Analysis of the XSL Attack

Coteanu Maria Gabriela[1] and Ţîflea Denisa-Ionela[1]

[1]Master of Information Security, Alexandru Ioan Cuza University of Iaşi

Abstract

In this paper, we examine the algebraic XSL attack on the Advanced Encryption Standard (AES). We begin with a brief introduction and we present an overview of AES, then, in Section 3, we present the algebraic attack on ciphers like AES, following with the XL and XSL algorithms in Section 4 and Section 5. Then, we present the XSL first and second attacks, also their aplicability on BES. We see how and if the algorithm has been improved since it firstly appeared. We conclude with Section 10.

1 Introduction

AES encryption is a widely used symmetric key encryption algorithm that is recognized for its security and efficiency. It is a standard for encrypting sensitive information, including financial transactions and confidential communications. Many cryptographers have tried to break the algorithm, but the only one that was thought to be successful is the XSL attack, to prove later that this is not effective either. In this article, we are going to analyze the types of XSL algorithms, attacks, their implementation, and use in the future.

2 Overview of block ciphers

2.1 AES

The **Advanced Encryption Standard**, also known as the *Rijndael* algorithm, is a 128 bits block cipher encryption for electronic data established by NIST in 2001[1] and has been adopted by the US government as a substitute for DES. It is performed on a 4x4 array of bytes:

$$\begin{bmatrix} b_0 & b_4 & b_8 & b_{12} \\ b_1 & b_5 & b_9 & b_{13} \\ b_2 & b_6 & b_{10} & b_{14} \\ b_3 & b_7 & b_{11} & b_{15} \end{bmatrix}$$

The key size determines the number of rounds required to convert the plaintext into ciphertext. For example, 10 rounds are needed for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys.

The **first round** is the *AddRoundKey*, where each byte of the state is XORed with a corresponding byte from the round key.

For the **remaining n-1 rounds**, the following operations are performed:

- *SubBytes* (replacing each byte in the state array with a corresponding value from a S-box table),
- *ShiftRows* (shifting the rows of the state array cyclically to the left),
- *MixColumns* (diffusing the cipher by multiplying each column of the current state with a fixed polynomial), and
- *AddRoundKey*.

The **final round** includes *SubBytes*, *ShiftRows*, and *AddRoundKey*, but not *MixColumns*, as not significantly improve the security of the algorithm and has a different structure in decryption compared to encryption.

2.2 Rijndael vs Serpent

In 2001, the finalists of AES selection process were *Rijndael* algorithm (proposed by Joan Daemen and Vincent Rijmen [2]) and *Serpent* algorithm (designed by Ross Anderson, Eli Biham, and Lars Knudsen [3]). In the end, Rijndael won and AES is an adaptation of it, of fixed 128 bits block size and a key size of 128, 192, or 256 bits, besides the original algorithm which has the block and key size multiple of 32 bits, starting from 128 up to 256 bits.

In contrast, the Serpent algorithm has the same block and key sizes as the standardized AES but has more negative votes because does not allow for an efficient future software implementation.

2.3 Block ciphers operation modes

A block cipher is a symmetric encryption algorithm that encrypts data in fixed-size blocks, typically with a block size of 128 bits. The encryption process involves dividing the plaintext message into blocks, and then each block is encrypted using the encryption key to produce the corresponding ciphertext block. The key is then used in reverse to decrypt the ciphertext and recover the original plaintext.

AES encryption algorithm can operate in different modes, each with its own strengths and weaknesses. The most commonly used modes are:

Electronic Codebook (ECB) mode: In ECB mode, the plaintext message is divided into blocks and each block is encrypted independently using the same encryption key. ECB mode is simple to implement but is not secure, as repeated blocks of plaintext will produce the same ciphertext, leading to patterns in the encrypted data.

Cipher Block Chaining (CBC) mode: In CBC mode, each block of plaintext is XORed with the ciphertext of the previous block before being encrypted. This makes it more secure than ECB mode, as it ensures that repeated blocks of plaintext will not produce the same ciphertext.

Counter (CTR) mode: In CTR mode, a counter is used to generate a stream of keystream bits, which are XORed with the plaintext to produce the ciphertext. CTR mode is highly efficient and can be used for parallel encryption and decryption, making it suitable for hardware acceleration.

Galois/Counter Mode (GCM): GCM is a mode of operation that provides both encryption and authentication. It uses a Galois field to encrypt the data and generate a unique tag that is sent along with the encrypted data to verify its authenticity upon decryption.

Output Feedback (OFB) mode: In OFB mode, a pseudorandom bit stream is generated and XORed with the plaintext to produce the ciphertext. OFB mode is particularly useful for streaming data, as it ensures that errors in the encrypted data will not propagate to subsequent blocks.

3 Algebraic Attack on AES-like Ciphers

AES can be represented as algebraically closed equations over $GF(2^8)$ [6]. It can also be represented as a system of multivariate quadratic equations over GF(2) with plaintext, ciphertext and key bits as variables. [4]

The Multivariate Quadratic (MQ) problem involves finding a solution to systems of equations that consist of multiple variables and are expressed as quadratics. This problem is considered NP-Hard for any general field [5], meaning there is no known efficient algorithm to solve it. In fact, solving a system of quadratic equations in any finite field is considered NP-Complete, which means it's a difficult computational problem. Despite the lack of a polynomial-time solution for the MQ problem in general, there are algorithms that can run efficiently for specific cases where the system of equations is over-defined.

The process of an algebraic attack on the MQ problem consists of two phases: generating equations and solving the resulting system.

4 XL Algorithm

The XL [6] (extended linearization) algorithm was introduced by Courtois, Klimov, Patarin, and Shamir in the year 2000 as an improvement to the relinearization method for solving large systems of overdefined multivariate quadratic polynomial equations. The XL algorithm works by expanding the initial system of equations through monomial multiplications, creating a new system that is viewed as a linear equation in the resulting monomials.

The XL (extended linearization) algorithm is effective in solving the MQ problem when the number of equations surpasses the number of variables. It achieves this by growing the initial equation system by incorporating new dependent equations that are not linearly related to the original set. The expansion process is performed using multiplications with monomials of restricted degrees.

The XL algorithm accepts as input the initial system of equations $\mathbb{A}$ (which has at least one solution), and a degree bound $D \in \mathbb{N}$.

If we consider a system of m quadratic equations and n variables over a finite field $\mathbb{K}$,

$$f_1(x_1, ..., x_n) = 0, ..., f_m(x_1, ..., x_n) = 0,$$

the algorithm simply multiplies the original equations by all monomials M_i

up to a prescribed degree $D - 2$, and attempts to solve the system of all resulting equations

$$M_i \cdot f_j(x_1, ..., x_n) = 0$$

of degree at most D by linearization.

One of the drawbacks of the XL algorithm is that as the degree bound D increases, so does the size of the expanded system, often leading to a significant growth. Additionally, a large number of linearly dependent equations are generated during the expansion, resulting in a larger system. Despite this, the equations produced by the XL algorithm tend to be sparse in general.

Unfortunately, the complexity of the XL method in general is not known [7], nor do we even know under what precise conditions the algorithm terminates. However, experimentally, it is more efficient than relinearization for proper choices of D. These experiments also show that even a little overdefinition helps a lot. Clearly the work depends a great deal on D.

5 XSL Algorithm

The XSL algorithm, which stands for "eXtended Sparse Linearization", operates differently from the XL algorithm in the way that equations are multiplied by monomials. In the XL algorithm, all monomials up to a certain degree are used for multiplication, whereas in the XSL algorithm, only specific, "carefully selected monomials" are utilized for this purpose.

There are different versions of the XSL algorithm. The first version was proposed in [8], where two different attacks were described: the first one eliminating the key schedule equations (but requiring a number of plaintext-ciphertext pairs), and a second, more specific attack, that used the key schedule equations (and should work with a single plaintext-ciphertext pair). Later a different version of the algorithm was introduced in [9] (called "compact XSL").

The objective of the XSL algorithm is to generate fewer new monomials during the equation expansion process, in comparison to the XL algorithm. This is achieved by being selective about the monomials used for multiplication. Additionally, the XSL algorithm includes a final step, known as the T' method, which aims to produce new linearly independent equations without creating additional monomials.

The XSL algorithm consists of four main steps [10]:

1. Process the existing set of equations, by choosing certain sets of monomials and equations that will be used during the later steps of the algorithm.

2. Select the value of the parameter P, and multiply the chosen equations by the product of $P-1$ selected monomials. This is the "core" of the XSL attacks and should generate a large number of equations whose terms are the product of the monomials chosen earlier.

3. Perform the T' method, in which some selected equations are multiplied by single variables. The goal is to generate new equations without creating any new monomials. Iterate with as many variables as necessary until the system has enough linearly independent equations to apply linearization.

4. Apply linearization, by considering each monomial as a new variable and performing Gaussian elimination. This should yield a solution for the system.

6 XSL Ciphers

By definition, an XSL-cipher is a composition of N_r similar rounds:

- **X**: The first round $i = 1$ starts with a XOR with the session key K_{i-1},
- **S**: we apply a layer of B bijective S-boxes in parallel, each on s bits,
- **L**: we apply a linear diffusion layer,
- **X**: we XOR with another session key K_i.
- If $i = N_r$ we finish, otherwise we increment i and go back to step **S**.

We denote the key bits used in an XSL-cipher by the variables K_{ij} with $i = 0 \ldots N_r$ and $j = 1 \ldots s * B$. There are $N_r + 1$ session keys, K_0 is the first and K_{N_r} is the last.

7 XSL Attack

7.1 First XSL Attack

The XSL attack is a method to determine the secret key of a cipher. For each S-box in the cipher with r equations and t terms, a set of quadratic equations will be written that fully defines the secret key.

Let A be an S-box of an XSL cipher, called an "active S-box". For this S-box A, we may write r equations of the form:

$$0 = \sum \alpha_{ijk} X_{ij} Y_{ik} + \sum \beta_{ij} X_{ij} + \sum \gamma_{ij} Y_{ij} + \delta$$

The number of monomials included in these equations is limited, only t in total (most of which have the form $X_{ij}Y_{ik}$). To further expand these equations, one of the t monomials from other "passive" S-boxes will be multiplied. The total number of S-boxes, S, equals $B * N_r * (N_r + 1)$, where B is the block size and $N_r + 1$ refers to the number of cipher executions. A critical parameter, P, is defined such that each equation from each "active" S-box will be multiplied by all possible terms for all subsets of (P-1) "passive" S-boxes. If P is large, the result will be similar to the general XL attack.

As a result, the total number of equations generated by this method will be about:

$$R \approx r * S * t^{P-1} * \binom{S-1}{P-1}$$

And the total number of terms in these equations:

$$T \approx t^{P} * \binom{S}{P}$$

However, a lot of them lack independence and the limitations were only to multiply a specific "active" equation by one of the monomials T_1 to T_{t-r}, for a particular "passive" S-box in the system. Additionally, the rules dictate that equations containing multiple "active" S-box products must also be included.

$$R \approx \binom{S}{P} \left(t^{P} - (t-r)^{P}\right)$$

As we can see, when P grows we will have R/T $\rightarrow$ 1. In addition, if we define $t' < t$ being the number of terms that can be multiplied by x_1 and

$$T' \approx t' * t^{P-1} * \binom{S-1}{P-1}$$

in order to solve such a system of equations, we need to have T - R < T' which is true, for a sufficiently large P.

In fact, the attack will always work for some fixed P because it seems [8] that P will increase with the number of rounds.

7.2 Second XSL Attack

In the second attack, the key schedule is utilized. Similar to before, a system of equations will be written with a separate variable for each input and output bit of each S-box, including those in the key schedule. We will have:

$$S = \Lambda * B * N_r + D + 1$$

where Λ is the number of plaintexts needed in order to completely determine the key used in the cipher and D is the number of S-boxes in the key schedule. But the number of equations is again equal to

$$R \approx \binom{S}{P} (t^P - (t-r)^P)$$

By adding the following equations: $X_{i+1j} = \sum \alpha_j Y_{ij} \oplus [K_{ij}]$ where $[K_{ij}]$ is the expression of K_{ij} as a linear combination of the S_k "true" key variable, and by multiplying with the products of terms of (P - 1) "passive" S-boxes, we obtain:

$$R' \approx \Lambda * s * B * (N_r + 1) * (t-r)^{P-1} * \binom{S}{P-1}$$

What is missing are the linear equations on the key schedule, which arise from the fact that the S_k key variables are not linearly independent. These equations are produced by multiplying the products of terms of (P-1) "passive" S-boxes and

$$R'' \approx (S_k - L_k) * (t-r)^{P-1} * \binom{S}{P-1}$$

The attack will work when P is:

$$\frac{R + R' + R''}{T - T'} > 1$$

The objective of XSL is to select an appropriate P so that we get enough equations. The following computations apply for AES-128, AES-192 and AES-256 [11]:

- **AES-128**: For the smallest $P = 7$, the parameters $R = 4.95 \times 10^{25}$, $R' = 4.85 \times 10^{24}$, $T = 5.41 \times 10^{25}$ and $(R + R')/T = 1.004$, the complexity of XSL attack is $T^{2.376} \approx 2^{203}$.
- **AES-196**: For the smallest $P = 7$, the parameters $R = 8.65 \times 10^{27}$, $R' = 8.50 \times 10^{26}$, $T = 9.46 \times 10^{27}$ and $(R + R')/T = 1.004$, the complexity of XSL attack is $T^{2.376} \approx 2^{221}$.
- **AES-256**: For the smallest $P = 7$, the parameters $R = 3.15 \times 10^{28}$, $R' = 3.02 \times 10^{27}$, $T = 3.45 \times 10^{28}$ and $(R + R')/T = 1.002$, the complexity of XSL attack is $T^{2.376} \approx 2^{225}$.

8 XSL Attack on BES

The Big Encryption System (BES) embedding for the AES cipher, introduced by Robshaw and Murphy, simplifies the quadratic equations that describe the S-box input-output relationship. This results in a significant decrease in the number of monomials present in the equation system, potentially leading to an exponential reduction in the complexity of the XSL attack. [12] AES performs operations in the binary field F_2, while BES achieves the same outcome through operations in the field F_{256}. This new representation is advantageous due to the simplicity of the S-box equation, as $xy = 1$ is immediately obtainable rather than having to use 8 quadratic equations for the input and output bits.

If we consider the set of non-reduced S-box equations and linear equations and fix the 8 input variables of an S-box, then the removal of each S-box results in 8 free variables. The number of linearly independent terms is then equal to or greater than the number of reduced monomials formed by these 8S free variables:

$$D_1 = \sum_{i=0}^{P} \binom{S}{i} 8^i$$

It is enough when solving the collection of extended linear equations with the extended S-box equations to be of the form $(v)(m_1) = (m_2)$, where m_1 and m_2 are reduced monomials of degree at most P - 1 and v is an S-box variable such that it or its dual occurs in m_1. If we set v' that occurs in m_1, we have the case when $v = v'$, so we have 16S choices and 8S choices for the case when *v'* is the dual of *v*, then the number of relevant equations is:

$$D_2 = 24S * \sum_{i=0}^{P-2} \binom{S-1}{i} 8^i$$

We need $D_2 > D_1$ to solve the secret key, then:

- **BES-128**: For the smallest P = 23, the complexity of XSL attack is $D_1^{2.376} = (5.9 \text{ x } 10^{50})^{2.376} \approx 2^{401}$
- **BES-196**: For the smallest P = 33, the complexity of XSL attack is $D_1^{2.376} = (5.857 \text{ x } 10^{78})^{2.376} \approx 2^{622}$
- **BES-256**: For the smallest P = 36, the complexity of XSL attack is $D_1^{2.376} = (3.798 \text{ x } 10^{87})^{2.376} \approx 2^{691}$

according to the table from [11], which gives worse complexity than the XL attack against AES-128.

9 Improvements

In Asiacrypt 2005, Cid and Leurent [10] gave an analysis of the compact XSL attack [9] on AES-128 and proved that it is equivalent to a substitution-thenXL (sXL) method. They concluded that XSL attack is essentially an XL attack on a system of equations larger than that of the original AES. Thus compact XSL is not an effective attack against the AES cipher. This partly answers some uncertainties of the compact XSL attack and suggests that it may not be an effective method against block ciphers. However, it does not give us the full answer on whether XSL is effective against AES. This is because in [10], only the compact XSL attack is analyzed.

Several modifications to the XL algorithm have been proposed, including the XL SGE (eXtended Linearization with Structured Gaussian Elimination) algorithm [4]. The XL SGE algorithm aims to reduce the number of linearized equations generated by the XL algorithm. This reduction is achieved through the use of structured Gaussian elimination (SGE) on the intermediate systems generated during the XL process. The resulting reduced systems are then multiplied with monomials to produce systems with higher algebraic degrees.

The XSL attack has a high computational cost, making it no more efficient in breaking AES than an exhaustive search. This means that it does not pose a significant threat to the security of block ciphers in the near future. Nevertheless, the simplicity of the XSL attack has raised concerns among experts about the algebraic security of AES.

10 Acknowledgement

We want to thank Prof. dr. Ferucio Laurențiu Ţiplea from the "Alexandru Ioan Cuza" University of Iaşi for his insightful suggestions over the subject and for providing us most of the relevant resources for writing this article.

We would also like to acknowledge the assistance of Prof. dr. Emil Simion, collaborator of the Information Security master's program of the "Alexandru Ioan Cuza" University of Iaşi, who supported us through the writing process.

11 Conclusions

There was some disagreement over the validity of the claims made in the original XSL paper [7]. Moh and Coppersmith initially claimed to have found issues with the effectiveness of the T' method and with the projections for the number of linear equations. However, Moh later withdrew part of his criticisms from his website.

Courtois, talking about the Courtois-Pieprzyk-Murphy-Robshaw attack, says that no one has yet shown that XSL will break AES. On the other hand, no evidence has been provided to show that the XSL algorithm cannot break AES either. The XL family of algorithms remains an area of ongoing research and the outcome is still uncertain regarding the possibility of AES being attacked through solving it as an MQ problem.

References

[1] I. T. L. N. I. of Standards and Technology), "Announcing the advanced encryption standard (aes)," 2001.

[2] J. Daemen and V. Rijmen, “The block cipher rijndael,” in *Smart Card Research and Applications*, J.-J. Quisquater and B. Schneier, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 277–284.

[3] R. Anderson, E. Biham, and L. Knudsen, “Serpent: A proposal for the advanced encryption standard,” 09 2000.

[4] S. Ghosh and A. Das, “An improvement of linearization-based algebraic attacks,” in *Security Aspects in Information Technology*, M. Joye, D. Mukhopadhyay, and M. Tunstall, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 157–167.

[5] E. Kleiman.

[6] N. Courtois, A. Klimov, J. Patarin, and A. Shamir, “Efficient algorithms for solving overdefined systems of multivariate polynomial equations,” in *Advances in Cryptology — EUROCRYPT 2000*, B. Preneel, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 392–407.

[7] H. Nover, “Algebraic cryptanalysis of aes: an overview,” *University of Wisconsin, USA*, pp. 1–16, 2005.

[8] N. T. Courtois and J. Pieprzyk, “Cryptanalysis of block ciphers with overdefined systems of equations,” in *Advances in Cryptology — ASIACRYPT 2002*, Y. Zheng, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 267–287.

[9] ——, “Cryptanalysis of block ciphers with overdefined systems of equations,” in *Advances in Cryptology — ASIACRYPT 2002*, Y. Zheng, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 267–287.

[10] C. Cid and G. Leurent, “An analysis of the xsl algorithm,” in *Advances in Cryptology - ASIACRYPT 2005*, B. Roy, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 333–352.

[11] C.-W. Lim and K. Khoo, “An analysis of xsl applied to bes,” in *Fast Software Encryption*, A. Biryukov, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 242–253.

[12] S. Murphy and M. J. Robshaw, “Essential algebraic structure within the aes,” in *Advances in Cryptology — CRYPTO 2002*, M. Yung, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 1–16.