SECURITY

Making It Easier to Encrypt Your Emails

JOHN S. KOH, STEVEN M. BELLOVIN, AND JASON NIEH

John S. Koh is a PhD candidate in computer science at Columbia University. John's interests lie in the intersection of applied cryptography, usability, and systems security with practicality in mind. koh@cs.columbia.edu

Steven M. Bellovin is Professor of Computer Science at Columbia University and affiliate faculty at its law school. His research specializes in security, privacy, and related legal and policy issues. He co-authored *Firewalls and Internet Security*, the first book on the subject. He is a member of the National Academy of Engineering and received the USENIX "Flame" award for co-inventing Netnews. smb@cs.columbia.edu

Jason Nieh is Professor of Computer Science and Co-Director of the Software Systems Laboratory at Columbia University. Professor Nieh has made research contributions in software systems across a broad range of areas, including operating systems, virtualization, thin-client computing, cloud computing, mobile computing, multimedia, web technologies, and performance evaluation. nieh@cs.columbia.edu

We've known for decades how difficult it is to encrypt email. We've developed E3, a client-side system that encrypts email at rest on mail servers to mitigate the most common cases of attacks today. E3 also demonstrates techniques for making key management simple enough for most users, including those who use email on multiple devices.

Email privacy is of crucial importance. Although email accounts and servers contain troves of valuable private information dating back years, they are easy to compromise. This makes them attractive targets for adversaries. Attackers often use methods such as spear-phishing, password recovery and reset, and social engineering attacks to obtain a victim's email credentials. With login details in hand, attackers then simply authenticate to the appropriate mail service like a normal user and siphon off all of the victim's emails.

We have seen this situation repeatedly in the news such as with the John Podesta, Sarah Palin, and John Brennan email hacks, among many more. Email encryption using a key inaccessible to the email service provider would have mitigated all these attacks. But none of these victims used encrypted email. If even prominent VIPs with access to top-notch advice are failing to use any kind of encrypted email, then everyday non-technical users are very unlikely to adopt email encryption. What makes this even worse is that a single breach of this kind is enough to compromise the entire history of affected users' emails. With the explosive growth of cloud storage, it is easy to keep gigabytes of old emails at no cost forever.

Existing email encryption approaches are comprehensive and effective against attackers but are seldom used due to their complexity and inconvenience. Examples include Pretty Good Privacy (PGP) [1] and Secure/Multipurpose Internet Mail Extensions (S/MIME), which are end-to-end encrypted email solutions. They are frankly too complicated to use, yet they represent the state of the art for secure email. The current paradigm for secure email places too much of a burden on its users, especially senders of email, who must correctly encrypt emails, manage keys, understand public key cryptography, and coordinate with other potentially non-technical users [5, 6]. The result is even technical users rarely encrypt their email.

End-to-end encrypted email is overkill for most users. Mail services are increasingly using SSL/TLS for email in transit between SMTP and IMAP servers, and are forcing clients to use SSL/TLS or STARTTLS. One example is Google's Gmail service, which completely disables plain IMAP and therefore requires clients to use TLS connections. This makes a large part of end-to-end encryption's benefits redundant since emails are already being encrypted in transit. What users are vulnerable to is an adversary who steals email account credentials, such as via a database leak or a phishing attack, or who compromises entire mail servers, such as when governments issue subpoenas for and seize entire servers belonging to mail services. But end-to-end encryption for email protects against a vast array of rarely encountered attacks other than these. This comes at the cost of usability, creating a chasm between end-to-end encryption's absolute security, which almost nobody uses, and regular plaintext email with no encryption, which everybody uses. There is thus room for change.

Designing for End Users

Any new secure email solution needs to be easy to use and also platform independent to help make it as amenable as possible to users. This has historically been a difficult problem. Various approaches, both academic and commercial, have tried to make it easier to use secure email but at the cost of sacrificing platform independence. They only work within closed ecosystems, such as Lavabit and Posteo, or with other people using the same solution, such as with traditional end-to-end encrypted email. But perhaps even more importantly, the more widely used secure mail services often encrypt emails or users' individual private keys on their servers using master private keys accessible to them.

What we need is a secure email solution that works on any mail service (yes, even Gmail) and that uses a private key that is inaccessible to the mail service but is accessible to all of a user's multiple devices for reading email. At the same time, users shouldn't need to know about key management concepts, public key cryptography, and public key infrastructure (PKI). Just as important is that this solution must work nearly identically to a regular email client to minimize the learning curve.

We developed Easy Email Encryption (E3) [4] as the first step to filling the void between unusable but secure email encryption and usable but insecure plaintext email. E3 provides a client-side encrypt-on-receipt mechanism that makes it easy for users since they do not need to rely on PKI or coordinate with recipients. The onus is no longer on the sender to figure out how to use PGP or S/MIME. Instead, email clients automatically encrypt received email without user intervention. E3 protects all emails received prior to any email account or server compromise for the emails' lifetime, using threat models similar to those of more complex schemes such as PGP and S/MIME.

E3 is designed to be compatible with existing IMAP servers and IMAP clients to ease adoption. No changes to any IMAP servers are necessary. Users require only a single E3 client program to perform the encryption, but multiple E3 clients are supported as well. Existing mail clients do not need to be modified and can be used as is alongside a separate E3 background app or add-on. If desired, existing mail clients can be retrofitted with E3 instead of relying on a separate app or on an add-on.

Users are free to use their existing, unmodified mail clients to read E3-encrypted email if they support standard encrypted email formats. The vast majority of email clients support encrypted emails either natively or via add-ons. Other than the added security benefits of encryption, all functionality looks and feels the same as a typical email client, including spam filtering and having robust client-side search capability.

Key management, including key recovery, is simplified by a scheme we call per-device key (PDK) management, which provides significant benefits for the common email use case of having two or more devices for accessing email, e.g., desktop and mobile device mail clients. Users with multiple devices leverage PDK with no reliance on external services. Users who truly only use a single device still benefit from PDK's key configuration and management capabilities but rely on free and reliable cloud storage for recovery. E3 as a whole is a usable solution for encrypted email that protects a user's history of emails while also providing a simple platform-independent key management scheme.

Encrypt on Receipt

Encrypt on receipt can be described as follows: when a user's E3 client detects that the mail server has received a new plaintext email, it downloads it, encrypts it, and replaces the original plaintext email with the encrypted version. In practice this is implemented entirely on the client side through the use of several existing IMAP commands, so E3 requires no modifications to the IMAP server and protocol. The encryption format is either standard PGP or S/MIME depending on implementation preference. Encrypt on receipt confers many benefits for usability while still retaining important security properties.

Self-generated, self-signed key pairs. Since the user isn't sending encrypted email but simply storing it for himself, the key pairs used for encrypting, decrypting, and signing don't need to be trusted by others. The user doesn't need to know about PKI and complicated key exchanges with other confused users. Self-generated and self-signed key pairs are also useful for E3's key management approach.

Support for all IMAP services. Encrypt on receipt is compatible with any IMAP mail service with no server modifications, including Gmail, Yahoo!, AOL, Yandex, and so on. It is also compatible with server-side spam filters, anti-virus scanners, and even indexing for ad-based services since emails are encrypted after they are received, giving the server a window of time to process email before it is encrypted.

Client implementation and compatibility. Encrypt on receipt requires only modest implementation changes for existing IMAP mail clients. We implemented E3 on multiple platforms, including on a popular open-source Android mail client, K-9 Mail, to show this. Furthermore, since E3 uses standard encrypted email formats, emails can be read on any unmodified mail client that supports them. Examples for S/MIME include Apple Mail, Mozilla Thunderbird, and Microsoft Outlook.

Secure against future compromises. Since all emails are encrypted on receipt, they remain secure against any future compromise of a user's account or IMAP server. To the attacker, all old emails would be encrypted and therefore unusable. However, if the attacker retains access to the account, newly arriving emails will be vulnerable. Encrypt on receipt therefore represents a much better-than-nothing approach to security. The current norm for email security is no security, so protecting a user's thousands of old emails is much better than protecting absolutely none of her emails.

Security against wiretapping. Encrypt on receipt is not end-to-end encryption, so email is not sent in encrypted form. This is actually not an issue. These days, especially after the Snowden revelations of widespread government surveillance of the Internet, practically all mail services use TLS for both client-server and server-server connections to protect email in transit.

Users don't need to know crypto. The user doesn't manually encrypt email because the client handles all encryption and decryption automatically. This is an issue observed in user studies, including our own—sometimes users can't figure out that they need to press the encrypt button when sending encrypted email to others.

Crypto algorithms can be updated. Once in a blue moon, a crypto algorithm or key length is discovered to have problems or simply has become too weak. Re-encrypting E3 emails to a newer crypto standard is simple: re-encrypt all emails using the user's new key. In contrast, this situation poses a problem for traditional PGP and S/MIME because re-encrypting emails received from other PGP or S/MIME users may not be possible. Perhaps the original sender can no longer be reached to re-sign the new copy, and thus his signature data would be lost. Even if he were reachable, the process of asking someone else to sign your old emails is a tedious task and also requires expert knowledge from all participants of how end-to-end encryption works.

Per-Device Keys

E3 eliminates manual public key exchanges. This simplifies the key management by removing half of it. What remains is the problem of private keys when using multiple devices. Traditional security best practices advise users to never transport private keys because doing so is insecure. This advice is almost never followed in practice because users often access email from multiple devices, all of which need the same private key when using common secure email usage models.

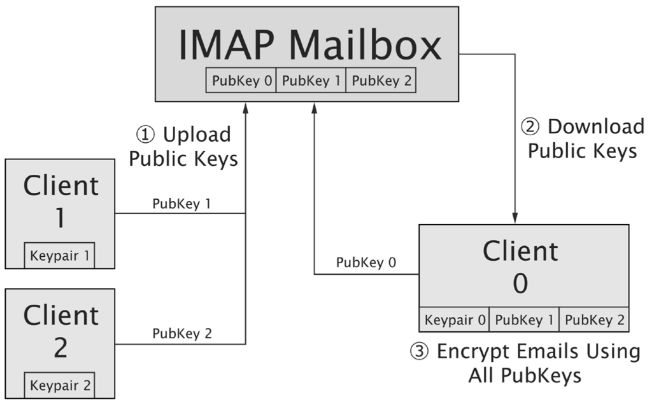

Figure 1: The per-device key (PDK) architecture

E3 returns to the traditional security advice of never transporting private keys. In contrast to most secure email schemes, which assume a user has a single private key, E3 asserts that a user should have a unique private key for every device. Then each device makes its public key available to the others. We call this the per-device key (PDK) scheme as depicted in Figure 1. PDK provides numerous benefits compared to traditional end-to-end encrypted email:

Complements self-generated, self-signed keys. One of the strengths of encrypt on receipt is that it can leverage self-generated, self-signed keys because it does not need to worry about third-party trust. PDK complements this scheme because each of a user's devices can generate its own self-signed key pair. This also greatly simplifies the process of adding a new device to a user's E3 ecosystem since it can just generate its own key pair.

Avoids moving private keys around. As we mentioned, traditional security best practices advise users to never move private keys around. Not only is this insecure, but most users have no concept of what a private key is and how it differs from its public key. With PDK, users don't need to know about these concepts and only know that their devices are encrypting their emails for them.

Eliminates manual public key exchanges. Instead of moving private keys, each of the user's E3 clients automatically makes available its public key to his other devices. Then any E3 client can encrypt the user's emails using the public keys from all of his devices. The principle is similar to when a traditional PGP or S/MIME user encrypts an email to multiple people. The email is not encrypted multiple times for each public key but is encrypted only once using a symmetric key, which in turn is encrypted to each public key. E3 takes this paradigm and applies it in a new way by encrypting emails on receipt using every verified public key belonging to the user. When a new key is added, clients re-encrypt already-encrypted emails to the new keys.

Requires no secondary communication channel. E3 maintains its requirement for platform independence even for its public-key exchanges. E3 clients upload their public keys to the mailbox as ordinary emails with the keys as attachments. Other E3 clients detect these key emails and store the public keys locally. These key emails contain a number of metadata fields to identify them but also to ensure security: for example, to support a secure key verification process.

Turns complicated key verification into simple device verification. In traditional end-to-end encrypted email, users must verify public keys, usually via trusted third parties. This places a burden on non-technical users who don't understand PKI. PDK also asks users to verify keys, but since E3 has different trust requirements compared to traditional end-to-end encrypted email, verifying PDK public keys is a much simpler process. Verifying a PDK public key means checking whether that key really belongs to one of the user's devices. E3 presents this as asking the user to verify whether she is adding a new device. Ideally, the method to do this should be compatible with any kind of device whether a desktop or mobile one. We therefore developed a process, which we refer to as a *two-way verification process*. A given client periodically scans for new keys, and when a new key is detected, the user is prompted to perform the two-way verification step.

The two-way verification process leverages a verification phrase that is easy for humans to recognize and match. When a client uploads its key for other devices to discover, it adds a randomly generated verification phrase to the key email, which is prominently displayed. The user then needs to confirm this verification phrase on one of his existing E3 clients. Once he completes the verification on any existing client, it will display a second verification phrase. The user then needs to confirm this second phrase on his new client to complete the two-way verification.

The catch is that when the user confirms a verification phrase, it must be selected from among two randomly generated incorrect phrases. The user must select the correct verification phrase in order to verify the key. This multiple-choice confirmation reduces the chances of a user accidentally accepting a key that isn't hers. The words in the phrases are selected from a curated pool such as the PGP Word List [3]. As shown in [2], this technique is effective and usable for quickly authenticating identities even with only three words.

Conclusion

Easy Email Encryption (E3) introduces new client-side encrypt-on-receipt and per-device key (PDK) mechanisms compatible with the existing IMAP standard and servers. E3 email clients automatically encrypt received email without user intervention, making it easy for users to protect the confidentiality of all emails received prior to any email account or server compromise. E3 uses keys that are self-generated and self-signed, and PDK makes it easy to use them to access encrypted email across multiple devices. Users no longer need to understand or rely on public key infrastructure, coordinate with recipients, or figure out how to use PGP or S/MIME.

E3 is also easy to implement, and we developed versions of it on a variety of platforms, including Android, Windows, Linux, and even Google Chrome. We also ensured that it works with popular IMAP-based email services, including Gmail, Yahoo!, AOL, and Yandex. Further, we conducted a user study to evaluate E3 usability, and results show that real users, even non-technical ones, consider E3 easy to use even when compared to using regular unencrypted email clients and vastly easier to use over the state of the art for PGP.

Twenty years ago, Whitten and Tygar's "Why Johnny Can't Encrypt" introduced Johnny to the research community as a representation of the average non-technical user who finds end-to-end encrypted email impossibly difficult to use [6]. However, we have seen an explosive growth of consumer-oriented technology since then. Always-on, always-connected mobile devices are ubiquitous, providing the necessary foundation for putting a new and usable spin on the idea of receiver-controlled encryption. Johnny may have been unable to encrypt, but Joanie in the modern age certainly can.

References

[1] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer, *OpenPGP Message Format*, IETF, RFC 4880, November 2007: http://www.rfc-editor.org/rfc/rfc4880.txt.

[2] M. Farb, Y.-H. Lin, T. H.-J. Kim, J. McCune, and A. Perrig, "SafeSlinger: Easy-to-Use and Secure Public-Key Exchange," in *Proceedings of the 19th Annual International Conference on Mobile Computing & Networking (MobiCom '13)*, ACM, pp. 417–428: https://doi.org/10.1145/2500423.2500428.

[3] P. Juola and P. Zimmermann, "Whole-Word Phonetic Distances and the PGPfone Alphabet," in *Proceedings of the 4th International Conference on Spoken Language Processing (ICSLP '96)*, vol. 1, IEEE, pp. 98–101: https://doi.org/10.1109/ICSLP.1996.607046.

[4] J. S. Koh, S. M. Bellovin, and J. Nieh, "Why Joanie Can Encrypt: Easy Email Encryption with Easy Key Management," in *Proceedings of the 14th EuroSys Conference 2019 (EuroSys '19)*, ACM, 2019, article no. 2: https://doi.org/10.1145/3302424.3303980.

[5] S. Ruoti, N. Kim, B. Burgon, T. Van Der Horst, and K. Seamons, "Confused Johnny: When Automatic Encryption Leads to Confusion and Mistakes," in *Proceedings of the 9th Symposium on Usable Privacy and Security (SOUPS 2013)*, ACM, article no. 5.

[6] A. Whitten and J. D. Tygar, "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0," in *Proceedings of the 8th USENIX Security Symposium (USENIX Security '99)*, USENIX Association, pp. 169–184: https://people.eecs.berkeley.edu/~tygar/papers/Why_Johnny_Cant_Encrypt/USENIX.pdf.