Symmetric Key Email Encryption

Data security has become an increasingly important issue as the world has come to rely more and more on the internet for data storage and transmission. Hacking, the act of exploiting weaknesses in computer security to gain unauthorized access to a device, originated almost simultaneously with the advent of the internet. Without proper security, computer users leave their devices susceptible to individuals who could obtain their personal information. However, it is not only private parties who are now attempting to access personal data uninvited. Former governmental employees have recently leaked evidence to the media of a widespread, secret surveillance program run by the United States federal government. Unbeknownst to private citizens, governmental agencies have amassed enormous amounts of data and metadata from inter-device communication.

Monitoring technological communication between parties is especially appealing because messaging devices are so widely employed. In particular, the use of email to communicate has greatly expanded in the recent decades. What was once originally a tool for academic communication has developed into a means of interaction across essentially all job fields and for nonprofessional, private parties. However, with the ever-increasing threat of unwanted surveillance, particularly by the federal government, the need for secure email has never been greater.

The most necessary requirement for keeping emails secure today is the guarantee that the email reaches the intended recipient, and only the intended recipient. To address this need for
secure email, several organizations and startups have created encrypted email protocols. These allow email communication between two parties that is protected from unauthorized access using encryption. The most well-known of these encrypted email protocols is OpenPGP (“Pretty Good Privacy”). OpenPGP employs a public key cryptosystem. During the encryption, OpenPGP generates a random key and encrypts the message using the key. Then, it encrypts the key using the public key associated with the intended recipient. After the message and key are encrypted, they are broadcasted, and the person with the associated private key is able to decrypt the key, and then the message. In this way, OpenPGP employs a public key cryptosystem to ensure the secure transmission of messages. However, there are several inefficiencies associated with this type of protocol. First, the use of a public key cryptosystem raises the problem trust; a sender must ensure that the private key associated with the public key actually belongs to the intended recipient and has been kept private. To do this, OpenPGP uses a “web of trust”; users of OpenPGP must get their keys signed by other user saying that they are trustworthy and who they claim to be. This introduces difficulties for users who are using OpenPGP for the first time, and necessitates face-to-face meeting such as “signing parties” to have new users’ keys signed.

Further, some implementations of OpenPGP are not entirely secure. Hushmail, for example, has recently been found to have collaborated with the federal government in its surveillance program.

A second email encryption protocol in use today is Bitmessage, which provides anonymity in addition to security. Bitmessage employs a protocol similar to bitcoin. A message is encrypted using a public key associated with the receiver’s private key, but before it can be broadcast, a proof of work must be completed. The message must also be time stamped to prevent flooding the system by broadcasting old messages. All users decrypt the broadcasted
message with their private keys to see if the message was intended for them. However, this 
system too has several inefficiencies. The proof of work is particularly time-consuming
(Bitmessage recommends that it take at least 4 minutes of computer operation), meaning that the 
system is extremely slow. The system also operates with best-effort delivery, meaning that 
messages are not guaranteed to be delivered on any given attempt; if a message is not delivered, 
the creator must re-time stamp and re-compute the proof of work before sending again. Finally, 
this cryptosystem is particularly challenging for the average, technologically inexperienced user 
to understand. Rather than email addresses, usernames are the hash of the user’s public key in 
base58, which is difficult to remember, and the user must also master time stamps and proofs of 
work.

For my senior project, I propose the creation of an easier, more efficient email 
cryptosystem that sacrifices some of the security of existing email security programs in favor of 
usability. Protocols like OpenPGP and Bitmessage are very secure, but that level of security is 
not necessary for most day-to-day email communication between most email users. Instead, the 
difficulty of navigating the OpenPGP and Bitmessage cryptosystems make them prohibitively 
challenging for people who need less than high-grade security. One of the greatest challenges to 
OpenPGP and Bitmessage comes from the use of a public key cryptosystem, which leads to a 
cumbersome identification process (like OpenPGP) or a hard-to-remember username (like 
Bitmessage). Instead, I will create a symmetric key cryptosystem, where the message is encoded 
and decoded using the same key. Specifically, each time a user wants to send a message, a 
random key will be generated. I will create and employ an encryption algorithm. The security 
from my system will come from the separation of message and key. The encrypted message will 
be sent over email as usual. The key, however, will be transmitted through Tor, an onion routing
network that provides online anonymity. The separation of key and message will provide extra security over the sending of unencrypted emails, but will provide a greater ease of use than public key email cryptosystems. As time permits, I will experiment with other methods of key transmission (such as using a cell phone text message, encryption of the key, etc.)

Project Steps:

1. Determine which encryption algorithm will be secure without using too much computing power
2. Implement an encryption and decryption algorithm to change plaintext to ciphertext and back
3. Implement a method for random key generation
4. Implement key transmission through Tor
5. Streamline process for usability
6. Explore other key transmission protocols

Sources:

https://bitmessage.org/wiki/Main_Page
http://www.openpgp.org/about_openpgp/