Cryptanalysis of the ESSENCE Family of Hash Functions

Nicky Mouha^{1,2,*}, Gautham Sekar^{1,2,**}, Jean-Philippe Aumasson^{3,***}, Thomas Peyrin^{4}, Søren S. Thomsen^{5}, Meltem Sönmez Turan^{6}, and Bart Preneel^{1,2}

^{1} Department of Electrical Engineering ESAT/SCD-COSIC, Katholieke Universiteit Leuven, Kasteelpark Arenberg 10, B-3001 Heverlee, Belgium.
^{2} Interdisciplinary Institute for BroadBand Technology (IBBT), Belgium.
^{3} FHNW, Windisch, Switzerland.
^{4} Ingenico, France.
^{5} Department of Mathematics, Technical University of Denmark, Matematiktorvet 303S, DK-2800 Kgs. Lyngby, Denmark.
^{6} Computer Security Division, National Institute of Standards and Technology, USA.

Abstract. ESSENCE is a family of cryptographic hash functions, accepted to the first round of NIST's SHA-3 competition. This paper presents the first known attacks on ESSENCE. We present a semi-free-start collision attack on 31 out of 32 rounds of ESSENCE-512, invalidating the design claim that at least 24 rounds of ESSENCE are secure against differential cryptanalysis. We develop a novel technique to satisfy the first nine rounds of the differential characteristic. Non-randomness in the outputs of the feedback function F is used to construct several distinguishers on a 14-round ESSENCE block cipher and the corresponding compression function, each requiring only 2^{17} output bits. This observation is extended to key-recovery attacks on the block cipher. Next, we show that the omission of round constants allows slid pairs and fixed points to be found. These attacks are independent of the number of rounds. Finally, we suggest several countermeasures against these attacks, while still keeping the design simple and easy to analyze.

Keywords: Cryptanalysis, hash function, ESSENCE, semi-free-start collision, distinguisher, key-recovery, slide attack.

1 Introduction

Recent attacks by Wang et al. on the widely used hash functions MD4 [16], MD5 [17], RIPEMD [16] and SHA-1 [18], as well as other hash functions, show that collisions for these hash functions can be found much faster than expected by the birthday paradox [15].

In search for a new secure hash function standard, NIST announced the SHA-3 hash function competition [12]. The ESSENCE family of cryptographic hash functions, designed by Martin [8], advanced to the first round of this competition. It is a family of block cipher-based hash functions using the Merkle-Damgård mode of operation. The ESSENCE family uses simple algorithms that are easily parallelizable and well-established mathematical principles. ESSENCE comes with a proof of security against linear and differential cryptanalysis, that until this paper remained unchallenged.

First, we describe several undesired properties of the ESSENCE L function. These can be used to build a semi-free-start collision attack [11, pp. 371–372] on 31 out of 32 rounds

* This author is funded by a research grant of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).
** This author is supported by an FWO project.
*** Supported by the Swiss National Science Foundation under project no. 113329.

of the ESSENCE-512 compression function using a differential characteristic. This directly invalidates the design claim that at least 24 rounds of ESSENCE are resistant against differential cryptanalysis [8]. To build our attack, we describe a novel technique to satisfy the conditions imposed by the characteristic in the first nine rounds. We do not know of a similar technique in existing literature.

Then, we find that the ESSENCE compression functions use a non-linear feedback function F that is unbalanced. We first exploit this to build efficient distinguishers on 14-round versions of the ESSENCE block ciphers as well as the compression functions. These distinguishers require only 2^{17} output bits. We then show how to use these results to recover the key with a few known plaintexts and a computational effort less than that of exhaustive search. We also show that, under some circumstances, the attacks on 14-round ESSENCE could be extended to the full 32-round block cipher and compression function.

Following this, we observe that the omission of round constants in ESSENCE leads to several attacks that cannot be prevented by increasing the number of rounds. A slide attack can be applied to any number of rounds of the ESSENCE compression function. We also find fixed points for any number of rounds of the ESSENCE block cipher, that lead to a compression function output of zero.

ESSENCE was not qualified to the second round of the SHA-3 competition; however, its appealing features (like design simplicity and hardware efficiency) make any effort on tweaking it appear worthwhile. Therefore, in this paper, we also suggest some countermeasures to thwart the aforesaid attacks.

In later work, Naya-Plasencia et al. [13] present different results on ESSENCE. Our paper presents not only differential cryptanalysis but also distinguishing attacks and slide attacks. Furthermore, some of our techniques can easily be generalized to other block ciphers and hash functions.

The paper is organized as follows. Section 2 describes the compression function of ESSENCE. In Sect. 3, we define and calculate the branching number of the linear L function for both linear and differential cryptanalysis. As the branching number turns out to be quite low, we use this observation to build a semi-free-start collision attack for 31 out of 32 rounds in Sect. 4. To satisfy the first nine rounds of the differential characteristic of the semi-free-start collision attack, we develop a technique in Sect. 5. Our distinguishers that exploit the weakness of F function are presented in Sect. 6. In the same section, we also show how our distinguishing attacks can be converted into key-recovery attacks on the block ciphers. Following this, we show how the omission of round constants allows us to find slid pairs (Sect. 7) and fixed points (Sect. 8) for any number of rounds. Finally, Sect. 9 enlists our countermeasures and Sect. 10 concludes the paper.

2 Description of the Compression Function of ESSENCE

The inputs to the compression function of ESSENCE are an eight-word chaining value and an eight-word message block, where each word is 32 or 64 bits in length, for ESSENCE-224/256 and ESSENCE-384/512 respectively. The compression function uses a permutation E, that in turn uses a nonlinear feedback function F, a linear transformation L, some XORs and word shifts.

The message block $m = (m_0, \ldots, m_7)$ forms the initial value of an eight-word state $k = (k_0, \ldots, k_7)$. In the case of the block cipher, m is the key $k = (k_0, \ldots, k_7)$. Similarly, the chaining value $c = (c_0, \ldots, c_7)$ is the initial chaining value of an eight-word state $r = (r_0, \ldots, r_7)$. In the case of the block cipher, c is the plaintext. Both states are iterated N times. The designer recommends N to be a multiple of 8, $N \geq 24$ for resistance to differential and linear cryptanalysis and $N = 32$ as a measure of caution [8]. Figure 1 illustrates one round of ESSENCE.

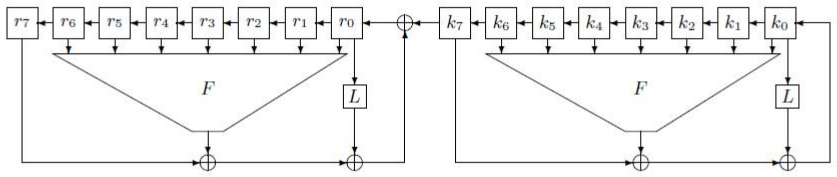

Fig. 1. One round of ESSENCE; each r_n and k_n $(n = 0, \ldots, 7)$ is a 32- or 64-bit word

The compression function uses a Davies-Meyer feed-forward (see Fig. 2). That is, at the end of N rounds, the value $r_7||r_6||r_5||r_4||r_3||r_2||r_1||r_0$ is XORed with the initial chaining value. The result is the $r_7||r_6||r_5||r_4||r_3||r_2||r_1||r_0$ for the next iteration.

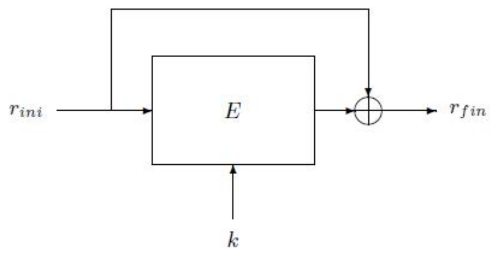

Fig. 2. The compression function of ESSENCE; E is the round function of ESSENCE when iterated N times, k denotes the message block, r_{ini} denotes the initial value of $r_7||r_6||r_5||r_4||r_3||r_2||r_1||r_0$ and r_{fin} denotes the value of r for the next iteration

3 Branching Number of the L Function

The L function of ESSENCE is a linear transformation from 32 (or 64) bits to 32 (or 64) bits and it is the only component that causes diffusion between the different bit positions of a word. Therefore, its properties are very important for both linear and differential cryptanalysis.

In this section, we focus on the branching number of the L function for both linear and differential cryptanalysis. Let the branching number for differential cryptanalysis be the minimum number of non-zero input and output differences for the L function. These branching numbers are 10 and 16 for the 32-bit and 64-bit L functions respectively. If we were to consider only one-bit differences at either the input or the output of L, these numbers would be 14 and 27 respectively.

The branching number for linear cryptanalysis can be defined as the (non-zero) minimum number of terms in a linear equation relating the input and output bits of the L function. These branching numbers are 10 and 17 for the 32-bit and 64-bit L function respectively. Considering linear relations that involve only one bit at the input or one bit at the output, we would find branching numbers of 12 and 26 respectively.

Although one-bit differences are spread out well by the L function, this is clearly not the case for differences in multiple bits. This problem is most severe with the 64-bit L function. In the next section, we will show how this property can be used to build narrow trails for all digest sizes of ESSENCE.

4 A 31-Round Semi-Free-Start Collision Attack For ESSENCE-512

In this section, we will focus only on ESSENCE-512 for the sake of brevity and clarity. As the strategy is not specific to any particular digest size, these results can easily be generalized to all digest sizes of ESSENCE.

Although the ESSENCE L function spreads out one-bit differences very well, the previous section showed that this is not the case for differences in multiple bits. We therefore propose to use the differential characteristic of Table 1, to obtain 31-round semi-free-start collisions for ESSENCE-512.

To construct narrow trails, we use the non-zero difference A with the lowest possible Hamming weight. For this difference, we impose the condition $(\neg A) \wedge L(A) = 0$, where $\neg$ represents the negation operation and all logical operations are to be performed bitwise. This can be formulated as follows: if there is a difference at the output of the L function at a particular bit position, there must be a difference at the input of L at this bit position as well. This requirement is necessary, as the F function can absorb or propagate an input difference at the output, but if no input difference is present, then there won't be an output difference either at this particular bit position. This places a restriction on the output difference of the L function for this bit position.

There exist exactly 8 differences A with a weight of 17 and lower weight differences A do not exist. These differences are available in Appendix A, along with a method to calculate them efficiently.

The last two columns of Table 1 provide an estimate of the probability that the characteristic is satisfied for every round. For these, we have assumed that the F function propagates or absorbs an input difference with equal probability. An more accurate calculation of these probabilities takes into account that the shift register causes input values of the F function to be reused.

We find that this probability is different for bit positions where A and $L(A)$ both contain a difference, and for bit positions where only A contains a difference. As such, of all differences A with weight 17, we select the difference that has the highest weight of $L(A)$. Five such differences exist, and we arbitrarily select the difference with the smallest absolute value, $A =$ `0A001021903036C3`. The corresponding $L(A) =$ `0200100180301283` has weight

11. As such, we find that rounds 10 to 16 of the key schedule, and rounds 18 to 24 of the compression function, each have a probability of $2^{-8.415 \cdot 6-8 \cdot 11} = 2^{-138.49}$. For rounds 18 to 23 of the key schedule, we find a probability of $2^{-7.193 \cdot 6-7 \cdot 11} = 2^{-120.16}$.

To find semi-free-start collisions, we first search for a message pair that satisfies the key expansion characteristic, and then afterwards search for a chaining value pair that satisfies the compression function characteristic. These two searches can be decoupled, as the chaining value does not influence the key schedule. As such, the probabilities for the message pairs and IV pairs can be summed up instead of multiplied.

As will be shown in the next section, only two round function calls are required to find a message (or IV) that satisfies the first nine rounds of the key expansion (or compression function). To find a pair of messages (or IVs) that satisfy the differential characteristic, we use the same depth-first search algorithm that was introduced for SHA-1 in [2]. The memory requirements of this search algorithm are negligible. We assume that the cost of visiting a node in this search tree is equivalent to one round function call, or 2^{-5} compression function calls. The complexity calculation of [2] then shows that a 31-round semi-free-start collision can be found using the characteristic of Table 1 after $2^{138.49+120.16+1-5} + 2^{138.49+1-5} = 2^{254.65}$ equivalent compression function calls. This is faster than a generic birthday attack, which requires about 2^{256} compression function evaluations.

5 Finding Message Pairs for the First Nine Rounds

To find messages that satisfy the first few rounds of the characteristic, single-message modification [17] cannot be used. This is because the entire message is loaded into the r-registers before the round function is applied, instead of injecting one message word every round. We therefore propose to use another technique, that turns out to be even more efficient than single-message modification. This concept is explained for the key schedule only, as it is completely analogous for the compression function.

In this section, we will adopt a stream-based notation for the round function. Denote the initial eight-word state $(k_7, k_6, k_5, k_4, k_3, k_2, k_1, k_0)$ as $(x_{-2}, x_{-1}, x_0, x_1, x_2, x_3, x_4, x_5)$. After clocking one for one round, the value of the register k_0 is represented by x_6, and so on. In this text, we will not make a distinction between linear and affine equations, and use the term "linear equation" for any equation that contains no monomials of a degree more than one.

Finding a pair of messages that satisfy the characteristic, can be seen as solving a set of non-linear equations defined by the round function. Solving a set of non-linear equations is a difficult problem in general. This is even more the case as we are not looking for a single solution, but for a very large set of solutions.

What we can do, however, is impose linear conditions on the variables x_0 to x_{12}, in such a way that the round function behaves as a linear function. We then obtain a set of linear equations, of which every solution corresponds to a message pair that follows the first nine rounds of the characteristic. Enumerating the solutions of this linear space has a negligible computation cost compared to a round function evaluation.

For every solution, we have to apply the round function twice to obtain x_{13} and x_{14}. These are guaranteed to follow the characteristic as well. They serve as a starting point to satisfy the conditions of the remaining characteristic in a probabilistic way. After reaching

Table 1. A 31-round semi-free-start collision differential characteristic for the ESSENCE-512 compression function; differences from R to Y are arbitrary, 0 represents the zero difference, $A =$ `0A001021903036C3`

Round	Register R	Register K	Pr for CV	Pr for m
0	0 0 0 0 0 0 0 0	A 0 0 0 0 0 0 0	1	1
1	0 0 0 0 0 0 0 A	0 0 0 0 0 0 0 A	1	1
2	0 0 0 0 0 0 A 0	0 0 0 0 0 0 A 0	2^{-17}	2^{-17}
3	0 0 0 0 0 A 0 0	0 0 0 0 0 A 0 0	2^{-17}	2^{-17}
4	0 0 0 0 A 0 0 0	0 0 0 0 A 0 0 0	2^{-17}	2^{-17}
5	0 0 0 A 0 0 0 0	0 0 0 A 0 0 0 0	2^{-17}	2^{-17}
6	0 0 A 0 0 0 0 0	0 0 A 0 0 0 0 0	2^{-17}	2^{-17}
7	0 A 0 0 0 0 0 0	0 A 0 0 0 0 0 0	2^{-17}	2^{-17}
8	A 0 0 0 0 0 0 0	A 0 0 0 0 0 0 0	2^{-17}	2^{-17}
9	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 A	1	1
10	0 0 0 0 0 0 0 0	0 0 0 0 0 0 A 0	1	2^{-17}
11	0 0 0 0 0 0 0 0	0 0 0 0 0 A 0 0	1	2^{-17}
12	0 0 0 0 0 0 0 0	0 0 0 0 A 0 0 0	1	2^{-17}
13	0 0 0 0 0 0 0 0	0 0 0 A 0 0 0 0	1	2^{-17}
14	0 0 0 0 0 0 0 0	0 0 A 0 0 0 0 0	1	2^{-17}
15	0 0 0 0 0 0 0 0	0 A 0 0 0 0 0 0	1	2^{-17}
16	0 0 0 0 0 0 0 0	A 0 0 0 0 0 0 0	1	2^{-17}
17	0 0 0 0 0 0 0 A	0 0 0 0 0 0 0 A	1	1
18	0 0 0 0 0 0 A 0	0 0 0 0 0 0 A 0	2^{-17}	2^{-17}
19	0 0 0 0 0 A 0 0	0 0 0 0 0 A 0 0	2^{-17}	2^{-17}
20	0 0 0 0 A 0 0 0	0 0 0 0 A 0 0 0	2^{-17}	2^{-17}
21	0 0 0 A 0 0 0 0	0 0 0 A 0 0 0 0	2^{-17}	2^{-17}
22	0 0 A 0 0 0 0 0	0 0 A 0 0 0 0 0	2^{-17}	2^{-17}
23	0 A 0 0 0 0 0 0	0 A 0 0 0 0 0 0	2^{-17}	2^{-17}
24	A 0 0 0 0 0 0 0	A 0 0 0 0 0 0 R	2^{-17}	1
25	0 0 0 0 0 0 0 0	0 0 0 0 0 0 R S	1	1
26	0 0 0 0 0 0 0 0	0 0 0 0 0 R S T	1	1
27	0 0 0 0 0 0 0 0	0 0 0 0 R S T U	1	1
28	0 0 0 0 0 0 0 0	0 0 0 R S T U V	1	1
29	0 0 0 0 0 0 0 0	0 0 R S T U V W	1	1
30	0 0 0 0 0 0 0 0	0 R S T U V W X	1	1
31	0 0 0 0 0 0 0 0	R S T U V W X Y	1	1

round 31, we can calculate x_{-2} and x_{-1} by applying two inverse round functions. These values will always satisfy the characteristic.

Let $A[j]$ denote the j-th significant bit ($j = 0$ denotes the least significant bit or LSB) of A. The only non-linear function of ESSENCE is the F function. As the F function operates on every bit in parallel, the linear conditions that have to be added, depend on the values $A[j]$ and $L(A)[j]$ at every bit position j. The equations we use are given in Appendix B. Note that for bit positions j where $A[j] = 0$, it is not a problem if $x_0[j]$ or $x_{12}[j]$ are represented by a non-linear expression, as these bits are not involved in any of the linear conditions anyway.

As the equations in Appendix B show, we need to add 10 equations for every bit position j where $A[j] = 1$, and 6 equations if $A[j] = 0$. Also, to represent the 64-bit values x_8 to x_{12} resulting from the round function, we need to add $5 \cdot 64$ additional equations for outputs of the round function. In total, we obtain a set of $10 \cdot 17 + 6 \cdot (64 - 17) + 5 \cdot 64 = 772$ linear equations in $13 \cdot 64 = 832$ binary variables.

We build this system of equations by successively adding $10 + 5 = 15$ or $6 + 5 = 11$ more equations for every bit position j. With some small probability, the system of equations becomes inconsistent. If this happens, we add a different set of linear equations for this bit position. Even this may fail with some probability, in which case we add a linearization of the F function using $7 + 5$ instead of $6 + 5$ equations for this particular bit position. This may or may not decrease the number of solutions slightly, but it allows us to avoid backtracking.

For one particular run, using only the equations mentioned in Appendix C, we find a consistent system of 772 linear equations in 832 binary variables. The number of linearly independent equations turns out to be 771. As such, we have found $2^{832-771}/2 = 2^{60}$ pairs of messages that satisfy the first 9 rounds. We divide by two to avoid counting the same pair twice. If more than 2^{60} pairs of messages needed, we can simply run this program again to find the next set of messages. As including these 771 equations would use up a lot of space, we give only one of the 2^{60} message pairs in Table 6.

This technique is very similar to the techniques of multi-message modification [17], tunneling [7], neutral bits [1] and the amplified boomerang attack [6]. These 2^{60} messages correspond to 60 auxiliary differential paths for the amplified boomerang attack. No results are known to us where these auxiliary differential paths were also obtained in a fully automated way.

6 Distinguishing Attacks

Our motivational observation is that the non-linear feedback function F is unbalanced. Exploiting this, we first construct distinguishers on 14-round ESSENCE (both the block cipher and the compression function) and then for the full 32-round ESSENCE. Towards the end of this section, we present key-recovery attacks on the ESSENCE family of block ciphers. These attacks can be seen as an immediate consequence of our distinguishing attacks.

6.1 Weakness in the Feedback Function of ESSENCE

In [8], the designer notes that the security of the algorithms is heavily dependent on F, as it is the only nonlinear function in ESSENCE. This gave us some motivation to study the properties of F. The function F takes seven 32-bit or 64-bit words (say, $a, \ldots, g$) as inputs and produces a 32-bit or 64-bit word as the output. The function works in a bitsliced manner. The exact description of F is largely irrelevant to our analysis; hence, we refer the interested reader to Appendix D.

Let $F(a, b, c, d, e, f, g)[j]$ denote the j-th significant bit ($j = 0$ denotes the LSB) of $F(a, b, c, d, e, f, g)$. Our motivational observation is the following (confirmed both experimentally and from the tables in Appendix D of [8]).

Observation 1: If $a, \ldots, g$ are uniformly distributed, then

$$Pr[F(a, b, c, d, e, f, g)[j] = 0] = \frac{1}{2} + \frac{1}{2^7} \ . \tag{1}$$

6.2 Distinguishers on 14-Round ESSENCE

In this section, we use Observation 1 to build distinguishers on 14-round ESSENCE. First, we consider the block cipher and then the compression function.

Let $k_n[j]$, $r_n[j]$ and $L(r_n)[j]$ respectively denote the j-th significant bits ($j = 0$ denotes the LSB) of k_n, r_n and $L(r_n)$. In the beginning, suppose the key k and the initial value r are such that $k_0[0] = r_0[0]$. Then, after 7 rounds, $k_7[0] = r_7[0]$. Now, if after the 7th round, $L(r_0)[0] = 0$ and $F(r_6, r_5, r_4, r_3, r_2, r_1, r_0)[0] = 0$ (from Observation 1, this occurs with $0.5 + 2^{-7}$ probability[7]), then after the 8th round, we will have $r_0[0] = 0$. Note that the condition $L(r_0)[0] = 0$ after the 7th round is the same as the condition $L(r_1)[0] = 0$ after the 8th round. Therefore, when the key and the plaintext are initially related in the form $k_0[0] = r_0[0]$, and when the outputs after 8 rounds satisfy the condition $L(r_1)[0] = 0$ (this occurs with probability 1/2), then $Pr[r_0[0] = 0] = 1/2 + 2^{-7}$. Now, r_0 and r_1 after the 8th round are respectively equal to r_6 and r_7 after the 14th round. Hence, when the key and the plaintext are related in the form $k_0[0] = r_0[0]$, and when the outputs after 14 rounds satisfy the condition $L(r_7)[0] = 0$, then

$$Pr[r_6[0] = 0] = \frac{1}{2} + \frac{1}{2^7} \ . \tag{2}$$

6.3 The Distinguisher

A distinguisher is an algorithm that distinguishes one probability distribution from another. In cryptography, a distinguisher is an algorithm that distinguishes a stream of bits from a stream of bits uniformly distributed at random (i.e., bitstream generated by an ideal cipher).

Our distinguisher on ESSENCE is constructed by collecting n outputs $r_6[0]$, after 14 rounds, generated by as many keys (so that the n samples are independent) such that $k_0[0] = r_0[0]$ initially. Let D_0 and D_1 denote the distributions of the outputs from 14-round ESSENCE block cipher and a random permutation, respectively. Given $L(r_7)[0] = 0$, let p_0 and p_1 respectively denote the probability that $r_6[0] = 0$ holds given the outputs

[7] The bit $L(r_0)[0]$ is the XOR-sum of $r_0[0]$ and several other bits of r_0. We assume that all $r_0[j]$ are independent and uniformly distributed. Then the condition $L(r_0)[0] = 0$ does not affect $Pr[r_0[0] = 0]$ and therefore the bias in $Pr[F(r_6, r_5, r_4, r_3, r_2, r_1, r_0)[0] = 0]$ is also unaffected.

are collected from 14-round ESSENCE and the probability that $r_6[0] = 0$ holds given the outputs are generated by a random source. That is, $p_0 = 1/2 + 2^{-7}$ (from (2)) and $p_1 = 1/2$. Then, $\mu_0 = n \cdot p_0$ and $\mu_1 = n \cdot p_1$ are the respective means of D_0 and D_1. Similarly, $\sigma_0 = \sqrt{n \cdot p_0 \cdot (1 - p_0)}$ and $\sigma_1 = \sqrt{n \cdot p_1 \cdot (1 - p_1)}$ denote the respective standard deviations of D_0 and D_1. When n is large, both these binomial distributions can be approximated with the normal distribution. Now, if $|\mu_0 - \mu_1| > 2(\sigma_0 + \sigma_1)$, i.e., $n > 2^{16}$, the output of the cipher can be distinguished from a random permutation with a success probability of 0.9772 (since the cumulative distribution function of the normal distribution gives the value 0.9772 at $\mu + 2\sigma$) provided $L(r_7)[0] = 0$. To test whether n is large enough for the normal approximation to the binomial distribution to hold, we use a commonly employed rule of thumb: $n \cdot p > 5$ and $n \cdot (1 - p) > 5$, where $p \in \{p_0, p_1\}$. A simple calculation proves that both the above inequalities hold when $n = 2^{16}$. Since the condition $L(r_7)[0] = 0$ holds with 0.5 probability, we need to generate $2 \cdot 2^{16} = 2^{17}$ samples of $r_6[0]$ from as many keys (such that $k_0[0] = r_0[0]$ initially) to build the distinguisher with a success probability of 0.9772. Our simulations support this result.

6.4 Distinguishers using Biases in Other Bits

Since the function F operates on its input bits in a bitsliced manner, it is easy to see that the distinguisher presented for the LSB of r_6 also works for more significant bits. In other words, if initially $k_0[j] = r_0[j]$, for any j in $\{0, \ldots, 31\}$, then with 2^{16} samples of $r_6[j]$ at the the end of 14 rounds, it is possible to distinguish 14-round ESSENCE block cipher from a random permutation with a success probability of 0.9772.

6.5 Distinguishers for the Compression Function

The ESSENCE compression function is a Davies-Meyer construction in which the output of the block cipher is XORed with the initial chaining value. In other words, the output of the compression function is the XOR-sum of the values of $r_7||r_6||r_5||r_4||r_3||r_2||r_1||r_0$ before and after applying the permutation E. This XOR-sum is the chaining value $r_7||r_6||r_5||r_4||r_3||$ $r_2||r_1||r_0$ for the next iteration. As we assume that an attacker can observe both the chaining value input and the compression function output, it is trivial to undo the Davies-Meyer feedforward and apply the distinguishers of the 14-round block cipher.

These observations are extended to 32-round ESSENCE in Appendix E.

6.6 Key-Recovery Attacks

In this section, we show that the distinguishing attacks on the ESSENCE family of block ciphers can be converted into key-recovery attacks.

Let us say that we have n known plaintexts. Considering that the plaintexts are initially loaded directly into the r-registers [9], we expect $n/2$ plaintexts to have $r_0[j] = 0$. Without loss of generality, let us consider this partition of the plaintext space where $r_0[j] = 0$. Now, from our analysis in Sect. 6.2, we can collect statistics on $L(r_7)[j] \oplus r_6[j]$ at the end of the 14 rounds and observe its tendency for sufficiently large n — if $L(r_7)[j] \oplus r_6[j] = 0$ more often, then the key bit $k_0[j] = 0$; likewise, if $L(r_7)[j] \oplus r_6[j] = 1$ more often, then the key bit $k_0[j] = 1$ (the results are swapped if we begin with plaintexts in which $r_0[j] = 1$).

Using a similar analysis, we can recover the rest of the key bits in k_0. The number of known plaintexts required is 2^{15}. This is obtained as follows, using standard linear cryptanalysis [10]. We are interested in finding whether, after 14 rounds, the number of times that $L(r_7)[j] \oplus r_6[j] = 0$ holds is greater than $n/4$. Accordingly, we determine the key bit $k_0[j]$. Unlike in the distinguishing attacks, a confidence interval for the uniform distribution is not required. From [10] we obtain that the success probability of this method is 0.9772 when $n/2 = |p - 1/2|^{-2}$, where p is the probability that $L(r_7)[j] \oplus r_6[j] = 0$ (or 1). Substituting $p = 1/2 \pm 2^{-7}$ in the above formula for n, we get $n = 2^{15}$. It follows that the probability that this recovered key word (k_0) is correct is $(0.9772)^{32} \approx 0.48$. The other 224 bits of the key can be exhaustively searched. Thereby, we expect that $2^{224}/0.48 \approx 2^{225.1}$ keys have to be tested before the correct key is obtained with guaranteed success. This key-recovery attack can also be applied on the block cipher of ESSENCE-224 (which is identical to the block cipher of ESSENCE-256) with the same complexities. For the block ciphers of ESSENCE-384/512, we require 2^{15} known plaintexts and a computational effort equivalent to testing $2^{448}/(0.9772)^{64} \approx 2^{450.1}$ keys (where exhaustive search requires testing 2^{512} keys) for guaranteed success.

These observations are extended to 32-round ESSENCE in Appendix F.

7 Slide Attack

In this part of the study, we provide an efficient method to find two inputs (c, m) and (c', m') such that their output (after feed-forward) r and r' are shifted versions of each other; i.e., if $r_i = r'_{i+1}$ for $0 \le i < 7$.

The necessary conditions on (c, m) and (c', m') are

1. $c_i = c'_{i+1}$ for $0 \le i \le 7$,
2. $c'_0 = m_7 \oplus c_7 \oplus F(c_6, \ldots, c_0) \oplus L(c_0)$,
3. $m_i = m'_{i+1}$ for $0 \le i \le 7$,
4. $m'_0 = m_7 \oplus F(m_6, \ldots, m_0) \oplus L(m_0)$.

If these conditions hold, then after 32 rounds (and XORing with the initial value), the output of the compression function satisfies $r_i = r'_{i+1}$ for $0 \le i < 7$.

As an example, let $m_i = 0$ for all i. Then we must choose $m'_i = 0$ for all $i > 0$, and $m'_0 = 1^n$ where 1^n represents the 32-bit or 64-bit unsigned integer of which all bits are set. Let $c_i = 0$ for all i, let $c'_i = 0$ for all $i > 0$, and let $c'_0 = 1^n$. Then, the two outputs of the compression function (with $N = 32$) are:

c	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
c'	FFFFFFFF	00000000	00000000	00000000	00000000	00000000	00000000	00000000
m	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
m'	FFFFFFFF	00000000	00000000	00000000	00000000	00000000	00000000	00000000
R	6B202EF2	BB610A07	97E43146	9BD34AE3	C8BC7CBF	B8EE4A3C	B6118DC5	775F7BBF
R'	C07ABCFA	6B202EF2	BB610A07	97E43146	9BD34AE3	C8BC7CBF	B8EE4A3C	B6118DC5

For every choice of (c, m), an input (c', m') such that this property on the compression function outputs is obtained can be found in time equivalent to about one compression function evaluation. Hence, in total about 2^{512} pairs of inputs producing slid pairs can be found by the above method. This observation can easily be extended to slide the output by $2, 3, \ldots, 7$ steps.

7.1 Slid Pairs with Identical Chaining Values

It is also possible to find slid pairs with $c = c'$. Let the initial state of the register R be of the form $(c_0, c_0, \ldots, c_0)$, where c_0 is selected randomly. For a message block m of the form $(m_0, m_1, \ldots, m_7)$ where $m_7 = F(c_0, \ldots, c_0) \oplus L(c_0)$ and the rest of the m_i's are selected arbitrarily, select m' as $(m'_0, m'_1, \ldots, m'_7)$, such that $m'_{i+1} = m_i$ for $i = 0, 1, 2, \ldots, 6$ and $m'_0 = m_7 \oplus F(m_6, \ldots, m_0) \oplus L(m_0)$. Then, the outputs of the compression function for m and m' also satisfy $r_i = r'_{i+1}$ for $0 \leq i < 7$. It is possible to select c in 2^{32} different ways, and for each selected c, we can choose $2^{7 \cdot 32}$ different message blocks, therefore the number of such slid pairs is 2^{256}. As an example, assume $c_0 =$ `243f6a88`, which is the truncated fractional part of π, and all "free" message words are zero.

c, c'	243F6A88	243F6A88	243F6A88	243F6A88	243F6A88	243F6A88	243F6A88	243F6A88
m	00000000	00000000	00000000	00000000	00000000	00000000	00000000	F6B1EB63
m'	094E149C	00000000	00000000	00000000	00000000	00000000	00000000	00000000
R	BE31AA01	EB6E9F07	EAD99889	6FE79B44	391CCD35	67FDB8B6	FC3AA0F6	6E80148E
R'	F86D77C6	BE31AA01	EB6E9F07	EAD99889	6FE79B44	391CCD35	67FDB8B6	FC3AA0F6

8 Fixed Points for the ESSENCE Block Cipher

If a fixed point for one round of the ESSENCE block cipher can be found, this automatically leads to a fixed point for all 32 steps of the block cipher. After applying the Davies-Meyer feed-forward, the resulting compression function output will then be zero.

If two different fixed points are found, this would lead to a free-start collision. This free-start collision is preserved after the output padding is applied.

For a fixed point for one round, $c_0 = c_1 = \ldots = c_7$ and $m_0 = m_1 = \ldots = m_7$ should hold. This is obvious: after one step, all register values move one place, but must have the same value as in the previous step to form a fixed point. Moreover, the round update functions should satisfy the following equations.

$$F(c_0, c_0, c_0, c_0, c_0, c_0, c_0) \oplus c_0 \oplus L(c_0) \oplus m_0 = c_0 \ ,$$
$$F(m_0, m_0, m_0, m_0, m_0, m_0, m_0) \oplus m_0 \oplus L(m_0) = m_0 \ .$$

Solving the equations, one gets the following values for ESSENCE-256 and ESSENCE-512:

	ESSENCE-256	ESSENCE-512
c_0	993AE9B9	D5B330380561ECF7
m_0	307A380C	10AD290AFFB19779

Using similar methods, we have found that the only fixed points for two, three or four rounds is the same fixed point for one round applied two, three or four times respectively.

We have not been able to extend this result for more rounds. As such, we have not been able to find a free-start collisions using this technique. Depending how the compression function is used, however, it might be undesirable that we can easily find inputs that fix the compression function output to zero.

9 Measures to Improve the Security of ESSENCE

From the analysis in Sect. 3–6, it is clear that ESSENCE has weaknesses in L and F.

The concatenation of both the input and output of the L function can be seen as an error-correcting code with $[n, k] = [64, 32]$ or $[128, 64]$. The branching number is then equal to the error-correcting code of these dimensions with the highest minimum weight. Best known results from coding theory [5] can be used to construct an L function with a branching number for both linear and differential cryptanalysis of 12 or 22 respectively. Better codes may exist according to currently known upper bounds for the minimum weight, but have so far not been found.

A search can be made for variants of these codes (possibly with a slightly lower branching number) that satisfy all design criteria for the L function. Although the resulting function will always be linear, it may however not be possible to implement it as an LFSR.

In (5), the function F is in algebraic normal form (ANF). We know that the coefficient of the maximum degree monomial in this ANF is equal to the XOR-sum of all the entries in the truth table of F. To thwart the attacks in Sect. 6 and Appendix F, it is necessary that F is balanced. Discarding the maximum degree monomial is a possible solution.

Other countermeasures include increasing the number of rounds and adding round constants. In Sect. 7 and Sect. 8, we saw how the omission of round constants allowed slid pairs and fixed points to be found. Increasing the number of rounds does not thwart these attacks, but it increases the security margin against the semi-free-start collision attacks of this paper.

10 Conclusions and Open Problems

In this paper, we first presented a semi-free-start collision attack on 31 out of 32 rounds with a complexity of $2^{254.65}$ compression function evaluations. We find messages that satisfy the first nine rounds of the differential characteristic of the semi-free-start collision attack as the solution of a large set of linear equations. We found that six linear input conditions are sufficient to make F behave as a linear function in Table 5. It is an open problem if solutions using fewer equations exist.

We also presented a set of distinguishers on 14-round ESSENCE. The distinguishers can be applied to the block cipher as well as the compression function. Each of the distinguishers on 14-round ESSENCE requires 2^{17} output bits. The distinguishers work on all digest sizes of ESSENCE with the same complexity. It has also been shown how the distinguishing attacks can be turned into key-recovery attacks.

We then showed how the omission of round constants allowed slid pairs and fixed points to be found. This cannot be prevented by increasing the number of rounds.

Finally, we suggested some measures to improve the security of ESSENCE. These suggestions are rather preliminary and need to be worked on further in order to obtain a more secure family of hash functions.

11 Acknowledgments

The authors would like to thank Christophe De Cannière, Sebastiaan Indesteege, Gaëtan Leurent, Willi Meier, Tomislav Nad, María Naya-Plasencia, Vincent Rijmen and Andrea Röck for their useful comments and suggestions.

Special thanks go out to the designer of ESSENCE, Jason Worth Martin, who not only gave us useful feedback, but was also very supportive when we wanted to make these results public. He has verified the correctness of the results in this paper.

Part of this work was performed at the Hash Function Retreat, hosted by the Graz University of Technology as an initiative of the SymLab group of the ECRYPT II project. We are very grateful to Mario Lamberger, Florian Mendel, Tomislav Nad, Christian Rechberger, Vincent Rijmen and Martin Schläffer for their excellent organization of this retreat.

María Naya-Plasencia, Andrea Röck, Thomas Peyrin, Jean-Philippe Aumasson, Gaëtan Leurent and Willi Meier have obtained other, non-overlapping results on ESSENCE [13] in parallel with these results. Their paper uses different characteristics and another way of finding conforming messages.

References

1. Eli Biham and Rafi Chen. Near-Collisions of SHA-0. In Matthew K. Franklin, editor, *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 290–305. Springer, 2004.
2. Christophe De Cannière and Christian Rechberger. Finding SHA-1 Characteristics: General Results and Applications. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT*, volume 4284 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2006.
3. Itai Dinur and Adi Shamir. Side Channel Cube Attacks on Block Ciphers. Cryptology ePrint Archive, Report 2009/127, 2009. Available at: `http://eprint.iacr.org/`.
4. Stefan Dziembowski and Krzysztof Pietrzak. Leakage-Resilient Cryptography. In R. Ravi, editor, *FOCS*, pages 293–302, 2008.
5. Markus Grassl. Tables of Linear Codes and Quantum Codes, June 2008. Available at: `http://www.codetables.de/`.
6. Antoine Joux and Thomas Peyrin. Hash Functions and the (Amplified) Boomerang Attack. In Alfred Menezes, editor, *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 244–263. Springer, 2007.
7. Vlastimil Klima. Tunnels in Hash Functions: MD5 Collisions Within a Minute. Cryptology ePrint Archive, Report 2006/105, 2006. Available at: `http://eprint.iacr.org/`.
8. Jason Worth Martin. ESSENCE: A Family of Cryptographic Hashing Algorithms. Submitted to the NIST SHA-3 hash function competition. Available at: `http://www.math.jmu.edu/~martin/essence/Supporting_Documentation/essence_compression.pdf` (2009/01/20).
9. Jason Worth Martin. Personal communication, 2009.
10. Mitsuru Matsui. Linear Cryptoanalysis Method for DES Cipher. In Tor Helleseth, editor, *EUROCRYPT*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer, 1993.
11. Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1996.
12. National Institute of Standards and Technology. Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA-3) Family. *Federal Register*, 27(212):62212–62220, November 2007. Available at: `http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf` (2008/10/17).
13. María Naya-Plasencia, Andrea Röck, Thomas Peyrin, Jean-Philippe Aumasson, Gaëtan Leurent, and Willi Meier. Cryptanalysis of ESSENCE. Unpublished, 2009. Available at: `http://www.131002.net/data/papers/NRALMP09.pdf`.
14. Krzysztof Pietrzak. A Leakage-Resilient Mode of Operation. In Antoine Joux, editor, *EUROCRYPT*, volume 5479 of *Lecture Notes in Computer Science*, pages 462–482. Springer, 2009.

15. Bart Preneel. *Analysis and design of cryptographic hash functions.* PhD thesis, Katholieke Universiteit Leuven, 1993.
16. Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu. Cryptanalysis of the Hash Functions MD4 and RIPEMD. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2005.
17. Xiaoyun Wang and Hongbo Yu. How to Break MD5 and Other Hash Functions. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 19–35. Springer, 2005.
18. Xiaoyun Wang, Hongbo Yu, and Yiqun Lisa Yin. Efficient Collision Search Attacks on SHA-0. In Victor Shoup, editor, *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2005.

A Finding the Lowest Weight Difference A

We wish to find a difference A that satisfies

$$(\neg A) \wedge L(A) = 0 \tag{3}$$

and

$$\mathrm{hw}(A) \leq w \enspace , \tag{4}$$

where $\mathrm{hw}(A)$ is the number of bits set in A and w is to be as small as possible.

We proceed as follows. Let w represent the (still unknown) weight of the lowest weight difference A. We then split w into two integers w_0 and w_1, such that $w_0 + w_1 = w$ and $|w_1 - w_0| \leq 1$. Let L^{-1} represent the inverse L function, such that $L^{-1}(L(x)) = x$. Let $M(x) = L(x) \oplus x$. The design of ESSENCE guarantees that M is invertible, as L is not allowed to have any eigenvalues in the ground field.

First step: We enumerate all x where $\mathrm{hw}(x) \leq w_0$. After calculating $A = L^{-1}(x)$, we check (3) and (4).

Second step: We enumerate all y where $\mathrm{hw}(y) \leq w_1$. After calculating $A = M^{-1}(y)$, we check if (3) and (4).

Equation (3) implies that the bit positions where $L(A)$ is 1, is always a subset of bit positions where A is 1. Therefore, we only have to consider two cases: the case where the set of bit positions where $L(A)$ is 1 contains no more than w_0 elements, and the case where the set bit positions where $L(A)$ is 0 and A is 1 contains not more than w_1 elements. As $w_0 + w_1 = w$, these two steps are guaranteed to find all A that satisfy (3) and (4). If no solution is found, we increase w by one and perform the two steps again, enumerating only the new values of x and y.

The total complexity of this search is $\left(\sum_{i=0}^{w_0} C_i^{64}\right) + \left(\sum_{j=0}^{w_1} C_j^{64}\right)$. As we find $w = 17$ here, the total number of 64-bit linear function evaluations is $\left(\sum_{i=0}^{8} C_i^{64}\right) + \left(\sum_{j=0}^{9} C_j^{64}\right) \approx 2^{35}$. This calculation can be performed in less than a minute on a recent desktop computer. The solutions are shown in Table 2.

B Making F Behave as a Linear Transformation

We consider three separate cases, depending on the values of A and $L(A)$ for a particular bit position j.

Table 2. All differences A with $\mathrm{hw}(A) = 17$ that satisfy (3); there are no solutions where $\mathrm{hw}(A) < 17$ and (3)

A
2461822430680025
48C3044860D0004A
91860890C1A00094
0A001021903036C3
1400204320606D86
2800408640C0DB0C
5000810C8181B618
A001021903036C30

If $A[j] = 1$, we can enumerate all possible input conditions, such that F behaves linearly and has the required differential behavior. Because we enumerate all possibilities, we obtain an optimal result: it is not possible to add fewer than 10 linear equations. All existing solutions where 10 linear equations are added, are shown in Table 3 (for $L(A)[j] = 1$) and Table 4 (for $L(A)[j] = 0$).

If $A[j] = 0$: the differential behavior is always satisfied: if there is no input difference, there will not be an output difference either. We found that adding 6 equations is sufficient. We do not rule out the possibility that fewer than 6 equations are sufficient. The solutions we found are given in Table 4.

We will omit the index j, so that x_0 to x_{12} represent one-bit variables. The expressions $F(x_0, \ldots, x_6)$ and $F(x_6, \ldots, x_{12})$ are not added to the system of linear equations of the attack, as this is not necessary. They are only mentioned to show that their differential behavior is correct.

Table 3. Making F linear and imposing the required differential behavior for position j where $A[j] = L(A)[j] = 1$ can be done by adding no more than 10 linear equations; exactly four such solutions exist

	Solution 1	Solution 2	Solution 3	Solution 4
	$x_0 \oplus x_2 = 1$	$x_1 = 1$	$x_1 = 1$	$x_1 = 1$
	$x_1 = 0$	$x_2 \oplus x_5 = 0$	$x_2 \oplus x_5 = 0$	$x_2 = 1$
	$x_3 = 1$	$x_2 \oplus x_7 = 1$	$x_2 \oplus x_7 = 1$	$x_3 = 0$
	$x_4 = 1$	$x_2 \oplus x_8 = 0$	$x_2 \oplus x_8 = 0$	$x_4 = 1$
	$x_5 = 1$	$x_2 \oplus x_9 = 0$	$x_2 \oplus x_9 = 0$	$x_5 = 1$
	$x_7 = 0$	$x_2 \oplus x_{12} = 1$	$x_3 = 0$	$x_7 = 0$
	$x_8 = 1$	$x_3 = 0$	$x_4 = 1$	$x_8 = 1$
	$x_9 = 0$	$x_4 = 1$	$x_{10} = 0$	$x_9 = 1$
	$x_{10} = 0$	$x_{10} = 0$	$x_{11} = 0$	$x_{10} = 0$
	$x_{12} = 1$	$x_{11} = 0$	$x_{12} = 1$	$x_{11} = 0$
$F(x_0, \ldots, x_6) =$	$x_6 \oplus 1$	$x_0 \oplus x_6$	$x_0 \oplus x_6$	$x_0 \oplus x_6$
$F(x_1, \ldots, x_7) =$	$x_2 \oplus 1$	$x_2 \oplus 1$	$x_2 \oplus 1$	0
$F(x_2, \ldots, x_8) =$	0	$x_2 \oplus 1$	$x_2 \oplus 1$	0
$F(x_3, \ldots, x_9) =$	0	x_5	x_5	1
$F(x_4, \ldots, x_{10}) =$	1	1	1	1
$F(x_5, \ldots, x_{11}) =$	1	0	0	0
$F(x_6, \ldots, x_{12}) =$	0	$x_7 \oplus 1$	0	$x_{12} \oplus 1$

Table 4. Making F linear and imposing the required differential behavior for position j where $A[j] = 1$ and $L(A)[j] = 0$ can be done by adding no more than 10 linear equations; exactly one such solution exists

	Solution 1
	$x_0 \oplus x_2 = 0$
	$x_1 = 0$
	$x_3 = 1$
	$x_4 = 1$
	$x_5 = 1$
	$x_7 = 0$
	$x_8 = 1$
	$x_9 = 0$
	$x_{10} = 0$
	$x_{12} = 1$
$F(x_0, \ldots, x_6) =$	1
$F(x_1, \ldots, x_7) =$	$x_2 \oplus 1$
$F(x_2, \ldots, x_8) =$	0
$F(x_3, \ldots, x_9) =$	0
$F(x_4, \ldots, x_{10}) =$	1
$F(x_5, \ldots, x_{11}) =$	1
$F(x_6, \ldots, x_{12}) =$	0

Table 5. Making F linear for position j where $A[j] = L(A)[j] = 0$ can be done by adding no more than 6 linear equations; at least six such solutions exist

	Solution 1	Solution 2	Solution 3	Solution 4	Solution 5	Solution 6
	$x_3 = 0$	$x_3 = 0$	$x_3 = 1$	$x_4 = 0$	$x_4 = 0$	$x_4 = 0$
	$x_4 = 0$	$x_4 = 0$	$x_4 = 1$	$x_5 = 1$	$x_5 = 1$	$x_5 = 1$
	$x_5 = 1$	$x_5 = 1$	$x_5 = 1$	$x_6 = 1$	$x_6 = 1$	$x_6 = 1$
	$x_6 = 0$	$x_6 = 1$	$x_6 = 1$	$x_7 = 0$	$x_7 = 0$	$x_7 = 0$
	$x_7 = 1$	$x_7 = 1$	$x_7 = 1$	$x_8 = 1$	$x_8 = 1$	$x_8 = 1$
	$x_9 = 1$	$x_8 = 1$	$x_8 = 1$	$x_9 = 0$	$x_{10} = 0$	$x_{11} = 1$
$F(x_1, \ldots, x_7) =$	$x_1 \oplus 1$	x_2	x_1	$x_1 \oplus x_2 \oplus 1$	$x_1 \oplus x_2 \oplus 1$	$x_1 \oplus x_2 \oplus 1$
$F(x_2, \ldots, x_8) =$	$x_2 \oplus x_8 \oplus 1$	$x_2 \oplus 1$	x_2	$x_3 \oplus 1$	$x_3 \oplus 1$	$x_3 \oplus 1$
$F(x_3, \ldots, x_9) =$	$x_8 \oplus 1$	$x_9 \oplus 1$	x_9	0	x_9	x_9
$F(x_4, \ldots, x_{10}) =$	x_8	0	$x_9 \oplus x_{10} \oplus 1$	$x_{10} \oplus 1$	$x_9 \oplus 1$	$x_9 \oplus x_{10} \oplus 1$
$F(x_5, \ldots, x_{11}) =$	$x_8 \oplus x_{10} \oplus 1$	$x_{10} \oplus x_{11} \oplus 1$	$x_{10} \oplus x_{11} \oplus 1$	$x_{10} \oplus 1$	$x_9 \oplus 1$	$x_9 \oplus x_{10} \oplus 1$

C A Message Pair for the First Nine Rounds

We give a message pair that satisfies the first 9 rounds of the characteristic of Table 1 in Table 6.

Table 6. A message pair satisfying the first 9 rounds of the characteristic of Table 1

i	m_i	m'_i	$m_i \oplus m'_i$
0	FFFFFFFFFFFFFFFF	FFFFFFFFFFFFFFFF	0000000000000000
1	1A001021983836CB	1A001021983836CB	0000000000000000
2	5809832A1DEA2458	5809832A1DEA2458	0000000000000000
3	8AEF5FEBEB9FDAAB	8AEF5FEBEB9FDAAB	0000000000000000
4	32F9D8578015D297	32F9D8578015D297	0000000000000000
5	0D031372423B91AC	0D031372423B91AC	0000000000000000
6	B804AC08CD97E348	B804AC08CD97E348	0000000000000000
7	E8BB8E649DC3B35F	E2BB9E450DF3859C	0A001021903036C3

D The Feedback Function *F*

We denote the field of two elements by $\mathbb{F}_2$. The nonlinear feedback function, F, of ESSENCE-224/256 (respectively ESSENCE-384/512) takes seven 32-bit (respectively 64-bit) input words and outputs a single 32-bit (respectively 64-bit) word as follows:

$$\begin{aligned} F(a,b,c,d,e,f,g) = \; & abcdefg + abcdef + abcefg + acdefg + \\ & abceg + abdef + abdeg + abefg + \\ & acdef + acdfg + acefg + adefg + \\ & bcdfg + bdefg + cdefg + \\ & abcf + abcg + abdg + acdf + adef + \\ & adeg + adfg + bcde + bceg + bdeg + cdef + \\ & abc + abe + abf + abg + acg + adf + \\ & adg + aef + aeg + bcf + bcg + bde + \\ & bdf + beg + bfg + cde + cdf + def + \\ & deg + dfg + \\ & ad + ae + bc + bd + cd + \\ & ce + df + dg + ef + fg + \\ & a + b + c + f + 1 \ , \end{aligned} \tag{5}$$

where the multiplication and addition are taken in $\mathbb{F}_2$ (i.e., they are the same as bitwise XOR and bitwise AND, respectively).

E Distinguishing Attacks on the Full 32-Round ESSENCE-256

The attacks described in Sect. 6.2 can be easily extended to the full ESSENCE-256 block cipher. Let us suppose the key k and the plaintext are related such that after 18 rounds,

$r_0[0] = k_0[0]$. Given this, using similar arguments as those used to derive (2), we obtain that at the end of 32 rounds, if $L(r_7)[0] = 0$, then

$$Pr[r_6[0] = 0] = \frac{1}{2} + \frac{1}{2^7} \ . \tag{6}$$

We can thus construct a distinguisher by collecting 2^{17} outputs $r_6[0]$, after 32 rounds, generated by as many keys (so that the samples are independent) given that after 18 rounds, $k_0[0] = r_0[0]$. In other words, the adversary first tests whether $k_0[0] = r_0[0]$ after 18 rounds. If this condition is satisfied, she collects the output $r_6[0]$ after 32 rounds provided $L(r_7)[0] = 0$. Therefore, this constitutes a known-key distinguishing attack which one may view as an attack on a large set of weak keys. Alternatively, the attack scenario may be such that two bits of the internal state after 18 rounds are leaked to the adversary. A similar assumption was made in [3], as a model for certain side-channel attacks. More generally, this scenario is captured by the notion of leakage resilience [4, 14], i.e., security when "even a bounded amount of arbitrary (adversarially chosen) information on the internal state (...) is leaked during computation" [4]. Although this assumption leads to trivial attacks (e.g., observe the full internal state of AES at the penultimate rounds), it assists to evaluate security against a wider range of adversaries, and to better understand the resilience of algorithms against "extreme" adversaries.

Since the condition $k_0[0] = r_0[0]$ (after 18 rounds) holds with 0.5 probability, the attacker would need to examine with $2^{17} \cdot 2 = 2^{18}$ randomly generated keys to mount the distinguishing attack with a success probability of 0.9772.

It is easy to see that distinguishers of the same complexity can be built by collecting any other bit of r_6 (after 32 rounds) because F operates in a bitsliced manner. As in Sect. 6.5, when the attacker can observe both the chaining value input and the compression function output, the above distinguishers can be applied onto the compression function as well.

F Key-Recovery Attacks on 32-Round ESSENCE

In Appendix E, we extended the distinguisher on 14-round ESSENCE-256 to 32 rounds by selecting plaintexts based upon the intermediate value of $r_0[j]$ and $k_0[j]$ at round 18. This result may be viewed in terms of a known plaintext key-recovery attack against a vulnerable implementation of the ESSENCE-256 block cipher. Let us say that we are attacking such an implementation of the 32-round ESSENCE-256 block cipher where through some means (side-channel analysis, cache pollution, etc.) we can read bit j of r_0 after 18 rounds. Like in Sect. 6.6, we focus on a subset of 2^{14} plaintexts where $r_0[j] = 0$ (or 1) for all 2^{14} texts after 18 rounds. Applying the same analysis as in Sect. 6.6 to the remaining 14 rounds gives us the value of $k_0[j]$ at round 18. If our vulnerable implementation allows us to read all the bit positions of r_0, then with probability 0.48, we can recover the full key-word k_0 at round 18. Since the key schedule is a bijection (and easily invertible) we can recover the original key with minimal effort. Again, a similar analysis can be applied to the other members of the ESSENCE family of block ciphers.