SendSecure: Encrypting and Decrypting Email Messages with 128-Bit AES

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

Abstract – *In this paper, I review the AES encryption and decryption in application to SendSecure, a desktop application which allows one to encrypt or decrypt text to be sent in an email.*

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 encrypting and decrypting email text, to add a layer of security to emailing.

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:

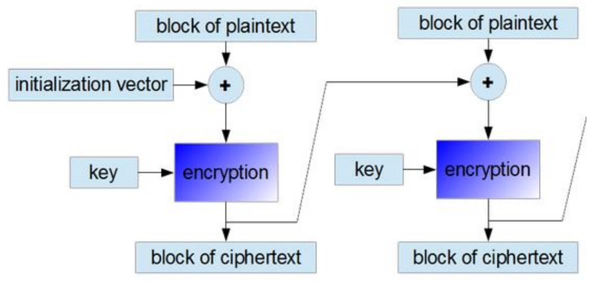


Figure 1: CBC Encryption

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

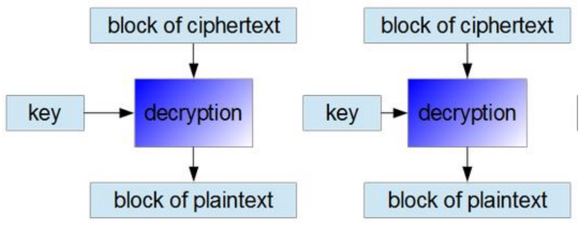


Figure 2: CBC Decryption

2. Methodology

SendSecure is a desktop application to be used by people who want to encrypt email messages they are going to send and decrypt ciphertext email messages they have received. Figure 3 shows the layout of the SendSecure. To encrypt plaintext, a user enters his message into the text box on the left, enters a 32-digit hex key, and presses Encrypt. The ciphertext for this plaintext-key combination will be output in the right text box. If one wishes to decrypt a message, he follows the same procedure: putting the encrypted message into the left text box, inputting the corresponding key, and pressing the Decrypt button. The plaintext will be output in the right textbox.

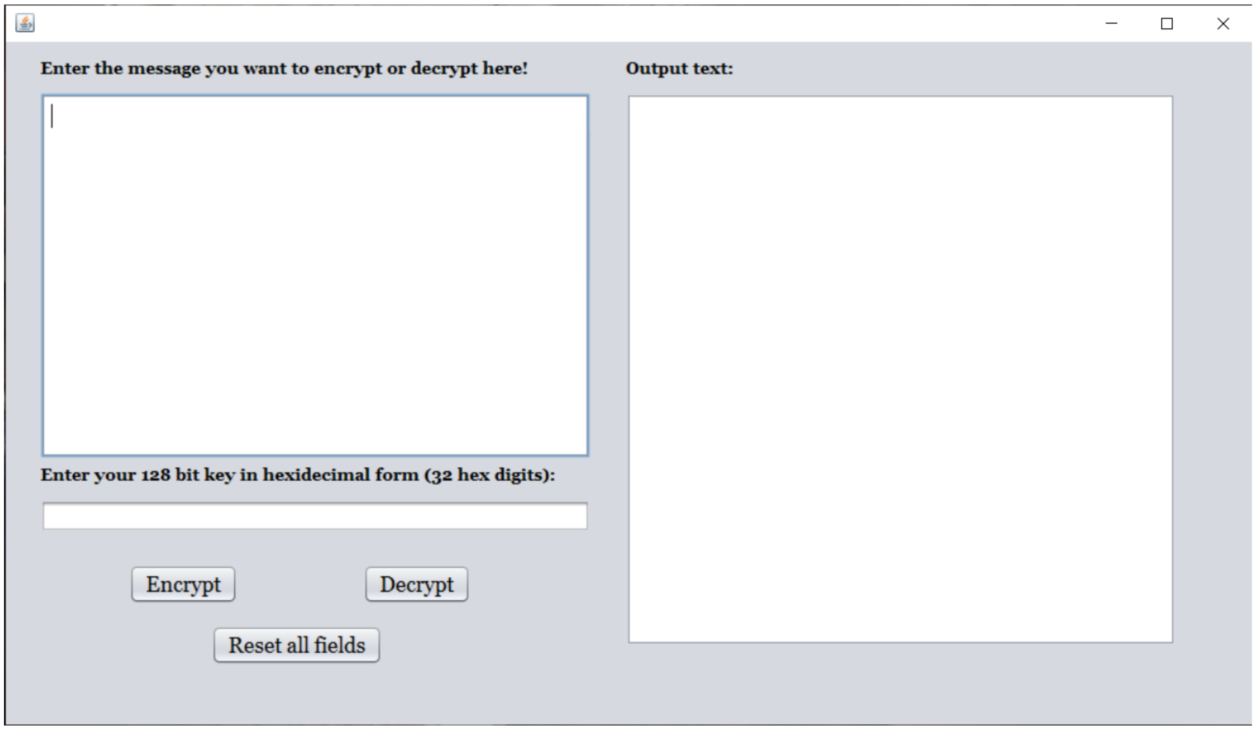

Enter the message you want to encrypt or decrypt here!
Output text:
Enter your 128 bit key in hexidecimal form (32 hex digits):
Encrypt
Decrypt
Reset all fields

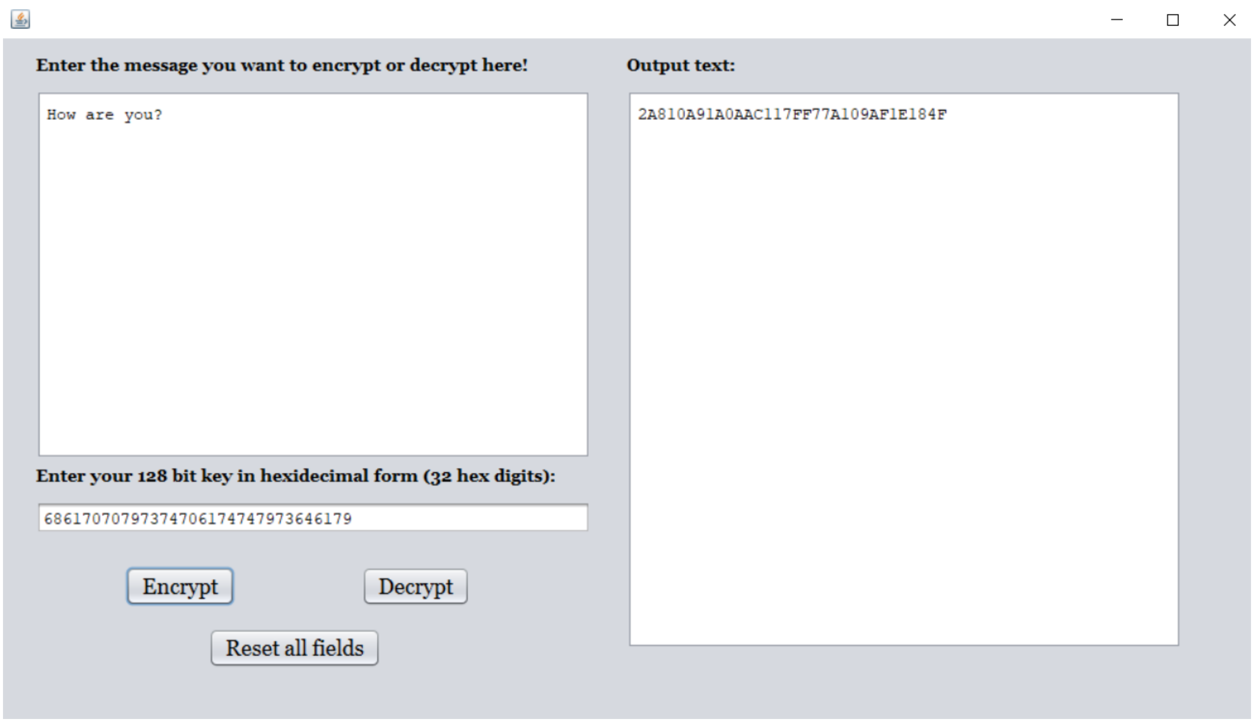
Figure 3: Blank SendSecure application

Encryption begins by taking in a plaintext message and a 32-digit hex key string. The key is used to produce the 11 round keys utilized in CBC. The entire plaintext message is taken in as one string. It is first altered by appending a padding character until the string is a multiple of length 16. This way, it can be broken into 128-bit blocks later. If the plaintext string is less than 16 ASCII characters, padding will be applied until it is 16 characters in length. If the string is already a multiple of 16, an additional 16 characters of padding will be appended. If the string is greater than 16 in length and is not a multiple of 16, it will be padded until it becomes a multiple of 16. The specified padding character for this application is “~”. This character was chosen because it is unlikely that an email message will end in this character. After the plaintext string has been padded, it is converted to its hexadecimal equivalent and broken down into 128-bit blocks. By way of a for loop, each block is encrypted using the AES algorithm, as detailed in Figure 1. Once a block is encrypted, its ciphertext is appended to a final output string. The final output string is displayed in the right textbox of SendSecure when encryption is complete.

Decryption works in a similar way, inverting the process performed on the plaintext. The same key used in encryption is input as the decryption key and produces the 11 round keys needed. These keys are the same as those produced through encryption. The rounds are implemented in the reverse order that they were used in encryption; encryption will begin with round 0, while decryption will begin with round 10. The ciphertext is taken from the input window and turned into 128-bit blocks, and then put through the decryption process, as detailed in Figure 2. Once the fully decrypted hex string is produced, it is converted back to ASCII and the padding is removed. Padding is removed by traversing the string backwards and deleting the specified padding character until it can no longer by found at the end of the string. This final string is output in the right textbox of SendSecure.

3. Experiments

SendSecure can successfully encrypt and decrypt a message of any length, while upholding the structural integrity of the text it is processing. Figure 4 and 5 demonstrate the encryption and decryption, respectively, of a message that is less than 16 characters.


Enter the message you want to encrypt or decrypt here!

```
How are you?
```

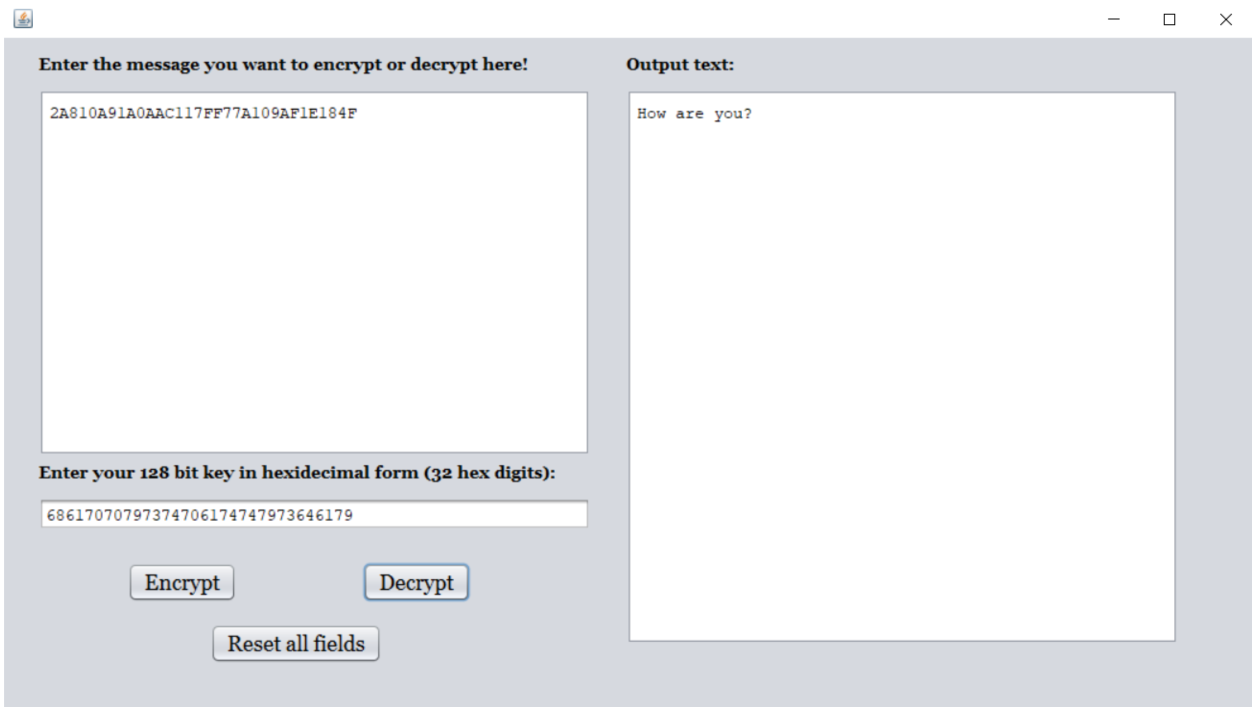
Enter your 128 bit key in hexidecimal form (32 hex digits):

```
68617070797374706174747973646179
```

Output text:

```
2A810A91A0AAC117FF77A109AF1E184F
```

Figure 4: Encryption of string less than 16 characters


Enter the message you want to encrypt or decrypt here!

```
2A810A91A0AAC117FF77A109AF1E184F
```

Enter your 128 bit key in hexidecimal form (32 hex digits):

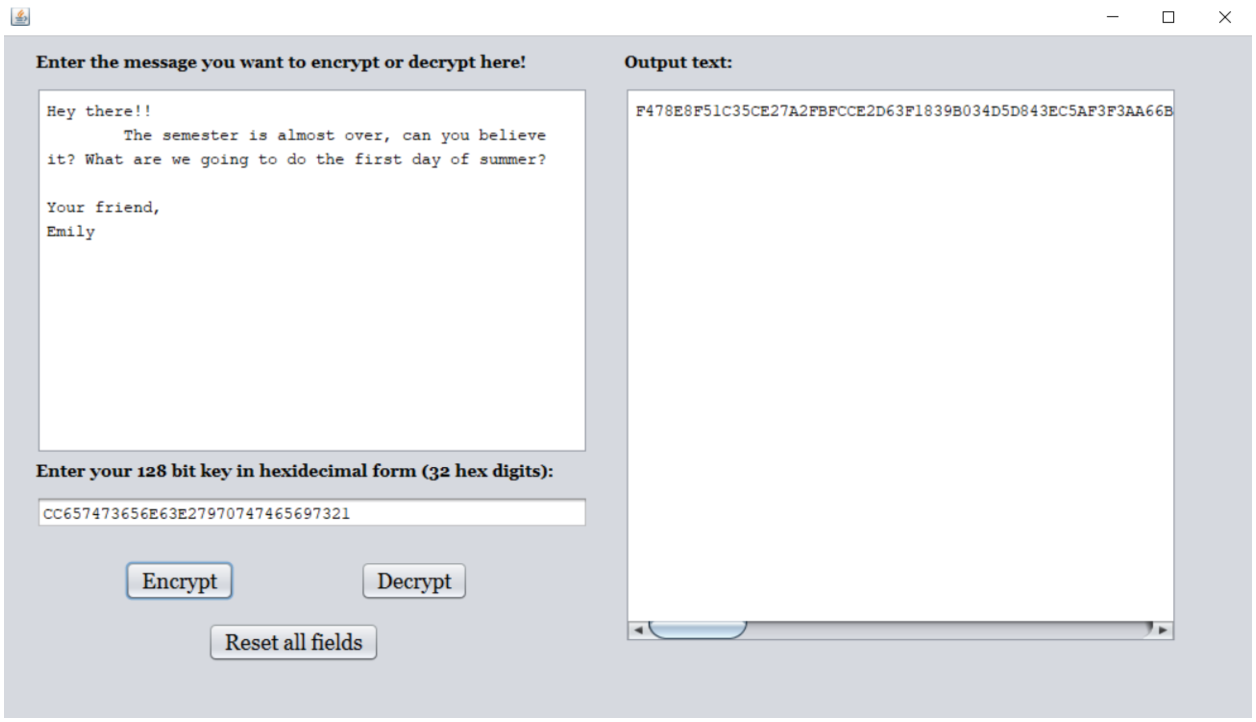
```
68617070797374706174747973646179
```

Output text:

```
How are you?
```

Figure 5: Decryption of string less than 16 characters

It is important that SendSecure upholds the structural integrity of any message it is fed. Therefore, if tabs, blank spaces, or symbols are used in the plaintext message, they will be maintained through the encryption and decryption process. This was implemented by ensuring that every ASCII character was mapped to its corresponding two-digit hex equivalent upon encryption, and vice versa during decryption. It was also important that the decryption process was completely autonomous in relation to the encryption process. No information, such as the round keys or expanded keys were to be shared between to these processes. It was designed this way so that two users could have this application on their separate desktops and encrypt and decrypt messages sent between one another over email without relying on any previously saved information. Figure 6 and 7 demonstrate the encryption and decryption, respectively, of a message that is greater than 16 characters, with newline characters and tabs.


Enter the message you want to encrypt or decrypt here!
Output text:
Enter your 128 bit key in hexidecimal form (32 hex digits):
Encrypt
Decrypt
Reset all fields

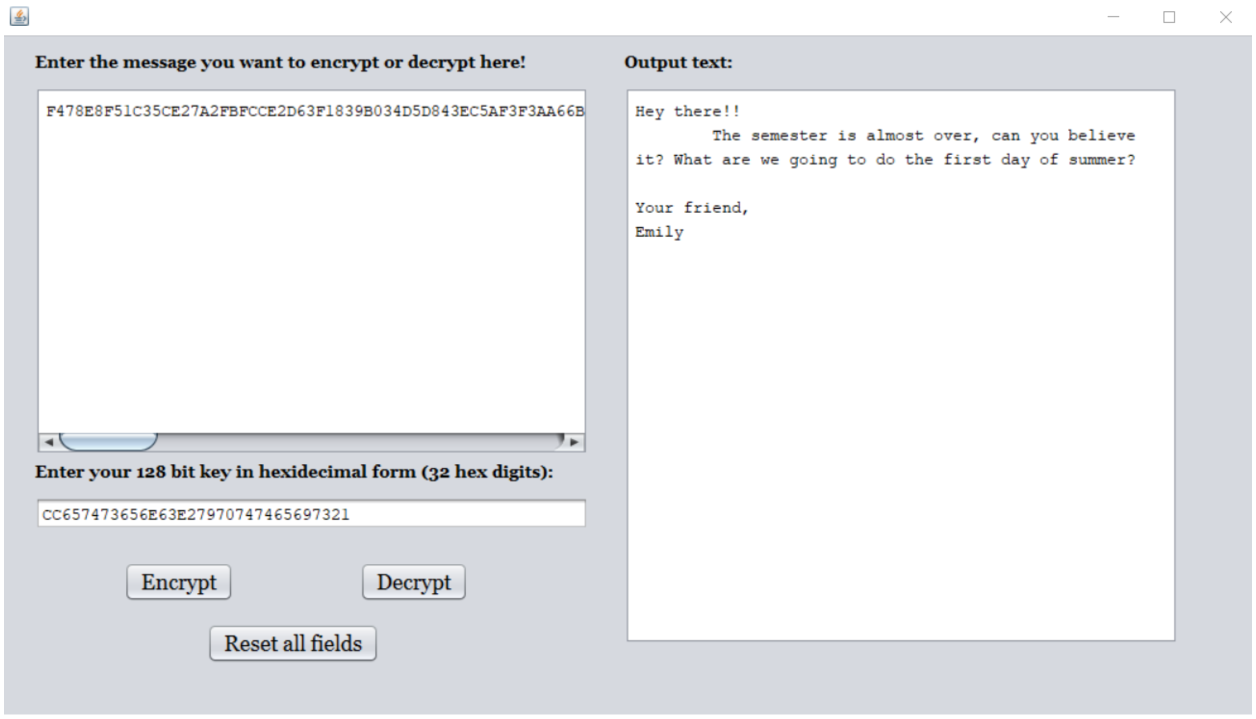
```
Hey there!!
	The semester is almost over, can you believe
it? What are we going to do the first day of summer?

Your friend,
Emily
```

```
CC657473656E63E27970747465697321
```

```
F478E8F51C35CE27A2FBFCCE2D63F1839B034D5D843EC5AF3F3AA66B
```

Figure 6: Encryption of string greater than 16 characters, with newline characters and tabs.


Enter the message you want to encrypt or decrypt here!
Output text:
Enter your 128 bit key in hexidecimal form (32 hex digits):
Encrypt
Decrypt
Reset all fields

```
F478E8F51C35CE27A2FBFCCE2D63F1839B034D5D843EC5AF3F3AA66B
```

```
CC657473656E63E27970747465697321
```

```
Hey there!!
	The semester is almost over, can you believe
it? What are we going to do the first day of summer?

Your friend,
Emily
```

Figure 7: Decryption of string greater than 16 characters, with newline characters and tabs.

4. Improvements

One improvement that could be brought to this application would be implementing it as an email plugin. This would be helpful because a user could transform his or her text directly in the email application, instead of having to cut and paste their text in and out of SendSecure. A second improvement that could be implemented would be providing a secure method by which the key is passed from user to user. Currently, a user must transmit the key via 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. This transmission could then be done in conjunction with the email sending.

5. Conclusion

In this paper I provided an overview of AES and how it is being implemented to encrypt and decrypt messages in SendSecure. 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