In How Many Ways Can You Write Rijndael?*

Elad Barkan Eli Biham

Computer Science Department
Technion – Israel Institute of Technology
Haifa 32000, Israel
Email: {barkan,biham}@cs.technion.ac.il
WWW: http://www.cs.technion.ac.il/~biham/
WWW: http://tx.technion.ac.il/~barkan/

Abstract. In this paper we ask the question what happens if we replace all the constants in Rijndael, including the replacement of the irreducible polynomial, the coefficients of the Mix-Column operation, the affine transformation in the S box, etc. We show that such replacements can create new *dual ciphers*, which are *equivalent* to the original in all aspects. We present several such dual ciphers of Rijndael, such as the square of Rijndael, and dual ciphers with the irreducible polynomial replaced by primitive polynomials. We also describe another family of dual ciphers consisting of the logarithms of Rijndael. We then discuss self-dual ciphers, and extend our results to other ciphers.

Keywords: Rijndael, AES, Galois Field, Dual Cipher, Self Dual, Logarithm.

1 Introduction

Recently, the cipher Rijndael [9] was selected as the Advanced Encryption Standard (AES) [19]. This cipher operates over the algebraic Galois field $GF(2^8)$. The motivation for this is computational efficiency, as $GF(2^8)$ elements can be represented by bytes, which can be very efficiently processed by modern computers, unlike bit-level operations that are usually more expensive in computer power. The drawback is that the algebraic structure inherited by the $GF(2^8)$ operations may be susceptible to algebraic relations, such as the relations in [10,22]. Algebraic structures may be used to develop cryptographic attacks that exploit the algebraic weakness of the cipher. An example for such attacks are interpolation attacks [12]. In attempt to avoid some of these difficulties, other mechanisms are introduced to these ciphers, such as bit level affine transformations.

In this paper we ask the question what happens if we replace all the constants in Rijndael, including the replacement of the irreducible polynomial, the coefficients of the MixColumn operation, the affine transformation in the S box, etc. We show that such replacements can create new *dual ciphers*, which are *equivalent* to the original in all aspects. Although their intermediate values during encryption are different than Rijndael's, we can show that they are *equivalent* to Rijndael. Examples of such ciphers include ciphers with a primitive polynomial (replacing the irreducible polynomial of Rijndael), the cipher *Square of Rijndael* that encrypts the square of the plaintext under the square of the key to the square of the ciphertext, and a cipher with a triangular affine matrix in the S box.

Definition 1 Two ciphers E and E' are called *Dual Ciphers*, if they are isomorphic, i.e., if there exist invertible transformations $f(\cdot)$, $g(\cdot)$ and $h(\cdot)$ such that

$$\forall P, K \qquad f(E_K(P)) = E'_{g(K)}(h(P)).$$

* An earlier version of this paper appears in Asiacrypt 2002.

Trivial dual ciphers are very easy to find for all ciphers. For example, every cipher is dual to itself with the identity transformations. Also, for any cipher, the addition of non-cryptographic invertible initial and final transformations creates a trivial dual cipher. We are not interested in these kinds of dual ciphers. The interesting question is whether there exists non-trivial dual ciphers of widely used ciphers.

An extension of dual ciphers, are semi-dual ciphers:

Definition 2 A cipher E' is called a *semi-dual cipher* of E, if there exist transformations $f(\cdot)$, $g(\cdot)$ and $h(\cdot)$ such that

$$\forall P, K \qquad f(E_K(P)) = E'_{g(K)}(h(P)).$$

where f,g and h are not necessarily invertible (and even not necessarily length-preserving).

Semi-dual ciphers potentially reduce the plaintext, the ciphertext, and the key spaces, and thus may allow to develop efficient attacks on their original cipher.

In this context we would like to mention that interpolation attacks [12] exploit the low order of interpolation polynomial of the cipher. However, they do not exploit other algebraic properties of the interpolation polynomial, such as these used in this paper.

Definition 3 In this paper we consider ciphers whose all operations are of the following types:

- Operations in $GF(2^8)$:
  1. Addition (i.e., XOR: $f(x,y) = x \oplus y$).
  2. XOR with a constant (e.g., $f(x) = x \oplus 3F_x$).
  3. Multiplication ($f(x,y) = x \cdot y$).
  4. Multiply by a constant (e.g., $f(x) = 03_x \cdot x$).
  5. Raise to any power (i.e., $f(x) = x^c$, for any integer c). This includes the inverse of x: x^{-1}.
  6. Any replacement of the order of elements (e.g., taking a vector containing the elements $[a,b,c,d]$, and changing the order to $[d,c,a,b]$).
- Non-$GF(2^8)$ operations:
  7. Linear transformations $L(x) = Ax$, for any boolean matrix A.
  8. Any unary operation over elements in $GF(2^8)$. (i.e., a look-up table, $S(x) = LookUpTable[x]$ or $F(x) : \{0,1\}^8 \longrightarrow \{0,1\}^8$).

We call these operations $EGF(2^8)$ *operations.*

Please note that this notation implies that in item 7, the variable x, which is an element in $GF(2^8)$, is converted to a vector of 8 bits (in $GF(2)^8$) before being multiplied by the matrix A. The result is converted back to be an element of $GF(2^8)$. It should be noted that since XOR with a constant is also allowed in item 2, any affine transformation is included in the operations we consider (i.e., $F(x) = Ax \oplus b$).

An example of such a cipher is Rijndael (AES) [9]. Many other ciphers are also built from these operations. Some examples of them are: Shark [7], Square [8], Scream [11], Crypton [15], Anubis [3], Khazad [4]. Our results can also be extended to Safer++ [16], E2 [18] and Camellia [1].

In this paper we also deal with the special case of self-duality. That is the case where a cipher is a dual of itself. We study this case and show that such ciphers can be attacked faster than exhaustive search. It is interesting to mention that RSA [20] is an example of a self-dual public key cipher. Let e and n be the RSA public key, and let $c = p^e \pmod{n}$ where p is the plaintext and c is the ciphertext. Then it follows that RSA is a dual of itself: $c^2 = (p^2)^e \pmod{n}$.

We also discuss the family of *Log Dual Ciphers.* In a log dual cipher, the logarithm of the plaintext is encrypted by the logarithm of the key to the logarithm of the ciphertext. We show that Rijndael has a family of log dual ciphers.

We indicate a variety of possible applications for dual ciphers, ranging from gaining insight for differential [5] and linear [17] cryptanalysis, to speeding up encryption, and to protect against power analysis [14] and fault analysis [6].

This paper is organized as follows: In section 2 we give a short description of Rijndael. Section 3 shows how to define square dual ciphers. Section 4 deals with changing the irreducible polynomial. Section 5 shows how to define logarithmic dual ciphers. In section 6 we discuss the special case of self-duality and show how to mount an attack on self-dual ciphers. Section 7 deals with application to other ciphers. Section 8 deals with other applications of dual ciphers. The paper is summarized in Section 9.

2 Description of Rijndael

In this section we give a short description of Rijndael. For a full description of Rijndael the reader may consult [9,19]. Rijndael is a block cipher with 128-bit blocks, and three key sizes of 128, 192 and 256 bits. The 128-bit blocks are viewed as either 16 bytes or as four 32-bit words. The bytes are organized in a square form:

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}

where b_i notes the i'th byte of the block.

Each column in this representation can be viewed as a 4-byte word. Rijndael has operations that work on columns, operations that work on rows, and operations that work on each byte separately.

The plaintexts are encrypted through successive operation of a round function 10 times (or 12 or 14 times for 192-bit and 256-bit keys, respectively).

A round is composed of 4 consecutive operations:

1. ByteSub: An S box is applied to each byte of the data (16 times in parallel).
2. ShiftRow: Changing the order of bytes in the data.
3. MixColumn: Every 4 consecutive bytes (column) are mixed by a linear operation.
4. AddRoundKey: The data is XORed with a 128-bit subkey.

The S box of Rijndael is taking the multiplicative inverse of the input in $GF(2^8)$ (modulo the irreducible polynomial of Rijndael $x^8 + x^4 + x^3 + x + 1$, which is denoted in binary notation by $11B_x$; for the purpose of inversion the inverse of 00_x is defined to be 00_x), the output of which is transformed by the affine transformation:

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

where the x_i's and the y_i's are coefficients of x and y (i.e., the bits of the bytes), and x_0 and y_0 are the least significant bits.

The ShiftRow operation is defined as changing the order of bytes in the data. When viewing the data in its square form, ShiftRow is:

– Leaving the first first row unchanged.
– Shifting the second row by one byte to the left (cyclically).
– Shifting the third row by two bytes to the left (cyclically).
– Shifting the fourth row by three bytes to the left (cyclically).

Taking the square form as the input of the ShiftRow operation, the ShiftRow operation has the following effect:

$$\begin{array}{|c|c|c|c|} \hline b_0 & b_4 & b_8 & b_{12} \\ \hline b_1 & b_5 & b_9 & b_{13} \\ \hline b_2 & b_6 & b_{10} & b_{14} \\ \hline b_3 & b_7 & b_{11} & b_{15} \\ \hline \end{array} \xrightarrow{\text{ShiftRow}} \begin{array}{|c|c|c|c|} \hline b_0 & b_4 & b_8 & b_{12} \\ \hline b_5 & b_9 & b_{13} & b_1 \\ \hline b_{10} & b_{14} & b_2 & b_6 \\ \hline b_{15} & b_3 & b_7 & b_{11} \\ \hline \end{array}$$

The MixColumn operation mixes every 4 consecutive bytes (every column) of the data. Therefore, there are 4 mix operations in each round. Let b_i, b_{i+1}, b_{i+2}, b_{i+3} be consecutive bytes of a column. The new state is defined by

$$\begin{array}{|c|c|c|c|} \hline b_0 & b_4 & b_8 & b_{12} \\ \hline b_1 & b_5 & b_9 & b_{13} \\ \hline b_2 & b_6 & b_{10} & b_{14} \\ \hline b_3 & b_7 & b_{11} & b_{15} \\ \hline \end{array} \xrightarrow{\text{MixColumn}} \begin{array}{|c|c|c|c|} \hline b'_0 & b'_4 & b'_8 & b'_{12} \\ \hline b'_1 & b'_5 & b'_9 & b'_{13} \\ \hline b'_2 & b'_6 & b'_{10} & b'_{14} \\ \hline b'_3 & b'_7 & b'_{11} & b'_{15} \\ \hline \end{array}$$

where $i \in \{0, 4, 8, 12\}$, and all the operations are in $GF(2^8)$:

$$\begin{pmatrix} b'_i \\ b'_{i+1} \\ b'_{i+2} \\ b'_{i+3} \end{pmatrix} = \begin{pmatrix} 02_x & 03_x & 01_x & 01_x \\ 01_x & 02_x & 03_x & 01_x \\ 01_x & 01_x & 02_x & 03_x \\ 03_x & 01_x & 01_x & 02_x \end{pmatrix} \begin{pmatrix} b_i \\ b_{i+1} \\ b_{i+2} \\ b_{i+3} \end{pmatrix}$$

This operation is actually a multiplication of the column by the polynomial $c(x) = 03_x x^3 + 01_x x^2 + 01_x x + 02_x$ in $GF(2^8)^4$ modulo the polynomial $x^4 + 1$.

The AddRoundKey simply XORs the 128-bit subkey to the data. The subkey is generated by the key expansion.

The key expansion of Rijndael generates the subkeys from the key using a blend of the same operations used in the rest of Rijndael, and using the round constants $Rcon[i] = (02_x)^{i-1}$ (i starts at 1).

The round-function of the first and the last rounds are slightly different than in other rounds: In the first round there is an additional AddRoundKey operation before the round starts, and in the last round the MixColumn operation is eliminated.

When the key size is 128 bits the round-function is repeated 10 times. The number of rounds is higher when longer keys are used: there are 12 rounds when the key size is 192 bits, and 14 rounds when the key size is 256 bits.

3 Square Dual Ciphers

Given a cipher E that uses only operations of $EGF(2^8)$, we define the cipher E^2 by modifying the constants of E. All the operations that do not involve constants remain unchanged. There are only four operations that involve constants:

1. $f(x) = c \cdot x$.
2. $f(x) = c \oplus x$.
3. $L(x) = Ax$, where A is a constant matrix.
4. $S(x) = LookUpTable[x]$, where the look-up table is constant.

In the first two operations we change the constant c in E to be c^2 in E^2, where c^2 is the result of squaring c in $GF(2^8)$. In the affine transformation A is replaced by QAQ^{-1}, where in the case of Rijndael Q and Q^{-1} are:

$$Q = \begin{pmatrix} 1\,0\,0\,0\,1\,0\,1\,0 \\ 0\,0\,0\,0\,1\,0\,1\,1 \\ 0\,1\,0\,0\,0\,1\,0\,0 \\ 0\,0\,0\,0\,1\,1\,1\,1 \\ 0\,0\,1\,0\,1\,0\,0\,1 \\ 0\,0\,0\,0\,0\,1\,1\,0 \\ 0\,0\,0\,1\,0\,1\,0\,0 \\ 0\,0\,0\,0\,0\,0\,1\,1 \end{pmatrix} \qquad Q^{-1} = \begin{pmatrix} 1\,0\,0\,1\,0\,1\,0\,1 \\ 0\,1\,1\,1\,0\,0\,0\,0 \\ 0\,0\,0\,1\,1\,1\,0\,0 \\ 0\,1\,0\,1\,0\,0\,1\,0 \\ 0\,1\,0\,0\,0\,0\,0\,1 \\ 0\,1\,0\,1\,0\,0\,0\,0 \\ 0\,1\,0\,1\,0\,1\,0\,0 \\ 0\,1\,0\,1\,0\,1\,0\,1 \end{pmatrix} \tag{1}$$

From now on we denote QAQ^{-1} by A^2, as we will show later that for any x, $QAQ^{-1}x^2 = (Ax)^2$. A^2 of Rijndael is given in Appendix A. The matrices Q and Q^{-1} depend on the irreducible polynomial of $GF(2^8)$. The matrices above suit Rijndael's irreducible polynomial $x^8 + x^4 + x^3 + x + 1$.

Finally, we replace look-up tables of the form $S(x)$ with $S^2(x)$, where $S^2(x)$ is defined as $S^2(x) = QS(Q^{-1}x)$.

Remark: To make it clear, in our notation, E^2 is not $E(E(\cdot))$ nor $(E(\cdot))^2$, A^2 is not the matrix A multiplied with itself, and $S^2(x)$ is not $(S(x))^2$, nor $S(S(x))$.

In general terms, to specify E^2, we take the specifications of the cipher, raise all the constants in the cipher to their second power, replace matrices A by $A^2 = QAQ^{-1}$ and replace look-up tables $S(x)$ by $S^2(x) = QS(Q^{-1}x)$. If we take Rijndael as an example of E, the polynomial $03_x x^3 + 01_x x^2 + 01_x x + 02_x$ of the mix column operation is replaced by $05_x x^3 + 01_x x^2 + 01_x x + 04_x$.[1] As a result of the above replacements, the affine transformation $Ax + b$ is replaced by the affine transformation $A^2 x + b^2 = QAQ^{-1}x + b^2$.

[1] In $GF(2^8)$, $03_x^2 = 05_x$.

The key expansion consists of S boxes, XORs, and XORs with constants in $GF(2^8)$ (called $Rcon$) which are powers of 02_x. These operations are replaced by the replacement operations as mentioned above, with the $Rcon$ constants being replaced by their squares.

We will now show that E and E^2 are dual ciphers:

Theorem 1 For any K and P, ${E^2}_{K^2}(P^2) = (E_K(P))^2$.

In the context of this paper, the notation K^2, and P^2 denote the square operation of each byte of K and P (and similarly for any data block).

This theorem states that if P is the plaintext, K is the key and the result of encryption with cipher E is C, then the result of encrypting P^2 under the key K^2 with the cipher E^2 is necessarily C^2.

Proof Any Galois field is congruent to a Galois field of the form of $GF(q^m)$, where q is a prime. The number q is called the characteristic of the field. It is well known that for any a, $b \in GF(q^m)$ it follows that: $(a+b)^q = a^q + b^q$. In $GF(2^8)$: $(a+b)^2 = a^2 + b^2$. That actually means that squaring an element in $GF(2^8)$ is a linear operation, which can be applied by a multiplication by a binary matrix Q of size 8×8. We computed the matrix Q of Rijndael, and described it in Eq. (1). For those unfamiliar with finite fields, we describe a brief proof of this property, and how to compute such Q in other representations and other Galois fields in Appendix B. It follows that Q^{-1} is the matrix that takes out the square root of an element in $GF(2^8)$.

To complete the proof, it suffices to show that for each operation $f(x)$ in E, and the corresponding operation in E^2, which we denote in this proof by $f^2(x)$, it follows that $f^2(x^2) = (f(x))^2$:

1. $f(x,y) = x \oplus y$. In this case $f^2(x^2, y^2) = x^2 \oplus y^2 = (x \oplus y)^2 = (f(x,y))^2$.
2. $f(x) = x \oplus c$. By definition $f^2(x^2) = x^2 \oplus c^2 = (x \oplus c)^2 = (f(x))^2$.
3. $f(x,y) = x \cdot y$. In this case $f^2(x^2, y^2) = x^2 \cdot y^2 = (x \cdot y)^2 = (f(x,y))^2$.
4. $f(x) = x \cdot c$. By definition $f^2(x^2) = x^2 \cdot c^2 = (x \cdot c)^2 = (f(x))^2$.
5. $f(x) = x^c$. In this case $f^2(x^2) = (x^2)^c = (x^c)^2 = (f(x))^2$.
6. It is clear that replacing the order of elements after they are raised to their second power is equal to raising elements to their second power, and then replacing their order.
7. $f(x) = L(x) = Ax$. By definition $f^2(x^2) = L^2(x^2) = QAQ^{-1}x^2 = QAx = (Ax)^2 = (f(x))^2$, as Q is the matrix which corresponds to the squaring operation in $GF(2^8)$.
8. $f(x) = S(x) = LookUpTable[x]$. By definition

$$f^2(x^2) = S^2(x^2) = QS(Q^{-1}x^2) = QS(x) = (S(x))^2 = (f(x))^2.$$

$\blacksquare$

The cipher $E^4 = (E^2)^2$ is a dual cipher of E^2, and thus also of E. Moreover, all ciphers E^{2^i} (for all i), which are E, E^2, E^4, E^8, E^{16}, E^{32}, E^{64} and E^{128}, are all dual ciphers of each other (there are 8 such ciphers as $E^{2^8} = E$).

It is interesting to note that Rijndael have these 7 dual ciphers, independently of the key size, the block size, the number of rounds, and even the arrangement of operations in the cipher. These dual ciphers exist for any cipher whose all operations are $EGF(2^8)$ operations.

Note that it is possible to define a trivial square dual cipher for any cipher by taking a cipher E and defining E^2 which apply Q^{-1} on the plaintext and K, calls E, and then applies Q on the result. However, we are interested in non-trivial dual ciphers with different cores.

4 Modifying the Polynomial

An $EGF(2^8)$ cipher E can include multiplication modulo an irreducible polynomial. The irreducible polynomial in Rijndael is used for the inverse computation in the S box and also in the multiplications in the MixColumn operation. Several researchers asked why the irreducible polynomial of Rijndael was not selected to be primitive. There are 30 irreducible polynomials of degree 8, of which 16 are primitive. In our discussion it is irrelevant if the irreducible polynomial is primitive or not, due to the isomorphism of all fields of $GF(2^8)$. The isomorphism transformation that takes one description of a cipher under an irreducible polynomial $g(x)$ to another description with a different irreducible polynomial $\hat{g}(x)$ is linear, and therefore can be represented as a binary matrix R such that $y = R \cdot x$, where x is the vector representation of an element under Rijndael's $g(x)$ polynomial, and y is the representation under the new polynomial $\hat{g}(x)$. The matrix R is used in a similar way to the matrix Q of the square dual cipher. The change of operations in the cipher is also similar to the square dual cipher. Note that the x^2 operation there is equivalent to $Q \cdot x$, and the same proof of duality follows.

The R matrix is always of the form $R = (1, a, a^2, a^3, a^4, a^5, a^6, a^7)$, where the a^i's are computed modulo the irreducible polynomial $\hat{g}(x)$. Note that in Appendix B we show that the matrix Q is actually one of these matrices R. For each irreducible polynomial we can define its 8 square dual ciphers. Since there are 30 irreducible polynomials, we get that there are 240 dual ciphers for each $EGF(2^8)$ cipher.

For example, we describe one of these 240 dual ciphers of Rijndael: the irreducible polynomial of Rijndael is replaced by the primitive polynomial $x^8 + x^4 + x^3 + x^2 + 1$ (denoted in binary notation by $11D_x$). In this example, the R matrix is

$$R = \begin{pmatrix} 1&1&1&1&1&1&1&1 \\ 0&1&0&1&0&1&0&1 \\ 0&0&1&1&0&0&1&1 \\ 0&0&0&1&0&0&0&1 \\ 0&0&0&0&1&1&1&1 \\ 0&0&0&0&0&1&0&1 \\ 0&0&0&0&0&0&1&1 \\ 0&0&0&0&0&0&0&1 \end{pmatrix} \qquad R^{-1} = \begin{pmatrix} 1&1&1&1&1&1&1&1 \\ 0&1&0&1&0&1&0&1 \\ 0&0&1&1&0&0&1&1 \\ 0&0&0&1&0&0&0&1 \\ 0&0&0&0&1&1&1&1 \\ 0&0&0&0&0&1&0&1 \\ 0&0&0&0&0&0&1&1 \\ 0&0&0&0&0&0&0&1 \end{pmatrix}.$$

The inverse matrix R^{-1} takes an element of the dual cipher to Rijndael's representation. It is interesting to note that the affine matrix of the S box becomes lower triangular in this case:

$$\hat{A} = \begin{pmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 1&0&0&1&0&0&0&0 \\ 1&1&0&0&1&0&0&0 \\ 0&1&1&0&0&1&0&0 \\ 0&0&1&1&0&0&1&0 \\ 0&0&0&1&1&0&0&1 \end{pmatrix}.$$

Also, the constant 63_x in the S box becomes 64_x, and the coefficients 03_x, 02_x of the MixColumn operation are interchanged (i.e., to 02_x, 03_x). The coefficients $0B_x$, $0D_x$, 09_x, $0E_x$ are also interchanged in pairs to $0D_x$, $0B_x$, $0E_x$, 09_x. The *Rcon* constants $(02_x)^{i-1}$ are replaced by $(03_x)^{i-1}$. The full description of these 240 dual ciphers of Rijndael can be found in *The Book of Rijndaels* [2].

Thus, we conclude that the choice of the irreducible polynomial of Rijndael is arbitrary, and in particular, there is no advantage to selecting a primitive polynomial over the current polynomial of Rijndael.

5 Log Dual Ciphers

In this section we discuss dual ciphers, to which we call *log dual ciphers*. We actually describe a family of log dual ciphers, which differ slightly from each other.

Let g be a generator in $GF(2^8)$. Since the cipher works on elements of $GF(2^8)$ we can write any element x as an exponent of g, i.e., $x = g^i$, except for $x = 0$, which we define as $g^{-\infty}$. In a logarithmic notation we write: $\log_g x = i$, where $log_g 0 = -\infty$. In the log cipher we use the logarithm representation of the elements, instead of the polynomial representation used in the original description of the cipher.

Let x and y be elements of $GF(2^8)$, and let $i = \log_g x$, $j = \log_g y$.

We use the notation $E^{\log_g}$, or shortly $E^{\log}$, to denote the log dual cipher. The log dual cipher is defined by taking the specifications of the cipher, and replacing the following operations:

1. The operation $f(x, y) = x \oplus y$ is replaced by the operation $f^{\log}(i, j) = j + T(i-j) \pmod{255}$ or by $f^{\log}(i, j) = i + T(j-i) \pmod{255}$, where the Zech logarithm [21] $T(i)$ is defined as $T(i) = \log_g(g^i \oplus 1)$. In cases where $-\infty$ appears in $f^{\log}$, we define $f^{\log}(-\infty, j) = j$, and $f^{\log}(i, -\infty) = i$.
2. The operation $f(x) = x \oplus c$ is replaced by the operation $f^{\log}(i) = k + T(k-i) \pmod{255}$ where $k = \log_g c$.
3. The operation $f(x, y) = x \cdot y$ is replaced by the operation $f^{\log}(i, j) = i + j \pmod{255}$. If either x or y is $-\infty$, then the result is $-\infty$.
4. The operation $f(x) = x \cdot c$ is replaced by the operation $f^{\log}(i) = i + k \pmod{255}$, where $k = \log_g c$.
5. The operation $f(x) = x^m$ is replaced by the operation $f^{\log}(i) = i \cdot m \pmod{255}$. If $i = -\infty$ then the result is $-\infty$.
6. Replacement of the order of elements remains the same replacement of order of the elements.
7. The operation $S(x) = LookUpTable[x]$ is replaced by the operation $S^{\log}(i) = \log_g(S(g^i))$.
8. The linear transformation $L(x) = Ax$ is written as a polynomial $\sum a_i \cdot x^{2^i}$, and treated as a combination of exponentiations , multiplications, and additions.

The definition of $\log_x(0) = -\infty$ is made carefully, ensuring that this definition is consistent: When applying the operation $j + T(i-j)$, a $-(-\infty)$ might result as an argument to T. We define that $j + (-\infty) = -\infty$ (which corresponds to multiplication by 0 in the original cipher or to XOR of a value with itself), $-\infty \cdot c = -\infty$ (which corresponds to an exponentiation of 0 in the original cipher), $-\infty - (-\infty) = 0$ (which corresponds to $0 \oplus 0$, or to $x \oplus 0$), $i - (-\infty) \neq j - (-\infty)$ for $i \neq j$ (meaning that $-(-\infty)$ does not consume i). $T(-\infty) = 0$, $T(0) = -\infty$. $T(i - (-\infty)) = -(-\infty) + i$. Note that the $-(-\infty)$ is always a result of an application of T. Then, another $+(-\infty)$ is always waiting to cancel it (as j). Therefore, the result of the T operation is always a number or a $-\infty$.

The following theorem proposes that if P is the plaintext, K is the key and the result of encryption of P under the key K with cipher E is C, the result of encrypting $\log_g(P)$ under the key $\log_g(K)$ with the cipher $E^{\log}$ is necessarily $\log_g(C)$.

Theorem 2 Let g be a generator in $GF(2^8)$. For any K and P:

$$E^{\log}_{\log_g K}(\log_g P) = log_g(E_K(P)).$$

In the context of this paper $\log_g X$ denotes the log of each byte of X.

Proof It suffices to show that for each operation $f(x)$ in E, and the corresponding operation in $E^{\log}$, which we denote by $f^{log}(x)$, it follows that $f^{\log}(\log_g x) = \log_g(f(x))$.

1. $f(x,y) = x \oplus y$. By definition $f^{\log}(i,j) = j + T(i-j) = j + \log_g(g^{i-j} \oplus 1) = \log_g(g^j \cdot (g^{i-j} \oplus 1)) = \log_g(g^i \oplus g^j) = \log_g(x \oplus y) = \log_g(f(x,y))$.
2. $f(x) = x \oplus c$, in the same way as the previous item.
3. $f(x,y) = x \cdot y$. In this case $f^{\log}(i,j) = i + j = \log_g(g^{i+j}) = \log_g(x \cdot y) = \log_g(f(x,y))$.
4. $f(x) = x \cdot c$, in the same way as the previous item.
5. $f(x) = x^c$. In this case $f^{\log}(i) = i \cdot c = log_g(x^c) = \log_g(f(x))$.
6. It is clear that replacing the order of elements after their log-value is taken is equal to replacing the order of elements and then taking their log-value.
7. $f(x) = S(x) = LookUpTable[x]$. In this case, by definition of $f^{\log}$ it follows that: $f^{\log}(i) = S^{\log}(i) = \log_g(S(g^i)) = \log_g(S(x)) = \log_g(f(x))$.
8. $L(x) = Ax$. This follows from items 1,2,4, and 5.

The above equations hold also in the case that $-\infty$ is an argument. ■

Note that the non-linear part of the ByteSub transformation of Rijndael in the log dual cipher becomes very simple (and linear). The non-linear part is finding the multiplicative inverse of an element. This operation is replaced by negation in the log dual cipher:

$$x^{-1} \longrightarrow -i.$$

The T transformation is non-linear. It has interesting properties. Here are some of the properties of the T transformation:

1. $T(x) - T(-x) = x$
2. $T(2x) = 2T(x)$
3. $T(T(x)) = x$

Additional properties of $T(x)$ can be found in Appendix C.

The log of the square dual cipher has the following property:

$$\log_g(E^2_{K^2}(P^2)) = 2E^{\log}_{\log_g K}(\log_g P).$$

This holds since:

$$\log_g(E^2_{K^2}(P^2)) = \log_g((E_K(P))^2) = 2\log_g(E_K(P)) = 2E^{\log}_{\log_g K}(\log_g P).$$

Note that this property resembles the property of the log function of an exponent (square in this case).

Let E^2 be a square dual cipher of E. The connection between the log dual cipher of E^2, namely $E^{2^{\log}}$ and the log dual cipher of E, namely $E^{\log}$ is

$$E^{2^{\log}}_{2K}(2P) = 2E^{\log}_K(P),$$

since for the change of variables $K = \log \hat{K}$, and $P = \log \hat{P}$

$$E^{2^{\log}}_{\log_g \hat{K}^2}(\log_g \hat{P}^2) = \log_g(E^2_{\hat{K}^2}(\hat{P}^2)) = 2E^{\log}_{\log_g \hat{K}}(\log_g \hat{P}).$$

Note that $2P$, and $2K$ are byte-wise multiplication by 2 modulo 255.

Each one of the 240 mentioned representations of Rijndael has the same set of 128 log dual ciphers.

6 Self-Dual Ciphers

We mention that any cipher is trivially dual to itself. However, it is possible to find ciphers that are self-dual in a non-trivial way. One such interesting family of dual ciphers is square dual ciphers. Let E be a square self-dual cipher. It follows that:

$$(E_K(P))^2 = E_{K^2}(P^2).$$

This means that each constant is the square of itself. In $GF(2^8)$ it means that the constants are either 0 or 1.

If we take Rijndael as an example, we need to change the constant 63_x in the affine transformation in the S box to either 00_x or 01_x. We would also need to change the constants of the mix column operation. A possible alternative matrix for the mix column operation, whose entries consist of only 0's and 1's is:

$$M = M^{-1} = \begin{pmatrix} 1\,1\,1\,0 \\ 1\,1\,0\,1 \\ 1\,0\,1\,1 \\ 0\,1\,1\,1 \end{pmatrix}.$$

In the key expansion we need to change the round constant. Any selection of values from $\{0_x, 1_x\}$ can be made for the *Rcon* constants. There are various such selections that can still prevent related key attacks.

We can replace the affine transformation to a self-dual one. We can easily find 8 affine transformations that are self-squares: The matrix Q (shown in Eq. (1)) is the square of itself under our definition, since $Q^2 = Q(Q)Q^{-1} = Q$. The order of Q is 8, therefore, we can easily find 8 self-square affine transformations: Q, $Q \cdot Q$, $Q \cdot Q \cdot Q, \ldots, Q \cdot Q \cdot Q \cdot Q \cdot Q \cdot Q \cdot Q$ and $Q \cdot Q \cdot Q \cdot Q \cdot Q \cdot Q \cdot Q \cdot Q = I$. Notice that all the linear combinations with coefficients from $\{0_x, 1_x\}$ of these matrices, are also self-squares matrices. Therefore, there are 256 such self-square matrices. Detailed analysis shows that these are all the self-square matrices. Of these 256 matrices only 128 matrices are involutions.

A property of such self-dual cipher is that if all the bytes of the key and all the bytes of the plaintext are in $\{00_x, 01_x\}$ then so are all the bytes of the ciphertext.

Note that the notion of simple relations, presented in [13], is related to self-dual ciphers.

6.1 Higher Order Self-Dual Cipher

In Section 6 we introduced the square self-dual cipher. In a similar way we can define the 4'th power self-dual cipher. Let E be a 4'th power self-dual cipher. It follows that:

$$(E_K(P))^4 = E_{K^4}(P^4).$$

This means that each constant is the 4'th power of itself. There are 4 such elements: the elements 0 and 1, and the two elements of order 3 (which are g^{85}, g^{170}, where g is a generator). If all the bytes of the plaintext and key are chosen from the set of these four elements, then so are all the bytes of the ciphertext.

We need the affine transformation to be self-dual, and therefore: $A^4 = Q \cdot Q \cdot (A) \cdot Q^{-1} \cdot Q^{-1} = A$. We can see that Q, $Q \cdot Q$, $Q \cdot Q \cdot Q$, …, $Q \cdot Q \cdot Q \cdot Q \cdot Q \cdot Q \cdot Q$ and $Q \cdot Q \cdot Q \cdot Q \cdot Q \cdot Q \cdot Q \cdot Q = I$ solve it much like for the square self-dual cipher, along with all their linear combinations, with coefficients from $\{0_x, 1_x, g^{85}, g^{170}\}$ (which is $GF(2^2)$). The total number of linear combinations is 2^{16}, of which $3 \cdot 2^{13}$ are involutions.

For the 16'th power square self-dual cipher all the constants should be 0, 1, and all the 14 elements of orders 3, 5, and 15. If all the bytes of the plaintext and key are chosen from the set of these 16 constants, then so are all the bytes of the ciphertext. The 16'th power square self-dual matrices are all the linear combinations of the Q^i matrices, with coefficients from the above constants. The total number of 16'th power square self-dual matrices is 2^{32}, of which $7 \cdot 5 \cdot 3^2 \cdot 2^{22}$ matrices are involutions. Fortunately, Rijndael's matrix is none of these matrices.

6.2 Cryptanalysis of Self-Dual Ciphers

The self-dual property of a cipher can be used to mount an attack which reduces the complexity of exhaustive search by a factor of about 8 in the case above (or by a factor of the number of the self-duals in the more general case). For example, if the key size is 128 bit, exhaustive search takes 2^{128} operations, and the attack we propose requires about 2^{125} operations. If we consider the expected time to complete the attack, exhaustive search takes about 2^{127}, and our attack takes about 2^{124} operations.

It is interesting to note that the number of rounds of the cipher does not affect the complexity of this attack.

By using the following chosen plaintext attack the key can be discovered in 2^{125} operations using 8 chosen plaintexts.

The attack takes advantage of cycles of keys under the squaring operation: A cycle is a set of keys where each key is the square of its predecessor, i.e., $\{K', K'^2, \ldots, K'^{2^7}\}$, and where the square of the last element equals the first element : $K' = K'^{2^8}$. Note that the possible cycle lengths are 8, 4, 2, and 1.

1. Choose a plaintext P, and compute $P_i = P^{2^i}$, for $i = 0, \ldots, 7$.
2. Ask for the encryption of $P_0, \ldots, P_7$, and denote the corresponding ciphertexts by $C_0, \ldots, C_7$. For every i, compute $\hat{C}_i = (C_i)^{2^{-i}}$, where the square root is defined to be the operation that finds for every byte its square root in $GF(2^8)$ (there is only one square root for each value).
3. Choose one key K' in each cycle, and compute $C = E_{K'}(P_0)$. If $C = \hat{C}_i$ for some $i \in \{0, \ldots, 7\}$, K'^{2^i} is a candidate to be K. Otherwise, K is not one of $\{K'^{2^i}\}$.

An equality $C = \hat{C}_i$ in step 3 ensures that encryption of P_i under the key K'^{2^i} gives C_i: If $C = \hat{C}_i$, then $C^{2^i} = \hat{C}_i^{2^i} = C_i$. Therefore, $C^{2^i} = (E_{K'}(P_0))^{2^i} = C_i = E_K(P_0^{2^i}) = E^{2^i}{}_K(P_0^{2^i})$. From the self-duality property it follows that: $K = K'^{2^i}$ (or that this is a false-alarm).

Note that the correct key is always found by this method, since for the correct key K: $E_K(P_0) = C_0$. The self-duality property implies that this happens if and only if for any i, $E_{K^{2^i}}(P_0^{2^i}) = C_0^{2^i}$. For each cycle, for example, for $\{K', K'^2, \ldots, K'^{2^7}\}$, we test only one key. If this key is K, then we would find it on the first equation. If one of the other keys is K, then the corresponding equation holds. So by checking one key out of a cycle we cover the whole cycle.

We test about 8 keys for every trial encryption. It is easy to choose the keys K in such a way that we choose only one key out of each cycle of keys. Therefore, this attack finds the key in about 2^{125} time. In Appendix D we show how to enumerate the keys (choose only one of each cycle), and show that the total number of cycles, and thus, the maximal complexity of this attack, is $2^{125}+2^{61}+2^{30}+2^{15}$, using 8 chosen plaintexts. The average case complexity is $2^{124}+\varepsilon$ where $\varepsilon=2^{-4}+2^{-67}+2^{-98}$.

We note that a similar attack can be designed for higher order self-dual ciphers.

7 Application to Other Ciphers

Square [8], Scream [11], Anubis [3], Crypton [15] and Khazad [4] are all $EGF(2^8)$ ciphers. Thus, our results hold to these ciphers as well. Our work can be extended to include ciphers such as E2 [18], Camellia [1] and Safer++ [16].

The only operation in Safer++ that is not a $EGF(2^8)$ operation is addition modulo 256: $f(x,y)=x+y \pmod{256}$. For the square dual cipher, we can define $f^2(x^2,y^2)=(f(x,y))^2$. f^2 can be implemented by $f^2(x,y)=Qf(Q^{-1}x,Q^{-1}y)$. This results in a substitution table of size 2^{16}.

It should be noted that since Safer++ does not use $GF(2^8)$ multiplications or exponentiations, the irreducible polynomial is irrelevant. For such a cipher, we can create a wide range of dual ciphers by using any invertible binary matrix Q of size 8×8. The operation $f^Q(x)$ is defined as $f^Q(x)=Qf(Q^{-1}x)$. For operations with two parameters $f^Q(x,y)=Qf(Q^{-1}x,Q^{-1}y)$. A constant c is replaced by Qc. This change does not fundamentally change the differential [5] properties of such functions, since Q and Q^{-1} are linear and invertible.

If we take E2, and remove the initial and final transformations, the affine operation, and also change the v_{-1} value of the key scheduling to be composed only from 0's and 1's, then E2 is a self-dual cipher. That means that the attack on self-dual ciphers we present in this paper is also applicable to this variant.

8 Other Applications

A possible application of dual ciphers is for developing differential [5] or linear [17] attacks. In such cases the insight gained from the dual ciphers can be used to attack the dual cipher, an attack which can be easily transformed to the original. A possible example for such insight might be the simplification of the affine transformation in the S box to a triangular matrix (see Section 4), which reduces the effect of modifying bits in the input on the resultant output of this transformation.

An other interesting application of dual ciphers might be an optimization of the speed of the cipher, as in some cases the dual cipher might actually be faster to compute than the original cipher! For example, many ciphers include multiplications by constants. The Hamming weight and the size of the constant has implications on the implementation efficiency. Thus, finding a more efficient dual cipher might be a good optimization strategy. Also, in some cases encryption might be fastest using one dual cipher, and decryption be fastest using another dual cipher.

The existence of dual ciphers can also be used to protect implementation against fault-analysis [6] and power-analysis [14], by selecting a different dual cipher at random each time an encryption or decryption is desired.

9 Summary

In this paper we show how to write many different implementations of Rijndael using its various dual ciphers. We describe hundreds of non-trivial dual ciphers of Rijndael, many of them differ from Rijndael only by the replacement of constants. We also discuss an attack on self-dual ciphers.

We conclude that the irreducible polynomial of Rijndael is chosen arbitrarily, and that it is possible to replace the irreducible polynomial of Rijndael by any other irreducible or primitive polynomial without changing the strength of cipher, and even without changing the cipher itself.

Acknowledgments

We are pleased to thank Ronny Roth for the various discussions which helped improving the results of this paper and to John Kelsey for observing that dual ciphers may be used to prevent power analysis.

The work described in this paper has been supported by the European Commission through the IST Programme under Contract IST-1999-12324.

References

1. Kazumaro Aoki, Tetsuya Ichikawa, Masayuki Kanda, Mitsuru Matsui, Shiho Moria, Junko Nakajima, Toshio Tokita, *Camellia: A 128-Bit Block Cipher Suitable for Multiple Platforms - Design and Analysis*, submitted to NESSIE, 2000.
2. Elad Barkan, Eli Biham, *The Book of Rijndaels*, Available online on http://eprint.iacr.org, 2002.
3. Paulo S.L.M. Barreto, Vincent Rijmen, *The Anubis Block Cipher*, submitted to NESSIE, 2000.
4. Paulo S.L.M. Barreto, Vincent Rijmen, *The Khazad Legacy-Level Block Cipher*, submitted to NESSIE, 2000.
5. Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, Springer-Verlag, 1993.
6. Eli Biham, Adi Shamir, *Differential Fault of Secret-Key Cryptosystems*, Advances in Cryptology, proceedings of Crypto'97, Lecture Notes in Computer Science 1294, Springer-Verlag, pp. 513–525, 1997.
7. Antoon Bosselaers, Joan Daemen, Erik De Win, Bart Preneel, Vincent Rijmen, *The Cipher Shark*, proceedings of Fast Software Encryption '96, Lecture Notes in Computer Science 1039, Dieter Gollmann, Ed., Springer-Verlag, pp. 99–112, 1996.
8. Joan Daemen, Lars R. Knudsen, Vincent Rijmen, *The Block Cipher Square*, proceedings of Fast Software Encryption '97, Lecture Notes in Computer Science 1267, Eli Biham, Ed., Springer-Verlag, pp. 149–165, 1997.
9. Joan Daemen, Vincent Rijmen, *AES Proposal: Rijndael*, submitted to the Advanced Encryption Standard (AES) contest, 1998.
10. Niels Ferguson, Richard Schroeppel, Doug Whiting, *A Simple Algebraic Representation of Rijndael*, proceedings of Selected Areas in Cryptography, Lecture Notes in Computer Science 2259, Serge Vaudenay and Amr Youssef, Eds., Springer-Verlag, pp. 103–111, 2001.
11. Shai Halevi, Don Coppersmith, Charanjit Jutla, *Scream: a Software-Efficient Stream Cipher*, preproceedings of Fast Software Encryption 2002, pp. 190–204, 2002.
12. Thomas Jakobsen, Lars R. Knudsen , *The Interpolation Attack on Block Ciphers*, proceedings of Fast Software Encryption '97, Lecture Notes in Computer Science 1267, Eli Biham, Ed., Springer-Verlag, pp. 28–40, 1997.
13. Lars R. Knudsen, *Practically Secure Feistel Ciphers*, proceedings of Fast Software Encryption '93, Lecture Notes in Computer Science 809, Ross Anderson, Ed., Springer-Verlag, pp. 211–221, 1994.
14. Paul Kocher, Joshua Jaffe, Benjamin Jun, *Differential Power Analysis*, Advances in Cryptology, proceedings of Crypto'99, Lecture Notes in Computer Science 1666, Springer-Verlag, pp. 388–397, 1999.

15. Chae Hoon Lim, *Crypton: a new 128-bit Block Cipher - Specifications and Analysis*, submitted to the Advanced Encryption Standard (AES) contest, 1998.
16. James L. Massey, Gurgen H. Khachatrian, Melsik K. Kuregian, *Nomination of Safer++ as Candidate Algorithm for the New European Schemes for Signatures, Integrity, and Encryption(NESSIE)*, submitted to NESSIE, 2000.
17. Mitsuru Matsui, *Linear cryptanalysis method for DES cipher*, Advances in Cryptology, proceedings of Eurocrypt'93, Lecture Notes in Computer Science 765, T. Helleseth, Ed., Springer-Verlag, pp. 386–397, 1994.
18. Nippon Telegraph and Telephone Corporation, *AES Proposal: E2*, submitted to the Advanced Encryption Standard (AES) contest, 1998.
19. National Institute of Standards and Technology, *FIPS-197:Advanced Encryption Standard*, Federal Information Processing Standard, FIPS-197, 2001.
20. Ronald L. Rivest, Adi Shamir, Leonard Adleman, *A Method for Obtaining Digital Signatures and Public Key Cryptosystems*, Communications of the ACM, 21(2):120–126, 1978.
21. Kenneth H. Rosen (Ed. in chief), *Handbook of Discrete and Combinatorial Mathematics*, CRC Press, 2000.
22. Serge Vaudenay, *Alert on Non-Linearity: Linearities in RIJNDAEL, KASUMI,...*, presented in the rump session of Crypto'01.

A The Affine Transformation of Rijndael and Rijndael^2

$$A = \begin{pmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{pmatrix} \qquad A^2 = \begin{pmatrix} 0&1&1&0&0&1&0&0 \\ 1&0&1&0&0&0&1&1 \\ 1&0&0&1&1&0&1&1 \\ 1&0&0&0&1&1&0&0 \\ 0&1&0&1&0&0&0&0 \\ 0&0&1&0&0&1&0&0 \\ 1&1&0&1&0&0&0&1 \\ 0&1&0&0&1&0&0&1 \end{pmatrix}$$

B The Matrix Q

We prove that from the equation $(a+b)^2 = a^2 + b^2$ it follows that the square operation can be done by multiplication by a matrix. While doing it we discover how to compute such a matrix Q for any irreducible polynomial as vectors of $GF(2)^8$.

Given a vectorial representation of an element $a \in GF(2^8)$, we can write the vector as $a = \sum_{i=1}^{8} a_i \cdot e_i$, where e_i is the i'th element of the basis (i.e., the vectors whose i'th bit is 1, and all the other bits are 0). a_i is the i'th bit of the vector a. Note that $a_i = a_i^2 \in GF(2^8)$, since a_i is either 0 or 1.

So $a^2 = (\sum_{i=1}^{8} a_i \cdot e_i)^2 = \sum_{i=1}^{8} a_i^2 \cdot e_i^2 = \sum_{i=1}^{8} e_i^2 \cdot a_i = Q \cdot a$, where Q is the matrix whose i'th column ($i \in \{0, \ldots, 7\}$) contain the vectorial representation of e_i^2. The matrix Q of Rijndael is given in Eq. (1). It can be seen there that the columns are powers of $4 (\equiv x^2)$, where the first element $1 = 4^0$, the next is 4^1, then 4^2, 4^3, etc.

C Properties of the $T(x)$ Transformation

Theorem 3 The following properties hold for the $T(x)$ transformation:

1. $T(x) - T(-x) = x$

2. $T(2x) = 2T(x)$ (therefore, $\forall i, T(2^i x) = 2^i T(x)$)
3. $T(T(x)) = x$
4. Let $g \triangleq g'^y, yT_g(x) = T_{g'}(yx)$
5. $T_g = T_{g^{2^i}}$
6. $T(x) = -T(-T(-x))$
7. $T(85) = 170$, $T(170) = 85$, and if $T(x) = x/2$ then $x \in \{85, 170\}$. Note that $85/2 \equiv 170 \pmod{255}$
8. $T(0) = -\infty$.
9. $T(-\infty) = 0$.
10. $T(x) = T(x \pm 255)$ - The cycle size of T is 255.

Proof Proofs of the properties:

1. $T(x) - T(-x) = x$. Since the addition is commutative it follows that: $j + T(i-j) = i + T(j-i) \Longrightarrow T(i-j) - T(j-i) = i - j$. If we substitute $x = i - j$ we get the first property.
2. $T(2x) = \log_g(g^{2x} + 1) =$ Using the equation $(a+b)^2 = a^2 + b^2$ we get
$$= \log_g((g^x+1)^2) = 2\log_g(g^x + 1) = 2T(x)$$
3. $yT_g(x) = y\log_g(g^x + 1) = y\log_g(g'^{yx} + 1) = log_{g'}(g'^{yx} + 1) = T_{g'}(yx)$
4. $T(T(x)) = \log_g(g^{\log_g(g^x+1)} + 1) = \log_g(g^x + 1 + 1) = \log_g(g^x) = x$
5. Combining properties 2 and 4. from 2: $T_g(2^i x) = 2^i T_g(x) =$ (from 4)$= T_{g'}(2^i x)$. When $g = g'^{2^i}$. Thus we get the property. $T_g = T_{g^{2^i}}$
6. Combining 1 and 3. from 1: $-T(-T(-X)) =$ (We substitute$y = T(-x)$ $= -T(-y) =$ (by property 1) $= y - T(y) =$(substitute back)$= T(-x) - T(T(-x)) = T(-x) - (-x) = T(-x) + x =$(property 1)$= T(x)$
7. First we prove that if there is x such that $T(x) = x/2$, then $x = 170$ or $x = 85$. After that, we show that there are such $x$es, and $T(170) = 85$, and $T(85) = 170$.
Claim 1: if $T(x) = x/2$ then $x = 170$. Proof: Let g be a generator. Then, under the assumption $y \triangleq T(x) = x/2 \Rightarrow x = 2y$. $T(x) = y \quad \Rightarrow \quad 1 = g^y + g^x$, and $g^y + 1 = g^x$. $1 = g^y + g^x = g^y(g^{x-y} + 1) = g^y(g^y + 1) = g^y g^x = g^{y+2y} = 1$ Therefore $3y = 0 \pmod{255}$. There are only two such elements, which are $y = 170$ or $y = 85$.
Claim 2: $T(85) = 170$ (And using property 3 it follows that $T(170) = 85$)
Let $y = T(85)$. Therefore $(*) g^{85} + 1 = g^y \Rightarrow (g^{85}+1)^4 = (g^y)^4$. The order of g is 255 since it is a generator. Therefore the order of g^{85} is $255/\gcd(255, 85) = 3$. $\Rightarrow \quad g^{85} + 1 = (g^y)^4 = g^y$. Therefore the order of g^y is either 1 or 3. It cannot be 1 since it would result that $g^{85} = 0$. Therefore the order is 3. In $GF(2^8)$ there are two elements with order 3. One of them is g^{85} as we already know. g^y must be a different element than g^{85}. Otherwise we get a contradiction in (*), since it becomes $1 = 0$. Therefore, g^y is the other element of order 3. This element is g^{170}, since $255/\gcd(170, 255) = 3$.
8. $T(0) = \log_g(g^0 + 1) = \log_g(0) = -\infty$.
9. $T(-\infty) = \log_g(g^{-\infty} + 1) = \log_g(1) = 0$.
10. $T(x) = \log_g(g^x + 1) = \log_g(g^{x \pm 255} + 1) = T(x \pm 255)$ That is true since g - The cycle of T is 255.

$\blacksquare$

Also if we define $S(x) = T(-x)$, we find the following properties:

1. $S(x) - S(-x) = -x$
2. $S(2x) = 2S(x)$
3. $S(S(S(x))) = x$
4. $S(-S(x)) = -x$
5. Let $g \triangleq g'^y, yS_g(x) = S_{g'}(yx)$

6. $S_g = S_{g^{2^i}}$
7. $S(170) = 170$, $S(85) = 85$, and there is no other x that satisfies S(X)=X.
8. $S(x) = S(x \pm 255)$. The cycle of S is 255.

The table of $T(x)$ with the generator 03_x is described in Table 1.

$T[x] =$

+	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	$-\infty$	25	50	223	100	138	191	112	200	120	21	245	127	99	224	33
16	145	68	240	92	42	10	235	196	254	1	198	104	193	181	66	45
32	35	15	136	32	225	179	184	106	84	157	20	121	215	31	137	101
48	253	197	2	238	141	147	208	63	131	83	107	82	132	186	90	55
64	70	162	30	216	17	130	64	109	195	236	103	199	113	228	212	174
80	168	160	59	57	40	170	242	167	175	203	62	209	19	158	202	176
96	251	190	139	13	4	47	221	74	27	248	39	58	161	71	126	246
112	7	76	166	243	214	122	164	153	9	43	117	183	180	194	110	12
128	140	239	69	56	60	250	177	144	34	46	5	98	128	52	218	150
144	135	16	217	53	206	188	143	178	226	119	201	159	169	41	93	155
160	81	108	65	182	118	227	114	87	80	156	85	211	229	232	79	88
176	95	134	151	37	124	29	163	123	38	249	61	204	149	219	97	6
192	247	28	125	72	23	49	26	75	8	154	94	89	187	207	148	205
208	54	91	241	171	78	233	116	44	67	146	142	189	252	102	237	3
224	14	36	152	165	77	172	231	230	173	213	244	22	73	222	51	129
240	18	210	86	115	234	11	111	192	105	185	133	96	220	48	24	—

and

$$T[-\infty] = 0.$$
$$T[i - (-\infty)] = -(-\infty) + i$$

Table 1. The Table $T(x)$ with the generator 03_x with the irreducible polynomial $11B_x$ (Rijndael).

D How to Enumerate the Keys of Self-Dual Ciphers

First thing to notice, is that square operation in $GF(2^8)$ organizes the elements in groups of different sizes. When taking an element a, it is in the group: $\{a, a^2, a^4, a^8, a^{16}, a^{32}, a^{64}, a^{128}\}$. It does not matter which element of this group that we choose as a, we would still get the same elements in the group. When we look at the number of (different) elements in the group, we know that this number is limited by 8, as in $GF(2^8)$ for any element a, $a^{256} = a$. Elements with order 255 will therefore organize themselves in groups of 8. This also follows for elements of orders: 17, 51 and 85. So in total we get 30 groups of 8 elements each. Elements with order 15 will organize themselves in groups of 4 elements. This is because if a is of order 15, then $a^{16} = a$. This is also true for elements of the order 5. In total we get 3 groups of size 4. There are two elements of order 3 which fall into one group of size 2. The remaining two elements are 0 and 1 that each one of them is in a group of size 1. This is summarized in Table 2.

This cycles induce cycles on the keys. For each key, the length of the cycle is the maximal length of the cycles of its bytes. We want to find the minimal subset of keys covering all the cycles, i.e., a set of exactly one key from each cycle of $\{K, K^2, \ldots, K^{2^7}\}$.

To find them, use the following algorithm:

1. Output a representative from each cycle of keys that has an element of order 8 in at least one of its bytes.

Group Size	Number of Groups	Total
1	2	2
2	1	2
4	3	12
8	30	240
Total	36	256

Table 2. The element cycles under the squaring operation.

2. Then, output a representative from each cycle of keys that has an element of order 4 in at least one of its bytes, but has no byte that belongs to a cycle of higher order.
3. Then, output a representative from each cycle of keys that has an element of order 2 in at least one of its bytes, but has no byte that belongs to a cycle of higher order.
4. Finally, output all the keys with cycle of size 1, in all their bytes.

It can be easily verified that the algorithm outputs exactly one representative key from each cycle.

The following algorithm outputs all the representative keys that have at least one byte that belongs to a cycle of size $c > 1$, such that there is no byte that belongs to a cycle with size higher than c.

This algorithm holds for $c \in \{2, 4, 8\}$:

For each byte $i = 0 \ldots 15$, For each cycle of size c:

1. fix the value of byte i to be an element of the cycle.
2. Output all the possibilities of keys such that
   (a) for bytes $j \in \{0, \ldots, i-1\}$, choose their values as all the combinations of bytes that belong to cycles of size smaller than c.
   (b) for bytes $j \in \{i+1, \ldots, 15\}$, choose their values as all the possible combinations of bytes that belong to cycles of size smaller or equal to c.

The algorithm for $c = 1$ is: output all the 2^{16} combinations of 16 bytes of $\{0, 1\}$ (whose order is 1).

The number of cycles of keys of each size is summarized in Table 3.

Key Cycle Order	Number of Keys in cycles
8	$\sum_{i=1}^{16} 30 \cdot (16)^{i-1} \cdot (256)^{16-i} = 2^{125} - 2^{61}$
4	$\sum_{i=1}^{16} 3 \cdot (4)^{i-1} \cdot (16)^{16-i} = 2^{62} - 2^{30}$
2	$\sum_{i=1}^{16} 1 \cdot (2)^{i-1} \cdot (4)^{16-i} = 2^{31} - 2^{15}$
1	2^{16}
Total	$(2^{125} - 2^{61}) + (2^{62} - 2^{30}) + (2^{31} - 2^{15}) + 2^{16} =$ $= 2^{125} + 2^{61} + 2^{30} + 2^{15}$

Table 3. The key cycles under the squaring operation.