SOME 30 YEARS HAVE PASSED SINCE I FIRST USED ENCRYPTED EMAIL, AND I WAS REFLECTING ON THE NEAR-ABSENCE OF ITS ADOPTION BY ORDINARY INDIVIDUALS, EVEN THOSE WHO ARE SECURITY CONSCIOUS. GIVEN THE KNOWLEDGE THAT WE NOW HAVE ABOUT EXTENSIVE GOVERNMENT SURVEILLANCE OVER INTERNET TRAFFIC, WE SHOULD CONCLUDE THAT END-TO-END ENCRYPTION IS THE ONLY TECHNOLOGY THAT CAN ASSURE EMAIL PRIVACY. THIS HAS LED ME TO PONDER SOME QUESTIONS, SUCH AS, “ARE THERE READILY AVAILABLE TOOLS FOR USING CRYPTOGRAPHY IN EMAIL? SHOULD WE BE ENCOURAGING OUR CORRESPONDENTS TO TURN ON THE ‘ENCRYPT’ FLAG?”

Fifteen years ago, Doug Tygar and Alma Whitten wrote an interesting paper, “Why Johnny Can’t Encrypt.” In it, they discussed the usability of encryption tools. They found that the predominant tool at the time, Pretty Good Privacy (PGP), failed to be “usable,” in part because users didn’t understand how cryptographic keys worked, where to get them, and how to use them. Are tools today more usable, and are there new usability issues? How do smartphones, tablets, and “bring your own device” policies change the picture?

The Internet is a much more complicated place than it was at the dawn of encrypted email, and despite its ubiquitous nature, perhaps email is passe. Should we worry more about new messaging apps, such as Facebook, Twitter, Skype, and Snapchat? What about Bitcoin?

I can’t answer all these questions in this small space, but this article intends to show that end-to-end encryption is somewhat understandable, accessible with some effort, possibly extensible to social media, and in the end, perhaps saved by the global reach of social networks.

**Why Do We Need End-to-End Encryption?**

Although email encryption standards were developed 30 years ago, other than for a few experimental purposes, only two people have ever asked to communicate with me via encrypted email. On the other hand, I have observed that some people routinely use digital signatures to ensure their email’s integrity and authenticity. I was curious about how much email is protected, and based on a simple analysis of non-spam messages that I’ve received in the past several months, I estimate that fewer than one person in a thousand signs messages. Because digital signatures are part of encrypted email, I assume that this represents the fraction of people who have configured their email software to use cryptography. This means that if I were to start encrypting email, I would have an uphill battle to convince my correspondents to start doing the same, because it seems that few of them have encryption enabled or available.

In talking about email privacy, I’m focusing on measures that defend email against any kind of snooping. The strongest guarantee of privacy for email is end-to-end encryption. This means that no one but the sender and the receiver can decrypt the communication. The email provider can’t decrypt the messages because only the two communicating parties have the necessary keys. No eavesdropper can understand the messages, and no one with access to files on the email server can read them.

Without end-to-end encryption, eavesdroppers have any number of ways to read email. They can read data from communication lines as the email goes from a user to an email server or from one server to another. They can copy data from email servers or the repositories that hold email in “folders.” Eavesdroppers can be unethical employees at datacenters, criminals who have gained illegal access to servers, or government agents with specific or blanket search warrants. Even if service providers try to protect communications using point-to-point encryption (for example, STARTTLS for the Simple
Email Privacy: What You Need to Know

First, let’s step back and review what’s needed for end-to-end email privacy. The sender and the receiver want to know that the message is actually from the sender, it is intended for the receiver, no one but the sender could have sent the message, and no one but the sender and the receiver can decrypt it. These goals are usually accomplished through a combination of public-key algorithms and symmetric encryption.

Six pieces of information are prerequisites for secure email communication: the sender’s identity, the intended recipient’s identity, a cryptographic binding between the message and the sender, a cryptographic binding between the encrypted message and the intended recipient, assurance that the message hasn’t been altered during transmission, and a transformation of the message from “plaintext” to “ciphertext” using a function that’s extremely difficult to reverse without a message key that only the sender and receiver know.

Why Won’t Johnny Encrypt?

Many companies have email systems that take care of key issuance and management, and all the complexity might be hidden from users through a gateway machine. If Bob wants to use his own reputation to establish a key, then he can create a self-signed certificate that says “Entity Bob, known as bob@example.com, asserts that the string of bits XYZ is the public key of the entity known as Bob.” This demonstrates that Bob has the private key that matches XYZ. For ordinary users, this level of trust is usually sufficient. After all, if you’ve been in the habit of sending email to “Bob,” then you’ll probably be happy enough to trust a certificate that he sends you.

End-to-end encryption is the solution to surveillance avoidance, but with great privacy comes great responsibility.

Why You Need a Certificate

Email systems generally represent public keys as part of a complicated data structure called a certificate. A certificate is a way of saying something like “Entity Alice asserts that the string of bits XYZ is the public key of the entity known as Bob with email address bob@example.com, and Alice has attached her digital signature S to this statement.” If an email user thinks that Alice is trustworthy, then the certificate binding bob@example.com to XYZ can be saved in the key-management store on the recipient’s machine. If Carol has Bob’s certificate, she can use his public key to send secure email to him.
The company might have a database of keys, and the user email clients might look these keys up and use them without having to bother the user. These solutions are especially important for organizations such as the US military; for example, I regularly get press releases from a local National Guard unit, and their messages always have a digital signature.

Do modern email clients have a “usable” interface to end-to-end encryption? In my observation, most of them support the first three items fairly well, though many users will have some trouble with the “understanding” requirement, because public-key technology isn’t an intuitive concept; it takes some experience to become comfortable with the components. Neophytes always have trouble with the next two items, because the notions of public and private keys, certificates, and email attachments can become mind-numbing. The caveats implicit in the sixth item will always haunt us; there are just too many ways to lose information.

Many users will have some trouble with the ‘understanding’ requirement, because public-key technology isn’t an intuitive concept.

Despite all this, “a few hours” will probably suffice to get almost anyone started on secure email.

**Configuration, Import, Export**

Linux, Apple OS X and iOS, and Microsoft all support signed and encrypted email through their widely used email clients. Google and Yahoo have announced that support for their email clients is on the way. So, in theory, nothing stands between a user and end-to-end privacy. But users probably won’t find a “Secure email” tab that does all the configuration automatically. It’s best to start by acquiring the cryptographic keys that establish an email identity. An identity is associated with an email account, and you will need separate cryptographic keys for each of your email accounts.

The first obstacle that encrypted email neophytes face is getting their email name connected to a pair of related cryptographic keys (the public and private keys discussed previously). These are sometimes called the certificate, but this is a terminology conflation. The key pairs consist of two large numbers that are generated by cryptographic software libraries and that must be saved in two separate files. The public key is encoded into a certificate, whereas the private key is usually protected using a passphrase that the user makes up. The passphrase is then turned into an encryption key, which encrypts the private key, and the result is stored in a file. It’s essential to make note of the passphrase!

Apple’s OS X operating system has a key manager with a reasonable GUI, and it can generate keys for email. If you’re already familiar with the key manager, you might naturally start there, but if you start with the email client, you’ll spend some time fumbling around without getting anywhere — so start with the key manager (hint: there’s an icon with a keychain graphic). Once you’ve generated the keys, you’ll need to “export” them into a format that an email client can read. The usual exchange format is called PKCS #12 (see [www.emc.com/emc-plus/rsa-labs/standards-initiatives/public-key-cryptography](http://www.emc.com/emc-plus/rsa-labs/standards-initiatives/public-key-cryptography)). The associated file extension is .p12. Because the file contains the all-important secret key, it’s encrypted with a key derived from the username and that user-supplied passphrase that you should not forget.

Once I had all that out of the way, I was able to import the key file into an email client on an Apple mobile device. Getting the client to actually sign messages wasn’t too hard, but I couldn’t have done it without a little help from the Web. The option is buried deep in “Settings” for the email app, under “Advanced” (of course), and at the bottom of a menu list that’s covered by the virtual keyboard. With this done, I could...
immediately send digitally signed messages. Moreover, other email clients on Apple devices detected the signature and marked the message as verified. Yay!

Using Microsoft Outlook started out as a more intuitive process, but it went awry. Having found email signing under “Tools,” then “Accounts,” and then the “Advanced” (of course) settings for one account, the application noticed that I didn’t have a public key, and it offered links to several certificate providers, including one that required no payment. All went well with the Web registration, and shortly after that I received an email with a link for my certificate and private-key package. Clicking on the link didn’t download a p12 file as I had expected, but it did cause the browser to consume the data into its key storage. When I tried to export the private key from there, I ran into access permission problems that seemed unsolvable. I gave up on that certificate and instead attached the p12 file from an Apple device to an email message to myself, and read it with Outlook, which happily consumed that data into its key storage (note that it needed to know the passphrase for the private key).

Linux supports two independent software bases that are very helpful for generating keys: the GNU Privacy Guard (GPG; www.gnupg.org) and OpenSSL (www.openssl.org), each with extensive cryptographic capabilities. These use a command-line interface, and the incantations are baroque (are classical musicians attracted to Linux?). Fortunately, on Ubuntu Linux, the Seahorse key manager does most of the work using GPG through a simple GUI.

Although Seahorse can generate keys for email, if you want to use them on other devices, you need to know that GPG prefers to use its own for keys. The armored ascii format is the default, using the file extension asc.

Armored ascii could be unpalatable to systems that default to the public-key cryptography standards. To export the key to that format, a user might need to invoke gnupg through the command line and specify the option -export-format pkcs12.

Users who want detailed control over their key generation can use either GPG or OpenSSL from the command line. The first step in OpenSSL is to create a certificate authority with a public and private key. This step is followed by creating a new key pair for email and signing it using the certificate authority. Finally, you can export the newly created email key pair to the p12 format. Microsoft Outlook seemed to consume that file happily.

Many Accounts, Many Keys
When I first dabbled with email encryption, I had only one computer and one email account to configure. Today I have five computers that I use regularly, each one with a different email app running on a different operating system. I’ve also got a handful of different email accounts. It’s hard enough to keep my contact lists in sync, and the idea of keeping my keys current is daunting. However, to have the flexibility of using whatever device is most convenient while still assuring end-to-end privacy, the keys associated with each account must reside in each computer. That’s why the information about importing and exporting the cryptographic identities is so important.

As mentioned earlier, end-to-end security requires secret keys on each device, but having more copies of secret keys increases the risk of their exposure. Recently, I was talking to experts at a computer forensics company, and they told me that in 99 percent of their investigations, passphrases or passwords are found lodged somewhere in the device memory. Should you trust your mobile phone or tablet with the keys for your email identity? The risk might be acceptable for your “social” email account, but what about the one you use for professional work? Or, you might trust a device that you use for work but not games. There’s no simple answer; maybe someday we will have verified, trustworthy operating systems on trustworthy hardware, but until then, caveat keyholder.

Maybe someday we will have verified, trustworthy operating systems on trustworthy hardware, but until then, caveat keyholder.

However, once you’ve decided on these measures and have loaded your p12 or asc files with your digital identities into your endpoint email clients, then you’re ready to start sending signed email. If you can convince your correspondents to do the same, then your email to one another will be signed, and as a side effect, the certificates will be email attachments. The PKCS #7 format for signatures has the extension p7s (usually smime.p7s), and this contains the certificate chain for your identity, your public key, and your digital signature of the whole message’s hash. Most email clients recognize these attachments and automatically incorporate them into your key manager. When you compose a new email message to a user whose public-key certificate is in your key storage, the email client will recognize it and can encrypt the email to that recipient.

Of course, there are some gotchas. You’ll need to read the email
on each of your devices to get your correspondents’ keys into the key storage, or else you will need additional software to add them to synchronized contact lists. After a year or so, the certificates will expire, and you’ll need to generate new key pairs and send them to all your correspondents, and they’ll need to do the same. Anytime you get a new email account, you’ll need to get new keys and let all your correspondents see your new certificate, and they’ll be doing the same thing. This is a real usability killer, one of the major reasons that people give up on strong authentication.

Long ago, there was an expectation that global directories would simplify the problem of finding keys. Some people envisioned a hierarchical structure based on organizational units of government and industry, and other people thought that a collection of easy-to-update servers and a “web of trust” would predominate. But neither one has caught on for the majority of Internet users today. Consequently, key management is relegated to enterprise solutions or to a few dogged, security-obsessed individuals.

Social Media to the Rescue?
Generally, social media doesn’t have end-to-end encryption, though any text-based messaging system could, in theory, be “crypto-ized.” Some third-party applications offer help with this by providing, for example, their own database for storing and retrieving encrypted Twitter messages and handling the keys and crypto operations automatically. We might need to wait a few decades for this to become an integral part of messaging apps. But there could be a place for social media in the salvation of key management. Instead of global servers for publishing public keys, why not a social network?

The not-yet-in-alpha Keybase system (https://keybase.io/) proposes maintaining a website for bootstrapping key advertisements from registered users, but their software will do much more than hold a directory. They will also provide open source software that can run independently of the website, using it only for bootstrapping and occasional checks for new key types.

The exciting thing about Keybase is that it recognizes that public keys are used for diverse applications on the Internet. You can announce a public key on Twitter, for example. If you were to register as a Keybase user, and then registered your Twitter handle with them, Keybase would look for your announcement of your public key in your Twitter feed and would verify the key by checking the signature on it.

Your friends, should they also be Keybase users, would contact the Keybase server and look up your Keybase entries. Seeing that you had a Twitter handle, their Keybase clients would learn that you had posted a public key. The clients don’t act as zombies; they independently fetch the posting with the key and validate it. Each client can sign these keys as “trusted” and save them on the Keybase server. Using a new device? You can pick up your trusted keys from the Keybase server and start watching for updates. It sounds as though you could manage trusted keys yourself, using other tools, but you’d want to keep announcing your new keys on the Keybase server and watching for your friends to start using new apps.

Because Keybase is built on GPG, it has all the crypto utility functions for key import and export and cipher suites available to its client software. Many operating systems, particularly Linux-based systems, have GPG installed by default; your mileage may vary with others.

Will this work for Bitcoin accounts? Keybase says it will, and they include this interesting statement on their webpage: “We’re now embedding signed announcements in the Bitcoin blockchain.” If everyone starts doing this, I think we’ll need to increase the world’s energy production dramatically. We want nuclear fusion now!

End-to-end encryption for email is or will soon be available for almost all commonly used computing devices. Users who are interested in it can start with signed email and move gradually to encrypted email if they can convince their friends and colleagues to join them in the quest for privacy. This is an era of social media, in which privacy takes a backseat to openness, but perhaps social media can help spread the word about the importance and accessibility of email communication privacy. Privacy — can it be the next viral movement?

Reference

Hilarie Orman is a security consultant and president of Purple Streak. Her research interests include applied cryptography, secure operating systems, malware identification, security through semantic computing, and personal data mining. Orman has a BS in mathematics from the Massachusetts Institute of Technology. She’s a former chair of the IEEE Computer Society’s Technical Committee on Security and Privacy. Contact her at hilarie@purplestreak.com.

Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.