SendSecure: Encrypting Email with 128-Bit AES

Emily Venuto
Department of Computer Science, Marist College, Poughkeepsie, New York, USA

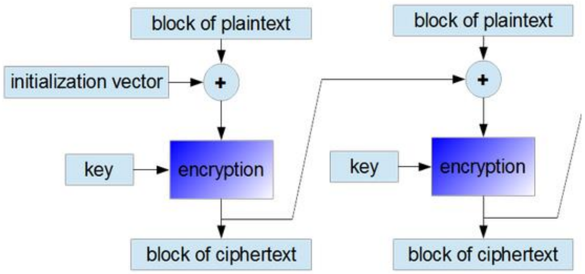
Abstract – *In this paper, I review the AES algorithm and its use in the SendSecure application, which is a Gmail plugin that allows encryption and decryption of email text using AES and DES, respectively.*

1. Introduction

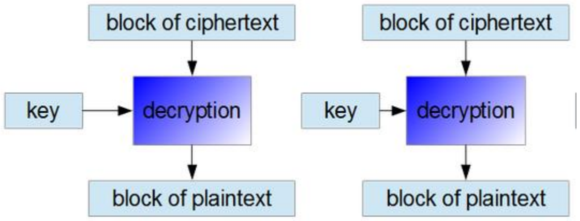
When data is sent from one place to another over an Internet connection, the biggest concern is if it is being done securely. I thought it would be interesting to explore sending an encrypted email using 128-bit AES where the text is transformed directly in the browser.

Advanced Encryption Standard (AES) is an algorithm that utilizes a single key to encrypt text in rounds. AES "rounds" refer to processing the input plaintext in blocks. The length of the block that is handled in a round is determined by the length of the key being used. For example, if a 128-bit key is used, blocks of text will be processed 128 bits at a time [1].

There are several modes under which AES can operate. The application discussed in this paper operates under cipher block chaining (CBC). With CBC, each block of plaintext is XOR-ed with the previously produced ciphertext, and then used as the next input into the AES algorithm [2]. This process is illustrated as follows:



The decryption process (DES) uses the single key to invert the encryption process and yield the plaintext from the ciphertext.



2. Methodology

The proposed encryption plugin has a simple workflow. The sender writes his email in the Gmail window. He enables the SendSecure plugin, which asks him to select if he wants to encrypt or decrypt. When the sender selects "encrypt," a randomly generated 128-bit key is displayed to the sender. The transmission of the key from sender to receiver is discussed later in this section. When the sender confirms the key, the AES algorithm runs, and the text written in the Gmail window is replaced with the encrypted text. The user then sends his email to his intended target. Note that in the encryption process, if the text does not fit directly into 128-bit blocks, filler text should be used. This is a feature that has not yet been implemented.

When a user receives his encrypted email, he will follow a similar workflow. He enables SendSecure and selects the encrypted email in his inbox. The plugin asks if he wants to encrypt or decrypt. When the receiver selects “decrypt,” he is prompted to input his secret key. Once he does so, the DES algorithm runs, and the decrypted text is shown to the viewer.

Since AES relies on a single key, it is important that the key is robust and that the method used to store or transmit it is secure. For the key itself to be secure, it should be randomly generated on every separate use of the application. The method by which this application will keep the key transfer secure is still being determined. The simplest option is for the sender to transmit the key to the receiver by text message, or a similar process that is separate from the email. Some possibilities to improve the key transmission process involve using encryption by way of public and private keys, such as in RSA, PGP, or Diffie-Hellman.

3. Conclusion

In this paper I provided an overview of AES and how it is being implemented into the SendSecure application. I discussed how the application is used from beginning to end, as well as features that are still in development.

References

[1] Townsend Security. “AES Encryption.” https://townsendsecurity.com/sites/default/files/AES_Introduction.pdf

[2] Yi-Li Huang, Fang-Yie Leu, Jung-Chun Liu, Jing-Hao Yang, Chih-Wei Yu, Cheng-Chung Chu, Chao-Tung Yang, “Building a block cipher mode of operation with feedback keys,” IEEE, Taipei, Taiwan, 28-31 May 2013, [Online]. https://ieeexplore.ieee.org/document/6563875/citations#citations *https://ieeexplore.ieee.org/document/6563875/authors#authors*