Adam Hayse

CS 492 – Computer Security

Survey paper

27 March 2017

Advanced Encryption Standard

The Advanced Encryption Standard(AES) is the most widely used encryption system used today. It is intended to be a replacement for the Data Encryption Standard(DES), which became outdated due to its small key size and the increase in computing power. DES did have a few weaknesses, but it was well-designed overall. However, AES was designed during a time when we had much more knowledge about the mathematics behind cryptography and is therefore more mathematical in its design. This mathematical design is meant to resist certain mathematics based attacks that DES was susceptible to. That being said, there are still some theoretical attacks that exist against AES. The security of AES has stood the test of time, and its security is trusted enough to be the standard used to encrypt top secret government documents.

**Overview of DES:**

DES is the predecessor to AES. DES is a block cipher that is also a Feistel cipher. A block cipher operates on plaintext by dividing it into fixed-size blocks that pass through a round function and produce ciphertext blocks that match the size of the original plaintext block. Feistel ciphers are block ciphers that process the plaintext by dividing it in half and generating the output left and right ciphertext blocks according to the following round function performed for n rounds:

P = (L0,R0) Li = Ri-1 Ri = Li-1 ⊕ F(Ri-1,Ki) C = (Ln,Rn) for i = 1, 2, 3, …, n

A basic understanding of boolean algebra makes it clear that these operations are invertible. This is a very important quality for an encrypting algorithm to have. Without it, you would be able to encrypt, but you wouldn’t be able to decrypt. This defeats the entire purpose of encrypting in the first place. A quick rearranging of the functions while preserving the equality yields these functions:

Ri-1 = Li Li-1 = Ri ⊕ F(Ri-1,Ki)

This shows that the ciphertext can be decrypted just as easily as it can be encrypted, which means it is invertible.

A large portion of the security offered by DES comes from the S-boxes in the round function. These S-boxes provide confusion while also being non-linear. When the NSA was consulted during the creation of DES, the S-boxes of the prototype round function were their main area of concern. The NSA had cryptographic knowledge of potential analytical attacks using the algorithm’s S-boxes that was not known at the time, specifically differential cryptanalysis, and they applied this knowledge to the design of the S-boxes. Differential cryptanalysis is a mathematics-based known plaintext attack that uses the computed difference between two known plaintext messages and their resulting ciphertext to deduce information about the key that was used to encrypt. In order for differential cryptanalysis to work, it needs to have complete knowledge of the encryption algorithm excluding the key that was used. History has shown that we cannot rely on the encryption algorithm being used to remain secret as a factor of security. The secrecy of the algorithm does not add to its security and is often a detriment to the security of the algorithm when relied on. Since this is the case, we must take precautions against known plaintext attacks, and this is exactly what the NSA did with DES.

In order for the DES S-boxes to be resistant against linear and differential cryptoanalytic attacks, it needed to follow some requirements. Here are some of the requirements:

- The output bits of the S-box cannot be too close to a linear function.

- Each possible 4-bit output is obtained exactly once if the leftmost and rightmost bits of the input remain fixed.

- If two inputs differ in exactly one bit, then their outputs must differ by at least 2 bits.

- If two inputs differ in bits 2 and 3 exactly (01**23**45), then the outputs must differ by at least 2 bits.

- If two inputs differ in bits 0 and 1, and are identical in bits 4 and 5, then their outputs must be different.

- For any nonzero 6-bit difference between inputs, no more than 8 of the 32 pairs with this difference can have a 6-bit difference between their outputs.

The influence that the NSA had on DES made it very secure for its time. It eventually became obsolete with the increase in computing power.

**Overview of AES:**

The advanced encryption standard was developed in the early 2000s during a time when we had much more knowledge of cryptography. Differential cryptanalysis was discovered a second time in the public realm, and the mathematics behind cryptography was more developed. Unlike DES, AES is not a Feistel cipher. However, it is a block cipher. There are multiple versions of AES that deal with different key sizes (128-, 192- and 256-bit), which results in differences in security. As expected, these differences in levels of security are a trade off for convenience. I’ll examine the 128-bit algorithm since the general algorithm is the same and its simplicity will help with explaining.

128-bit AES contains 10 rounds: 1 initialization round, 9 intermediate rounds (the first of the intermediate rounds and the initialization round are considered one round), and 1 final round. Each of these rounds contain a composition of four different functions: SubBytes, ShitRows, MixColumns, and AddRoundKey. The initial round contains only AddRoundKey, which performs addition modulo 2 (also known as XOR) between the input and a subkey provided by the Rijndael keyschedule algorithm. The AddRoundKey function at the start and end of this iterated block cipher implements a concept called key whitening, which is a modern technique that adds to the security of the encryption. The intermediate rounds perform all 4 functions in their respective orders as listed above. The final round performs the same operations as the intermediate round in the same order excluding the MixColumns function. I will examine each of these functions and take a brief look into the mathematics behind each of them.

**Byte Substitution:**

Byte substitution is performed in the SubBytes function. It provides confusion. Similar to the S-boxes in DES, it can be thought of as a look-up table. Unlike the S-boxes in DES, which reduce the input from 6 bits to 4 bits, this function does no reduction at all. It instead maps the input byte to a different byte, which is effectively a lookup table with 28 = 256 values. The SubBytes function in 128-bit AES divides the 128-bit input block into 16 bytes. These 16 bytes are processed in in parallel and passed through the 16 identical SubBytes functions. The idea behind byte substitution is incredibly simple. However, the math behind the selected byte mappings is anything but simple and it yields some very interesting mathematical properties.

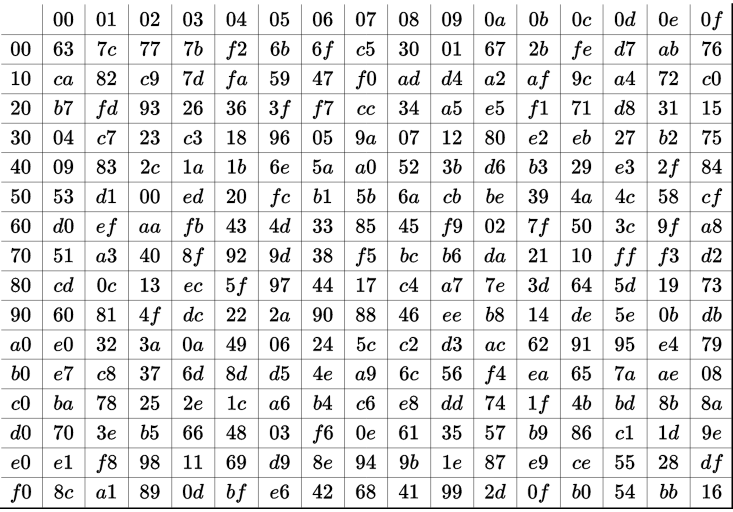
In order to properly explain how the mapping for the SubBytes function was chosen, we need to understand finite fields (also known as a Galois field). A finite field is a set in which addition, subtraction, multiplication, and division all yield results that exist within the set. In other words, there is closure for all of the elements within the set. A theorem in mathematics states that:

A finite field only exists if it has pm elements, where p is a prime and m is a nonnegative integer.

For example, the Galois field GF(2) exists because there exists a p and m such that pm = 2. In this case, p = 2 and m = 1. This is the most basic case. The Galois field GF(2) consists of 2 elements: 0 and 1.

There are 2 Galois fields that we are concerned about for performing computations in AES. These are the finite fields GF(2) = {0, 1} and GF(28) = GF(256) = {0, 1, x, x+1, x2, x2+1, x2+x, x2+x+1, … , x7+x6+x5+x4+x3+x2+x+1}. The elements of these fields, which are all polynomials, all follow the rules stated above for closure among the four operations. This is accomplished by using modular arithmetic. For addition and subtraction, which happen to be exactly the same for p=2, the operation performed is equal to a + b mod p. It becomes slightly more complicated for multiplication in division. For AES, we are not concerned with division, so I’ll stick to explaining multiplication.

In order for multiplication to be a closed operation within this finite field, we need to have a polynomial that behaves like a prime number to divide by. One such polynomial for GF(256) is P(x) = x8 + x4 + x3 + x + 1. We use this polynomial to reduce the produce of polynomial multiplication to fit within the Galois field and thus maintain closure. Here is a picture of the lookup table that is the ByteSub function:



Here is an example of how one of the values in this S-box is computed. Let’s take the value 0xc2, which is mapped to 0x25.

The first step is to represent 0xc2 as a polynomail. A(x) = 0xc2 = 1100 0010 = x7 + x6 + x + 1. The next step is to compute the inverse of this polynomial. The inverse of the polynomial is another polynomial such that the multiplication between the two polynomials results in 1. This value is B’(x) = x5 + x3 + x2 + x + 1= 0010 1111 = 0x2f.

After this value is obtained, the next step is to perform an affine mapping. The purpose of this mapping is to take away from the mathematical nature of how B’(x) was computed, since this could lead to an attack. This final step is computed by performing matrix multiplication between an 8x8 matrix and B’(x), and then performing modulo 2 addition between the result and another 8x1 matrix. An interesting note about these matrices is that they consist of only 1s and 0s.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | X | 0 | + | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

b0 = (x7+x3+x2+x+1) x (x5 + x3 + x2 + x + 1) = x12 + x10 + x9 + x8 + x7 + x8 + x6 + x5 + x4 + x3 + x7 + x5 + x4 + x3 + x2 + x6 + x4 + x3 + x2 + x + x5 + x3 + x2 + x + 1 = x12 + x10 + x9 + (1+1)x8 + (1+1)x7 + (1+1)x6 + (1+1+1)x5 + (1+1+1)x4 + (1+1+1+1)x3 + (1+1+1)x2 + (1+1)x + 1 = x12 + x10 + x9 + x5 + x4 + x2 + 1

x12 + x10 + x9 + x5 + x4 + x2 + 1 : x8 + x4 + x3 + x + 1 = x4

x12 + x8 + x7 + x5 + x4

(1+1)x12 + x10 + x9 + x8 + x7 + (1+1)x5 + (1+1)x4 + x2 + 1 = x10 + x9 + x8 + x7 +x2 + 1

x10 + x9 + x8 + x7 +x2 + 1 : x8 + x4 + x3 + x + 1 = x4 + x2

x10 + x6 + x5 + x3 + x2

(1+1)x10 + x9 + x8 + x7 + x6 + x5 + x3 + x2 + 1 = x9 + x8 + x7 + x6 + x5 + x3 + x2 + 1

x9 + x8 + x7 + x6 + x5 + x3 + x2 + 1 : x8 + x4 + x3 + x + 1 = x4 + x2 + x

x9 + x5 + x4 + x2 + x

(1+1)x9 + x8 + x7 + x6 + (1+1)x5 + x4 + x3 + (1+1)x2 + x + 1 = x8 + x7 + x6 + x4 + x3 + x + 1

x8 + x7 + x6 + x4 + x3 + x + 1 : x8 + x4 + x3 + x + 1 = x4 + x2 + x + 1

x8 + x4 + x3 + x + 1

(1+1)x8 + x7 + x6 + (1+1)x4 + (1+1)x3 + (1+1)x + (1+1) = x7 + x6 = 1100 0000 = 0xc0

0xc0 + 0xc6 = 0x06

I computed the incorrect answer. As you can see, the math behind S-box of AES is very tedious. This does not come into play at all however since all of these values are computed beforehand. I wish I had more time to look into Galois fields and the S-box of AES because they seem very interesting.

**Row Shift:**

The row shift function provides diffusion. It simply rearranges the output bytes resulting from the 16 byte substitutions. The purpose of this operation is to prevent the individual columns from being linearly independent. It simply adds a little bit of complexity to the algorithms. This operation can be better understood by organizing the 16 bytes into a 4x4 matrix and performing the shift operations. The bytes produced from the previous step are numbered a0 through a15. They are entered into a 4x4 matrix as shown below.

|  |  |  |  |
| --- | --- | --- | --- |
| a0 | a4 | a8 | a12 |
| a1 | a5 | a9 | a13 |
| a2 | a6 | a10 | a14 |
| a3 | a7 | a11 | a15 |

The following operations are performed:

- Do nothing to row 1.

- Shift row 2 to the left 1.

- Shift row 3 to the left 2.

- Shift row 4 to the left 3.

The result is as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| a0 | a4 | a8 | a12 |
| a5 | a9 | a13 | a1 |
| a10 | a14 | a2 | a6 |
| a15 | a3 | a7 | a11 |

The resulting output is a rearrangements of the bytes. The new ordering is a0, a5, a10, a15, a4, a9, a14, a3, a8, a13, a2, a7, a12, a1, a6, a11. This function combined with the MixColumns function provide very strong diffusion to the encryption algorithm.

**Mix Columns:**

The mix columns function takes advantage of matrix multiplication to add complexity to the algorithm. This function makes it possible for a single change in a set of 4 bytes affects the entire output of 16 bytes. Here is a quick demonstration of how it works:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 02 | 03 | 01 | 01 | X | a0 | = | C0 |
| 01 | 02 | 03 | 01 | a5 | C1 |
| 01 | 01 | 02 | 03 | a10 | C2 |
| 03 | 01 | 01 | 02 | a15 | C3 |

For example, the value of C0 is equal to 02 x a0 + 03 x a5 + 01 x a10 + 01 x a15. a0 is a factor in the computation of C0. As you can see, it will also be a factor in C1, C2, and C3. This is the nature of matrix multiplication and it adds to the security of the algorithm.

The matrix multiplication performed above isn’t as simple as regular matrix multiplication between simple numbers. What is happening above is matrix multiplication of polynomials. The values in the 4x4 matrix are hexadecimal representations of polynomials that occupy the 256 element Galois field. 01 is equal to the polynomial 1. 02 is equal to the polynomial x. 03 is equal to the polynomial x+1. These are then multiplied with the polynomial representations of each of the input bytes (a0, a5, a10, and a15), and modded with an irreducible polynomial P(x). An irreducible polynomial has the same properties as a prime number, but it is in polynomial form. The irreducible polynomial P(x) of GF(256) for AES is:

P(x) = x8 + x4 + x3 + x +1

**Add Round Key:**

This function simply performs modulo 2 addition between its input and the subkey provided by the Rijndael key schedule algorithm. It is an XOR operation. This function used at the start and end of the entire encryption function adds key whitening to the algorithm. This process increases the security of the encryption by hiding the input and output going in and out of the first and last round of the block cipher. This additional level of security helps to prevent a meet-in-the-middle attack (MITM). A MITM attack attempts to use the plaintext and ciphertext to find a forward mapping of the plaintext through the first functions that matches a backward mapping of the ciphertext through the last functions in order to determine the key that is used. This is one of the security flaws of the original DES, which went unaddressed until the tenth iteration of DES.

**Possible Attacks:**

People have attempted to find weaknesses in AES since its inception. The first controversial attack was called the XSL attack. This stands for extended sparse linearization. This attack happened to have a high work factor, so it actually did not perform better than a brute force attack.

Another attack was discovered in 2009 that took advantage of the Rijndael key schedule algorithm. This attack has a complexity of 296, which is much better than a brute force method. However, the attack only worked for 1 out of every 235 keys. Selected at random, a given key would have a 1 out of 34,359,738,368 chance of being susceptible to this attack. This problem is easily avoided by following the proper protocol.

As the most widely used encryption algorithm used in the world, people are extremely motivated to investigate this algorithm and look for potential weaknesses to be worked around. The fact that it still remains in use after 16 years is a clear demonstration of its security. This does not mean that a major weakness will not be found, so it is important to constantly look for new ways to break it.

References

Paar, Christof. "Advanced Encryption Standard (AES)." *YouTube*. Web. 26 Mar. 2017.

<https://www.youtube.com/watch?v=NHuibtoL\_qk>.

Paar, Christof. "Introduction to Galois Fields for the AES." *YouTube*. Web. 26 Mar. 2017.

<https://www.youtube.com/watch?v=x1v2tX4\_dkQ>.

Coppersmith, D. "The Data Encryption Standard (DES) and its strength against attacks." *IBM Journal of Research and Development* 38.3 (1994): 243-50. Web.

Stamp, Mark. *Information security: principles and practice*. Hoboken, NJ: Wiley, 2011. Print.