man-in-the-middle. Email is also subject to privacy violations because it does not easily provide anonymity, deniability, freedom from tracing, and ephemerality.

C. How Criminals Use Email

As long as unsolicited emails reliably reach the inboxes of potential victims, email is destined to remain a dominant vehicle for crime. Two prominent examples are spam (both for sales and for malware) and social engineering.

In the early 2010s, a successful campaign might see a spammer employ a botnet of 3,000 compromised machines to send 25 billion emails to 10 million addresses [63]. Each aspect of the pipeline—from the compromised machines to the email list to the software tools—might be sold by specialists [82], and the campaign itself is typically run by third-party advertisers earning pay-per-click revenue for directing traffic to a third-party site (*e.g.,* storefronts for unregulated pharmaceuticals constitute over a third of spam) [91].

Social engineering may be crafted as a generic attack or a targeted attack against specific individuals. The openness of email enables direct contact with targets and an opportunity to mislead the target through the content of the email, a spoofed or look-alike send address, and/or a malicious file attachment [96], [57]. As an illustrative example, HR department employees of the security company RSA were sent an email with a malicious Excel file (exploiting a zero-day embedded Flash vulnerability) masquerading as a recruitment plan; at least one opened the file,

and their computer was compromised, eventually leading to a deep infiltration of the company's network [109]. Spam filters are trained to detect bulk email delivery, which presents new challenges for classifying bespoke spear phishing emails [77] and training employees [18].

III. How Service Providers Secure Email

Email service providers have adopted a number of technologies to improve the security of email, including link encryption, domain and sender authentication, and spam and malware mitigation. Here we review current and planned efforts, the protection they offer, and measurements that reveal that these technologies have had a large variance of effectiveness to date.

A. Link Encryption

Email providers have adopted methods for encrypting email while it is in transit between mail servers or between a mail server and a client. We call this *link encryption* to distinguish it from end-to-end encryption, which occurs between the email sender and recipient. Link encryption is designed to prevent eavesdropping and tampering by third parties that may own untrusted routers along the path that email is being delivered [59].

By default, mail sent using SMTP is sent in the clear, but the connection may be upgraded to use TLS with the STARTTLS command [59]. This is known as *opportunistic encryption*. Because this is negotiated in plaintext, it is trivial for an adversary to mount a downgrade attack by corrupting or stripping the STARTTLS command [31].

To fix this problem, a March 2016 Internet draft introduced Strict Transport Security for SMTP [87]. With STS, email providers may use DNS records to advertise a policy requiring TLS connections for SMTP. A receiving mail server can authenticate the policy it receives (ensuring it is valid for the sending domain) using a variety of mechanisms, such as with the Certificate Authority system or with DANE [**?**], [**?**]. Sending mail servers can report on or refuse to deliver messages that cannot be delivered using TLS.

For communication between mail client and a mail server, a variety of encryption methods can be used. A client using POP or IMAP can start encryption using STARTTLS [100]. However, many servers are now disabling plain IMAP and POP and instead requiring clients to connect over TLS. Clients using webmail can use HTTPS to provide encryption between the recipient mail server and the client. Clients may also submit email using SMTP and the STARTTLS command.

One additional concern is that SMTP reveals significant metadata as email messages are relayed, such as sender and recipient email addresses. Some work tries to address this concern [84], [125], but no significant work beyond draft stages exists as of this writing.

B. Domain Authentication

Email providers have invested significant effort in providing *domain authentication*, which ensures that an email originated from a specific domain. Consider the case when Alice receives an email from `bob@gmail.com`. Domain authentication indicates that the email was sent by a server authorized to send email from `gmail.com`. This is contrasted with *sender authentication*, §III-D, which occurs when `gmail.com` authenticates `bob` to access and send email. A third step is user authentication, which occurs when Alice ensures that a human, such as Bob Smith, owns the `bob@gmail.com` account.

Domain authentication has a long history rooted in identifying spam and filtering malware [73], [37], [76], [6]. These authentication methods provide assurance that the originating domain of an email, either as listed in the header or the envelope, actually sent the email.

1) DKIM: DomainKeys Identified Mail (DKIM) [26], [75] allows a server that originates email to include a digital signature over some portions of the email. The signature can include the entire message (header and body) or just selected fields in the header, but signatures are not applied to the envelope. The sending server hashes the fields to be signed and digitally signs the hash with the private key of the sender's domain. The sending server also inserts a message header field indicating which domain should be used for checking the signature. A receiving email provider uses this header to determine the domain, retrieves the public key of the sending provider via DNS, then verifies the signature. Assuming email providers use DNSSEC to ensure that public keys received via DNS are legitimate, this provides assurance that the sending email provider did originate the email and that the signed fields have not been modified in transit.

2) SPF: Sender Policy Framework (SPF) [73] allows an organization to publish a DNS record that specifies which IP addresses are allowed to originate email for their domain. A receiving provider can verify that the IP address of the server that originates the email is on the list of approved addresses for the domain in the envelope MAIL FROM field. These IP addresses do not necessarily need to be owned by the domain; they can, for example, be IP addresses from a separate email provider. SPF breaks mail forwarding, but a mail server can instead use remailing, which changes the envelope sender domain to match that of the forwarder.

3) DMARC: Domain Message Authentication, Reporting, and Conformance (DMARC) [76] builds on these technologies by allowing an organization to publish a DNS record indicating which of these services (DKIM, SPF) they support, along with a policy indicating what action should be taken if authentication fails. DMARC also requires identifier alignment. For SPF, this means that the domain in the envelope *From* address (which is authenticated with SPF) must match the domain in the header *From* address. For DKIM, this means that the domain used for signing must match the domain in the header *From* address. This links the authentication or signature verification done by SPF and DKIM to the *From* address seen by the user. DMARC also provides a mechanism for receiving email providers to send reports back to sending email providers regarding the outcomes of DMARC policy enforcement. This helps email providers identify misconfigurations and abuse.

4) ARC: Authenticated Received Chain (ARC) [6], [65] extends email authentication to handle cases when messages are potentially modified and then forwarded, such as by a mailing list. With ARC, a forwarder adds headers indicating the status of validity checks done on authentication added by the originator or other intermediate forwarder (i.e., SPF, DKIM, DMARC, or ARC). The forwarder then adds a signature over the message body and header as it received it, then adds another signature (called a *seal*) over any ARC fields added by this or previous forwarders. When there are multiple forwarders involved, the set of ARC fields added forms an ARC chain.

C. Measurement Studies of Adoption and Effectiveness

A number of papers [31], [39], [90], [60] have measured the level of adoption and effectiveness of the encryption and domain authentication used by email providers. These papers conduct a variety of active measurements of email servers and passive measurements of ongoing email traffic. The general picture they paint is that top email providers encrypt messages with STARTTLS and use SPF and DKIM for authentication, but there is a long tail of organizations that are lagging in deploying these mechanisms.

Even when security solutions are deployed, they are often compromised by insecure practices. The above studies find that of those servers that offer encryption, many use self-signed certificates, expired certificates, or broken chains, all of which cause the validation of the certificate to fail. There are numerous other cases in which email traffic uses weak cipher suites, weak cryptographic primitives and parameters, weak keys, or plain authentication over unencrypted connections. Of the techniques that rely on DNS, basic attacks such as DNS hijacking, dangling DNS pointers [86], and modifying non-DNSSEC lookups can enable circumvention. Stripping attacks can compromise STARTTLS, with Durumeric et al. [31] illustrating how these attacks lead to 20% of inbound Gmail messages being sent in cleartext for seven countries. Use of SPF is common, but enforcement is limited, and DNS records often aren't protected with DNSSEC. There is little use of DKIM, and few servers reject invalid DKIM signatures [39]. As Mayer et al. [90] conclude, *"the global email system provides some protection against passive eavesdropping, limited protection against unprivileged peer message forgery, and no protection against active network-based attacks."*

D. Sender Authentication

The methods discussed above authenticate only the sending domain and do not guarantee that the sending user was authenticated by the domain. Most email domains do authenticate their users [58]. For example, if the sender is using webmail, then she may authenticate by logging into her webmail account over HTTPS. If the sender is using a desktop client, the mail domain can authenticate her with SMTP Authentication, which provides a number of methods that enable the sender to authenticate with the mail server by a username and password [122], [123], [121]. However, the measures a domain uses to authenticate a sender are not communicated to the recipient of an email message, nor can they be verified by the recipient.

E. Spam and Malware Mitigation

Providers and enterprises have invested significant effort in spam and malware filtering. Malware filtering is often performed by comparing email attachments to signatures of known malware. Spam filtering has evolved from IP blacklists to highly sophisticated classifiers that examine content, meta-information including origin, user reports, and protocol details such as SMTP header fingerprints [127], [128]. Spammers use a variety of evasion techniques, including sending from the IP addresses of malware-compromised computers [41], spoofing sender addresses, and encoding text as images. A more esoteric approach to spam prevention is requiring the sender to compute a costly function to send an email [32], [8]—an approach that never caught on [79].

IV. End-to-End Encryption

While provider-based solutions have focused on preventing spam, filtering malware, and stopping passive eavesdropping, end-to-end encryption is designed to provide security guarantees that enable users themselves (whether individuals or corporations) to exchange secure email without revealing plaintext to third parties. In addition to encryption, these efforts usually include digital signatures and integrity as well. Here we review these initiatives to gain insight into why these efforts have failed to enjoy widespread use.

A. PEM (Privacy Enhanced Mail)

The first major effort to secure email was PEM [71], which was designed in the late 1980s and early 1990s, resulting in proposed Internet standards [85], [70], [10], [66] that are now designated "Historic". The primary goals of PEM were end-to-end email security with confidentiality, data origin authentication, connectionless integrity (order not preserved), non-repudiation with proof of origin, and transparency to providers and to SMTP.

PEM offers important lessons as the first completely architected, full-function and operational end-to-end secure email system. It identified and solved many—but not all—of the difficult challenges still confronting today's proposals:

1) Interoperability: PEM was designed to be interoperable with existing email transport mechanisms and non-PEM recipient clients. For example, signed messages could be sent so that intervening mail servers could modify the message during processing; this invalidated the signature but allowed non-PEM clients to read the message.

2) Hierarchical Trust: PEM used X.509 certificates and a hierarchical trust model with a single root CA, which simplified trust resolution and restricted namespace authority (thus largely precluding rogue certificate issues haunting later PKI systems).

3) Revocation: PEM had a centralized certificate revocation design, which placed primary responsibility for distributing revocation information on the infrastructure rather than end-users. Every CA was required to distribute CRLs and provide their users access to global CRL information, coordinated with the root CA.

4) Validation: PEM specified that clients originating mail could include all certificates that recipient needed to process it; caching by recipients could expedite and/or accommodate missing certificates. A PEM client could thus automatically map email addresses or local aliases to distinguished names in certificates and thereby public keys.

5) Anonymity: PEM accommodated anonymous users with *persona certificates*, which could provide assurances of continuity of a pseudonymous user ID.

A contributing factor cited [102] in PEM's demise was its slow progress in evolving for Multipurpose Internet Mail Extensions (MIME) [42], the standard for including attachments, multi-part bodies, and non-ASCII character sets. Industry supported S/MIME, while privacy advocates favored PGP because it was free from the restrictions imposed by PEM's centralized and hierarchical organization.

B. S/MIME

S/MIME [105] is a standards suite for securing MIME data with both encryption and digital signatures. It was originally developed during the early 1990s by RSA Data Security, then later adopted by the IETF, resulting in standards in 1999 [106], [105]. Of note, S/MIME makes use of the Cryptographic Message Syntax (CMS) [61], which had origins in PEM and PKCS. S/MIME uses a supporting suite of certificate management protocols, including RFC 5280 [23], which defines an IETF subset of X.509v3 certificates. S/MIME has wide support on major platforms and products, such as IBM (Lotus) Notes and Microsoft Outlook mail clients [102, p.60-62].

S/MIME has many similarities with PEM but also important differences. Its X.509v3 certificates, designed to support email interoperability (among other application uses), are generally issued by third-party Certificate Authorities (CAs) operating under a centralized trust model, with CAs also responsible for certificate revocation information. X.509v3 extensions define information such as whether a certificate holds the public key of a CA or a non-CA entity (e.g., useful for properly verifying certificate chains), and what the key may be used for (e.g., signatures vs. encryption). Unlike PEM, S/MIME does not mandate a hierarchy with a single root CA—any organization can act as an independent, trusted root for its certificates, and this is its most common usage today. Recent industry initiatives to facilitate interoperability between key management clients and key management servers are being advanced under the OASIS banner KMIP (Key Management Interoperability Protocol) [101].

The centralized certificate management supported by S/MIME is a good match for internal communication within enterprises and governments. This has led to some large deployments of S/MIME in commercial or government silos [19]. However, this has led to limited adoption by individual users. As noted by Garfinkel and Miller [45], individuals that do not belong to an organization must manage their own public key pairs and certificates and also find the correct public key for a recipient, without the benefit of a dedicated IT staff.

A number of works have examined usability deficiencies with S/MIME implementations. Kapadia [69] conducted a case study of S/MIME and reported that vendors install too many trusted CAs, opening users to the potential for man-in-the-middle attacks. He notes it is difficult for users to know which CAs to trust and discusses comparing key fingerprints via a phone call as one alternative. Fry et al. [43] used a cognitive walkthrough of three S/MIME clients to show that usability challenges persist, particularly involving certificate management. Orman [102, p.60–67] notes that the inconsistent handling of certificates in clients—depending on whether the operating system, the mail client, or special certificate manager software receives a certificate—can be a barrier to deployment and causes usability issues. One solution used is to integrate identity with other enterprise systems, then automatically create and distribute signing and encryption keys whenever a new account is created. This is the case with IBM (formerly Lotus) Notes and Grove [45].

C. PGP

PGP eschews a centralized trust model for an egalitarian *web of trust*, in which every user independently decides which public keys to associate with a given email address and whom to trust to vouch for those keys. Ever since its 1991 release by Phil Zimmermann as "public key cryptography for the masses" [140], supporters have promoted PGP as a de facto world standard for encryption. Its founding motivation as "guerrilla cryptography" to counter authorities [140] has made it a combined religion/cult and political movement especially favored by privacy advocates who reject big-business and government. This mindset, and the tendency of enterprise organizations to favor S/MIME, has led to a market bifurcation. Arguably, the resulting product indecision and non-interoperability has negatively impacted the deployment of secure email in general.

PGP's history is a fascinating 25-year tale of controversy, architectural zig-zags, name ambiguity, and patent disputes, with changes in algorithms, formats and functionality; commercial vs. non-commercial products; corporate brand ownership; and circumvention of U.S. crypto export controls.[2] The current standard for the PGP message format is OpenPGP [16], [33], a patent-unencumbered variation of PGP.

Despite evolving formats or encryption algorithms, PGP enthusiasts until recently have largely remained faithful to PGP's distinguishing concepts:

1) PGP key packets and lightweight certificates: Key packets hold the session key used to encrypt a message; the session key is encrypted with the recipient's public key. Public and

[2]PGP was distributed as freeware on the Internet in 1991, leading to an investigation of Zimmermann by the United States Customs Office for allegedly violating U.S. export laws. He published the PGP source code in book form in 1995 [138], and the case was subsequently dropped in 1996 [80].

private keys are kept in *key certificates* [140], which are not certificates in the X.509 sense, but instead contain keys that are associated with a User ID, in the form of a user name and email address.

To allow client software to properly use PGP key packets (bare keys), they are accompanied by further fields composing *transferable public keys* [16], thereby reconstructing X.509 certificate functionality in many aspects. These include one or more *User ID packets* each followed by zero or more *signature packets*. The latter attest the signing party's belief that the public key belongs to the user denoted by User ID.

2) PGP keyrings: Key rings are collections of one or more keys in a file or database. Public keyrings store collected public keys; private keyrings store a user's own private key(s), with each individual key encrypted under a symmetric key derived from a user-chosen *passphrase*.

3) PGP's web of trust: The web of trust is a model in which users personally decide whether to trust public keys of other users, which may be acquired through personal exchanges or from public servers, and which may be endorsed by other users they explicitly designate to be *trusted introducers*, a sort of ad hoc Certification Authority [139]. Client software tracks the resulting trust level within a user's keyring(s).

4) PGP key servers: Users publish their public key to either closed or publicly accessible key servers, which contain a mapping of email address to the public key. Clients query to locate the public key associated with an email address.

PGP's design around the web of trust has allowed quick deployment in small groups without bureaucracy or costs of formal Certification Authorities [92], but leads to other significant obstacles. A variety of limitations in PGP are evident when examining *The official PGP user's guide*, written by Phil Zimmermann in 1995, and the history of PGP.

a) Scalability beyond small groups: Zimmermann notes [140, p.23], that *"PGP was originally designed to handle small personal keyrings"*. Scaling PGP requires acquiring large numbers of keys, along with a manual trust decision for each key, plus manual management of keyrings and the key lifecycle.

b) Design failure to address revocation: Zimmermann writes [140, p.31], *"If your secret key is ever compromised, you'd better get the word out quickly to all interested parties (good luck!) before someone uses it to make signatures in your name ... You just have to spread the word and hope everyone hears about it"*. PGP does have *compromise certificates*—a user signature declares their key to be revoked. Creating these requires the private key (which is unavailable if lost), and distributing them to others is ad hoc. After creating one, [140, p.32] *"Then you must somehow send this compromised certificate to everyone else on the planet..."* Trusted introducers may also sign revocation certificates, but expecting ordinary users set this up appears naively optimistic.

c) Usability by non-technical users: Users must understand the differences between trusting a public key for personal use, endorsing a public key for other users, and designating others as trusted introducers. There is no system help or recovery if users fail to back up their private key, or forget their passphrase. The difficulty that this imposes on users is nicely summarized by the PGP slogan that *"PGP is for people who prefer to pack their own parachutes"* [140, p.31].

d) Trust model mismatch: "PGP tends to emphasize [an] organic decentralized non-institutional approach" reflecting personal social interaction rather than organizational relationships [140, p.25]. The PGP web of trust was designed to model social interaction, rather than decision-making processes in governments and large enterprises. It is thus not a one-size-fits-all trust model.

Of these problems, the poor usability of PGP has received significant attention. In 1999, a formal user study of PGP 5 [133] demonstrated users were unable to successfully send encrypted email in the context of a hypothetical political campaign. The study uncovered serious usability issues with key management and helped shape modern usable security research. A number of subsequent laboratory studies [120], [113] in the decades since then have continued to highlight the poor usability of PGP. Indeed, many PGP clients continue to expose the details of key management to users. Typical user interactions include generating keys, creating and saving a revocation certificate (in case of key loss), and importing keys.

D. Dissemination of Public Keys

A *certificate directory* publishes public keys. There are two basic implementation approaches. The first approach is a directory operated by an untrusted server that publishes certificates signed by trusted issuers (*e.g.,* CAs) or users (*e.g.,* web of trust). The issuer is responsible for authenticating the owner of the certificate. Because the certificate directory is untrusted, clients are responsible for determining the validity of a retrieved certificate, which includes verifying that the certificate was signed by an entity whose verification public key the client trusts. In PGP, this is called a key server.

The second approach is a directory operated by a trusted server, and clients assume the keys have been verified by the directory. Since the trust is in the party hosting the directory rather than signed data as in the first approach, the information must be pulled over a secure channel. The public key and associated meta-data can be unsigned or placed in self-signed certificates. This approach is sometimes referred to as a *key directory*, but we use the term certificate directory for both approaches.

Certificate ledgers are a subclass of certificate directories that allow the directory's actions to be audited by third parties. Certificate Transparency (CT) [78] enables public auditing of CA issuers to detect the issuance of malicious certificates. An adaptation [116] of CT supports revocation and a secure email architecture where the email provider hosts the ledger for its users. CONIKS [93] is an end-user key verification system that builds on ledger-based CT proposals for web server certificates. CONIKS directories manage disjoint namespaces, end users can monitor their keys, and the security properties of the ledger are designed so the directory can never *equivocate* by delivering the real key to the user monitoring it, while sending a bogus

public key to other users querying the directory. ARPKI [11] is a somewhat more complex public key infrastructure that delivers stronger security guarantees, with certificate operations that are fully transparent and auditable.

E. Storage of Private Keys

Issues of private key management have been discussed in the PKI community since the 1990s [5, Chapter 7]. Private keys used for decrypting email must be stored safely as long as the encrypted email requires protection [94, Chapter 13.7]. Companies using secure email often retain access to private keys in order to maintain access to all emails, to comply with regulations, to scan email for spam or malware, or to detect fraud or insider trading. A variety of enterprise software helps companies manage these assets. For non-enterprise users, for example using PGP, private keys are often stored online and encrypted with a password, though hardware tokens are also available. Operating systems provide seamless synchronization of passwords, bookmarks, and contacts; similar methods could be used to provide migration of private keys or certificates for non-enterprise users [12]. NIST [19, p.76] recommends that keys used for S/MIME and PGP be used only for messages in transit, and that arriving mail that is intended to be archived should be decrypted and then stored locally and re-encrypted with new keys; *e.g.,* in a cryptographic file system.

A related issue is the debate over government mandated access to plaintext, which originally surfaced in the U.S. in the 1990s and has been rekindled regularly. Proponents cite fears that widespread use of end-to-end encryption will enable criminals and terrorists to "go dark" and evade law enforcement. In response, privacy advocates decry growing mass surveillance, point to a history of abuses of wiretapping [28], and suggest that market forces will ensure there is plenty of unencrypted data for use by law enforcement regardless [46]. A 2015 paper from Abelson et al. [1] highlights risks of regulatory requirements in this area, reiterating many issues discussed in their earlier 1997 report [2]. Identified risks include reversing progress made in deploying forward secrecy, leading to weaker privacy guarantees when keys are compromised; substantial increases to system complexity, making systems harder to secure; and the concentration of value for targeted attacks. Their report also highlights jurisdictional issues that create significant complexity in a global Internet. More broadly, whenever service providers have access to keys that can decrypt customer email, this allows plaintext to be revealed due to incompetent or untrustworthy service providers, by disillusioned employees, by government subpoena, or by regulatory coercion.

F. Improving the Usability of Email PKI

Due to the limited success of secure email, a variety of efforts have explored improving the usability of PKI for email, primarily focusing on automating discovery of public keys. This work applies to both S/MIME and PGP, since either the centralized infrastructure (S/MIME) or the web of trust (PGP) could be replaced with the methods we discuss here. An important observation is that all of this work abandons the web of trust around which PGP was originally designed.

1) Automatic key directories: Recently, the usable security community has studied automated use of certificate directories. Ruoti et al. [114], [112] showed that automated key management with identity-based encryption (IBE) resulted in successful task completion and strong, positive user feedback. Atwater et al. [7] showed that automated certificate directories likewise have high usability when integrated into a PGP-based email system. Bai et al. [9] found users prefer directories to manual key exchange, even after being taught about the security limitations of a directory.

2) Social Authentication: Another way to disseminate public keys is to associate them with public social media accounts. The Keybase project helps users to post a signed, cryptographic proof to their account, simultaneously demonstrating ownership of a public key and ownership of the account. By aggregating proofs across multiple accounts for the same person, a client can establish evidence that ties a public key to an online persona, under the assumption that it is unlikely that a person's social media accounts are all compromised simultaneously. The Confidante email system [81] leverages Keybase for distribution of encryption public keys, with a user study finding it was usable for lawyers and journalists. The authors also find differences in use cases and threat models, indicating that a one-size-fits-all approach may not work.

3) Trust-on-first-use (TOFU): Another approach is to exchange keys in-band and have clients blindly trust them on first use. This was integrated with an email system by Roth et al. [110], using a novel postcard and letter metaphor to distinguish between plaintext and encrypted email, and Garfinkel and Miller [45], in a system designed to simplify S/MIME for non-enterprise users. Masone and Smith [89] studied this "first use" case by building a system to help users decide whether to trust signed email from someone that they don't know. Since 2016, the developer community has been integrating TOFU into PGP implementations in the MailPile, PEP [13], LEAP [125], and autocrypt projects. A common critique of TOFU is that users cannot distinguish valid key changes from an attack. Recent work by developers in the PEP and LEAP projects are aiming to address this problem with additional methods to authenticate public encryption keys, such as downloading the public key for a recipient from the recipient's email provider, checking that providers advertise consistent keys for their users, and asking the user to compare key fingerprints [62], [27].

4) Short-lived keys and forward secrecy: Schneier and Hall [117] explored the use of short-term private keys to minimize the damage resulting from the compromise of a private key. Brown et al. [15] discuss timeliness in destroying a short-lived key and how short-lived keys complicate usability by requiring more frequent key dissemination. Off-the-Record Communication (OTR) [14] expanded on this vision with a protocol that provides perfect forward secrecy on instant messaging platforms. The authors also discussed the use of OTR for email, indicating that it could be used if two individuals

communicate frequently and don't expect their initial message to be encrypted. This work led to the double ratchet algorithm used in Signal [104].

5) Encrypting Proxies: One approach to making interactions with PKI transparent to users is to layer encryption and signing below client software, as suggested by Levien et al. [83] and Wolthusen [135]. A related approach for webmail systems is hosted S/MIME, discussed in §V-D.

G. Why Users Don't Encrypt Email

Much of the above work assumes that improving the usability of end-to-end encryption will by itself speed the adoption of secure email clients. However, this remains unclear. Surveys and interviews with users have found reasons for non-adoption include lack of concern, lack of perceived need, misconceptions about how to obtain security, and poor awareness of effective security practices [44], [107]. Similar themes have been found in a study of adoption of secure messaging [3]. Other work reports that users are unsure about when they would need secure email [111] and are skeptical that any system can secure their information [115]. It is not clear that users want to use digital signatures or encryption for daily, non-sensitive messages [36], and one study of encryption inside of an activist organization found that many users felt routine use of secure email was paranoid [47]. Related work in mental modelling has been used to understand how users approach home security [132], why users follow security advice [35], and how users perceive privacy threats [68].

V. Securing Webmail

We now consider efforts to secure web-based email (or webmail) clients. The distinguishing characteristic of webmail is that email is accessed using a web browser, with HTTPS generally used to send and retrieve email between the browser and a web server. Much of this work can be generalized to desktop or mobile clients, though there are some issues related to browsers that require special attention.

A. Information Depots

One alternative to sending secure email is to send a plaintext email with a link to a message that can be read on a separate web-based system, an approach we refer to as an *information depot*. All sensitive information is conveyed in the separate message, which the recipient can access by authenticating herself to the web system. This approach is common with banks, insurance companies, universities and other entities needing to communicate securely with their customers or students.

B. Secure Webmail Systems

A secure webmail system integrates email encryption (and sometimes digital signatures) into a typical webmail interface. Encryption between the service's users is handled automatically; the public key for a recipient is generally retrieved from a certificate directory run by the email provider, and the email is encrypted with this key. This process is entirely transparent to the user, and there is typically no method to verify the fingerprint of the public keys.[3] To send email to a user of a different email provider, the sender typically chooses a password and the system emails the recipient a link to the email provider's information depot. The recipient typically decrypts the message in the web browser, using JavaScript to download and decrypt the message with the password. Out-of-band distribution of the password is left to the user. With rare exception, most secure webmail systems do not offer secure email that is interoperable with other secure email clients.[4] Examples of secure webmail systems include ProtonMail, Tutanota, and Hushmail. Some services, such as ProtonMail, also offer mobile applications.

As compared to traditional desktop email software, secure webmail systems introduce additional potential attacks. Each time a user visits the site, their browser can potentially download a different script, enabling an email provider (perhaps under subpoena from a government) to insert malicious JavaScript that reveals their password, encryption keys, or plaintext. A standard is being developed by the W3C that enables JavaScript to be signed, so that browsers can verify downloaded code. However, even in this case, it is possible for other JavaScript loaded on the same page to undermine the security of any JavaScript library through DOM interactions or manipulation of the execution environment. Additional methods are needed to provide privilege separation for JavaScript in the browser [95], [131].

Another security concern for webmail systems is private key storage. In many systems, the private key is stored on the email provider's servers, encrypted with a password chosen by the user, so that the user can easily access secure email from a variety of devices. This requires the user to choose a password strong enough to resist offline password attacks [38], in case the private key storage is compromised or the service provider is malicious. In addition, this same password is often used to authenticate the user to the webmail system. In this case, the browser should not transmit the password to the service provider when authenticating the user.[5] Possible approaches are hashing and salting the password in the browser before sending it to the server to retrieve the encrypted private key (Tutanota) or using the Secure Remote Password protocol [136] (ProtonMail).

C. Browser Extensions

Another approach to providing secure webmail is to develop a browser extension that enables users to send signed and encrypted email with their existing webmail provider. One category of extensions extends webmail to include PGP functionality, typically automating some key management

[3]Fingerprint comparison is common with secure messaging applications but the feature is often ignored by users [118]

[4]Hushmail allows users to send email to a PGP user at a different email provider if the recipient uploads their public key to Hushmail's key server

[5]The original Lavabit service, which offered a secure webmail system and was famous for having Edward Snowden as a client, sent the user's password to the server to unlock the private key. This meant Lavabit could use the password to unlock the private key and read all the user's emails, even if they were stored in an encrypted format.

tasks. Extensions in this category include Mailvelope and FlowCrypt. There are also browser extensions that provide automated S/MIME-based encryption and signing (Fossa Guard), encryption with a symmetric key held by the service (Virtu), and encryption using a password shared out of band (SecureGmail).

Some security issues that must be addressed for secure webmail systems likewise apply to browser extensions. The use of a browser extension provides greater control over JavaScript execution [12], reducing the need for auditing and privilege separation. Extensions sometimes store end-user private keys on a server managed by the extension, encrypted with a password, while others use the browser's *localStorage* implementation, which typically results in key material being stored on the client's local disk. However, *localStorage* can be read by any JavaScript loaded from the same server. One issue unique to extensions is that they need a way of preventing the webmail provider from accessing plaintext of emails and drafts (many webmail providers sync drafts as a user types). An extension can prevent this by using a secure browser overlay such as an iframe [112].

D. Hosted S/MIME

A different approach to securing webmail is for the webmail provider to handle all encryption for the client, including storing the user's private keys. This was suggested by Garfinkel et al. [44] and Farrell [36] and implemented by Google in early 2017 as hosted S/MIME. An organization or user uploads end-user public key certificates that have been signed by a publicly trusted anchor CA, along with the private key for each certificate. The provider then automatically encrypts and signs messages to other users also participating in the service. Because the provider can decrypt incoming emails, it can also provide spam and malware filtering. This does not provide end-to-end encryption, but provides more reliable confidentiality than opportunistic link encryption.

VI. Privacy

Message confidentiality is an email privacy enhancement that has received considerable attention, however researchers and developers have explored other privacy aspects including anonymity, email traceability, deniability, and message ephemerality. In terms of deployment and academic study, these areas are further behind message confidentiality and subsequently, we do not discuss them much outside of this section. We highlight them here as important considerations beyond the typical discussion of securing email.

A. Anonymity

Email senders may wish to remain anonymous: *i.e.,* unlink the contents of the email from their true email address, their IP address, and/or the identity of their mail server. Senders may also opt for pseudonymity [54] where a set of emails are known to originate from the same person but no further information is known.[6] Generally speaking, email is sent to chosen individuals or groups; thus, email receivers are typically not fully anonymous but might be pseudonymous—*e.g.,* the recipient should be the same person who sent a prior anonymous email.

One common template for anonymous email is to route messages through multiple non-colluding servers and to layer on cryptography to ensure all servers in the chain need to be compromised to link together the message's sender and receiver [21]. Cryptographic return addresses could also be used to send a message back from the receiver to the original sender, even though no single server knows who the sender is. This idea was championed by the cypherpunk movement [98], [99] and adapted to the email protocol with remailers [53], [51], [52]. Not all remailers have the ability to reply to a received email, however this limitation dovetailed with the prominent use case of posting anonymous messages to email-based mailing lists, where replies were collected as new messages sent to the mailing list. This approach is more widely used for anonymous web browsing (onion routing [55] and Tor [30]) than email.

A second template is much simpler: register a webmail account under a pseudonymous address. As some webmail providers will forward the user's IP address with the message, while others will not, users might opt to use Tor to access their mailbox to hide their IP address. One real-world example of this technique is Satoshi Nakamoto, the inventor of Bitcoin [97], who corresponded over webmail for many months while remaining anonymous.[7] A less successful example was a Harvard undergraduate student who emailed a bomb threat to the university's administration via webmail accessed over Tor. In this case, the student was found to be the only individual accessing Tor on the university's own network during the time the email was sent—while strictly circumstantial, the student confessed [56]. This illustrates an important consideration: anonymity is realized as indistinguishability from a *set* of plausible candidates—the set of other users at the time of use [29]. Remailers have always had a small user-base which leads to weak anonymity, despite the cryptographic protections.

B. Traceability

Modern email clients have browser-like features, weaving together passive content from multiple sources and displaying it to the user as a single message. Email senders have for some time abused this functionality and encoded the recipient's information (such as their email address) into external resources [34]: *e.g.,* the URL of a single pixel image that will be invisibly requested by the user's mail client when they load the email. Links in emails can also contain embedded user information as URL parameters that are passed to the server when users click. Outgoing requests may also include eligible cookies and the user's IP address. In total, email

[6]Note that pseudonymity and message authentication techniques, such as DKIM or digital signatures, are not contradictory. The latter techniques bind messages to keys and keys become pseudonyms. Pseudonymity forgoes the further binding (via a PKI) of keys to identities.

[7]'His' anonymity continues to the time of writing.

tracking techniques can enable senders to determine when each specific recipient views an email, from where, when a link is clicked, and to tie (via third-party trackers) the recipient to other collected information about their browsing habits.

As of 2018 there is a patchwork of solutions for traceability in use by mail providers, servers, clients, and browsers (for webmail) that is typically not comprehensive [34]. Prevention of email tracking can be handled by the mail server which might automatically fetch and cache all resources for the user upon receipt of the mail. This hides the user's access pattern to the email. Mail clients (both client-side and web-based) might blacklist requests to known trackers, and strip out sensitive request headers like cookies and the referrer.

C. Deniability

An important component of end-to-end secure email is the ability to sign messages. Signatures provide a degree of non-repudiation: publicly verifiable evidence that a message originated from the asserted sender. In some cases, such as anonymous email above, the sender wants no evidence they are the sender beyond the plaintext of the message itself. Deniability considers a middle case where the recipient wants to authenticate the sender but the sender does want the evidence to be convincing to anyone else. Deniability is often considered in instant messaging, where both parties are simultaneously online and can engage in an interactive protocol [129]. By contrast, email is generally non-interactive. Cryptographers have suggested new signature types [20], [64], [108] to provide deniability but with near-zero deployment in email clients. The fine-print of these primitives include trusted third parties and/or a robust PKI—the same issues limiting end-to-end secure email.

D. Ephemerality

Once sent, a sender loses control over an email and the extent to which its contents will be archived. In recent years, companies and organizations have been damaged by unencrypted email archives being leaked to the press or the public. The decrease in storage costs, both client-side and cloud-based, have removed a previous need for many to delete old emails. In order to automate a shorter retention period, emails might contain a link to the message body which is deposited with and automatically deleted by a trusted service provider or a distributed network [48], [134]. This mechanism can be composed with email but is not specific to email in any way, and so we do not describe it further.

VII. Cryptographic Enhancements for Email

In the previous sections, we reviewed a spectrum of security and privacy issues with email protocols, systems, and architectures. We discussed the foundations of prominent security issues in the protocols, the failure of secure email to gain significant traction, and efforts to improve usability. In this section, we evaluate—in a side-by-side, systematic way—the prominent mechanisms and architectures proposed for improving email through the use of cryptography: namely encryption, signatures, and key/certificate management. A summary is given in Table I, which categorizes the benefits of each according to Security, Deployability, and Usability. We evaluate these proposals as general approaches rather than specific technologies that might change over time (*e.g.,* we evaluate *end-to-end encryption and signatures* rather than PGP or S/MIME). Concrete deployments mix and match from these proposals; we show compositions for common systems in Table II.

A. Evaluation Framework

In Table I, we break email enhancements (rows) into two categories—*crypto primitives* refer to specific cryptographic approaches for securing email, while *key management primitives* indicate how keys are managed. We evaluated four cryptographic primitives and eight key management primitives.

To rate each primitive, we identified fifteen key properties (columns) which collectively describe its security, deployability, and usability. Properties are worded so that the fulfillment of a column is considered beneficial. For each property, enhancements are assigned a score regarding to how well they satisfy that property—full support (●), partial support (◐), or no support (○). In some cases, the rating a primitive receives will depend on the key management used (▼). In other cases, the property will not apply to that primitive, and no rating is given (empty cell). In Appendix B we describe the properties, cryptographic primitives, and key management primitives in greater detail.

B. Summary findings

Based on our Table I, we identify the following highlights:

1) No one-size-fits-all solution: Email provider-based approaches (R2, R3, R5) are easily deployed and highly usable, but they offer only modest security gains. End-to-end encryption can provide significant security benefits (*e.g.,* R4+R10), but has significant deployment and usability hurdles—for example, requiring updates to client software and disrupting the ability of servers to provide content processing (*e.g.,* spam and malware filtering). As such, it is unlikely that any single secure email system will be suitable for all users and their various use cases.

2) Limitations of no third parties: R7, R8, R12, and R13 all reject the use of third parties. While there is a security benefit to removing third parties, it also makes it difficult to quickly and fully disseminate revocation information. When a user's private key has been compromised, they must notify all of their contacts of this compromise and provide them with the compromised user's new certificate. This process is potentially highly time consuming and opens up a significant avenue for social engineering. Compromised users must also update any medium they have used to share their compromised certificate—for example, on a website or a business card. Critically, users who previously obtained the compromised certificate, but who have not yet been in contact with the compromised party, will remain unaware of the revocation.

The lack of a third party also makes it difficult to retrieve certificates for individuals with whom the user does not regularly interact. In many cases, it may be impossible to establish an out-of-band (i.e., non-email) communication

	#	*Primitive*		*Security*								*Deployability*					*Usability*		
				P1: Protection from network eavesdropping	P2: Protection from network tampering	P3: Sender anonymity	P4: No third parties required	P5: Private key only accessible to user	P6: Provides a public key audit trail	P7: Responsive public key revocation	P8: No email server modifications needed	P9: No email client modifications needed	P10: No new non-email infrastructure needed	P11: Simple initial communication	P12: Supports server-side content processing	P13: Effortless encryption key discovery	P14: Effortless signing key validation	P15: Supports private-key recovery
	R1	Plaintext email		○	○	○	○	○	○	○	●	●	●	●	●	●	●	●
Crypto Primitives	R2	Link encryption	III-A	◐	○	○	○	○	▼	▼	●	●	▼	◐	●	●	●	●
	R3	Signature by originating mail server	III-B	○	◐	○	○	○	▼	▼	○	●	▼	◐	●	●	●	●
	R4	End-to-end encryption and signing	IV	●	●	○	▼	▼	▼	▼	●	○*	▼	▼	▼	▼	▼	▼
	R5	Hosted encryption and signing	V-D	◐	◐	○	○	○	▼	▼	○	●	▼	▼	●	▼	▼	●
Key Management Primitives	R6	Certificate authority	IV-B				○	●	○	●			●	○	○	○	●	○
	R7	Manual certificate distribution†	IV-C				●	●	○	○			●	○	○	○	○	○
	R8	Web-of-trust†	IV-C3				◐	●	○	○			●	○	○	○	●	○
	R9	Certificate directory	IV-D				○	●	◐	●			○	◐	○	●	●	○
	R10	Certificate ledger	IV-D				◐	●	●	●			○	◐	○	●	●	○
	R11	Directory with private key recovery	IV-E				○	○	◐	●			○	●	◐	●	●	●
	R12	Trust on First Use	IV-F3				●	●	○	○			●	○	○	○	●	○
	R13	Shared secret†	V-B				●	●	○	○			●	●	○	○	○	●

● Full support ◐ Partial Support ○ No support ▼ Depends on key management

*There is already significant commercial client support for end-to-end encryption and signing (*e.g.,* S/MIME).

†In practice, these key management schemes do not compose well with R2 or R3.

TABLE I
A COMPARATIVE EVALUATION OF PRIMITIVES USED TO ENHANCE EMAIL SECURITY.

System	Crypto	Key Management
TLS between client and server	R2	R6
TLS between servers	R2	R12
DKIM	R3	R9
Corporate S/MIME	R4	R6, R11
PGP	R4	R7, R8
Secure Webmail	R4	R9, R13
Google's Hosted S/MIME	R5	R6
Confidante	R4	R10

TABLE II
MAPPING BETWEEN EXISTING SYSTEMS AND TABLE I.

channel to exchange certificates, potentially preventing two users from communicating securely with email.

3) Usability benefits of key recovery: As discussed in §IV-E, there are significant security limitations for systems that rely on key recovery and important reasons why privacy-sensitive end users may not trust any third party to hold their private key. On the other hand, Table I also demonstrates (see R11) important usability benefits (countered by empty-dot P5). In particular, key recovery allows for support of server-side processing of email message contents (P12) and ensure that users' private keys are always recoverable (P15). The lack of support for these features in other key management primitives is a significant impediment to their adoption in practice.

4) Certificate ledgers offer promise: Of all the key management primitives, certificate ledgers (R10) rate most strongly in terms of security through a more complex design. While they do rely on third parties, they diffuse this trust across multiple third parties. They provide both effortless encryption key discovery (P13) and signing key validation (P14), something none of the other non-directory schemes (R6–R8, R12–R13) supply.

5) Areas warranting additional research: End-to-end encryption conflicts with server-side processing of email. Unfortunately, this interferes with core functionalities of email—for example, search, message threading, and spam filtering. If end-to-end encryption is to be used more widely, this limitation will need to be addressed (*cf.* [124], [50], [67]). In addition, none of the primitives Table I evaluated address anonymity or meta-data privacy, warranting further research (see §VI).

VIII. DISCUSSION

Moving beyond our comparative evaluation, we offer some further observations on the general state of secure email.

A. Email today offers little protection

While email protocols support link encryption and domain authentication, lack of adoption and the availability of easy attacks mean that plaintext email is subject to passive and active eavesdropping, as well as message forgery. If providers could solve these issues and offer provider-to-provider encryption and authentication, then the resulting service could work without user interaction and could satisfy users who trust their email provider with plaintext. This would complement the spam and malware filtering that providers already offer and which appears largely effective. Filtering remains a somewhat more daunting task for smaller email providers, due to the complex protocols that must be used and the lack of a large userbase that makes collaborative filtering or machine learning feasible. Phishing remains a major problem [72], particularly spearphishing attacks against high-value targets, as evidenced by high-profile breakins.

B. Secure email works within closed communities

The centralized trust model in S/MIME matches the needs of organizations, as indicated by its deployment within a variety of companies and in major government departments [19, p.67]. S/MIME removes the burden of fine-grained trust decisions from users. Its centrally-managed and supported architecture facilitates holding private keys centrally; despite objections to plaintext access, this allows an organization to implement content processing and filtering. Secure webmail services offer advantages to non-enterprise users, with ProtonMail and Tutanota reporting over a million users of their services. PGP works well for small groups, such as a collection of journalists or activists, who prefer not to trust any third party, despite the burden of manually exchanging keys. In all of these cases, the group using secure email primarily forms a closed community, and secure communication outside the community is often not supported or plaintext is preferred.

C. The PGP web of trust remains unsuccessful after 25 years

The web of trust that is so central to the original design of PGP—including manual key exchange and trusted introducers—has largely failed. Its use is generally limited to isolated, small communities. Its appeal is that it allows quick deployment in small groups without bureaucracy or costs of formal Certification Authorities, but in practice the downside is poor scalability and poor usability. Many long-term PGP supporters have abandoned the web of trust, and developers are pursuing alternative methods for distributing public keys. As such, PGP has essentially become more about the format of messages and keys, rather than the methods used to distribute and verify keys. This points to a future where most PGP clients will adapt to handle a variety of key formats, automated distribution methods, and automated authentication steps, with the traditional manual trust decisions left to a small minority with specialized needs.

D. A global, end-to-end encrypted email system is far-off

No single party controls the email ecosystem, and widespread deployment of secure email appears to need cooperation of numerous stakeholders. An analysis of stakeholders is provided in Appendix A, and our conclusion is that no individual stakeholder has the capability to build (or the desire to demand) a secure email system that provides seamless interoperability for all of the billions of email users and for all the diverse uses of email. Furthermore, our synthesis and evaluation of existing work indicates that no system comes close to scaling to worldwide demand, particularly with respect to the PKI, including supporting certificate directories as needed for obtaining, distributing, and certifying keys and their mappings to email addresses.

IX. Research and Development Directions

A single secure email solution suitable across the full spectrum of email users and scenarios appears unlikely; different use cases require different solutions [47]. Here we sketch some possible paths forward and open questions.

A. Linking existing communities

The operators of secure email communities could work toward interoperability mechanisms that would enable their members to send secure email to members of other communities. Organizations would need to standardize methods for resolving (*e.g.,* via DNS) the certificate directory of a given email domain and a design where either certificates are trusted by virtue of the directory's endorsement or by the signature of an external certificate authority. If the latter, it is unclear how to create and share a trust anchor store of the various authorities involved without replicating the challenges that come with the certificate authority system in use with the web [22]. As an alternative, certificate directories themselves could be authoritative over a particular namespace (the most natural being the domain name in the email address), perhaps with a key transparency system like CONIKS to audit directories. It is unclear how many organizations would open their directories to outside queries (this may increase encrypted spam and malware) or otherwise cooperate in providing this functionality.

B. Bootstrapping end-to-end encryption

Given the ongoing adoption challenges for secure email, additional attention could be given to helping end users transition from plaintext to secure email. One possibility is to start with highly usable software that allows users to easily send occasional encrypted emails to each other. This could be done with trust-on-first use exchange of encryption public keys among a user's existing circle of contacts, an automated certificate directory, or other methods that score high on usability and deployability. An intriguing option is to deploy a system similar to Let's Encrypt, but for email certificates; email clients can, by means similar to how domain validated (DV) web certifcates are currently issued, automatically prove ownership of an email address in order to receive a certificate for the user. An open question is how to help users continue this transition to additional, stronger methods for authenticating public keys as they use secure email more frequently. It may be possible to build a system that migrates from one key

management row in Table I to another, or to build systems that combine the benefits of multiple rows.

C. Changing secure email functionality

Most efforts to provide secure email have tried to maintain all of current email functionality. One of the primary tradeoffs illustrated in our evaluation is between end-to-end encryption and server access to plaintext, the latter being important for searching archives and filtering spam and malware. Rather than trying to meet both needs simultaneously, email clients could explore offering different functionality for secure email. One possibility is to mimic secure messaging applications, which can automatically expire messages after a short lifetime (configurable by the user). If users view this as a privacy-enhancing feature, it would remove the expectation for searchable archives. An open question is whether this or any other change in functionality meets the needs of some users for some use cases and might lead to higher adoption.

D. Removing key management barriers

Much of the work in the academic and developer communities has focused on public key discovery. There are still open questions regarding how non-enterprise users will manage the full key life cycle, including private key storage, expiration, backup, and recovery [94, §13.7]. These questions are complicated by issues such as whether to use separate keys for encrypting email during transmission as opposed to those for long-term storage (see §IV-E), how much automation can be used to simplify key management, and how much users need to know in order to safeguard them. Use of short-term keys could alleviate some of these problems but create new ones. Many of these issues are not unique to secure email and are still unsolved for applications like secure messaging.

E. Addressing archive vulnerability

In recent years, the security of email has received routine media coverage for numerous cases of email 'hacking': the digital theft of the extensive information available in the long-term email archives of various individuals, companies, and organizations. It is ironic that the most active areas of research into securing email are largely orthogonal to this widely reported threat. While this issue might be categorized as a general data security issue, the way email products and architectures are designed (*e.g.,* emails archived by default, mail servers accessible by password) are contributing factors toward this vulnerability. It is unclear if technical solutions, revised social norms about email retention, or other approaches could be helpful in addressing this issue.

F. Overcoming adoption factors

The literature we reviewed in §IV-G indicates that usability may not be the primary issue preventing users from adopting secure email products. Researchers have examined human behavior, as it relates to security and privacy decisions, more broadly than lab-based usability studies of specific technologies. This literature includes, as a small sampling, studies of mental models [17], behavior and micro-economics [4], risk perception [126], and modelling for human error [24]. While research summarized previously sheds light on why users claim to not need secure email [3], [36], [44], [47], [107], [111], [115], it is less clear if there are underlying factors that can be better assessed through understanding human behavior or if increased adoption can be designed for.

X. Conclusion

Deployment and adoption of end-to-end encrypted email continues to face many challenges. Usability issues related to the day-to-day use of encrypted email remain in many free PGP-based tools, although they were reasonably addressed by enterprise S/MIME products starting in the 1990s (*e.g.,* IBM/Lotus Notes). Other barriers are important as well. For example, the issue of decryption key backup, and partially related, controversial key escrow issues, continues to divide communities. This results in diverging opinions on the "right" architectural solution, and progress is frozen as both software vendors and purchasing organizations try to avoid "wrong" decisions by maintaining the status quo (plaintext email). Those affirming the view that all secure email products should allow warranted law enforcement access by design should never be expected to agree with the traditional supporters of end-to-end encryption. Email service providers and organizations placing high value on malware and spam filtering methods that require plaintext access are also implicitly opposing (current) end-to-end encryption technologies, whether or not they are philosophically opposed. Ironically, even those favoring end-to-end encryption might put up road-blocks due to their own internal divisions; *e.g.,* on whether S/MIME is better than PGP or vice-versa. This stalls progress and, worse, adds technical complications to support different message, certificate, and key formats. These examples demonstrate that numerous forces often directly, or indirectly, hinder the adoption of end-to-end encryption. Perhaps the lesson is that we should be less surprised by its lack of deployment.

It also appears increasingly unlikely that a one-size-fits-all (*i.e.,* for all target scenarios, environments, and user classes) solution architecture will emerge. Enterprise organizations and individual consumers have different views of priorities and requirements, as well as widely differing levels of technical support. In the end, perhaps the biggest barrier is the implicit belief that we must not move forward until finding a one-size-fits-all solution. A common manifestation of this is those who oppose the roll-out of products or services that don't meet their own particular needs (rather than acknowledging the need for alternate approaches, and supporting advancement of their agenda in parallel without blocking that of others). Divided communities and differing visions are thus a big part of the problem. Welcome to the real world.

References

[1] H. Abelson, R. Anderson, S. M. Bellovin, J. Benaloh, M. Blaze, W. Diffie, J. Gilmore, M. Green, S. Landau, P. G. Neumann, et al. Keys under doormats: Mandating insecurity by requiring government access to all data and communications. *Journal of Cybersecurity*, 1(1), 2015.

[2] H. Abelson, R. J. Anderson, S. M. Bellovin, J. Benaloh, M. Blaze, W. Diffie, J. Gilmore, P. G. Neumann, R. L. Rivest, J. I. Schiller, et al. The risks of key recovery, key escrow, and trusted third-party encryption. *World Wide Web Journal*, 2(3), 1997.

[3] R. Abu-Salma, M. A. Sasse, J. Bonneau, A. Danilova, A. Naiakshina, and M. Smith. Obstacles to the adoption of secure communication tools. In *IEEE S&P*, 2017.

[4] A. Acquisti, L. Brandimarte, and G. Loewenstein. Privacy and human behavior in the age of information. *Science*, 347(6221):509–514, 2015.

[5] C. Adams and S. Lloyd. *Understanding PKI: concepts, standards, and deployment considerations*. Addison-Wesley Professional, 2003.

[6] K. Andersen, B. Long, S. M. Jones, and M. Kucherawy. Authenticated Received Chain (ARC) protocol. Internet-Draft draft-ietf-dmarc-arc-protocol-09, Internet Engineering Task Force, July 2017. work in progress.

[7] E. Atwater, C. Bocovich, U. Hengartner, E. Lank, and I. Goldberg. Leading Johnny to water: Designing for usability and trust. In *SOUPS*, 2015.

[8] A. Back. Hashcash-a denial of service counter-measure, 2002.

[9] W. Bai, M. Namara, Y. Qian, P. G. Kelley, M. L. Mazurek, and D. Kim. An inconvenient trust: User attitudes toward security and usability tradeoffs for key-directory encryption systems. In *SOUPS*, 2016.

[10] D. Balenson. Privacy enhancement for internet electronic mail: Part III: Algorithms, modes, and identifiers. RFC 1423, RFC Editor, February 1993.

[11] D. Basin, C. Cremers, T. H.-J. Kim, A. Perrig, R. Sasse, and P. Szalachowski. ARPKI: Attack resilient public-key infrastructure. In *CCS*, 2014.

[12] S. M. Bellovin. Easy email encryption. *IEEE S&P Magazine*, 14(6), 2016.

[13] V. Birk, H. Marques, Shelburn, and S. Koechli. pretty Easy privacy (pEp): Privacy by default. Internet-Draft draft-birk-pep-00, Internet Engineering Task Force, June 2017. work in progress.

[14] N. Borisov, I. Goldberg, and E. Brewer. Off-the-record communication, or, why not to use PGP. In *WPES*, 2004.

[15] I. Brown and B. Laurie. Security against compelled disclosure. In *ACSAC*, 2000.

[16] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer. OpenPGP message format. RFC 4880, RFC Editor, November 2007.

[17] L. J. Camp. Mental models of privacy and security. *IEEE Technology and Society Magazine*, 28(3), 2009.

[18] D. D. Caputo, S. L. Pfleeger, J. D. Freeman, and M. E. Johnson. Going spear phishing: Exploring embedded training and awareness. *IEEE S&P Magazine*, 12(1), 2014.

[19] R. Chandramouli, S. L. Garfinkel, S. J. Nightingale, and S. W. Rose. Trustworthy email. *Special Publication (NIST SP) 800-177*, 2016.

[20] D. Chaum. Designated confirmer signatures. In *EUROCRYPT*. Springer Berlin Heidelberg, 1995.

[21] D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *CACM*, Feb. 1981.

[22] J. Clark and P. v. Oorschot. SSL and HTTPS: Revisiting past challenges and evaluating certificate trust model enhancements. In *IEEE S&P*, 2013.

[23] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile. RFC 5280, RFC Editor, May 2008.

[24] L. F. Cranor. A framework for reasoning about the human in the loop. In *Usability, Psychology, and Security (UPSEC)*, 2008.

[25] D. Crocker. Standard for the format of ARPA internet text messages. RFC 822, RFC Editor, August 1982.

[26] D. Crocker, P. Hallam-Baker, and T. Hansen. DomainKeys Identified Mail (DKIM) service overview. RFC 5585, RFC Editor, July 2009.

[27] S. Dechand, D. Schürmann, T. IBR, K. Busse, Y. Acar, S. Fahl, and M. Smith. An empirical study of textual key-fingerprint representations. In *USENIX Security Symposium*, 2016.

[28] W. Diffie and S. Landau. *Privacy on the Line: The Politics of Wiretapping and Encryption (2/e)*. The MIT Press, 2007. Second edition 2007 (472 pages), first edition 1998 (352 pages).

[29] R. Dingledine and N. Mathewson. Anonymity loves company: Usability and the network effect. In *WEIS*, 2006.

[30] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *USENIX Security Symposium*, 2004.

[31] Z. Durumeric, D. Adrian, A. Mirian, J. Kasten, E. Bursztein, N. Lidzborski, K. Thomas, V. Eranti, M. Bailey, and J. A. Halderman. Neither snow nor rain nor MITM...: An empirical analysis of email delivery security. In *IMC*, 2015.

[32] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In *CRYPTO*, 1992.

[33] M. Elkins, D. D. Torto, R. Levien, and T. Roessler. MIME security with OpenPGP. RFC 3156, RFC Editor, August 2001.

[34] S. Englehardt, J. Han, and A. Narayanan. I never signed up for this: privacy implications of email tracking. PETS, 2018.

[35] M. Fagan and M. M. H. Khan. Why do they do what they do?: A study of what motivates users to (not) follow computer security advice. In *SOUPS*, 2016.

[36] S. Farrell. Why don't we encrypt our email? *IEEE Internet Computing*, 13(1), 2009.

[37] J. Fenton. Analysis of threats motivating DomainKeys Identified Mail (DKIM). RFC 4686, RFC Editor, September 2006.

[38] D. Florêncio, C. Herley, and P. C. Van Oorschot. An administrator's guide to internet password research. In *LISA*, 2014.

[39] I. D. Foster, J. Larson, M. Masich, A. C. Snoeren, S. Savage, and K. Levchenko. Security by any other name: On the effectiveness of provider based email security. In *CCS*, 2015.

[40] L. Franceschi-Bicchierai. Even the inventor of PGP doesn't use PGP, September 2015.

[41] J. Franklin, A. Perrig, V. Paxson, and S. Savage. An inquiry into the nature and causes of the wealth of internet miscreants. In *CCS*, 2007.

[42] N. Freed and N. S. Borenstein. Multipurpose Internet Mail Extensions (MIME) part one: Format of internet message bodies. RFC 2045, RFC Editor, November 1996.

[43] A. Fry, S. Chiasson, and A. Somayaji. Not sealed but delivered: The (un) usability of S/MIME today. In *ASIA*, 2012.

[44] S. L. Garfinkel, D. Margrave, J. I. Schiller, E. Nordlander, and R. C. Miller. How to make secure email easier to use. In *CHI*, 2005.

[45] S. L. Garfinkel and R. C. Miller. Johnny 2: a user test of key continuity management with S/MIME and Outlook Express. In *SOUPS*, 2005.

[46] U. Gasser, N. Gertner, J. L. Goldsmith, S. Landau, J. S. Nye, D. O'Brien, M. G. Olsen, D. Renan, J. Sanchez, B. Schneider, et al. Don't panic: Making progress on the "going dark" debate, 2016.

[47] S. Gaw, E. W. Felten, and P. Fernandez-Kelly. Secrecy, flagging, and paranoia: Adoption criteria in encrypted email. In *CHI*, 2006.

[48] R. Geambasu, T. Kohno, A. A. Levy, and H. M. Levy. Vanish: Increasing data privacy with self-destructing data. In *USENIX Security Symposium*, 2009.

[49] R. Gellens and J. Klensin. Message submission for mail. RFC 6409, RFC Editor, November 2011.

[50] C. Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009.

[51] I. Goldberg. Privacy-enhancing technologies for the Internet, II: Five years later. In *PETS*, 2003.

[52] I. Goldberg. Privacy enhancing technologies for the Internet III: Ten years later. In A. Acquisti, S. Gritzalis, C. Lambrinoudakis, and S. De Capitani di Vimercati, editors, *Digital Privacy: Theory, Technologies and Practices*. Auerbach Press, 2007.

[53] I. Goldberg, D. Wagner, and E. Brewer. Privacy-enhancing technologies for the Internet. In *IEEE COMPCON. Digest of Papers*, Feb 1997.

[54] I. A. Goldberg. *A Pseudonymous Communications Infrastructure for the Internet*. PhD thesis, UC Berkeley, 2000.

[55] D. Goldschlag, M. Reed, and P. Syverson. Onion routing. *CACM*, Feb. 1999.

[56] D. Goodin. Use of tor helped FBI ID suspect in bomb hoax case. Ars Technica, December 2013.

[57] C. Hadnagy. *Social Engineering: The art of human hacking*. Wiley, 2010.

[58] P. Hoffman. Allowing relaying in SMTP: A series of surveys. *Internet Mail Consortium Report*, 16, 2002.

[59] P. E. Hoffman. SMTP service extension for secure SMTP over Transport Layer Security. RFC 3207, RFC Editor, February 2002.

[60] R. Holz, J. Amann, O. Mehani, M. Wachs, and M. A. Kaafar. TLS in the wild: An internet-wide analysis of TLS-based protocols for electronic communication. In *NDSS*, 2016.

[61] R. Housley. Cryptographic Message Syntax (CMS). RFC 5652, RFC Editor, September 2009.

[62] H.-C. Hsiao, Y.-H. Lin, A. Studer, C. Studer, K.-H. Wang, H. Kikuchi, A. Perrig, H.-M. Sun, and B.-Y. Yang. A study of user-friendly hash comparison schemes. In *ACSAC*, 2009.

[63] J. Iedemska, G. Stringhini, R. Kemmerer, C. Kruegel, and G. Vigna. The tricks of the trade: What makes spam campaigns successful? In *SPW*, 2014.

[64] M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In *EUROCRYPT*, 1996.

[65] S. M. Jones, J. Rae-Grant, J. T. Adams, and K. Andersen. Recommended Usage of the Authenticated Received Chain (ARC). Internet-Draft draft-ietf-dmarc-arc-usage-02, Internet Engineering Task Force, June 2017. Work in progress.

[66] B. Kaliski. Privacy enhancement for internet electronic mail: Part iv: Key certification and related services. RFC 1424, RFC Editor, February 1993.

[67] S. Kamara. Encrypted search. *XRDS*, 21(3):30–34, Mar. 2015.

[68] R. Kang, L. Dabbish, N. Fruchter, and S. Kiesler. "My data just goes everywhere:" user mental models of the internet and implications for privacy and security. In *SOUPS*, 2015.

[69] A. Kapadia. A case (study) for usability in secure email communication. *IEEE S&P Magazine*, 5(2), 2007.

[70] S. Kent. Privacy enhancement for internet electronic mail: Part ii: Certificate-based key management. RFC 1422, RFC Editor, February 1993.

[71] S. T. Kent. Internet privacy enhanced mail. *CACM*, 36(8), 1993.

[72] M. Khonji, Y. Iraqi, and A. Jones. Phishing detection: a literature survey. *IEEE Communications Surveys & Tutorials*, 15(4), 2013.

[73] D. S. Kitterman. Sender Policy Framework (SPF) for authorizing use of domains in email, version 1. RFC 7208, RFC Editor, April 2014.

[74] J. C. Klensin. Simple Mail Transfer Protocol. RFC 5321, RFC Editor, October 2008.

[75] M. Kucherawy, D. Crocker, and T. Hansen. DomainKeys Identified Mail (DKIM) signatures. RFC 6376, RFC Editor, September 2011.

[76] M. Kucherawy and E. Zwicky. Domain-based Message Authentication, Reporting, and Conformance (DMARC). RFC 7489, RFC Editor, March 2015.

[77] A. Laszka, Y. Vorobeychik, and X. D. Koutsoukos. Optimal personalized filtering against spear-phishing attacks. In *AAAI*, 2015.

[78] B. Laurie. Certificate transparency. *CACM*, 57(10), Sept. 2014.

[79] B. Laurie and R. Clayton. Proof-of-work proves not to work; version 0.2. In *WEIS*, 2004.

[80] E. Lauzon. The philip zimmerman investigation: The start of the fall of export restrictions on encryption software under first amendment free speech issues. *Syracuse L. Rev.*, 48:1307, 1998.

[81] A. Lerner, E. Zeng, and F. Roesner. Confidante: Usable encrypted email: A case study with lawyers and journalists. In *IEEE EuroS&P*, 2017.

[82] K. Levchenko, A. Pitsillidis, N. Chachra, B. Enright, M. Félegyházi, C. Grier, T. Halvorson, C. Kanich, C. Kreibich, H. Liu, et al. Click trajectories: End-to-end analysis of the spam value chain. In *IEEE S&P*, 2011.

[83] R. Levien, L. McCarthy, and M. Blaze. Transparent internet e-mail security. In *NDSS*, 1996.

[84] L. Levison. Dark Internet Mail Environment architecture and specifications, March 2015.

[85] J. Linn. Privacy enhancement for internet electronic mail: Part i: Message encryption and authentication procedures. RFC 1421, RFC Editor, February 1993.

[86] D. Liu, S. Hao, and H. Wang. All your DNS records point to us: Understanding the security threats of dangling DNS records. In *CCS*, 2016.

[87] D. Margolis, M. Risher, N. Lidzborski, W. Chuang, D. Long, B. Ramakrishnan, A. Brotman, J. Jones, F. Martin, K. Umbach, and M. Laber. SMTP Strict Transport Security, Mar. 2016. work in progress.

[88] M. Marlinspike. GPG and me, February 2015.

[89] C. Masone and S. W. Smith. Abuse: PKI for real-world email trust. In *EuroPKI*. Springer, 2009.

[90] W. Mayer, A. Zauner, M. Schmiedecker, and M. Huber. No need for black chambers: Testing TLS in the e-mail ecosystem at large. In *IEEE ARES*, 2016.

[91] D. McCoy, A. Pitsillidis, J. Grant, N. Weaver, C. Kreibich, B. Krebs, G. Voelker, S. Savage, and K. Levchenko. PharmaLeaks: Understanding the business of online pharmaceutical affiliate programs. In *USENIX Security Symposium*, 2012.

[92] S. E. McGregor, E. A. Watkins, M. N. Al-Ameen, K. Caine, and F. Roesner. When the weakest link is strong: Secure collaboration in the case of the Panama papers. In *USENIX Security Symposium*, 2017.

[93] M. S. Melara, A. Blankstein, J. Bonneau, E. W. Felten, and M. J. Freedman. CONIKS: Bringing key transparency to end users. In *USENIX Security Symposium*, 2015.

[94] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone. *Handbook of applied cryptography*. CRC press, 1996.

[95] L. Meyerovich and B. Livshits. ConScript: Specifying and enforcing fine-grained security policies for JavaScript in the browser. In *IEEE S&P*, 2010.

[96] K. D. Mitnick and W. L. Simon. *The art of deception: Controlling the human element of security*. John Wiley & Sons, 2011.

[97] S. Nakamoto. Bitcoin: A peer-to-peer electionic cash system. Unpublished, 2008.

[98] A. Narayanan. What happened to the crypto dream?, part 1. *IEEE S&P Magazine*, 11, 2013.

[99] A. Narayanan. What happened to the crypto dream?, part 2. *IEEE S&P Magazine*, 11, 2013.

[100] C. Newman. Using TLS with IMAP, POP3 and ACAP. RFC 2595, RFC Editor, June 1999.

[101] Key management interoperability protocol specification version 1.3. OASIS Standard, 2016.

[102] H. Orman. *Encrypted Email: The History and Technology of Message Privacy*. Springer, 2015.

[103] C. Partridge. The technical development of internet email. *IEEE Annals of the History of Computing*, 30(2), 2008.

[104] T. Perrin and M. Marlinspike. Double ratchet algorithm, revision 1, 2016.

[105] B. Ramsdell and S. Turner. Secure/Multipurpose Internet Mail Extensions (S/MIME) version 3.2 message specification. RFC 5751, RFC Editor, January 2010.

[106] B. C. Ramsdell. S/MIME version 3 message specification. RFC 2633, RFC Editor, June 1999.

[107] K. Renaud, M. Volkamer, and A. Renkema-Padmos. Why doesn't Jane protect her privacy? In *PETS*, 2014.

[108] R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *ASIACRYPT*, 2001.

[109] U. Rivner. Anatomy of an attack. RSA blog.

[110] V. Roth, T. Straub, and K. Richter. Security and usability engineering with particular attention to electronic mail. *International Journal of Human-Computer Studies*, 63(1), 2005.

[111] S. Ruoti, J. Andersen, S. Heidbrink, M. O'Neill, E. Vaziripour, J. Wu, D. Zappala, and K. Seamons. "We're on the same page": A usability study of secure email using pairs of novice users. In *CHI*, 2016.

[112] S. Ruoti, J. Andersen, T. Hendershot, D. Zappala, and K. Seamons. Private webmail 2.0: Simple and easy-to-use secure email. In *UIST*, 2016.

[113] S. Ruoti, J. Andersen, D. Zappala, and K. Seamons. Why Johnny still, still can't encrypt: Evaluating the usability of a modern PGP client, 2015. arXiv preprint arXiv:1510.08555.

[114] S. Ruoti, N. Kim, B. Burgon, T. Van Der Horst, and K. Seamons. Confused Johnny: when automatic encryption leads to confusion and mistakes. In *SOUPS*, 2013.

[115] S. Ruoti, T. Monson, J. Wu, D. Zappala, and K. Seamons. Weighing context and trade-offs: How suburban adults selected their online security posture. In *SOUPS*, 2017.

[116] M. D. Ryan. Enhanced certificate transparency and end-to-end encrypted mail. In *NDSS*, 2014.

[117] B. Schneier and C. Hall. An improved e-mail security protocol. In *ACSAC*, 1997.

[118] S. Schröder, M. Huber, D. Wind, and C. Rottermanner. When SIGNAL hits the fan: On the usability and security of state-of-the-art secure mobile messaging. In *EuroUSEC*, 2016.

[119] A. Shamir. Identity-based cryptosystems and signature schemes. In *Crypto*, 1984.

[120] S. Sheng, L. Broderick, C. Koranda, and J. Hyland. Why Johnny still can't encrypt: Evaluating the usability of email encryption software. In *SOUPS - Poster Session*, 2006.

[121] R. Siemborski and A. Gulbrandsen. IMAP extension for Simple Authentication and Security Layer (SASL) initial client response. RFC 4959, RFC Editor, September 2007.

[122] R. Siemborski and A. Melnikov. SMTP service extension for authentication. RFC 4954, RFC Editor, July 2007.

[123] R. Siemborski and A. Menon-Sen. The Post Office Protocol (POP3) Simple Authentication and Security Layer (SASL) authentication mechanism. RFC 5034, RFC Editor, July 2007.
[124] D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *IEEE S&P*, 2000.
[125] E. Sparrow, H. Halpin, K. Kaneko, and R. Pollan. LEAP: A next-generation client vpn and encrypted email provider. In *CANS*, 2016.
[126] G. Stewart and D. Lacey. Death by a thousand facts: Criticising the technocratic approach to information security awareness. *Information Management & Computer Security*, 20(1), 2012.
[127] G. Stringhini, M. Egele, A. Zarras, T. Holz, C. Kruegel, and G. Vigna. B@bel: Leveraging email delivery for spam mitigation. In *USENIX Security Symposium*, 2012.
[128] G. Stringhini, O. Hohlfeld, C. Kruegel, and G. Vigna. The harvester, the botmaster, and the spammer: On the relations between the different actors in the spam landscape. In *CCS*, 2014.
[129] N. Unger, S. Dechand, J. Bonneau, S. Fahl, H. Perl, I. Goldberg, and M. Smith. SoK: Secure messaging. In *IEEE S&P*, 2015.
[130] F. Valsorda. Op-ed: I'm throwing in the towel in PGP, and I work in security. Ars Technica, December 2016.
[131] S. Van Acker, P. De Ryck, L. Desmet, F. Piessens, and W. Joosen. WebJail: Least-privilege integration of third-party components in web mashups. In *ACSAC*, 2011.
[132] R. Wash. Folk models of home computer security. In *SOUPS*, 2010.
[133] A. Whitten and J. D. Tygar. Why Johnny can't encrypt: A usability evaluation of PGP 5.0. In *USENIX Security Symposium*, 1999.
[134] S. Wolchok, O. S. Hoffman, N. Henninger, E. W. Felten, J. A. Haldermann, C. J. Rossback, B. Waters, and E. Witchel. Defeating vanish with low-cost sybil attacks against large DHTs. In *NDSS*, 2010.
[135] S. D. Wolthusen. A distributed multipurpose mail guard. In *IAW*, 2003.
[136] T. D. Wu. The secure remote password protocol. In *NDSS*, 1998.
[137] K. Zetter. Researchers uncover RSA phishing attack, hiding in plain sight. Wired Magazine, August 2011.
[138] P. Zimmermann. *PGP source code and internals*. MIT Press, 1995.
[139] P. Zimmermann. PGP marks 10th anniversary, 5 June 2001.
[140] P. R. Zimmermann. *The official PGP user's guide*. MIT press, 1995.

APPENDIX

A. Stakeholder analysis: who cares about secure email?

It's worth considering whether anybody cares enough about secure email to make it worthwhile putting in the effort that would be required to design and deploy it. Do users perceive enough risk from plaintext email, and enough confidence in a secure email solution (and their ability to use it), that they will make the effort to adopt and use it? A brief analysis of stakeholders illustrates some of the complexities involved:

a) Enterprises: Enterprises have an incentive to deploy secure email systems for internal communication, however corporate email security often prioritizes scanning incoming email for malware and ensuring outgoing email does not reveal company intellectual property. For external communication, many companies compete on service and costs rather than privacy, and security often reduces perceived level of service (due to usability) and increases costs. Thus, use of secure email is likely to be driven more by regulation or liability/lawsuit, than by companies voluntarily undertaking the burdens it imposes. This helps explain why consumer communication often takes place in information depots, in fields such as banking and healthcare. The ease with which information depots can be deployed and used reduces the incentive for businesses to help end users adopt end-to-end encryption.

b) Product vendors: When security products are marketed to end users, the "market for lemons" problem arises. Due to asymmetry in information between vendors and consumers, users have no way of measuring the security of a product accurately, and so they can't distinguish between secure and insecure services. The natural result is a stampede toward low-cost, more usable products and services; secure email loses in this environment.

c) End users: Non-enterprise end users have the power to choose among different secure email options provided to them or to continue using plaintext email. Most users to date have chosen plaintext and demand for secure email appears weak. Although some secure webmail systems report having nearly a million subscribers, this is small compared to billions of worldwide email users. As mentioned in IV-G, primary reasons for lack of adoption of secure email include lack of concern and lack of perceived need. Another complication is that any email products that don't support server-side spam/malware filtering and search features are unlikely to be chosen, at least until new techniques that work on encrypted archives are practical [124], [50], [67]. Users are more likely to notice the impact of lack of filtering and search than lack of privacy, since privacy violations are often silent.

d) Email providers: Email providers, driven by end user demand, have focused primarily on spam and malware filtering, along with link encryption. Deploying end-to-end encryption conflicts with filtering, and the need for interoperability between email providers further hampers deployment.

e) Privacy evangelists: Organizations that advocate for increased consumer privacy encourage greater end user adoption of secure email. These organizations typically sponsor development projects and educate end-users about privacy concerns and available products. However, advocacy methods are limited in their effectiveness. A technocratic approach, which involves experts telling the general public what they think the public should know, is considered harmful [126].

f) Governments: Governments have enacted regulations that both safeguard email privacy and hamper it, so it is difficult to predict how they may impact the adoption of secure email. For example, the General Data Protection Regulation (GDPR) in the European Union governs use of collected personal data (such as email contents), which could provide an incentive to encrypt data at rest in order to avoid liability. Conversely, in the U.S., the Electronic Communications Privacy Act of 1986 (ECPA) has been interpreted to mean that email stored online for over 180 days is abandoned and can be accessed without a warrant. The Email Privacy Act proposes to close this loophole, but this legislation is not yet adopted, despite broad support from the tech industry.

It's clear from this analysis that no stakeholder has the capability to build a secure email system that provides seamless interoperability for all of the billions of email users and for all of the diverse uses of email. No single party controls the email ecosystem, and widespread deployment of secure email appears to require cooperation of numerous stakeholders.

B. Evaluation Framework

We describe here the properties, cryptographic primitives, and key management primitives that form the columns and

rows of Table I. An extended version of this paper is available[8] that includes a dot-by-dot explanation of each cell of the table. We first describe the properties (columns) we use in Table I.

Security Properties (P1–P7)

P1 *Protection from network eavesdropping.* The sender and recipient may wish to keep email confidential during delivery. A full dot indicates that the message is kept confidential from all parties other than the sender and the recipient. A half dot indicates that the sender and receiver's mail servers are also able to read the message, but not other relaying servers or network middleboxes. Importantly, this column does not evaluate whether envelope metadata and message headers, including subject lines, are protected.

P2 *Protection from network tampering.* In addition to inspecting email content, a mail server or other network middlebox might alter the contents of email. A full dot is given if email cannot be modified undetectably by any entity except the sender, while a half dot allows modification by the sender and the sender's mail server.

P3 *Sender anonymity.* The sender of an email may wish to remain anonymous. We award a full dot if the sender's identity cannot be determined by any individual party, and a half dot if the sender's identity is linked to a reusable, pseudonymous identifier.

P4 *No third parties required.* The sender and recipient of an email may want to achieve confidentiality, integrity, and anonymity without reliance on third parties. If this is the case, the primitive is given a full dot. If third parties are required but the trust is distributed (*i.e.,* distinct third parties must be simultaneously compromised) and/or agile (*i.e.,* the user can choose which third-parties to trust) we award a half dot. An empty dot indicates that third parties are required and trust is neither distributed nor agile.

P5 *Private key only accessible to user.* From a security perspective, it is ideal if end-user signing and/or decryption keys reside only with the user. Architectures that support this are awarded a full dot. A half dot indicates that the private key resides with a third party, however the third party is denied direct access to the key by a password. Since passwords might not resist an offline guessing attack, this is weaker than denying the party any access to the private key. An empty dot indicates a third party has direct access to the private key.

P6 *Provides a public key audit trail.* In a successful impersonation attack, an adversary's public key is accepted by others in place of the user's real key. These attacks can be detected if a user is able to audit which public keys have been associated with their identity. If an audit log is available that provides non-equivocation (see §IV-D), we award it a full dot. If the audit log does not provide non-equivocation, then a half dot is given, and if no audit log is available than an empty dot is assigned.

[8]To be published after anonymous review stage; available by request.

P7 *Responsive public key revocation.* If compromise of a private key is detected, or a key is otherwise invalidated, its validity status should be signaled to other users in a timely manner. Architectures that allow for immediate and automatic checking of up-to-date revocation information earn a full dot. Revocation status updates that are untimely, not guaranteed to eventually arrive, fail open, or are not supported earn an empty dot.

Deployability Properties (P8–P12)

P8 *No email server/client modifications needed.* Primitives
P9 that need not change the implementation of email servers or clients (two columns) have the greatest potential for deployment. A full dot indicates there are no required software changes in the server/client, while an empty dot indicates changes are required. We do not award a half dot to highlight that the decision to change is the main hinderance, and the magnitude of the change is less consequential.

P10 *No new infrastructure needed.* Some primitives rely on new server infrastructure to be deployed, *e.g.,* to host a directory. Such primitives receive an empty dot, while ones that can be deployed within existing email infrastructure receive a full dot. For the same reason as directly above, we do not award a half dot.

P11 *Simple initial communication.* If a user can send an encrypted email to a recipient without needing the recipient to first adopt the primitive, then a full dot is awarded. If the sender must first wait for the recipient to adopt the primitive, then a half dot is awarded. If in addition to waiting for the recipient to adopt the primitive, the sender must also wait for the recipient to send their public key to the sender, then an empty dot is given.

P12 *Supports server-side content processing.* Primitives potentially limit the ability of email servers to perform services for the user—*e.g.,* spam and malware filtering, search, labels. A full dot is given if the primitive does not interfere with the inline server's ability to process content. A half dot is given if content processing can be enabled through cooperation between the email server and a third party without either party learning the plaintext contents of encrypted email. Note that research into computing on encrypted data is active and enables the composition of both message confidentiality and content processing [67], however we assume in our evaluation that only existing techniques are utilized.

Usability Properties (P13–P15)

P13 *Effortless public key discovery/validation.* The two
P14 columns, effortless encryption key discoverability and signing key validation, award a full dot when it is possible for an online user to automatically acquire *any* existing secure email user's public key. An empty dot indicates that this is not possible. We do not award a half dot. We consider encryption and signing independently because

encryption keys are needed *a priori* to sending a message, while signing keys can be validated *a posteriori*.

P15 Supports private key recovery. In large-scale deployments of secure email, private key loss is inevitable. If a primitive allows private key recovery, then a full dot is awarded, otherwise an empty dot is given. Note that this property also supports synchronization of a user's private keys between their devices, although this is uncommon at present.

We next describe the primitives (rows) used in Table I.

R1 Plaintext email. Email without any cryptographic protections.

Cryptographic Primitives (R2–R5)

Cryptographic primitives improve the security of email through the use of encryption and signatures. Each of these primitives must be paired with one or more key management primitives.

R2 Link encryption (§III-A). Encrypting the email content during transmission (i.e., TLS) prevents passive attackers from reading email messages. Protection from active man-in-the-middle attacks depends on which key management primitives are composed with this primitive. Note that this primitive only offers strong protection if all links along the transmission path are properly encrypted.

R3 Signature by originating mail server (§III-B). The sender's outgoing mail server signs email to identify the origin server and provide message integrity.

R4 End-to-end encryption and signing (§IV). Email is encrypted and signed with cryptographic keys associated with end-users. Cryptographic operations are executed at the end-user (client) devices. The security, deployability, and usability of this primitive depends largely on the key management primitive it is composed with.

R5 Hosted encryption and signing (§V-D). Instead of having end-user devices handle end-to-end encryption and signing, users upload their key-pairs to their respective mail servers and allow those mail servers to handle encryption and signing. This primitive can interoperate with end-to-end encryption and signing, albeit with impact on property P12.

Key Management Primitives (R6–R13)

R6 Certificate authority (§IV-B). After generating a key pair, a user submits their public key and distinguished name (i.e., email address) to a certificate authority. After validating the user's ownership of the distinguished name and the private key associated with the submitted public key, the certificate authority issues a signed certificate that binds the public key to the distinguished name. When receiving a certificate—either through an out-of-band communication channel or using another primitive (i.e., R9, R10)—a relying client validates the certificate, including verifying the signature, with trust anchored by a public key in the system's trusted root store.

R7 Manual certificate distribution (§IV-C). Users are responsible for exchanging and validating certificates amongst themselves. Most simply, this happens in person—for example, at a key signing party. Alternatively, a user could retrieve a certificate from a public source (e.g., a personal website), then validate that certificate by contacting the other user over an out-of-band channel (e.g., phone call).

R8 Web-of-trust (§IV-C3). This primitive builds on manual certificate distribution by adding trusted introducers. In addition to accepting certificates that they have personally exchanged, relying clients will also accept certificates that have been signed (endorsed) by one or more trusted introducers. This primitive does not aid in certificate discovery for encryption.

R9 Certificate directory (§IV-D). Users' certificates are stored and disseminated by a third-party. Disseminated certificates are trusted either because they have been signed by a trusted issuer (i.e., a certificate authority) or because the certificate directory itself is trusted to only disseminate verified certificates. Here "certificate" is either a signed structure or (unsigned) public key with metadata.

R10 Certificate ledger (§IV-D). This primitive extends the certificate directory primitive by making the directory's operations auditable by outside third-parties [93], [116]. In particular, clients can examine a history of all certificates that the directory has disseminated for the user's identity, allowing them to detect impersonation attacks.

R11 Directory with private key recovery (§IV-E). This primitive extends the certificate directory primitive by also including a server that stores users' decryption private keys. If a user loses their decryption private key, they can retrieve a copy from the key recovery server. While not included in Table I, identity-based encryption [119] is functionally similar to this primitive and if included would receive the same evaluation (i.e., dots).

R12 Trust on first use (§IV-F3). With this primitive, users attach their certificates to all outgoing messages, regardless of whether signed or encrypted. When a recipient receives a certificate for the first time, the recipient's client associates that certificate with the sender's identity. In the future, if a new certificate is received, the recipient's client must decide whether it is a valid update or an attack (an issue warranting further research).

R13 Shared secret (§V-B). Instead of using key-pairs (i.e., public key cryptography), clients can encrypt messages using a shared secret (i.e., symmetric key cryptography). These secrets could be a symmetric key or more likely a string both users know (e.g., password) that is hashed to a symmetric key. Secrets are shared in person or over an out-of-band channel that provides confidentiality.