Symmetric Key, Two-Factor Email Encryption

Danielle Zucker
Yale University
Project Advisor: Michael Fischer
April 30, 2015

Abstract

Email message privacy is under frequent attack from various types of adversaries ranging from malicious attackers and spammers to large companies seeking to mine email data. However, many casual email users neglect to use email security protocols that would protect their messaging sessions, largely due to the difficulties and inefficiencies they introduce. Existing email security protocols implement public key cryptography, requiring users to exchange keys or certificates prior to messaging. I propose a symmetric key email encryption protocol as a solution to the usability issue. A random key specific to each encrypted message is generated, and the key and message are transmitted through two separate channels. Only a recipient with access to both channels is able to decrypt the message. Although this system is not secure against powerful and determined attackers, it provides casual security for informal email users while prioritizing usability and efficiency.

1 Introduction

Email security has become an increasingly important issue, with more and more privacy threats revealed each year. Current email encryption protocols employ public key infrastructure, requiring that individuals create and import certificates and then exchange these certificates with other users prior to messaging. Though extremely secure, these types of protocols are particularly challenging and inefficient for the average, non-technical email user who is more concerned with preventing casual eavesdropping or email scraping.

I propose an email security protocol that prioritizes usability while still maintaining a moderate degree of message privacy. The system implements a symmetric key encryption scheme with two-factor-like authentication users must possess both the ciphertext and the key to decrypt a message. A 128-bit key is randomly generated for each email exchange, and the message is encrypted using Advanced Encryption Standard (AES). The ciphertext and key are then encoded with base64 to ensure the transmission of ASCII printable characters. The ciphertext and key are split and transmitted separately, with the encrypted
message body sent to the recipient’s email account and the key conveyed through one of any number of other channels, including alternate email accounts or SMS text messages. The recipient must obtain the ciphertext and key to decrypt the email and read the intended message.

Though this protocol does not provide the unbreakable security of alternate two key systems, it does protect emails from all but the most powerful and determined adversaries who must have access both channels of transmission in order to read the message illicitly. The separate channels provide security similarly to two-factor authentication. Further, the protocol enhances usability for communicators, as no exchange of information or certificate creation is required prior to a messaging session.

2 Security Today

Information security is growing to be a more crucial and more salient topic with each passing year. Our online data, it can commonly appear, is under constant threat of attack. In 2014 alone, hundreds of millions of consumers’ personal and financial information was compromised, often through highly publicized security breaches of industry giants like Home Depot, JPMorgan, and EBay [1]. In addition, 2014 saw the discloser of the Heartbleed security vulnerability that compromised any web traffic encrypted with OpenSSL. Of this security bug, cryptography Bruce Schneier [2] wrote, ”'Catastrophic' is the right word. On a scale of 1 to 10, this is an 11.” Yet malicious adversaries are not only conducting their attacks to gain personal data: they are also targeting private messages. The recent and highly publicized Sony hacks released, among other pieces of sensitive information, the private email communications of dozens of employees, giving the public access to troves of embarrassing and unflattering messages [3]. Evidence gathered by the FBI indicates that these attacks on Sony were likely coordinated by a powerful adversary, North Korea. However, private email security is not only threatened by large, skilled adversaries like enemy countries; private messages are also compromised by non-malicious entities. For example, Google and other email providers monitor users’ private mail for the purpose of targeted advertising and tailored search engine results. In April 2014, Google even updated the language of its privacy policies to clarify how the company monitored emails: ”Our automated systems analyze your content (including emails) to provide personally relevant product features, such as customized search results, tailored advertising, and spam and malware detection. This analysis occurs as the content is sent, received, and when it is stored,” [4]. Thus, whether from evil adversaries or service providers themselves, an individual’s private messages are not entirely free from third-party intrusion.

It is clear that standard email protocols do not adequately protect messages from third-party access for those who are concerned about privacy of communication. There are, however, existing email encryption standards that offer high levels of security to users. Among the most popular of these standards are S/MIME [5], OpenPGP [6], BitMessage [7].
S/MIME allows senders to both encrypt messages and to attach a digital signature verifying their identities. Those who wish to use S/MIME must generate a certificate and distribute to all people whom they would like to message. They must also obtain their recipients’ certificates in order to message them securely.

OpenPGP employs a two-key cryptosystem, relying on PKI (Public Key Infrastructure). The sender and recipient must each generate a public and private key and then exchange them prior to sending emails. These keys must also be verified, or signed, and imported into the users’ email clients.

BitMessage functions in a similar manner, also requiring each user to obtain public and private keys and exchange them prior to sending messages. However, BitMessage has an added layer of anonymity; senders and recipients are not required to know each other’s identities. To allow for this, sent messages are broadcasted to all BitMessage users to decrypt, though only those users with the correct private key can decode the message into readable text. Senders are also required to compute a proof-of-work, which prevents malicious attackers from flooding the system with spam which would preclude the transmission of genuine messages.

With any of these three messaging protocols comes inherent inefficiencies. The reliance on PKI requires that users must exchange information prior to sending messages. A sender must possess the certificates of all his intended recipients, a process which requires added levels of effort on behalf of all involved in the message exchange. Furthermore, the process for obtaining one’s own certificate is not straightforward or easily navigable. For those who want to guarantee high levels of message security, S/MIME, OpenPGP and BitMessage can provide thus-far unbreakable encryption. However, the average, everyday email user likely does not need this level of security.

I propose a mid-level email security protocol that prioritizes both privacy and usability. Rather than relying on PKI or some other two key system, my protocol will implement a symmetric cryptosystem, with a unique, random key generated for each message sent. This will enhance usability by allowing users to send messages without exchanging any information prior to the message transmission. Further, users will not have to go through the process of generating and importing certificates, a procedure that is challenging especially for regular, non-sophisticated user. Instead, the application will automatically generate a session key and encrypt the message using Advanced Encryption Standard (AES-128) [8].

Security is enhanced by sending them through two separate channels. Similar to a two-factor authentication scheme [9], the recipient must have two pieces of information, the ciphertext and the key, to decrypt the intended message. Because these pieces of information are sent to the intended recipient separately, a malicious attacker would have to have the capability to monitor two channels in order to eavesdrop or modify messages.

It is important to note that this is not the first time that two channel encryption has been proposed as a security protocol. A similar concept using a one-time pad cipher was patented in 2001 [10]. However, this proposal is
presented as a solution to the usability issue, particularly the necessity of exchanging information prior to messaging.

While this email messaging protocol is not as secure as the two key systems described above and will not stop a determined and powerful adversity, it provides significant improvements in security for the average user who is trying to protect himself from email scraping or prying, and it maintains the efficiency and ease of use that many find appealing about email communication.

3 Implementation

This email encryption protocol relies on a symmetric cryptosystem. Messages are encrypted using Advanced Encryption Standard (AES), as implemented with Python’s pycrypto library. The key length is 128 bits, and a new key is generated for each message sent. Security comes from a splitting of the key and the encrypted message, with the ciphertext message sent through email and the key transmitted through either an SMS text message or a separate email. This implementation involves a two-part process: an encrypt-and-send function and a decrypt function.

To encrypt, the user must first run the Encrypt Message GUI. The user is prompted to supply his email address and his email password, which is masked by asterisks. It is important that, prior to attempting to run this app, the user has enabled the app to connect to his email account; otherwise, most modern email domains will block attempts to access the account through unverified apps. The user then has the option to select how the key will be sent to the message recipient: through email or through SMS text message to a cell phone.

If the user chooses to transmit the key through email, he is asked to supply the email address of his intended recipient. If he chooses to transmit the key through a cell phone message, he must additionally provide the recipient’s phone number and phone service provider. Then, the user types the subject and message body of his intended email, and has the option to encrypt the subject in addition to the message.

After all information has been accurately entered, the user sends the message by clicking a button on the GUI. After the message has been sent, the GUI remains open and does not clear the user’s email address or password, effectively keeping the email session open so that the user is free to send multiple encrypted messages efficiently.

After the user sends the message, the app generates a random 128-bit string session key. The message body (and subject, if the user indicates this) is encrypted with an AES cipher in ECB mode using the session key, and then both the encrypted message and key are converted to base64 text to prevent distortions in transmissions. The encrypted message is stored as a MIMEText object, along with its subject, sender, and recipient. A tag, "(enc)", is appended to the end of the subject to indicate that it has been encrypted. An SMTP client session object is then created and connected to a port for mail transmission. The connection uses TLS for increased transmission security.
Once the SMTP object is created, the user is logged in and the MIMEText object is sent. Then, if the user has indicated that the key should be sent through email, an additional MIMEText object is created, with a message body containing the session key in converted to base64 and a subject containing the message’s subject and a "Key for:" tag. This MIMEText object is also sent to the recipient’s email account. Otherwise, if the user selects to send the key through SMS, the provided cell phone number is parsed and the domain address of the provider is appended. This becomes the recipient’s address in the key’s MIMEText object. Then, the key is stored as the MIMEText object’s message body and the object is sent. After transmission, the SMTP object is closed.

To decrypt, the recipient must run the Decrypt Message GUI. He is presented with the option of manually entering the information needed to decrypt or automating the process, although automation is available only when the key has been transmitted through email. If the recipient chooses to enter the information manually, he must provide the key that was sent to him through email or SMS and the encrypted message (and subject, if it is also encrypted). He may either copy and paste this information into the GUI or type it. After the information is entered, the recipient clicks a decryption button. The app pulls the key and ciphertext, changes them from base64 to binary output. Then, the ciphertext is decrypted with an AES cipher. The plaintext is output to the GUI and is then readable by the recipient.

Otherwise, if the recipient chooses to allow the app to automatically decrypt, he must provide his email address and password, as well as the subject of the message he wants to decrypt. Once this has been provided, the recipient clicks a button to decrypt. The app creates an IMAP4 connection over SSL to remain encrypted, and the recipient’s email is accessed using the provided password. The inbox is then searched for two messages with the given subject. The message with the (enc) tag is stored as an email message object, and the body is parsed out and stored as the ciphertext. The message with the "Key for:" tag is also stored as an email message object, with the body stored as the key. After the ciphertext and key have been retrieved, the IMAP4 connection is closed. Then, the ciphertext and key are converted from base64 to binary and decrypted using AES. The plaintext message is similarly output to the GUI for the recipient to read.

4 Discussion

This implementation of two-factor, symmetric email encryption is designed to be moderately secure and easy to use. Because it is intended for everyday use, this protocol sacrifices some of the security of more advanced protocols, like S/MIME, OpenPGP or BitMessage, in exchange for usability. Rather than exchanging keys prior to message transmission, users are able to communicate through their email accounts without contacting each other before a conversation. Messages are encrypted with AES-128 using a randomly generated session key, and are then converted to base64 encoding to ensure that only printab
characters are transmitted. The encrypted message is sent to the recipient’s email account, while the key is split and sent separately. The current implementation allows for the transmission of the key through a separate email to the same address or through SMS text messaging to a mobile phone. Other possible channels of key transmission include:

1. A second email address also belonging to the recipient,
2. An attachment sent with the encrypted email message,
3. An email message sent a significant amount of time after the initial encrypted message,
4. A voicemail to a mobile phone or an audio message to an email account
5. Or a captcha or some other sort of message embedded in an image [11].

Any of these transmission options require that an adversary have access to two separate channels to obtain the two pieces of information required to decrypt an email message: the ciphertext and the key. While this security will not prevent powerful and determined attackers, such as the NSA [12], from conducting email surveillance, it will likely add a prohibitively challenging obstacle for casual eavesdroppers or scrapers looking to target their advertising. For most email users who send messages for business or personal reasons (rather than Top Secret information exchanges), the level of security provided by this protocol is sufficient to deter common adversaries. A greater level of privacy offers little advantage to these users.

While the two-factor system makes decryption extremely difficult for a hacker, a legitimate recipient who does have access to both channels of transmission can decode the message in one simple step. Users must submit the key and ciphertext to the application, which automatically performs decryption by converting from base64 to bit stream and then applying AES decoding. This implementation also allows for automation when the ciphertext and key are both sent to the same email account; the application needs an input of only the message subject, and can then search for the two factors required for decryption. Future updates may include decryption automation that works with any channel of key submission, where users must input the key but the ciphertext is found through a search conducted by the application. Alternatively, automation can be employed through a client-specific plugin that can be accessed when an email message is opened.

In this way, this symmetric key email encryption protocol allows for message security while also prioritizing usability for the average sender and recipient. Other popular email security protocols require the exchange of information before messaging can occur, adding an extra step that complicates the process. Furthermore, for the non-sophisticated user, the procedure for obtaining, downloading, and importing email certificates can be prohibitively complicated and difficult. The higher level of security provided by two key protocols is unnecessary for most users. Instead, a two factor-like, symmetric key system increases the usability with a small decrease in security. Hackers are unable to decrypt messages without access to two channels of communication, and users must
perform only one extra step in the messaging process combining the ciphertext and key. Thus, the proposed system offers an extremely usable and reasonably secure procedure for transmitting email message free from attacks from most typical adversaries.

References


https://www.schneier.com/blog/archives/2014/04/heartbleed.html


http://www.businessinsider.com/google-warns-we-are-scanning-your-email-2014-4


http://www.openpgp.org/

https://bitmessage.org/wiki/Main_Page

http://searchsecurity.techtarget.com/definition/Advanced-Encryption-Standard

http://www.google.com/patents/US20020146119

http://www.captcha.net/

http://www.theguardian.com/world/2013/jun/06/us-tech-giants-nsa-data