Tim Gass

John Thompson

Zach Reiss

**Email Encryption App**

A symmetric cipher algorithm was decided upon because there is no way for a handshake process between emails. Our application uses a 2 step algorithm with AES. AES requires two, 16 byte strings, one is the key and the other is the initialize Vector. The application begins with the original key, K1 and the vector V1 and uses a SecureRandom to generate 24 bytes. These 24 bytes undergo a similar expansion process as the mini-DES algorithm that Andrew taught the class.

Below, the randomly generated bytes are expanded as such to make the 32 bytes necessary.

0 1 2 3 | 4 5 | 6 7 8 9 | 10 11 | 12 13 14 15 | 16 17 | 18 19 20 21 |22 23

0 1 2 3 | 5 4 5 4 | 6 7 8 9 | 11 10 11 10 | 12 13 14 15 | 17 16 17 16 | 18 19 20 21 | 23 22 23 22

The first 16 bytes of the expanded form is key 2, and the remaining 16 bytes are vector 2. The actual email sent is the 24 bytes encrypted with (K1,V1) concatenated with the hash value of the message and the message itself both encrypted with (K2,V2).

Message sent: 24 characters (K1,V1) + (Hash value + plaintext)(K2,V2)

The decryption process starts with decrypting the first 24 bytes, then expanding those 24 bytes into (K2,V2) and using (K2,V2) to decrypt the message. The hash value received from this is then compared with the hash value of the received message to check if there was any tampering. Furthermore to prevent tampering, the application filters your inbox by MIME type, if an email was sent from gmail in a browser it wouldn’t be shown within the application. This prevents even us from easily demonstrating a man in the middle attack being detected with the hash value checking because the modified email that was sent from a browser won’t be shown in our application.

Pros:

* This allows for every email to be encrypted according to its own randomly generated keys, which means that even identical emails will be have vastly different encrypted forms.
* Only a portion of the key needs to be sent along with the email
* If a malicious actor brute forced their way to (K1,V1), this would only allow them to unlock the first 24 characters keeping the message encrypted unless the actor was able to successfully determine those first 24 characters can be expanded into (K2,V2)
* A man in the middle attack will usually be filtered out so that you can’t try to open it, in the case that you can open it, the application will check whether or not its hash is a valid match

Cons:

* The sender and receiver both need a copy of this program
* The casting of bytes to strings limits the amount of possibilities for (K2,V2) to

**Citations**

“JavaMail.” *Javaserverfaces-Spec*, Oracle, javaee.github.io/javamail/. *Accessed 4 Oct 2018.*

Chumbley, Alex. “Symmetric Ciphers.” *Brilliant Math & Science Wiki*, Brilliant,

brilliant.org/wiki/symmetric-ciphers/. *Accessed 4 Oct 2018.*

Ray, Dhiraj. “AES Encryption and Decryption in Java(CBC Mode).” *Java Code Geeks*, Java Code Geeks, 12 Mar. 2018, [www.javacodegeeks.com/2018/03/aes-encryption-and-decryption-in-](http://www.javacodegeeks.com/2018/03/aes-encryption-and-decryption-in-) javacbc-mode.html. *Accessed 19 Nov 2018.*

“Example of Sending Email in Java through Gmail Server - Javatpoint.” *Www.javatpoint.com*, Java T Point, [www.javatpoint.com/example-of-sending-email-using-java-mail-api-through-gmail-](http://www.javatpoint.com/example-of-sending-email-using-java-mail-api-through-gmail-) server. *Accessed 7 Nov 2018.*

*“*City University Di New York State University Di New York College a Cortland, Università Di

Albany SUNY SUNY Politecnico e Università Di Stato Di New York Di Sistema - Studiare

All'estero - Unlimited Download. Kisspng.com.” *Kisspng.com*, it.kisspng.com/kisspng-1ypfnh/. *Accessed 7 Nov 2018*