Batch Encryption with AES in CBC Mode

Samantha Berry

Marist College, 3399 North Road Poughkeepsie, NY 12601, USA
`{samantha.berry1}@marist.edu`

Abstract. The goal of this paper is to document the process behind my final project. The background section outlines the research done on AES encryption and data at rest. The methodology section outlines my experience implementing my AES encryption and decryption, with the following testing section illustrating the process by which I ensured all code elements worked correctly. Finally, the conclusions section gives a summary of the complete project and my final thoughts.

1 Introduction

For this project I decided to make a software that preforms encryption and decryption using AES in CBC mode. The software is intended to encrypt and decrypt files in large batches, allowing the user to perform operations on groups of files in a directory instead of having to select plaintext sources individually. I built this project of off my experience in developing AES encryption with a 128-bit key from the labs. For this project I went on to further my understanding of AES by researching and implementing AES decryption and AES CBC mode. During my research I gained knowledge of the practical uses of encryption and how different algorithms can be best suited for certain usages. I became aware that the attributes of AES make it a good fit for my chosen implementation.

2 Background

The AES (Advanced Encryption Standard) was established in 2001 by the U.S. National Institute of standards and Technology (NIST) [1]. It was described by Daemen and Rijmen in their 1999 submission to the NIST in response to their search for a secure cypher for government data. In the modern world, AES is commonly used for encryption by both government and private groups. AES is a block cypher algorithm, meaning it takes in blocks of plaintext of a fix length. Each block of data in AES is 128-bit, and the key can be 128-bit, 192-bit, or 256-bit, which each increase in key size adding additional security. Another attribute of AES is that it is a symmetric key algorithm, meaning that decrypting a cyphertext requires the identical key as the one use to encrypt the original plaintext.

The exact mode of AES used in my project is Cipher Block Chaining (CBC). The principles for CBC implementation in computer ciphers was invented in 1976 and described in a patent by Ehrsam, Meyer, Smith, and Tuchman [2]. In CBC mode each subsequent block of plaintext is XORed to the ciphertext from the last round, and since there is no previous input for the first round, it is XORed to a randomly generated input vector.

When describing data that needs to be encrypted, it is common to describe the state of the data. Two common descriptors of data are data in transit, and data at rest, and both have different factors that determine what encryption approach is most effective. My project most closely resembles an encryption tool for data at rest, taking in large amounts of data to be transformed and then archiving the data in its new state.

The properties of AES make is much more suited for encrypting data at rest when compared to data in transit. AES is not preferred for encrypting data in transit due to the fact it is a symmetric key algorithm. Both the sender and receiver need the same key in order to decrypt the data at the other end of transit. However, this brings up the issue of how to send that key securely. Data in transit most commonly uses asymmetrical key algorithms, which have public keys that can be safely sent without revealing the data. AES, as a symmetric key algorithm, is better put to use in a system where keys are less a risk, making it a good choice for encrypting data at rest, as well as my project.

3 Methodology

For this project I created a program that encrypts and decrypts files using AES in CBC mode. My first step was to implement AES decryption, since I had already implemented encryption in the labs. After testing that the decryption worked correctly, my next addition was a way to create an array of inputs to be fed into the CBC mode I would add later. To this end I added a method that converts standard plain text to an array 128-bit hex strings and another method to transform them back. This also accomplished my project objectives of implementing a method of to transform text into a format readable by AES and a method to return the output of AES decryption to a human-readable format. At this point I added a method to randomly generate hex strings for use as input vectors for CBC. In the final version this method is also used to generate random encryption keys.

I moved on to adding CBC mode encryption and decryption. One quirk of CBC mode is that, since the first block of plaintext is XORed to a random input vector, it cannot be transformed back to readable plaintext by decryption. In order to mediate this issue, I would fill the first index of my generated hex array with a randomized 128-bit string and then remove these first 128-bits after decryption. This ensured that none of the critical data from the file would be lost.

The complete project includes a command line user interface for input. The user can select to perform either encryption or decryption and choose how to input the encryption key. The user can enter the key in the command line, have it read from the file key.txt in the key directory, or, for encryption only, have it randomly generated. The random key is written to the key.txt for later use. The option is not available for decryption since this operation requires a known key. In any case, the input key is checked to make sure to matches the format and length required for AES key. Once a key is obtained, then the appropriate operation is preformed, reading files from the plaintext directory is encryption is performed and reading files from the cyphertext directory if decryption is performed.

4 Testing

Testing the AES encryption and decryption was done using Junit. For AES encryption I fed in randomly generated 128-bit strings of hex for the key and plaintext and compared the output to a known correct cyphertext. Testing AES decryption is done using the same set of data, but the cyphertext is fed in and it is checked to the plaintext. As for the data set for these tests, some of it I generated myself using the random.org hex generator, and some I gathered from my classmates' tests for lab 5.

Testing of String to Hex conversion is also done using Junit. Known plaintext strings were converted to hex, converted back to plaintext and were compared to the original string. I also constructed tests for using AES and my hex methods together. I would take a string, change it to hex, encrypt it, decrypt it, change it back to a string, and then ensure that it matched the original string.

Testing of file reading and file writing was preformed manually. Simply, I compared program logs to the files' contents in order to ensure all reads were obtaining correct data and all writes were inserting the correct data.

5 Conclusions

Through the experience of working on this project I expanded my knowledge of encryption and its many uses. Despite the logical complexity, the AES algorithm was the simplest part to complete. Since AES is a common encryption algorithm, I had many resources to reference to learn its construction, all that I had to do was research and implement it. The difficultly came when figuring out how to transform file input into something the algorithm could use.

The string to hex method I came up with is very simple, since it is only able to read standard English characters. If I were given more time, I would have created a more robust algorithm capable of interpreting other character sets and possibly also other file types.

I chose to make a batch encryption software because I wanted to expand my abilities with AES and because I was interested in various uses for encryption. I was already were of the used of encryption concerning data in transit due to taking an internet security class. Through my research I gained knowledge on how different uses of encryption and how the properties of encryption algorithms effect what uses they are best suited for.

References

1. Daemen, J., Rijmen, V.: AES Proposal: Rijdael. National Institute of Standards and Technology (1999).
2. Ehrsam, W., Meyer, C., Smith, J., Tuchman, W.: Message Verification and Transmission Error Detection by Block Chaining. US Patent 4074066 (1976).