Integral Cryptanalysis on reduced-round Safer++

– A way to extend the attack? –*

Gilles Piret and Jean-Jacques Quisquater

Université catholique de Louvain, Crypto Group
Place du Levant, 3 1348 Louvain-la-Neuve, Belgium
{piret, jjq}@dice.ucl.ac.be – http://www.dice.ucl.ac.be/crypto

Abstract. In this paper we describe an integral distinguisher over 2 rounds of Safer++. It allows a practical attack against 3 rounds of Safer++$_{128}$, as well as attacks on 4 rounds of Safer++$_{128}$ and Safer++$_{256}$ (without the last key addition layer), under the chosen-plaintext hypothesis. These results achieve much lower complexity than the currently known best attacks on Safer++, namely weak-key linear cryptanalysis by Nakahara[9]. As a side result, we prove that the byte-branch number of the linear transform of Safer++ is 5.
We also discuss a way for further research in order to extend integral cryptanalysis.

1 Introduction

The integral cryptanalysis (or square attack) was first presented as a dedicated attack when J. Daemen, L. Knudsen, and V. Rijmen published the SQUARE algorithm[2]. Square attacks were then applied to reduced-round Rijndael, which is not surprising, as its structure is very close to the one of Square. Four years later however, S. Lucks applied the square technique to a non-square-like cipher, namely Twofish[7]. This way he managed to break up to eight rounds of Twofish, which is at present the best known attack on it. Since then, it was applied to many other algorithms: IDEA[5], Camellia[10], Skipjack[4], Misty[6],...

In this paper, we will apply the integral cryptanalysis technique to the Safer++ algorithm. Although the diffusion layer of this cipher is not very strong (more precisely, it has a small branch number), the best known attack on it is a linear cryptanalysis against weak-key classes breaking 3 rounds of Safer++$_{128}$ (the 128-bit keys version) for a fraction of 2^{-6} of the

* The authors recently learned about better attacks on Safer++ than theirs[1]: Biryukov and al. are able to attack up to 5.5 out of 7 rounds. These results will probably be published in the near future.

keys, and 4 rounds of Safer++$_{256}$ (the 256-bit keys version) for a fraction of 2^{-11} of the keys(see [9]). We present attacks on 3 rounds of Safer++$_{128}$ and 4 rounds of Safer++$_{256}$ performing much better than those of [9], and an attack on 4 rounds of Safer++$_{128}$. We must however emphasize that our attacks works only under the *chosen*-plaintext assumption, contrary to linear cryptanalysis that only requires *known*-plaintexts.

Finally, we discuss a way of improving integral cryptanalysis in order to attack more rounds. It implies to solve a problem whose resolution will need further research.

2 Some Definitions regarding Integral Cryptanalysis

In this paper, we will use the notion of "multiset" to describe an n-bit data channel (roughly speaking, a multiset is a set whose elements may appear several times). A **n-bit multiset** is a multiset whose elements belong to the set $\{0,1\}^n$. The following definitions are fundamental regarding to integral cryptanalysis:

- A n-bit multiset with $k \cdot 2^n$ entries is said **active** if any value in $\{0,1\}^n$ is found exactly k times.
- A multiset is said **passive** if it contains only one fixed value.
- A multiset $\{x_i\}_{i=0..2^n-1}$ is said to be **balanced** with respect to some group operation whenever

$$\sum_{i=0}^{2^n-1} x_i = 0$$

(where $\sum$ and 0 refer to the considered group operation).

Note the following properties:

- An active multiset is balanced.
- A passive multiset is balanced.
- The sum of two balanced multisets is a balanced multiset.

3 Description of Safer++

The Encryption Algorithm. The algorithm makes frequent use of addition mod 256, denoted by $\boxplus$. It also uses exclusive or, denoted by $\oplus$. Safer++$_{128}$ has 7 rounds, while Safer++$_{256}$ is compound of 10 rounds. One round is depicted in Fig. 1. It is made out of four layers: successively,

one key addition layer, one s-boxes layer (L and X are two 8×8 s-boxes, that are inverses of each other), another key addition layer, and finally the PHT diffusion layer. We denote the composition of the first three layers by γ, and the fourth layer by θ. We sometimes index them by the number r of the round: γ_r, θ_r. Note that both key addition layers alternate the $\boxplus$ and $\oplus$ operations. We denote by $F_{r,p}$ the action of γ restricted to the p^{th} byte at round r:

$$\begin{cases} F_{r,p}(x) = X(x \oplus K_{2r-1}^{(p)}) + K_{2r}^{(p)} & \text{if } p \equiv 0,1 \mod 4 \\ F_{r,p}(x) = L(x + K_{2r-1}^{(p)}) \oplus K_{2r}^{(p)} & \text{if } p \equiv 2,3 \mod 4 \end{cases} \quad (1)$$

where $K_i^{(p)}$ denotes the p^{th} byte of the i^{th} subkey. Note that the 7 (resp. 10) rounds are followed by one final key addition layer.

The Key Schedule of Safer++$_{128}$. Let $K = (K^{(1)}, K^{(2)}, ..., K^{(16)})$ denotes the key. To understand our attack, it is sufficient to know the following properties of the key schedule (the reader interested by a detailed description can refer to [8]):

- A 17^{th} byte is computed as:

$$K^{(0)} = \bigoplus_{i=1}^{16} K^{(i)} \quad (2)$$

- $K_1^{(p)} = K^{(p)} (p = 1...16)$
- $K_i^{(p)}$ can be computed as a function of $K^{(i+p-1 \mod 17)}$.

We use the notation $K_i^{(p)} \approx K_j^{(q)}$ to denote the fact that two subkey bytes are derived from the same master key byte.

About the Linear Transform θ. Whereas the designers of SAFER++ claimed it to have a high diffusion PHT layer, it was proven in [9] that its byte branch number was ≤ 7. The byte branch number is defined as follows (see [3]):

Definition 1. *The **byte branch number** $B(.)$ of a linear transform F is defined as*

$$B(F) = min_{a \neq 0}\{W(a) + W(F(a))\}$$

where $W(a)$ is the byte Hamming Weight of the vector a (i.e. the number of non-zero bytes in a).

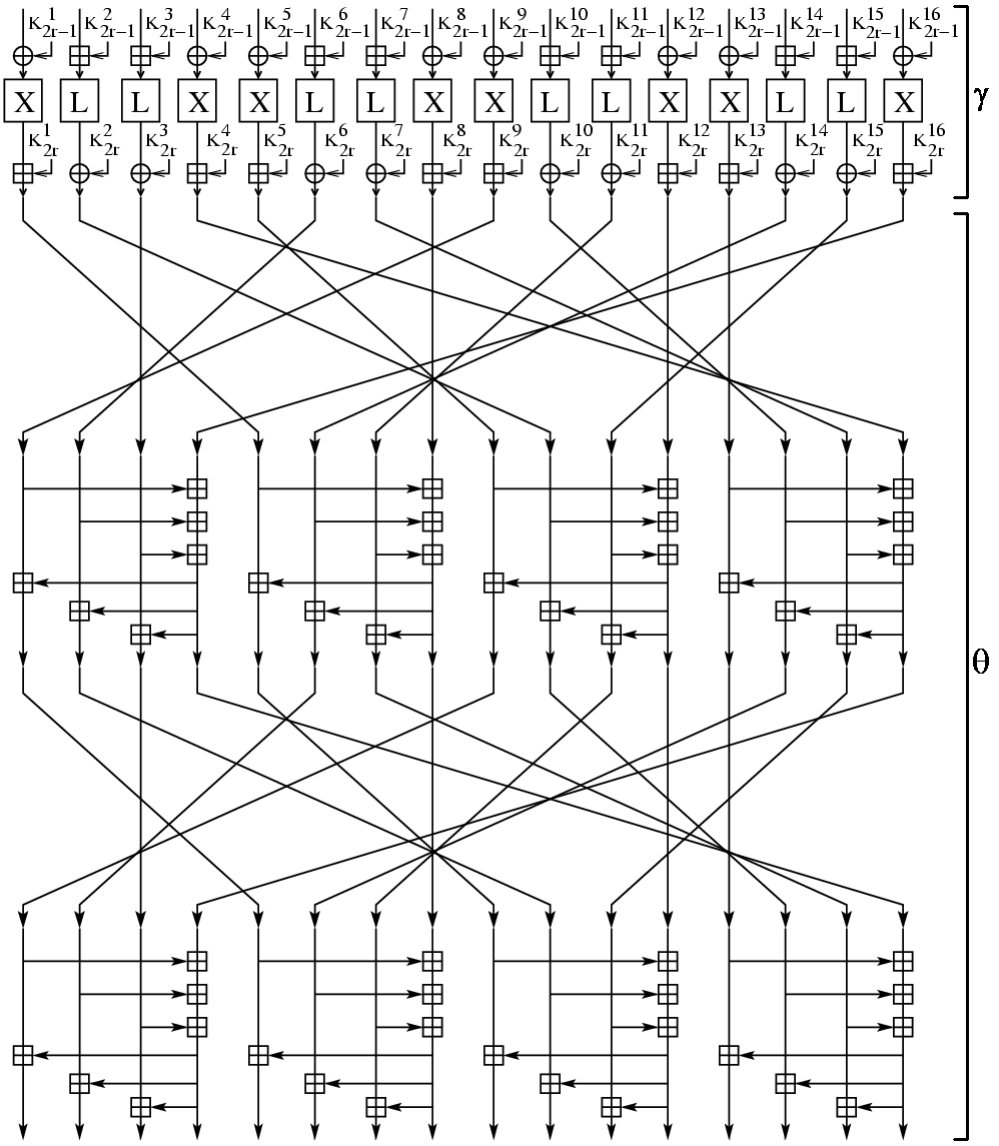

Fig. 1. One round of Safer++

We prove that the byte branch number of θ is in fact precisely 5:

- In order to prove that $B(\theta) \leq 5$, it is sufficient to exhibit an input a achieving this bound. For example,

$$a = 00008080000000000000000000000000_x$$

  implies

$$\theta(a) = 80800080000000000000000000000000_x$$

  And $W(a) = 2$, while $W(\theta(a)) = 3$.
- It is not difficult to prove that there is no input a achieving $B(\theta) = 4$. As the demonstration is purely technical, we give it in appendix A.

It is often a good idea to try integral attacks against ciphers with such a relatively weak diffusion layer. Indeed, such attacks can exploit weak diffusions, while the cryptographic strength of the s-boxes and key addition used are of no relevance against them.

4 A distinguisher on 2 rounds of Safer++

Consider a set of plaintexts of the form

$$\{(A, B, a_i, D; E, F, G, H; I, J, K, L; b_i, N, O, P)\}_{i \in \{0,\ldots,2^{16}-1\}}, \quad (3)$$

where $\{(a_i, b_i)\}_{i \in \{0,\ldots,2^{16}-1\}}$ is a 16-bit active multiset, the other bytes being constants.

As all functions $F_{r,p}$ are permutations, the action of γ does not change the active/passive properties. Thus the data after it can be described by (3) as well. Without loss of generality, we may assume the constants to be 0. Then at the end of first round, we obtain:

$$\begin{array}{c}\{2a_i + b_i, 2a_i + b_i, 4a_i + b_i, 2a_i + b_i; 2a_i + b_i, a_i + 2b_i, a_i + b_i, a_i + b_i; \\ a_i + b_i, 2a_i + b_i, a_i + 2b_i, a_i + b_i; a_i + 4b_i, a_i + 2b_i, a_i + 2b_i, a_i + 2b_i\}\end{array} \quad (4)$$

After the γ layer of the second round, the data can be written as:

$$\begin{array}{rl}\{X_i\}_i = \{&F_{2,1}(2a_i + b_i), F_{2,2}(2a_i + b_i), F_{2,3}(4a_i + b_i), F_{2,4}(2a_i + b_i); \\ &F_{2,5}(2a_i + b_i), F_{2,6}(a_i + 2b_i), F_{2,7}(a_i + b_i), F_{2,8}(a_i + b_i); \\ &F_{2,9}(a_i + b_i), F_{2,10}(2a_i + b_i), F_{2,11}(a_i + 2b_i), F_{2,12}(a_i + b_i); \\ &F_{2,13}(a_i + 4b_i), F_{2,14}(a_i + 2b_i), F_{2,15}(a_i + 2b_i), F_{2,16}(a_i + 2b_i)\}\end{array} \quad (5)$$

Finally, the first byte of the output can be expressed as:

$$\begin{aligned}4F_{2,5}(2a_i+b_i)+2F_{2,2}(2a_i+b_i)+2F_{2,15}(a_i+2b_i)+2F_{2,12}(a_i+b_i)+\\ 2F_{2,14}(a_i+2b_i)+F_{2,1}(2a_i+b_i)+F_{2,11}(a_i+2b_i)+F_{2,8}(a_i+b_i)+\\ 2F_{2,3}(4a_i+b_i)+F_{2,9}(a_i+b_i)+F_{2,6}(a_i+2b_i)+F_{2,16}(a_i+2b_i)+\\ F_{2,13}(a_i+4b_i)+F_{2,10}(2a_i+b_i)+F_{2,7}(a_i+b_i)+F_{2,4}(2a_i+b_i)\end{aligned} \tag{6}$$

We can summarize the successive states through these 2 rounds by: (3) $\stackrel{\gamma_1}{\rightarrow}$ (3) $\stackrel{\theta_1}{\rightarrow}$ (4) $\stackrel{\gamma_2}{\rightarrow}$ (5) $\stackrel{\theta_2}{\rightarrow}$ (6).

As all bytes of (5) are active, they satisfy $\sum_i X_i^{(j)} \equiv 0 \pmod{256}$. Thus so does the sum (6); this is the distinguishing criteria. It is satisfied with probability 2^{-8} for a random permutation. As the same reasoning applies to any byte of the output, by checking whether all of them are balanced we obtain a distinguisher that is satisfied with probability 2^{-128} for a random permutation.

Note finally that there are other distinguishers similar to the one we described, but corresponding to sets of 2^{16} plaintexts different from the one described in (3).

5 Three attacks on Safer++ using this distinguisher

5.1 An attack on 3 rounds of Safer++$_{128}$

In this section we present an attack on 3 rounds of Safer++, without the final key addition layer however. We make a straightforward use of the distinguisher we just described; it is applied to the two first rounds.

We consider 2^{16} chosen plaintexts following pattern (3). Then we guess the subkey bytes of K_5 and K_6 one after the other, and use the fact that the data we obtain by decrypting θ_3 and γ_3 must be balanced, in order to eliminate a large proportion of bad guesses. More precisely, the attack works this way:

1. Ask for encryption of 2^{16} plaintexts of the form (3).
2. – Go through all values of $(K_5^{(1)}, K_6^{(1)}) \approx (K^{(5)}, K^{(6)})$; for each of them compute the first byte of the data after round 2, corresponding to all 2^{16} ciphertexts. Check balance of these bytes. Out of the 2^{16} candidates, about 2^8 pass the test.
   – Guess successively on $K^{(7)}$, $K^{(8)}$, ..., $K^{(0)}$, $K^{(1)}$, ..., $K^{(5)}$. Note that $K^{(i)} \approx K_5^{(i-4)} \approx K_6^{(i-5)}$ $(i = 6...16)$ and $K^{(i)} \approx K_5^{(i+13)} \approx$

$K_6^{(i+12)}$ $(i = 0...4)$. Thus each guess allows us to check that one more byte is balanced. After checking balance, the mean number of candidates for the part of the key already guessed is 2^8. Note also that $K^{(4)} \approx K_6^{(16)}$ is obtained "for free" after all other key bytes have been guessed, by relation (2). Thus after checking balance of the last byte we can expect to have a very small number of remaining candidates (about 1 bad key along with the right one).

3. If necessary, exhaustive search allows to distinguish the right key from the few wrong ones.

The most time-consuming step of this algorithm is the first one. Thus global time complexity is about 2^{16} encryptions. Memory requirements are of the same order of magnitude. This attack is thus quite practical.

5.2 An attack on 4 rounds of Safer++$_{128}$

In this section we adapt our previous attack, if one more round is added at the beginning of the cipher. In other words, the distinguisher is now applied to the two middle rounds. Thus we must find a set of 2^{16} plaintexts such that the corresponding inputs to the second round look like (3). Applying the inverse permutation layer to the 2^{16} data of pattern (3), and assuming constants to be 0, we obtain:

$$\{(0, 0, a_i + b_i, -4a_i - b_i; 0, b_i, a_i, 0; b_i, a_i, 0, 0; a_i + b_i, 0, 0, -a_i - 4b_i)\}_i \quad (7)$$

Knowing that $K_1^{(i)} \approx K_2^{(i-1)}$ $(i = 2...16)$, it is easy to see that the corresponding set of plaintexts can be obtained by guessing on $13 \cdot 8 = 104$ key bits. More precisely, $K_1^{(2)} \approx K_2^{(1)}$, $K_1^{(12)} \approx K_2^{(11)}$, and $K_1^{(15)} \approx K_2^{(14)}$ do not need to be guessed (remember that the 0's in (7) could as well be replaced by any constant).

In order to test the output of the third round, we first apply to the ciphertext the inverse θ_4^{-1} of the permutation layer. Thanks to the guess made on 13 key bytes, it is then possible to compute 8 bytes at the end of the third round. Table 1 illustrates this. As for a false key guess the test on each byte has a probability 2^{-8} to succeed, a proportion of about 2^{-64} of the bad guesses will not be discardable. Let us detail how the complete attack works:

1. Ask for encryption of all 2^{64} plaintexts of the form

$$\{(A, B, \alpha_i^{(1)}, \alpha_i^{(2)}; C, \alpha_i^{(3)}, \alpha_i^{(4)}, D; \alpha_i^{(5)}, \alpha_i^{(6)}, E, F; \alpha_i^{(7)}, G, H, \alpha_i^{(8)}\}_i \quad (8)$$

where $A, B, .., H$ denote arbitrary constants.

Table 1. Bytes that can be computed at the end of round 3 (attack on Safer++$_{128}$). The key bytes in bold are those that were guessed at step 2 of the attack.

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$K_7^{(i)} \approx$	$\mathbf{K^{(7)}}$	$\mathbf{K^{(8)}}$	$\mathbf{K^{(9)}}$	$\mathbf{K^{(10)}}$	$\mathbf{K^{(11)}}$	$K^{(12)}$	$\mathbf{K^{(13)}}$	$\mathbf{K^{(14)}}$	$K^{(15)}$	$\mathbf{K^{(16)}}$	$K^{(0)}$	$\mathbf{K^{(1)}}$	$K^{(2)}$	$\mathbf{K^{(3)}}$	$\mathbf{K^{(4)}}$	$\mathbf{K^{(5)}}$
$K_8^{(i)} \approx$	$\mathbf{K^{(8)}}$	$\mathbf{K^{(9)}}$	$\mathbf{K^{(10)}}$	$\mathbf{K^{(11)}}$	$K^{(12)}$	$\mathbf{K^{(13)}}$	$\mathbf{K^{(14)}}$	$K^{(15)}$	$\mathbf{K^{(16)}}$	$K^{(0)}$	$\mathbf{K^{(1)}}$	$K^{(2)}$	$\mathbf{K^{(3)}}$	$\mathbf{K^{(4)}}$	$\mathbf{K^{(5)}}$	$\mathbf{K^{(6)}}$
	✓	✓	✓	✓			✓							✓	✓	✓

2. Guess on 104 key bits, not including $K^{(2)}$, $K^{(12)}$, and $K^{(15)}$.
3. Using the key bits just guessed, compute 2^{16} plaintexts from (7).
4. From the corresponding ciphertexts, compute the 8 bytes 1,2,3,4,7,14,15,16 at the end of round 3.
5. Sum up all 2^{16} 8-byte data obtained, and checks balance of these 8 bytes.
6. If the test succeed, checks whether it is the right key e.g. by exhaustive search[1].
7. If it is not, go back to step 2.

The time complexity of this attack is 2^{64} encryptions and $2^{104} \cdot 2^{16} = 2^{120}$ additions (these are the biggest part of the work). It requires 2^{64} chosen plaintexts, and 2^{64} memory. By not performing step 1, it is possible to make the amount of memory needed negligible; in this case, time complexity becomes $2^{104} \cdot 2^{16} = 2^{120}$ encryptions.

5.3 An attack on 4 rounds of Safer++$_{256}$

The attack is very similar to the one on 4 rounds of Safer++$_{128}$. Only some details in the key guess change due to a different key schedule. Given a 256-bit key $(K^{(1)}, K^{(2)}, ..., K^{(32)})$, the key schedule of Safer++$_{256}$ has the following properties:

- Two parity bytes are computed: $K^{(0)} = \bigoplus_{i=1}^{16} K^{(i)}$ and $K^{(33)} = \bigoplus_{i=17}^{32} K^{(i)}$.
- Odd subkeys only depends on bytes $K^{(0)}, ..., K^{(16)}$. More precisely, $K_i^{(p)} \approx K^{(i+p-1 \mod 17)}$.
- Even subkeys only depends on bytes $K^{(17)}, ..., K^{(33)}$. More precisely, $K_i^{(p)} \approx K^{((i+p-2) \mod 17+17)}$.

Table 2 details the subkeys of layers γ_1 and γ_4. We can see that the 16 bytes we need to guess in order to compute the plaintexts (step 3 of the attack) allows to compute 7 bytes at the end of round 3.

The attack is:

[1] In fact it is possible to do better. See next section(more precisely step 6 of the attack) for details.

Table 2. Bytes that can be computed at the end of round 3 (attack on Safer++$_{256}$), using the key bytes guessed at step 2 of the attack(in bold).

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$K_1^{(i)} \approx$	$K^{(1)}$	$K^{(2)}$	$\mathbf{K^{(3)}}$	$\mathbf{K^{(4)}}$	$K^{(5)}$	$\mathbf{K^{(6)}}$	$\mathbf{K^{(7)}}$	$K^{(8)}$	$\mathbf{K^{(9)}}$	$\mathbf{K^{(10)}}$	$K^{(11)}$	$K^{(12)}$	$\mathbf{K^{(13)}}$	$K^{(14)}$	$K^{(15)}$	$\mathbf{K^{(16)}}$
$K_2^{(i)} \approx$	$K^{(18)}$	$K^{(19)}$	$\mathbf{K^{(20)}}$	$\mathbf{K^{(21)}}$	$K^{(22)}$	$\mathbf{K^{(23)}}$	$\mathbf{K^{(24)}}$	$K^{(25)}$	$\mathbf{K^{(26)}}$	$\mathbf{K^{(27)}}$	$K^{(28)}$	$K^{(29)}$	$\mathbf{K^{(30)}}$	$K^{(31)}$	$K^{(32)}$	$\mathbf{K^{(33)}}$
$K_7^{(i)} \approx$	$\mathbf{K^{(7)}}$	$K^{(8)}$	$\mathbf{K^{(9)}}$	$\mathbf{K^{(10)}}$	$K^{(11)}$	$K^{(12)}$	$\mathbf{K^{(13)}}$	$K^{(14)}$	$K^{(15)}$	$\mathbf{K^{(16)}}$	$K^{(0)}$	$K^{(1)}$	$K^{(2)}$	$\mathbf{K^{(3)}}$	$\mathbf{K^{(4)}}$	$K^{(5)}$
$K_8^{(i)} \approx$	$\mathbf{K^{(24)}}$	$K^{(25}$	$\mathbf{K^{(26)}}$	$\mathbf{K^{(27)}}$	$K^{(28)}$	$K^{(29)}$	$\mathbf{K^{(30)}}$	$K^{(31)}$	$K^{(32)}$	$\mathbf{K^{(33)}}$	$K^{(17)}$	$K^{(18)}$	$K^{(19)}$	$\mathbf{K^{(20)}}$	$\mathbf{K^{(21)}}$	$K^{(22)}$
	✓		✓	✓			✓			✓				✓	✓	

1. Ask for encryption of all 2^{64} plaintexts looking like (8).
2. Guess the 16 key bytes in bold in Table 2.
3. Using the key bytes just guessed, compute 2^{16} plaintexts from (7).
4. From the corresponding ciphertexts, compute the 7 bytes 1,3,4,7,10,14,15 at the end of round 3.
5. Sum up all 2^{16} 7-byte data obtained, and checks whether these 7 bytes are balanced.
6. If the test succeeds, guess two more key bytes such as to check balance on one more byte at the end of round 3. Keep guessing new key bytes until one check has failed or the entire key has been guessed (as was done in section 5.1). In the second case, with all 16 bytes balanced, check by a trial encryption whether it is the right key .
7. If the right key has not found, go back to step 2.

As a fraction of about $1/2^{128}$ of the keys passes the test, 2^{128} trial encryptions must be done in order to discard wrong keys. However the biggest part of the work consists in about $2^{136} \cdot 2^{16} = 2^{152}$ additions (most of them during step 6).

6 Towards a more efficient attack?

Let us now analyze what happens if we try to build a 3-round distinguisher following the same principle as in section 4 (i.e. tracing the same set of plaintexts described by (3) through one more round). The expressions we obtain at the output (corresponding to equation (6) of section 4) can be written as functions of a and b that look like:

$$\Phi^{(j)}(a,b) = \sum_{k=1}^{16} \lambda_k^{(j)} F_{3,k}\Big(\sum_{l=1}^{16} \mu_{k,l}^{(j)} F_{2,l}(\xi_{k,l}^{(j)} a + \xi_{k,l}^{\prime(j)} b)\Big) \qquad (j = 1...16) \quad (9)$$

where $\lambda_k^{(j)}$, $\mu_{k,l}^{(j)}$, $\xi_{k,l}^{(j)}$ and $\xi_{k,l}^{\prime(j)}$ are constants. To be formal, note that some $F_{r,p}$ have a slightly more general form than given in (1), as they must incorporate the constants we neglected in section 4.

Just like we did with equation (6), we would like to find a criteria (possibly probabilistic) allowing to distinguish functions $\Phi^{(j)}$ that can be written as (9) from random functions. Balance does not longer work, as it is not preserved by permutations (such as $F_{3,k}$): while $\sum_{l=1}^{16} \mu_{k,l}^{(j)} F_{2,l}(\xi_{k,l}^{(j)} a + \xi_{k,l}^{'(j)} b)$ is balanced, $F_{3,k}(\sum_{l=1}^{16} \mu_{k,l}^{(j)} F_{2,l}(\xi_{k,l}^{(j)} a + \xi_{k,l}^{'(j)} b))$ is not. It is true however that very few functions $\mathbb{Z}_{256} \times \mathbb{Z}_{256} \to \mathbb{Z}_{256}$ can be written as (9). Indeed the number of such functions is $2^{2^{19}}$, while the number of 32-uples of functions $\mathbb{Z}_{256} \to \mathbb{Z}_{256}$ (16 functions $F_{2,l}$ and 16 functions $F_{3,k}$) is only $2^{2^{16}}$.[2]

Despite these considerations, it is not clear whether such a criteria exists, with affordable complexity: after all, the general problem of constructing a distinguisher for a block cipher also amounts to, given the input-output behavior of a function, deciding whether it matches with a given expression or not. The difference is that the size of the input and output sets is much more affordable in our problem than in the general distinguisher problem, which maybe could make time and data complexity more reasonable.

If a solution to this problem is found, it could potentially apply to all ciphers vulnerable to integral cryptanalysis. Of course the performance achievable for this type of attacks would strongly depends on the complexity of the underlying solution.

7 Conclusion

In this paper we gave three attacks, respectively on 3 rounds of Safer++$_{128}$, 4 rounds of Safer++$_{128}$, and 4 rounds of Safer++$_{256}$. All three are based on the same 2-rounds integral distinguisher, and are thus *chosen*-plaintexts attacks. We summarize our results and compare them to the existing ones in Table 3, where we assume one encryption takes the same time as 2^8 additions.

We also sketched a way to extend the idea of integral distinguishers. Although we are not convinced this track will effectively lead to practical attacks, we thought this point was worth to be mentioned.

[2] If we take into account the fact the $F_{r,l}$ are permutations, this number reduces to $2^{2^{15.7}}$. In fact, due to their particular form, there is far less possibilities for each function $F_{r,l}$. But an hypothetical distinguisher would probably not be able to take these particularities into account.

Table 3. Summary of existing attacks on Safer++$_{128}$ and Safer++$_{256}$. CP=Chosen Plaintext; KP=Known Plaintext. Time complexity is measured in number of encryptions, space complexity in number of plaintexts

Attack Type	Key Size	#rounds	#Plaintexts	Time Complexity	Space Complexity	Fraction of Keys
Integral(CP)	128	3	2^{16}	2^{16}	2^{16}	All
Integral(CP)	128	4	2^{64}	2^{112}	2^{64}	All
			2^{64}	2^{120}	2^{16}	All
Integral(CP)	256	4	2^{64}	2^{144}	2^{64}	All
Linear(KP)[9]	128, 256	3	2^{33}	2^{121}		2^{-6}
Linear(KP)[9]	256	4	2^{81}	2^{178}		2^{-13}
Linear(KP)[9]	256	4	2^{91}	2^{167}		2^{-11}

References

1. A. Biryukov, C. De Cannière, and G. Dellkrantz. Cryptanalysis of Safer++. Internal NESSIE Report, 2003.
2. J. Daemen, L. Knudsen, and V. Rijmen. The block cipher square. In E. Biham, editor, *Fast Software Encryption 1997*, pages 149–165. Springer-Verlag, 1997. Lecture Notes in Computer Science Volume 1267.
3. J. Daemen and V. Rijmen. AES proposal: Rijndael. In *Proc. first AES conference*, August 1998. Available on-line from the official AES page: `http://csrc.nist.gov/CryptoToolkit/aes/rijndael/Rijndael.pdf`.
4. K. Hwang, W. Lee, S. Lee, S. Lee, and J. Lim. Saturation attacks on reduced round skipjack. In J. Daemen and V. Rijmen, editors, *Fast Software Encryption (FSE '02)*, pages 100–111, Berlin, 2002. Springer-Verlag. Lecture Notes in Computer Science Volume 2365.
5. J. Nakahara Jr, P. S.L.M. Barreto, B. Preneel, and al. Square attacks on reduced-round pes and idea block ciphers. Available at http://eprint.iacr.org/2001/068/.
6. L. Knudsen and D. Wagner. Integral cryptanalysis. In J. Daemen and V. Rijmen, editors, *Fast Software Encryption (FSE '02)*, pages 112–127, Berlin, 2002. Springer-Verlag. Lecture Notes in Computer Science Volume 2365.
7. S. Lucks. The saturation attack - a bait for twofish. In M. Matsui, editor, *Fast Software Encryption (FSE '01)*, pages 1–15, Berlin, 2001. Springer-Verlag. Lecture Notes in Computer Science Volume 2355.
8. J.L. Massey, G.H. Khachatrian, and Kuregian M.K. Nomination of SAFER++ as Candidate Algorithm for NESSIE. Available at http://www.cryptonessie.org.
9. J. Nakahara, B. Preneel, and al. Linear cryptanalysis of reduced-round safer++. In *Proceedings of the second NESSIE Workshop*, 2001.
10. Y. Yeom, S. Park, and I. Kim. On the security of camellia against the square attack. In J. Daemen and V. Rijmen, editors, *Fast Software Encryption (FSE '02)*, pages 89–99, Berlin, 2002. Springer-Verlag. Lecture Notes in Computer Science Volume 2365.

A Proof that $B(\theta) = 5$.

We prove that it is impossible to find an input a achieving $B(\theta) = 4$. There are three possibilities:

– $W(a) = 1$ and $W(\theta(a)) = 3$: Let us describe the linear transformation θ by a matrix M_θ, such that $b = \theta(a) \Leftrightarrow b = a \cdot M_\theta$:

$$M_\theta = \begin{pmatrix}
1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 4 & 2 & 2 & 2 & 1 & 1 & 2 & 1 \\
2 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 2 & 4 & 2 & 2 \\
2 & 2 & 4 & 2 & 2 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 2 & 1 & 2 & 1 & 1 & 1 \\
4 & 2 & 2 & 2 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 \\
1 & 1 & 2 & 1 & 2 & 1 & 1 & 1 & 2 & 4 & 2 & 2 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 2 & 2 & 4 & 2 & 2 & 1 & 1 & 1 \\
1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 2 & 1 \\
1 & 1 & 2 & 1 & 4 & 2 & 2 & 2 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 2 & 4 & 2 & 2 & 1 & 1 & 2 & 1 & 2 & 1 & 1 & 1 \\
1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 2 & 2 & 4 & 2 \\
2 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 2 & 1 & 4 & 2 & 2 & 2 \\
2 & 4 & 2 & 2 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 2 & 1 \\
2 & 1 & 1 & 1 & 2 & 2 & 4 & 2 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 \\
1 & 1 & 2 & 1 & 2 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1
\end{pmatrix}$$

We can see that every row contains at least ten 1's. Thus for an input a whose one only byte is $\neq 0$, at least ten bytes of the output are $\neq 0$. Thus $W(a) + W(\theta(a)) \geq 11$.

– $W(a) = 3$ and $W(\theta(a)) = 1$: Let us now write the matrix M_θ^{-1} corresponding to θ^{-1}:

$$M_\theta^{-1} = \begin{pmatrix}
0 & 0 & 0 & -4 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & -1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & -4 & 0 & 0 & 1 & -1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & -4 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & -1 \\
0 & 0 & -1 & 16 & -1 & 0 & -4 & 1 & 0 & -4 & 0 & 1 & -4 & -1 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & -4 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & -1 \\
1 & 0 & 0 & -1 & 0 & 0 & 0 & -4 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & -4 & 0 & 0 & 1 & -1 & 0 & 1 & 1 & 0 \\
-4 & 0 & 0 & 1 & 0 & 0 & 0 & 16 & -1 & -1 & -4 & 1 & 0 & -4 & -1 & 1 \\
1 & 1 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & -4 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & -4 & 0 & 0 & 1 & -1 \\
0 & 1 & 0 & -1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & -4 & 0 & 0 & 1 & 0 \\
-1 & -4 & 0 & 1 & -4 & -1 & -1 & 1 & 0 & 0 & 0 & 16 & 0 & 0 & -4 & 1 \\
0 & 0 & 1 & -1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & -4 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & -4 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & -1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & -4 \\
0 & -1 & -4 & 1 & 0 & -4 & 0 & 1 & -4 & 0 & -1 & 1 & -1 & 0 & 0 & 16
\end{pmatrix}$$

As every row contains at least 5 1's or -1's, $W(\theta(a)) = 1$ implies $W(a) \geq 5$.

- $W(a) = 2$ and $W(\theta(a)) = 2$: Finding a achieving these conditions amounts to find $\alpha, \beta \in \mathbb{Z}_{256}$ and $i, j \in \{1, 16\}$ such that the linear combination $\alpha \cdot L_i + \beta \cdot L_j$ contains (at most) two non-zero elements (where L_i denotes the i^{th} row of matrix M_θ). As every row contains at least ten 1's, for any pair of rows, there is at least four positions where both rows have a 1. Thus a 0 must appear at these positions in the linear combination. We conclude that $\alpha \equiv -\beta$. Without loss of generality, we can assume $\alpha = 1$. By computing difference of all 120 pairs of rows, and by noting that none of them has a byte hamming weight lower than 3, we reach the conclusion.