Software Quality Engineering

- DT510A -

Quality of Symmetric Encryption

Marcus Östmo

October 2021

Contents

1 Introduction

This report will focus on a overview of encryption, with a focus of how encryption is important in a perspective of software quality and why it might be more important than ever. However, first a brief introduction of the subject cryptography is in order to understand where these techniques emerge from and why they are so important.

The necessity for keeping information and messages secret has existed throughout time. One of the earliest findings of techniques to obscure messages where found in ancient Egypt as well as in Sparta [1], where one of the earliest variants of cryptography has been found, such as the Ceasar cipher[2].

Moving on throughout history, in the beginning of the 20Th century as the second world war hit Europe, the need for encrypted messages were greater than ever. During this time machines purpose built for encrypting and decrypting messages were developed with the purpose of sending secret messages. Perhaps the most famous example is the *Enigma Machine* which were used by Nazi Germany in the second world war. Although these machines did not use software in order to encrypt the messages, the *quality* of the cryptography were of great importance in order to guarantee that the message only were interpretative for the intended reader.

As the world came to use computers, communicating over the internet and also wanting to keep local information secure, the need for encryption of data raised. Nowadays, we take for granted that every day internet based tasks, like saving family photos online or performing banking errands, are carried out in a secure manner. If the data would not be encrypted, all of the information would be sent in plain text over the public internet. Therefore, security in form of encryption is of great importance.

This report will describe how one of the most used encryption algorithms used today works, as well as analyze it in the purpose of determine whether or not it is a good option regarding data encryption. It will also be discussed how these techniques will withstand the future.

2 Background

The Ceasar cipher and the Enigma machine mentioned in the introduction works in a similar way, although Enigma off course is much more advanced. The main point is however the same, to encrypt a message. Encryption of data is a term used for techniques of obscuring data or information. Decryption is the reversed process, to receive the data from the encrypted message.

Both these methods function by substituting the individual characters of a plain text message to another. For example the Ceasar cipher shifts each letter either left or right some number of positions of the alphabet[2]. The Enigma machine does the same thing but in a much more sophisticated way. Although both these methods have been superseded for a long time, it will be shown in this report that some of the fundamental ideas of substituting data is still being used in the advanced algorithms used today.

2.1 Key concepts

For the subject of computer encryption, there are some important terms and concepts that is reoccurring in all approaches. This section will provide a brief explanation of these terms.

The first term is **plain text** which is the unencrypted data that need so to be encrypted in order for it to not be interpretable for anyone that are not authorized to access the data.

The **encryption algorithm** is the algorithm used to obscure the data. The number of bits that are used as input to the encryption algorithm is known as the **block size**.

The **secret key** is used as input to the algorithm, and based upon the key, the plain text is transferred into **cipher text**. The number of bits that the key consists of in known as the **key size**

An **XOR gate** is also something commonly used in cryptography. XOR stands for exclusive OR, and is a bit-wise operation that uses two inputs, compares them using the exclusive OR-rule. That is, if either of the inputs are 1, the output is 1, the exclusive part means that if both inputs are 1, the output is 0 as demonstrated in table 1.

A	B	Output
0	0	0
1	0	1
0	1	1
1	1	0

Table 1: XOR truth table.

A **brute force** attack is were an attacker exhaustively tries to break a system by trying all different combinations. This is the least insightful way of

performing an attack, but is sometimes a successful option if the number of possible combinations to test is feasible.

2.2 Symmetric encryption

Symmetric encryption is the most common form of encryption and historically it has been the only technique. The way that symmetric encryption works, and why it is known as symmetric, is that the encryption algorithm and the decryption algorithm essentially is the same. The difference between the two is that the decryption algorithm is the reverse of the encryption[3]. This is shown in figure 1.

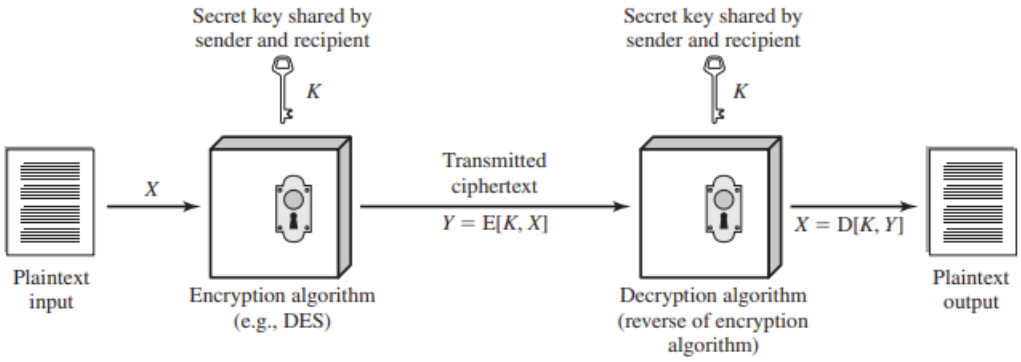

Figure 1: The flow of a symmetric encryption algorithm from Stallings[3].

2.3 DES

For a long time the standard encryption algorithm was the *DES* algorithm, which stands for Data Encryption Algorithm. DES was developed in the 1970s by IBM and was the method used from the 1970s through the end of the 1990s. The problem with DES, and the reason why it needed a replacement was that the key size used was 56 bits. This might have been plenty in 1970 but was simply not enough as computers became more powerful. This made the encryption too weak, as brute force attacks was able to break the encryption [4].

In order to make DES more robust, a variant known as Triple DES were introduced. This is, as the name suggests, DES ran three times, making it much more resistant to brute force attacks. However, the algorithm still has some theoretical weaknesses[5].

2.4 AES

Because of the problems with DES, a new standard where needed in the end of the 1990s. A competition where held by NIST, the United States National

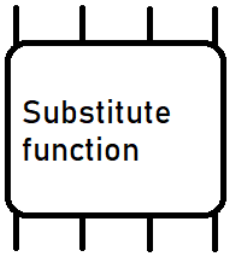

Figure 2: Example of four bit substitution box.

Institute of Standards and Technology, were cryptography researchers could submit their solutions for the new standard, known as AES (Advanced Encryption Standard). The winning algorithm is known as Rijndael, which were submitted by the dutch scientist Vincent Rijmen and Joan Daemen in 1999 [6]. Rijndael, or AES as it is more commonly referred as, is like DES a symmetric algorithm, although it uses a larger **block size** to strengthen the encryption and greatly reduce the chances of the encryption being broken by brute force attacks. This is a fixed size, and the larger the block is, the harder it becomes to break the encryption using brute force. This was one of the reasons that DES was no longer considered safe[4], as it used a block size of 64 bits, whereas AES uses a block size of 128 bits.

2.5 SP network

The AES algorithm uses a SP-network in order to encrypt data. A SP-network consists of two parts, *substitution* and *permutation*, hence the name SP. A SP-network works by first substituting the data and then perform permutations of said data. Substitutions builds upon the same principles as used by the Enigma machine discussed earlier. The difference here however, is that Enigma substitutes each character individually. This is known as one-to-one mapping. One character, in the Enigma case, is encrypted into another character. In the substitution made by AES however, the whole block size of 128 bits is encrypted[6]. Because of this there is no one-to-one mapping, making the encryption much safer, since one can not focus on a single set of characters when trying to break the encryption.

The substitution works as following; a block of x bits is sent in to the substitution function as input. The function then, using some internal rules *substitutes* the bits, and x bits are then presented as output. An example is found in figure 2, were a block of four bits are substituted. A basic example would be that all of the 16 different combinations, 2^4, would be statically mapped to another number, e.g $0 \rightarrow 5, 1 \rightarrow 7, 7 \rightarrow 15$ and so on.

It is important that this mapping is made with caution regarding the numbers, if say 5 would map to 7, then having 7 map back to 5 yields in a weaker

cryptography as opposed to having 7 mapped to a different number [6]. Going back to the example in figure 2, if encrypting the number 7, which is 0111 in binary, these bits are what would be sent as input to the substitution function. This number is then received via the look up table and for this example 1111, 15 in decimal, would be the output. This process can then be repeated several times for increased obscurity, but this alone will not be sufficient. More is needed, leading to the next part of the SP-network, the permutation.

Permutation essential move things around as it changes place of the bits provided as input. As with substitution, a block of X bits are provided, and X bits are then presented as output. In the output however, the bits will have changed place with each other. Therefore, for a block consisting of 8 bits, the first bit might have been switched to the eighth place, the fifth bit might be in the second position and so on. This is demonstrated in figure 3[7].

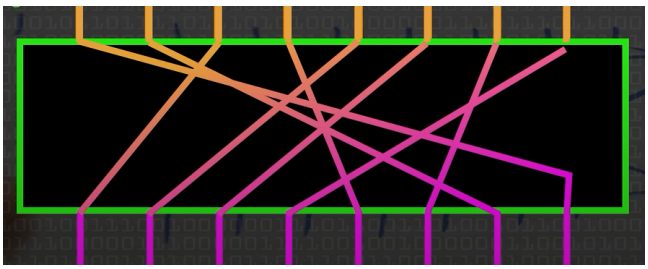

Figure 3: Example of a permutation within a SP network.

To create the SP network, the aforementioned components are then connected together as shown in figure 4. Each run through this SP boxes is known

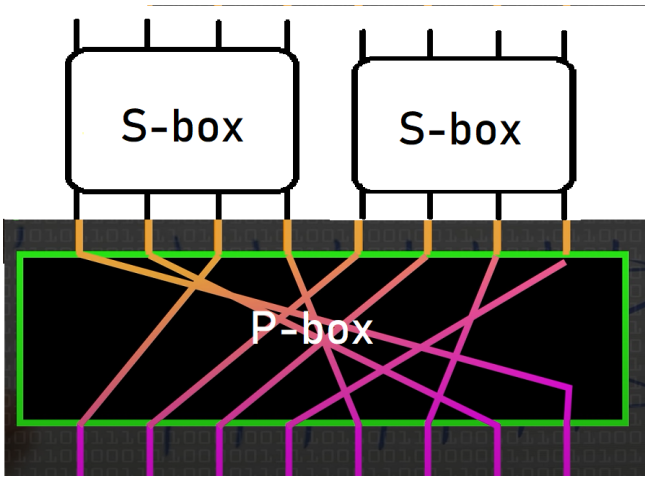

Figure 4: SP network.

as a **round**[6]. Multiple rounds are performed to further obscure the message.

This is however not enough, as these steps alone would be able to be back tracked as just running everything in reverse would give the plain text back. This is were the key has its purpose.

The key is separated into multiple chunks, known as round keys, that are then mixed with the data using XOR. The result of the XOR is then inserted as input into the next round, as seen in figure 5.

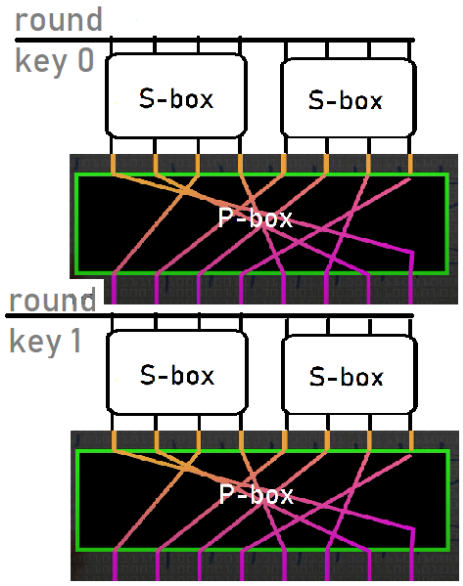

Figure 5: Illustration of a SP network with inserted keys.

This makes it much more difficult to reverse the encryption. Without the key it is not possible to reverse, even if both the S-box and P-box are known. The number of rounds performed depends on how many that are deemed necessary for it to not be breakable. This is a trade-off between security and performance, as too few rounds leads to weak security, whereas too many rounds will lead to the encryption being too slow[6].

3 Technical Details

3.1 AES in detail

With the basic concepts of AES explained in section 2, a closer look at AES can be done. This in order to analyze how it manages to withstand attacks and still be the standard encryption algorithm, even more than 20 years after its initial release.

AES, also known as the Rijndael algorithm, uses a block size of 128 bits, and the size of the secret key can either be 128, 192 or 256 bits long. This means that AES encrypts a block of 16 bytes (128 bits). These block are sent through a variant of the SP network that is described in section 2.4. However, instead of having the block organized in a long line of bits, AES organizes the block as a matrix of column major order. The matrix is a 4x4 matrix, with 16 cells, one for each byte.

$$\begin{bmatrix} b0 & b4 & b8 & b12 \\ b1 & b5 & b9 & b13 \\ b2 & b6 & b10 & b14 \\ b3 & b7 & b11 & b15 \end{bmatrix}$$

All of the operation, or steps, in the encryption are then performed upon this matrix. These steps are quite similar to the simple SP network discussed earlier with some key differences. The concept of doing rounds are the same although here the number of rounds is decided by the size of the key being used. If using a 128 bit key, 10 rounds are performed. When using a key size of 192 bits, the number of rounds is 12, and finally if using a 256 bit key, 14 rounds is performed. So, how does a round of AES look like?

The first step of the algorithm is to perform bitwise XOR with a piece of the key, known as a round key [6]. The outcome of this XOR is then used as input to the SP network.

The first step here is substitution, where the bytes are replaced. This is done as in the simple example in figure 2, with a lookup table. However, this is a non-linear substitution and not just a simple mapping, for example 5→7. This non-linear substitution is made possible by using a *finite field*[8].

A finite field consists of some number of elements on which a set of operations can be performed. These operations are addition, multiplication, division and subtraction. One interesting property with a finite field is that the operations can be applied to elements within the field, and one can be certain that the outcome also will lie within the field. In Rijndael, a finite field of 2^8 elements is used, leading to each of the elements being one byte, going from 0000 0000 to 1111 1111. The non-linearity of this substitution is made by taking the multiplicative inverse of the field[6].

The next step in the round is the permutation, which is performed in multiple steps. Firstly, the rows are shifted. The first row is unchanged, the second row is shifted by one to the left, the third row is shifted by two to the left. Finally

the fourth row is shifted by three to the left, which gives the following matrix:

$$\begin{bmatrix} b0 & b4 & b8 & b12 \\ b5 & b9 & b13 & b1 \\ b10 & b14 & b2 & b6 \\ b15 & b3 & b7 & b11 \end{bmatrix}$$

With the rows shifted, the second step in the permutation is to mix the columns. Mixing the columns is done via matrix multiplication, where each of the columns of the matrix is multiplied with a matrix that is pre-determined in the algorithm:

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} b0 \\ b5 \\ b10 \\ b5 \end{bmatrix}$$

This matrix is selected in a way that the numbers are big enough to make a sufficient change to the input, but small enough to still provide good performance. Each of the columns will be multiplied with this matrix from the right, which yields a new column vector that is the same size as the input.

Once this step is complete, a round has been performed. This will then be repeated 10, 12 or 14 times depending on the size of the key.

3.2 Security

There have been some attacks towards AES that have been somewhat successful, such as side channel attacks, including timing and cache attacks[9] [10]. These attacks does not attack the algorithm itself however. Instead it targets the implementation which sometimes in certain applications have been lacking. This makes it possible for an attacker to be able to receive fractions of, or the whole, key. In later years this has become less of a problem as almost all modern processors have an implementation of AES built-in to hardware[11][12][13]. This protects the encrypted information against these kind of implementation based attacks. These hardware implementations are also *very* fast on modern computer architectures such as Intel Core and AMD Ryzen, with speeds of multiple gigabytes per second[14].

As of now, there are no known attacks that can break AES and receive the plain text from a encrypted message. In the year 2000 Bruce Schneier, an author of an competing algorithm in the competition for the AES algorithm, wrote:[15]

> "I do not believe that anyone will ever discover an attack that will allow someone to read Rijndael traffic."

This is still true at this day. Brute forcing AES is an impossible task with the computational power of today, even if one would use the worlds fastest super computer of 2021, the Fugaku[16]. The Fugaku is capable of 442 Petaflops/s but it would require:

442×10^{15} Flops (floating point operations per second)
Assume 1000 Flops required per combination
The number of combinations check per second is then:
$442 \times 10^{15}/1000 = 442 \times 10^{12}$
The number of seconds in a year is:
$365 \times 24 \times 60 \times 60 = 31536000$
The number of combinations of AES with a 128-bit key is:
$2^{128} \approx 3.4 * 10^{38}$
$\hookrightarrow \frac{3.4*10^{38}}{442 \times 10^{12} \times 31536000}$
$\implies \frac{3.4*10^{26}}{13938912000}$
$\implies 2.4 \times 10^{16}$

Dividing by two, since on average that is where a brute force attack would find the answer, still gives 10^{16} years, which is ten million billion years to break the encryption.

4 Conclusion

To conclude, it has been shown that although the AES algorithm, also known as Rijndael, was proposed 22 years ago, the encryption it provides is still more than strong enough to endure brute force attacks. This also applies to the very powerful computers, as apposed to 20 years ago, that we posses today. Some more sophisticated attacks have been able to make some progress faster than brute force[10], although these attacks have focused on certain implementations and not the algorithm itself.

Several reports can be found on different approaches evaluating different aspects of Rijndeal. However, the fact is that so far there exists no attack that would enable an attacker to get access to the plain text encrypted by AES without knowing the key. As seen above, retrieving the data from brute force, even if it was encrypted using the smallest key of 128 bits, would take more time than the age of the earth.

References

[1] Cypher Research Laboratories. A brief history of cryptography, 2006.

[2] Dennis Luciano and Gordon Prichett. Cryptology: From caesar ciphers to public-key cryptosystems. *The College Mathematics Journal*, 18(1):2–17, 1987.

[3] W. Stallings and L. Brown. *Computer Security: Principles and Practice*, pages 284, 311. Pearson Education, Incorporated, 2018.

[4] W. Diffie and M.E. Hellman. Special feature exhaustive cryptanalysis of the nbs data encryption standard. *Computer*, 10(6):74–84, 1977.

[5] Bernardo Bátiz-Lazo. *Cash and Dash: How ATMs and Computers Changed Banking*. Oxford University Press, 2018.

[6] Joan Daemen and Vincent Rijmen. Aes proposal: Rijndael. 1999.

[7] Michael Pound. Sp networks, computerphile. `https://www.youtube.com/watch?v=DLjzI5dX8jc`. Accessed: 2021-10-28.

[8] Christoforus Juan Benvenuto. Galois field in cryptography. *University of Washington*, 1(1):1–11, 2012.

[9] Gorka Irazoqui, Mehmet Sinan Inci, Thomas Eisenbarth, and Berk Sunar. Wait a minute! a fast, cross-vm attack on aes. In Angelos Stavrou, Herbert Bos, and Georgios Portokalidis, editors, *Research in Attacks, Intrusions and Defenses*, pages 299–319, Cham, 2014. Springer International Publishing.

[10] C. Ashokkumar, Ravi Prakash Giri, and Bernard Menezes. Highly efficient algorithms for aes key retrieval in cache access attacks. In *2016 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 261–275, 2016.

[11] Shay Gueron. Intel advanced encryption standard (aes) instruction set white paper. `https://www.intel.com/content/dam/doc/white-paper/advanced-encryption-standard-new-instructions-set-paper.pdf`. Accessed: 2021-10-28.

[12] Qualcomm. Cryptographic module in snapdragon 805 is fips 140-2 certified. `https://www.qualcomm.com/news/onq/2014/11/07/cryptographic-module-snapdragon-805-fips-140-2-certified`. Accessed: 2021-10-28.

[13] AMD. Amd geode lx processor family technical specifications. `https://www.amd.com/us/products/embedded/processors/geode-lx/Pages/geode-lx-processor-family-technical-specifications.aspx`. Accessed: 2021-10-28.

[14] Tony Le Bourne. "amd ryzen 7 1700x review". https://www.vortez.net/articles_pages/amd_ryzen_7_1700x_review,7.html. Accessed: 2021-10-28.

[15] Bruce Schneier. Aes announced. http://www.schneier.com/crypto-gram-0010.html. Accessed: 2021-10-28.

[16] Tim Greene. Fugaku still reigns as the world's fastest supercomputer. https://www.networkworld.com/article/3622923/fugaku-still-reigns-as-the-world-s-fastest-supercomputer.html. Accessed: 2021-10-28.