Attacks on Hash Functions based on Generalized Feistel Application to Reduced-Round *Lesamnta* and *SHAvite-3*$_{512}$

Charles Bouillaguet[1], Orr Dunkelman[2],
Gaëtan Leurent[1], and Pierre-Alain Fouque[1]

[1] Département d'Informatique
École normale supérieure
45 Rue D'Ulm
75320 Paris, France
{charles.bouillaguet, gaetan.leurent, pierre-alain.fouque}@ens.fr
[2] Faculty of Mathematics and Computer Science
Weizmann Institute of Science
P.O. Box 26
Rehovot 76100, Israel
orr.dunkelman@weizmann.ac.il

Abstract. In this paper we study the strength of two hash functions which are based on Generalized Feistels. We describe a new kind of attack based on a cancellation property in the round function. This new technique allows to efficiently use the degrees of freedom available to attack a hash function. Using the cancellation property, we can avoid the non-linear parts of the round function, at the expense of some freedom degrees.
Our attacks are mostly independent of the round function in use, and can be applied to similar hash functions which share the same structure but have different round functions. We start with a 22-round generic attack on the structure of *Lesamnta*, and adapt it to the actual round function to attack 24-round *Lesamnta* (the full function has 32 rounds). We follow with an attack on 9-round *SHAvite-3*$_{512}$ which also works for the tweaked version of *SHAvite-3*$_{512}$.

1 Introduction

Many block ciphers and hash functions are based on generalized Feistel constructions. In this paper we treat such generalized Feistel constructions and especially concentrate on the case where an n-bit round function is used in a $4n$-bit structure. Two of these constructions, shown at Figure 1,[1] used in the *Lesamnta* and the *SHAvite-3*$_{512}$ hash functions, are the main focus of this paper.

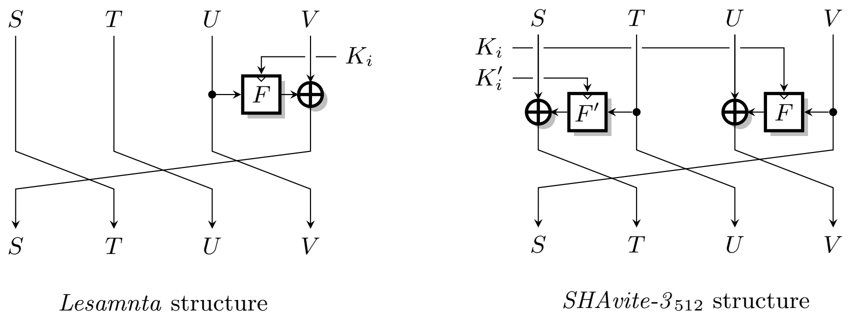


Fig. 1. The Generalized Feistel Constructions Studied in this paper

While in the ideal Luby-Rackoff case, the round functions are independent random functions, in practice, most round functions $F(k, x)$ are usually defined as $P(k \oplus x)$, where P is a fixed

[1] Note that the direction of the rotation in the Feistel structure is not really important: changing the rotation is equivalent to considering decryption instead of encryption.

permutation (or function). Hence, we introduce several attacks which are based on *cancellation property*: if the fixed function P accepts twice the same input, it produces twice the same output. In a hash function setting, as there is no secret key, the adversary may actually make sure that the inputs are the same.

For *Lesamnta* we start with generic attacks that work independent of the actual P in use, but then use the specific properties of *Lesamnta*'s round functions to offer better attacks. The attack on *SHAvite-3*$_{512}$ is a more complicated one, following the more complex round functions (and the structure which uses two functions in each round), but at the same time, is of more interest as *SHAvite-3*$_{512}$ is still a SHA3 candidate.

1.1 Overview of the Attacks

Our attacks are based on a partial preimage attack, *i.e.* we can construct specific inputs where part of the output H is equal to a target value $\overline{H}$. To achieve such a partial preimage attack, we use truncated differentials built with the cancellation property, and we express the constraints needed on the state of the Feistel network in order to have the cancellation with probability one. We use degrees of freedom in the inputs of the compression function to satisfy those constraints. Then, we can compute some part of the output as a function of some of the remaining degrees of freedom, and try to invert the equation. The main idea is to obtain a simple equation that can be easily inverted using cancellations to limit the diffusion.

A partial preimage attack on the compression function allows to choose k bits of the output for a cost of 2^t (with $t < k$), while the remaining $n - k$ bits are random. We can use such an attack on the compression function to target the hash function itself, in several scenarios.

Preimage Attacks By repeating such an attack 2^{n-k} times, we can obtain a full preimage attack on the compression function, with complexity 2^{n+t-k}. This preimage attack on the compression function can be used for a second preimage attack on the hash function with complexity $2^{n+(t-k)/2}$ using a standard unbalanced meet-in-the middle [8]. Note that $2^{n+(t-k)/2} < 2^n$ if $t < k$.

Moreover, we point out that *Lesamnta* is built following the Matyas-Meyer-Oseas construction, *i.e.* the chaining value is used as a key, and the message enters the Feistel rounds. Since our partial preimage attack does not use degrees of freedom in the key (we only need the key to be known, not chosen), we can use a chaining value reached from the IV as the key. We have to repeat the partial preimage attack with many different keys in order to build a full preimage, but we can use a first message block to randomize the key. This gives a second preimage attack on the hash function with complexity 2^{t+n-k}.

Collision Attacks The partial preimage attack can also be used to find collisions in the compression function. By generating $2^{(n-k)/2}$ inputs where k bits of the output are fixed to a common value, we expect a collision thanks to the birthday paradox. This collision attack on the compression function costs $2^{t+(n-k)/2}$. If $t < k/2$, this is more efficient than a generic birthday attack on the compression function.

If the compression function is built with the Matyas-Meyer-Oseas mode, like *Lesamnta*, this attack translates to a collision attack on the hash function with the same complexity. However, if the compression function follows the Davies-Meyer mode, like *SHAvite-3*, this does not translate to an attack on the hash function.

1.2 Our results

The first candidate for the technique is the *Lesamnta* hash function. The best known generic attack against this structure is a 16-round attack by Mendel described in the submission document of *Lesamnta* [6]. Using a cancellation property, we extend this attack to a generic attacks on 22-round *Lesamnta*. The attack allows to fix one of the output words for an amortized cost of 1, which gives collisions in time $2^{3n/8}$ and second preimages in time $2^{3n/4}$ for *Lesamnta-n*. Moreover, the preimage attack can be extended to 24 rounds using $2^{n/4}$ memory. We follow with adaptations of the 24-round attacks without memory using specific properties of *Lesamnta*'s round function.

The second target for our technique is the hash function *SHAvite-3*$_{512}$. We show a 9-round attack using a cancellation property on the generalized Feistel structure of *SHAvite-3*$_{512}$. The attack also works for the tweaked version of *SHAvite-3*$_{512}$, and allows fixing one out of the four output words. This allows a second preimage attack on 9-round *SHAvite-3*$_{512}$ that takes about 2^{448} time. Note that this attack has recently been improved in a follow-up work [5]. First a new technique was used to add one extra round at the beginning, leading to attacks on 10 rounds of the compression function. Second, a pseudo-attack against the full *SHAvite-3*$_{512}$ is described, using additional degrees of freedom in the salt input. The follow-up work has been published first because of calendar issues, but it is heavily based on this work which was available as a preprint to the authors of [5]. Moreover, in this paper, we describe a more efficient way to find a suitable key for the attack, which improves the 10-round attack of [5].

Finally, we show some applications to block ciphers. We describe an integral attack on 21 rounds of the inner block cipher of *Lesamnta* using a cancellation property, and a new truncated differential for *SMS4*.

The paper is organized as follows. Section 2 explains the basic idea of our cancellation attacks. Our results on *Lesamnta* are presented in Section 3, while application to *SHAvite-3*$_{512}$ is discussed in Section 4. Finally, application to the inner block cipher of *Lesamnta* is shown in Appendix A, while an attack on *SMS4* is described in Appendix B.These results are summarized in Section 5.

2 The Cancellation Property

In this paper we apply cancellation cryptanalysis to generalized Feistel schemes. The main idea of this technique is to impose constraints on the values of the state in order to limit the diffusion in the Feistel structure. When attacking a hash function, we have many degrees of freedom from the message and the chaining value, and it is important to find efficient ways to use those degrees of freedom.

Table 1. Cancellation property on *Lesamnta*.
On the left side, we have full diffusion after 9 rounds.
On the right side, we use the cancellation property to control the diffusion of the differences.

i	S_i	T_i	U_i	V_i		S_i	T_i	U_i	V_i	
0	x	-	-	-		x	-	-	-	
1	-	x	-	-		-	x	-	-	
2	-	-	x	-		-	-	x	-	
3	y_1	-	-	x	$x \to y_1$	y	-	-	x	$\underline{x \to y}$
4	x	y_1	-	-		x	y	-	-	
5	-	x	y_1	-		-	x	y	-	
6	z	-	x	y_1	$y_1 \to z$	z	-	x	y	$y \to z$
7	y'	z	-	x	$x \to y_2,\ y' = y_1 \oplus y_2$	$\underline{-}$	z	-	x	$\underline{x \to y}$
8	x	y'	z	-		x	$\underline{-}$	z	-	
9	w	x	y'	z	$z \to w$	w	x	$\underline{-}$	z	$z \to w$

Table 1 shows the diffusion of a single difference in *Lesamnta*. After 9 rounds, all the state words are active. However, we note that if the transitions $x \to y_1$ at rounds 3 and $x \to y_2$ at round 7 actually go to the *same* y, *i.e.* $y_1 = y_2$, this limits the diffusion. In the ideal case, the round functions are all independent, and the probability of getting the same output difference is very small. However, in practice, the round functions are usually all derived from a single fixed permutation (or function). Therefore, if we add some constraints so that the input *values* of the fixed permutation at round 3 and 7 are the same, then we have the same output values, and therefore the same output difference with probability one. This is the *cancellation property*.

Table 2. Cancellation property on *SHAvite-3*$_{512}$.
On the left side, we have full diffusion after 4 rounds.
On the right side, we use the cancellation property to control the diffusion.

i	S_i	T_i	U_i	V_i		S_i	T_i	U_i	V_i	
0	x	-	-	-		x	-	-	-	
1	-	x	-	-		-	x	-	-	
2	-	y_1	x	-	$x \to y_1$	-	y	x	-	$\underline{x \to y}$
3	-	z	y_1	x	$y_1 \to z$	-	z	y	x	$y \to z$
4	x	w	z	y'	$x \to y_2,\ y' = y_1 \oplus y_2$	x	w	z	$\underline{-}$	$\underline{x \to y}$

Similarly, Table 2 shows the diffusion of a difference in *SHAvite-3*$_{512}$. If the transitions $x \to y_1$ at round 2 and $x \to y_2$ at round 4 actually go to the *same* y, we can limit the diffusion.

Our attacks use an important property of the Feistel schemes of *Lesamnta* and *SHAvite-3*$_{512}$: the diffusion is relatively slow. When a difference is introduced in the state, it takes several rounds to affect the full state and two different round functions can receive the same input difference x. Note that the slow diffusion of *Lesamnta* is the basis of a 16-round attack in [6] (recalled in Section 3.2), and the slow diffusion of *SHAvite-3*$_{512}$ gives a similar 8-round attack [4]. Our new attacks can be seen as extensions of those.

We now describe how to enforce conditions of the state so as to have this cancellation with probability 1. Our attacks are independent of the round function, as long as all the round functions are derived from a single function as $F_i(X_i) \triangleq F(K_i \oplus X_i)$.

2.1 Generic Properties of $F_i(X_i) = F(K_i \oplus X_i)$

We assume that the round functions F_i are built by applying a fixed permutation (or function) F to $K_i \oplus X_i$, where K_i is a round key and X_i is the state input. This practice is common in many primitives like DES, *SMS4*, GOST, or *Lesamnta*.

This implies the followings, for all i, j, k:

(i) $\exists c_{i,j} : \forall x,\ F_i(x \oplus c_{i,j}) = F_j(x)$.
(ii) $\forall \alpha,\ \#\big\{x : F_i(x) \oplus F_j(x) = \alpha\big\}$ is even.
(iii) $\bigoplus_x F_k\big(F_i(x) \oplus F_j(x)\big) = 0$.

Property (i) is the basis of our cancellation attack. We refer to it as the *cancellation property*. It states that if the inputs of two round functions are related by a specific fixed difference, then the outputs of both rounds are equal. The reminder of the paper is exploring this property.

Properties (ii) and (iii) can be used in an integral attack, as shown in Appendix A. Note that Property (ii) is a well known fact from differential cryptanalysis.

Proof.

(i) Set $c_{ij} = K_i \oplus K_j$.
(ii) If $K_i = K_j$, then $\forall x,\ F_i(x) \oplus F_j(x) = 0$. Otherwise, let x be such that $F_i(x) \oplus F_j(x) = \alpha$. Then $F_i(x \oplus K_i \oplus K_j) \oplus F_j(x \oplus K_i \oplus K_j) = F_j(x) \oplus F_i(x) = \alpha$. Therefore x is in the set if and only if $x \oplus K_i \oplus K_j$ is in the set, and all the elements can be grouped in pairs.
(iii) Each term $F_k(\alpha)$ in the sum appears an even number of times following (ii). $\square$

Table 3. Values of the Registers for Five Rounds of *Lesamnta*.

i	S_i	T_i	U_i	V_i
2	a	b	c	d
3	$F_2(c) \oplus d$	a	b	c
4	$F_3(b) \oplus c$	$F_2(c) \oplus d$	a	b
5	$F_4(a) \oplus b$	$F_3(b) \oplus c$	$F_2(c) \oplus d$	a
6	$F_5(F_2(c) \oplus d) \oplus a$	$F_4(a) \oplus b$	$F_3(b) \oplus c$	$F_2(c) \oplus d$
7	$\cancel{F_6(F_3(b) \oplus \underline{c})} \oplus \cancel{F_2(\underline{c})} \oplus d$	$F_5(F_2(c) \oplus d) \oplus a$	$F_4(a) \oplus b$	$F_3(b) \oplus c$

2.2 Using the Cancellation Property

To better explain the cancellation property, we describe how to use it with the truncated differential of Table 1. In Table 3, we show the *values* of the registers during the computation of the truncated differential, starting at round 2 with $(S_2, T_2, U_2, V_2) = (a, b, c, d)$. To use the cancellation property, we want to make S_7 independent of c. Since we have $S_7 = F_6(F_3(b) \oplus \underline{c}) \oplus F_2(\underline{c}) \oplus d$, we can cancel the highlighted terms using property (i). The dependency of S_7 on c disappears if $F_3(b) = K_2 \oplus K_6$, *i.e.* if $b = F_3^{-1}(K_2 \oplus K_6)$:

$$\begin{aligned} S_7 &= F_6(F_3(b) \oplus c) \oplus F_2(c) \oplus d \\ &= F(K_6 \oplus F_3(b) \oplus c) \oplus F(K_2 \oplus c) \oplus d \\ &= F(K_2 \oplus c) \oplus F(K_2 \oplus c) \oplus d = d. \end{aligned}$$

Therefore, we can put any value c in U_2, and it does not affect S_7 as long as we fix the value of T_2 to be $F^{-1}(K_2 \oplus K_6) \oplus K_3$. Note that in a hash function, we can compute $F^{-1}(K_2 \oplus K_6) \oplus K_3$ since the keys are known to the adversary (or controlled by him), and we can choose to have this value in T_2.

This shows the three main requirements of our cancellation attacks:

- The generalized Feistel structures we study have a relatively slow diffusion. Therefore, the same difference can be used as the input difference of two different round functions.
- The round functions are built from a fixed permutation (or a fixed function), using a small round key. This differs from the ideal Luby-Rackoff case where all round functions are chosen independently at random.
- In a hash function setting the key is known to the adversary, and he can control some of the inner values.

Note that some of these requirements are not strictly necessary. For example, we show a 21-round integral attack on *Lesamnta*, without knowing the keys in Section A. Moreover, in Section 4 we show attacks on 9-round *SHAvite-3*$_{512}$, where the round functions use more keying material.

3 Application to *Lesamnta*

3.1 A Short Description of *Lesamnta*

Lesamnta is a hash function proposal by Hirose, Kuwakado, and Yoshida as a candidate in the SHA-3 competition [6]. It is based on a 32-round unbalanced Feistel scheme with four registers used in MMO mode. The key schedule is also based on a similar Feistel scheme. The round function can be written as:

$$S_{i+1} = V_i \oplus F\left(U_i \oplus K_i\right) \qquad T_{i+1} = S_i \qquad U_{i+1} = T_i \qquad V_{i+1} = U_i$$

Alternatively, we can write it with a single register X, equivalent to the original S

$$X_{i+4} = X_i \oplus F\left(X_{i+1} \oplus K_{i+3}\right)$$

where $K_0, \ldots, K_{31}$ are round keys derived from the chaining value, and the state register X is initialized with the message in X_{-3}, X_{-2}, X_{-1}, X_0. The output of the compression function is $X_{-3} \oplus X_{29}, X_{-2} \oplus X_{30}, X_{-1} \oplus X_{31}, X_0 \oplus X_{32}$.

3.2 Previous Results on *Lesamnta*

The best known attack on *Lesamnta* is the self-similarity attack of [2]. Following this attack, the designers have tweaked *Lesamnta* by changing the round constants [12]. In this paper we consider attacks that work with any round constants, and thus are applicable to the tweaked version as well.

Several attacks on reduced-round *Lesamnta* are presented in the submission document [6]. A series of 16-round attacks for collisions and (second) preimage attacks are presented, all of which are based on the following 16-round truncated differential with probability 1:

i	S_i	T_i	U_i	V_i	
0	x_1	x_2	x_3	$x \oplus x_4$	
1	x	x_1	x_2	x_3	$x_3 \to x_4$
2	-	x	x_1	x_2	$x_2 \to x_3$
3	-	-	x	x_1	$x_1 \to x_2$
4	-	-	-	x	$x \to x_1$
5	x	-	-	-	
6	-	x	-	-	
7	-	-	x	-	
8	?	-	-	x	
9	x	?	-	-	
11	-	x	?	-	
11	?	-	x	?	
12	?	?	-	x	
13	x	?	?	-	
14	?	x	?	?	
15	?	?	x	?	
16	?	?	?	x	
FF	?	?	?	x_4	

where[2]

$$\begin{aligned}
x_3 &= M_2 \oplus F^{-1}(F(M_2 \oplus K_0) \oplus x_4) \oplus K_0, && \textit{i.e. } F_0(U_0) \oplus F_0(U_0 \oplus x_3) = x_4 \\
x_2 &= M_1 \oplus F^{-1}(F(M_1 \oplus K_1) \oplus x_3) \oplus K_1, && \textit{i.e. } F_1(U_1) \oplus F_1(U_1 \oplus x_2) = x_3 \\
x_1 &= M_0 \oplus F^{-1}(F(M_0 \oplus K_2) \oplus x_2) \oplus K_2, && \textit{i.e. } F_2(U_2) \oplus F_2(U_2 \oplus x_1) = x_2 \\
x &= (M_3 \oplus F(M_2 \oplus K_0)) \oplus F^{-1}(F(M_3 \oplus K_3 \oplus F(M_2 \oplus K_0)) \oplus x_1) \oplus K_3, \\
& && \textit{i.e. } F_3(U_3) \oplus F_3(U_3 \oplus x\ ) = x_1
\end{aligned}$$

and M_i are the corresponding message words of the message block.

This truncated differential allows fixing the fourth output word to a constant value determined by the adversary using two queries to the compression function. One first picks a random

[2] Note that the expression given for x in [6] is incorrect.

message m, and computes the difference x_4 between the desired value $\overline{H}_4$ and the actual value $H_4 = V_o \oplus V_{16}$ of the fourth output word. Since the key is known, it is easy to compute x_3 from x_4, and similarly x_2, x_1 and x, as shown above. Then, by picking $m' = m \oplus (x_1, x_2, x_3, x \oplus x_4)$, it is assured that the fourth output word is equal to $\overline{H}_4$.

This allows a collision attack (of expected time complexity 2^{97}) and second preimage attack (of expected time complexity 2^{193}). We note that this property is independent of F (as long as F is bijective), and can be applied even when the round functions are ideal independent permutations.

In the next sections we show new attacks using the cancellation property. We first show some attacks that are generic in F, as long as the round functions are defined as $F_i(X_i) = F(K_i \oplus X_i)$, and then improved attacks using specific properties of the round functions of *Lesamnta*.

3.3 Generic Attacks

Our attacks are based on the differential of Table 4, which is an extension of the differential of Table 1. In this differential we use the cancellation property three times to control the diffusion. Note that we do not have to specify the values of y, z, w, r and t. This specifies a truncated differential for *Lesamnta*: starting from a difference $(x, -, -, -)$, we reach a difference $(?, ?, ?, x_1)$ after 22 rounds. In order to use this truncated differential in our cancellation attack, we use two important properties: first, by adding constraints on the state, the truncated differential is followed with probability 1; second, the transition $x \to x_1$ is known because the key and values are known. Therefore, we can actually adjust the value of the last output word.

Table 4. Cancellation Property on 22 Rounds of *Lesamnta*

i	S_i	T_i	U_i	V_i	
0	x	-	-	-	
1	-	x	-	-	
2	-	-	x	-	
3	y	-	-	x	$\underline{x \to y}$
4	x	y	-	-	
5	-	x	y	-	
6	z	-	x	y	$y \to z$
7	$\underline{-}$	z	-	x	$\underline{x \to y}$
8	x	$\underline{-}$	z	-	
9	w	x	$\underline{-}$	z	$\underline{z \to w}$
10	z	w	x	$\underline{-}$	
11	x_1	z	w	x	$x \to x_1$
12	r	x_1	z	w	$w \to x \oplus r$
13	$\underline{-}$	r	x_1	z	$\underline{z \to w}$
14	?	$\underline{-}$	r	x_1	
15	$x_1 + t$	?	$\underline{-}$	r	$\underline{r \to t}$
16	r	$x_1 + t$	?	$\underline{-}$	
17	?	r	$x_1 + t$	?	
18	?	?	r	$x_1 + t$	
19	$\underline{x_1}$	?	?	r	$\underline{r \to t}$
20	?	$\underline{x_1}$	?	?	
21	?	?	$\underline{x_1}$	?	
22	?	?	?	$\underline{x_1}$	
FF	?	?	?	x_1	

In order to express the constraints that we need for the cancellation properties, we look at the *values* of the registers for this truncated differential. In Table 5, we begin at round 2 with $(S_2, T_2, U_2, V_2) = (a, b, c, d)$, and we compute the state values up to round 19. This is an extension of the values computed in Table 3.

We can see that we have

$$X_{19} = F(c \oplus \alpha) \oplus \beta,$$

where

$$\alpha = K_{10} \oplus F_7(F_4(a) \oplus b) \oplus F_3(b) \qquad \text{and} \qquad \beta = d$$

provided that (a, b, d) is the unique triplet satisfying the following cancellation conditions:

Round 7: we have $F_6(F_3(b) \oplus \underline{c}) \oplus F_2(\underline{c})$. They cancel if:
$F_3(b) = c_{2,6} = K_2 \oplus K_6$ *i.e.* $b = F_3^{-1}(K_2 \oplus K_6)$
Round 13: we have $F_{12}(F_9(d) \oplus \underline{F_5(F_2(c) \oplus d) \oplus a}) \oplus F_8(\underline{F_5(F_2(c) \oplus d) \oplus a})$. They cancel if:
$F_9(d) = c_{8,12} = K_8 \oplus K_{12}$ *i.e.* $d = F_9^{-1}(K_8 \oplus K_{12})$
Round 19: we have $F_{18}(F_{15}(F_4(a) \oplus b) \oplus \underline{X_{12}}) \oplus F_{14}(\underline{X_{12}})$. They cancel if:
$F_{15}(F_4(a) \oplus b) = c_{14,18} = K_{14} \oplus K_{18}$ *i.e.* $a = F_4^{-1}(F_{15}^{-1}(K_{14} \oplus K_{18}) \oplus b)$

Note that a, b, d and α, β are uniquely determined from the subkeys. Hence, one can set X_{19} to any desired value X_{19}^* by setting $c = F^{-1}(X_{19}^* \oplus \beta) \oplus \alpha$.

Table 5. Values of the Register for the 22-round Cancellation Property of *Lesamnta*

i	$X_i (= S_i)$
-1	d
0	c
1	b
2	a
3	$F_2(c) \oplus d$
4	$F_3(b) \oplus c$
5	$F_4(a) \oplus b$
6	$F_5(F_2(c) \oplus d) \oplus a$
7	$\cancel{F_6(F_3(b) \oplus \underline{c})} \oplus \cancel{F_2(\underline{c})} \oplus d$
8	$F_7(F_4(a) \oplus b) \oplus F_3(b) \oplus c$
9	$F_8(F_5(F_2(c) \oplus d) \oplus a) \oplus F_4(a) \oplus b$
10	$F_9(d) \oplus F_5(F_2(c) \oplus d) \oplus a$
11	$F_{10}(F_7(F_4(a) \oplus b) \oplus F_3(b) \oplus c) \oplus d$
12	$F_{11}(F_8(F_5(F_2(c) \oplus d) \oplus a) \oplus F_4(a) \oplus b) \oplus F_7(F_4(a) \oplus b) \oplus F_3(b) \oplus c$
13	$\cancel{F_{12}(F_9(d) \oplus \underline{F_5(F_2(c) \oplus d) \oplus a})} \oplus \cancel{F_8(\underline{F_5(F_2(c) \oplus d) \oplus a})} \oplus F_4(a) \oplus b$
14	?
15	$F_{14}(X_{12}) \oplus F_{10}(F_7(F_4(a) \oplus b) \oplus F_3(b) \oplus c) \oplus d$
16	$F_{15}(F_4(a) \oplus b) \oplus X_{12}$
17	?
18	?
19	$\cancel{F_{18}(F_{15}(F_4(a) \oplus b) \oplus \underline{X_{12}})} \oplus \cancel{F_{14}(\underline{X_{12}})} \oplus F_{10}(F_7(F_4(a) \oplus b) \oplus F_3(b) \oplus c) \oplus d$

22-round Attacks The truncated differential of Table 4 can be used to attack 22-round *Lesamnta*. We start with the state at round 2 $(S_2, T_2, U_2, V_2) = (a, b, c, d)$ satisfying the cancellation properties, and we can compute how the various states depend on c, as shown in Table 6. A dash (-) is used to denote a value that is independent of c. We know exactly how c affects the last output word, and we can select c in order to get a specific value at the output. Suppose we are given a set of subkeys, and a target value $\overline{H}$ for the fourth output word. Then the attack proceeds as follows:

1. Set a, b, and d to the values that allow the cancellation property.
   Then we have $V_0 \oplus V_{22} = \eta \oplus F(c \oplus \alpha) \oplus \beta$, as shown in Table 6.

Table 6. Collision and Preimage Characteristic for the 22-Round Attack

i	S_i	T_i	U_i	V_i
0	c	-	-	η
1	-	c	-	-
2	-	-	c	-
2–19	Repeated Cancellation Property: Table 5			
19	$F(c \oplus \alpha) \oplus \beta$	?	?	?
20	?	$F(c \oplus \alpha) \oplus \beta$	?	?
21	?	?	$F(c \oplus \alpha) \oplus \beta$	?
22	?	?	?	$F(c \oplus \alpha) \oplus \beta$
FF	?	?	?	$\eta \oplus F(c \oplus \alpha) \oplus \beta$

η, α and β can be computed from a, b, d and the key:
$\eta = b \oplus F_0(a \oplus F_3(d))$, $\alpha = K_{11} \oplus F_8(F_5(a) \oplus b) \oplus F_4(b)$, $\beta = d$.

Table 7. Computing Values Backwards from the State $(S_4, T_4, U_4, V_4) = (a, b, c, d)$

i	X_i
-3	$d \oplus F_0(c \oplus F_1(b \oplus F_2(a \oplus F_3(d))))$
-2	$c \oplus F_1(b \oplus F_2(a \oplus F_3(d)))$
-1	$b \oplus F_2(a \oplus F_3(d))$
0	$a \oplus F_3(d)$
1	d
2	c
3	b
4	a

$V_0 = X_{-3} = \lambda \oplus F_0(c \oplus \gamma)$, with $\lambda = d$

2. Compute c as $F^{-1}(\overline{H} \oplus \eta \oplus \beta) \oplus \alpha$.
3. This sets the state at round 2: $(S_2, T_2, U_2, V_2) \triangleq (a, b, c, d)$.
   With this state, we have $V_0 \oplus V_{22} = \overline{H}$.
4. Compute the round function backwards up to round 0, to get the input $(S_0, T_0, U_0.V_0)$.

This costs less than one compression function call, and does not require any memory.

For a given chaining value (*i.e.* a set of subkeys), this algorithm can only output one message. To build a full preimage attack or a collision attack on the compression function, this has to be repeated with random chaining values. Since the attack works for any chaining value, we can build attacks on the hash function using a prefix block to randomize the chaining value. This gives a collision attack with complexity 2^{96} (2^{192} for *Lesamnta*-512), and a second-preimage attack with complexity 2^{192} (2^{384} for *Lesamnta*-512).

24-round Attacks We can add two rounds at the beginning of the truncated differential at the cost of some memory. The resulting 24-round differential is given in Table 8. The output word we try to control is equal to $F(c \oplus \gamma) \oplus F(c \oplus \alpha)$, for some constants α, and γ that depend on the chaining value (note that $\beta = \lambda$ in Table 8). We define a family of functions $h_\mu(x) = F(x) \oplus F(x \oplus \mu)$, and for a given target value $\overline{H}$, we tabulate $\varphi_{\overline{H}}(\mu) = h_\mu^{-1}(\overline{H})$. For each μ, $\varphi_{\overline{H}}(\mu)$ is a possibly empty set, but the average size is one (the non-empty values form a partition of the input space). In the special case where $\overline{H} = 0$, $\varphi_0(\mu)$ is empty for all $\mu \neq 0$, and $\varphi_0(0)$ is the full space.

We store $\varphi_{\overline{H}}$ in a table of size $2^{n/4}$, and we can compute it in time $2^{n/4}$ by looking for values such that $F(x) \oplus F(y) = \overline{H}$ (this gives $\varphi_{\overline{H}}(x \oplus y) = x$). Using this table, we are able to choose one output word just like in the 22-round attack.

We start with a state $(S_4, T_4, U_4, V_4) = (a, b, c, d)$ such that a, b, d satisfy the cancellation conditions, and we compute α, β, γ, λ. If we use $c = u \oplus \alpha$, where $u \in \varphi_{\overline{H}}(\alpha \oplus \gamma) = h^{-1}_{\alpha \oplus \gamma}(\overline{H})$,

Table 8. Collision and Preimage Path for the 24-round Attack

i	S_i	T_i	U_i	V_i
0	-	-	$c \oplus \gamma$	$F(c \oplus \gamma) \oplus \lambda$
1	-	-	-	$c \oplus \gamma$
2	c	-	-	-
3	-	c	-	-
4	-	-	c	-
4–21	Repeated Cancellation Property: Table 5			
21	$F(c \oplus \alpha) \oplus \beta$	?	?	?
22	?	$F(c \oplus \alpha) \oplus \beta$	?	?
23	?	?	$F(c \oplus \alpha) \oplus \beta$	?
24	?	?	?	$F(c \oplus \alpha) \oplus \beta$

α, β, γ and λ can be computed from a, b, d and the key by:
$\alpha = K_{13} \oplus F_{10}(F_7(a) \oplus b) \oplus F_6(b)$, $\beta = d$ and
$\gamma = F_1(b \oplus F_2(a \oplus F_3(d)))$, $\lambda = d$

we have:

$$\begin{aligned} V_0 \oplus V_{24} &= F(c \oplus \gamma) \oplus F(c \oplus \alpha) \\ &= F(u \oplus \alpha \oplus \gamma) \oplus F(u) = h_{\alpha\oplus\gamma}(u) = \overline{H} \end{aligned}$$

On average this costs one compression function evaluation to find a $n/4$-bit partial preimage. If the target value is 0, this only succeeds if $\alpha \oplus \gamma = 0$, but in this case it gives $2^{n/4}$ solutions. This gives a preimage attack with complexity $2^{3n/4}$ using $2^{n/4}$ memory.

Note that it is possible to make a time-memory trade-off with complexity 2^{n-k} using 2^k memory for $k < n/4$.

3.4 Dedicated 24-round Attacks on *Lesamnta*

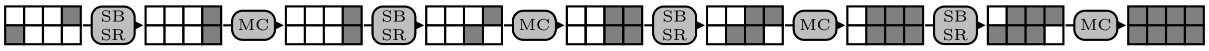
We now describe how to use specific properties of the round functions of *Lesamnta* to remove the memory requirement of our 24-round attacks.

Neutral Subspaces in F_{256} The F function of *Lesamnta*-256 has a property that limits the difference propagation to linear subspaces. Namely, we identified two linear subspaces $\mathit{\Gamma}$ and $\mathit{\Lambda}$ for which

$$x \oplus x' \in \mathit{\Gamma} \Rightarrow F(x) \oplus F(x') \in \mathit{\Lambda}$$

The subspaces $\mathit{\Gamma}$ and $\mathit{\Lambda}$ have dimensions of 16 and 48, respectively.

The AES-like round function of *Lesamnta*-256 achieves full diffusion of the values after its four rounds, but some linear combinations of the output are not affected. Starting from a single active diagonal, we have:



All the output bytes are active, but there are some linear relations between them. More precisely, the inverse MixColumns operation leads to a difference with two inactive bytes. Therefore, we can equivalently say that there are 16 linear relations of the output bits that are not affected by 16 input bits.

Collision and Second Preimage Attacks on Lesamnta-256 Using this property, we can choose 16 linear relations of the output of the family of function h_μ, or equivalently, choose 16 linear relations of the output of the compression function.

Let $\bar{\Lambda}$ be a supplementary subspace of Λ. Any 64-bit value x can be written as $x = x_\Lambda + x_{\bar{\Lambda}}$, where $x_\Lambda \in \Lambda$ and $x_{\bar{\Lambda}} \in \bar{\Lambda}$. We show how to find values x such that $h_\mu(x)_{\bar{\Lambda}} = \overline{H}_{\bar{\Lambda}}$ for an amortized cost of one, without memory:

1. Compute $h_\mu(u)$ for random u's until $h_\mu(u)_{\bar{\Lambda}} = \overline{H}_{\bar{\Lambda}}$
2. Far all v in Γ, we have $h_\mu(u+v)_{\bar{\Lambda}} = \overline{H}_{\bar{\Lambda}}$

This gives 2^{16} messages with 16 chosen relations for a cost of 2^{16}. It allows a second-preimage attack on 24-round *Lesamnta*-256 with complexity 2^{240}, and a collision attack with complexity 2^{120}, both memoryless.

Symmetries in F_{256} and F_{512} The AES round function has strong symmetry properties, as studied in [9]. The round function F of *Lesamnta* is heavily inspired by the AES round, and has similar symmetry properties. More specifically, if an AES state is such that the left half is equal to the right half, then this property still holds after any number of SubBytes, ShiftRows, and MixColumns operations. Explicitly, after one AES-like round of *Lesamnta*-256, we have:

$$\begin{array}{|c|c|c|c|}\hline w & x & w & x \\ \hline y & z & y & z \\ \hline\end{array} \xrightarrow{AES\text{−}like} \begin{array}{|c|c|c|c|}\hline w' & x' & w' & x' \\ \hline y' & z' & y' & z' \\ \hline\end{array} \text{ where } \begin{cases} w' = 2 \bullet S[w] \oplus S[z] & x' = 2 \bullet S[x] \oplus S[y] \\ y' = S[w] \oplus 2 \bullet S[z] & z' = S[x] \oplus 2 \bullet S[y] \end{cases}$$

And similarly in *Lesamnta*-512:

$$\begin{array}{|c|c|c|c|}\hline w & x & w & x \\ \hline y & z & y & z \\ \hline s & t & s & t \\ \hline u & v & u & v \\ \hline\end{array} \xrightarrow{AES} \begin{array}{|c|c|c|c|}\hline w' & x' & w' & x' \\ \hline y' & z' & y' & z' \\ \hline s' & t' & s' & t' \\ \hline u' & v' & u' & v' \\ \hline\end{array}$$

When we consider the F functions of *Lesamnta*, we have that: *if $x \oplus K_i$ is symmetric, then $F_i(x) = F(x \oplus K_i)$ is also symmetric.* More precisely, if we denote the set of symmetric values by $\mathcal{S}$, we use the following property:

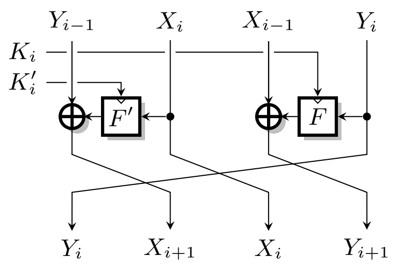
$$x, x' \in \mathcal{S} \Rightarrow F(x) \oplus F(x') \in \mathcal{S}$$

*Collision Attacks on Lesamnta-*256 *and Lesamnta-*512 This property can be used for an improved collision attack. As seen earlier we have $V_0 \oplus V_{24} = F(c \oplus \gamma) \oplus F(c \oplus \alpha)$. In order to use the symmetry property, we first select random chaining values, and we compute the value of α and γ until $\alpha \oplus \gamma$ is symmetric ($\alpha \oplus \gamma \in \mathcal{S}$). Then, if we select c such that $c \oplus \gamma \in \mathcal{S}$, we also have $c \oplus \alpha \in \mathcal{S}$, and this gives $V_0 \oplus V_{24} \in \mathcal{S}$.

We need to try 2^{32} (respectively, 2^{64} for *Lesamnta*-512) random chaining values in order to get $\alpha \oplus \gamma \in \mathcal{S}$, but once we have a good chaining value, we can use it with 2^{32} (respectively, 2^{64}) messages, and in each one $V_0 \oplus V_{24} \in \mathcal{S}$. So we can have 32 equalities between output bits (respectively, 64 equalities) for an average cost of one compression function call. This leads to a collision attack with complexity 2^{112} for *Lesamnta*-256, and 2^{224} for *Lesamnta*-512. In Appendix C, we give an example of inputs where 64 bits of the output are set to 1 using this property.

4 Application to *SHAvite-3*$_{512}$

SHAvite-3 is a hash function designed by Biham and Dunkelman for the SHA-3 competition [1]. It is based on a generalized Feistel construction with an AES-based round function, used in Davies-Meyer mode. In this section we study *SHAvite-3*$_{512}$, the version of *SHAvite-3* designed for output size of 257 to 512 bits. The cancellation property can not be used on *SHAvite-3*$_{256}$ because the Feistel structure is different and has a faster diffusion. We describe an attack on the *SHAvite-3*$_{512}$ hash function reduced to 9 rounds out of 14. An earlier variant of our attack was later extended in [5] to a 10-round attack. We note that our improved 9-round attack can be used to offer an improved 10-round attack.



$$X_{i+1} = Y_{i-1} \oplus F_i(X_i)$$
$$Y_{i+1} = X_{i-1} \oplus F'_i(Y_i)$$

$$X_{i-1} = Y_{i+1} \oplus F'_i(Y_i)$$
$$Y_{i-1} = X_{i+1} \oplus F_i(X_i)$$

Fig. 2. The Underlying Block Cipher of C_{512}

4.1 A Short Description of *SHAvite-3*$_{512}$

The compression function of *SHAvite-3*$_{512}$ accepts a chaining value of 512 bits, a message block of 1024 bits, a salt of 512 bits, and a bit counter of 128 bits. As this is a Davies-Meyer construction, the message block, the salt, and the bit counter enter the key schedule algorithm of the underlying block cipher. The key schedule algorithm transforms them into 112 subkeys of 128 bits each. The chaining value is then divided into four 128-bit words, and at each round two words enter the nonlinear round functions and affect the other two:

$$S_{i+1} = V_i \qquad T_{i+1} = S_i \oplus F'_i(T_i) \qquad U_{i+1} = T_i \qquad V_{i+1} = U_i \oplus F_i(V_i)$$

The nonlinear function F and F' are composed of four full rounds of AES, with 4 subkeys from the message expansion:

$$F_i(x) = P(k^3_{0,i} \oplus P(k^2_{0,i} \oplus P(k^1_{0,i} \oplus P(k^0_{0,i} \oplus x))))$$
$$F'_i(x) = P(k^3_{1,i} \oplus P(k^2_{1,i} \oplus P(k^1_{1,i} \oplus P(k^0_{1,i} \oplus x))))$$

where P is one AES round (without the AddRoundKey operation).

In this section we use an alternative description of *SHAvite-3*$_{512}$ with only two variables per round, as shown in Figure 2. We have

$$S_i = Y_{i-1} \qquad T_i = X_i \qquad U_i = X_{i-1} \qquad V_i = Y_i$$

The message expansion is detailed in Appendix D.

Note that with the original key schedule of *SHAvite-3*, a specific set of message, salt, and counter leads to all the subkeys being zero [10,11]. Thus, the key schedule was tweaked for the second round of the SHA-3 competition. Our attack applies both to the original message expansion and to the tweaked version.

4.2 Cancellation Attacks on *SHAvite-3*$_{512}$

The cancellation path is described in Table 9, which is an extension of Table 2. We use the cancellation property twice to control the diffusion. Note that we do not have to specify the values of y, z, and w. Like in the *Lesamnta* attack, this path is a truncated differential, and we use constraints on the state to enforce that it is followed. Moreover, the transitions $x \to x_1$ and $x_1 \to x_2$ are known because the key is known.

Note that the round functions of *SHAvite-3*$_{512}$ are not defined as $F(k, x) = P(k \oplus x)$ for a fixed permutation P. Instead, each function takes 4 keys and it is defined as:

$$F(k^0_i, k^1_i, k^2_i, k^3_i, x) = P(k^3_i \oplus P(k^2_i \oplus P(k^1_i \oplus P(k^0_i \oplus x))))$$

where P is one AES round. In order to apply the cancellation property to *SHAvite-3*$_{512}$, we need that the subkeys k^1, k^2, k^3 of two functions be equal, so that $F_i(x)$ collapses to $P'(k^0_i \oplus x)$ and F_j to $P'(k^0_j \oplus x)$, where $P'(x) \triangleq P(k^3_i \oplus P(k^2_i \oplus P(k^1_i \oplus P(x)))) = P(k^3_j \oplus P(k^2_j \oplus P(k^1_j \oplus P(x))))$.

Table 9. Cancellation Property on 9 Rounds of *SHAvite-3*$_{512}$

i	S_i	T_i	U_i	V_i	
0	?	x_2	?	x	
1	x	-	x_2	x_1	
2	x_1	x	-	-	$x_1 \to x_2$
3	-	-	x	-	$x \to x_1$
4	-	-	-	x	
5	x	-	-	y	$\underline{x \to y}$
6	y	x	-	z	$y \to z$
7	z	$\underline{-}$	x	w	$\underline{x \to y}, \underline{z \to w}$
8	w	z	$\underline{-}$	?	
9	?	$\underline{-}$	z	?	$\underline{z \to w}$
FF	?	x_2	?	?	

Table 10. Values of the Registers for the 9-round Cancellation Property of *SHAvite-3*$_{512}$

i	X_i/Y_i
X_0	$b \oplus F_3(c) \oplus F'_1(c \oplus F_2(d \oplus F'_3(a)))$
Y_0	$d \oplus F'_3(a) \oplus F_1(a \oplus F'_2(b \oplus F_3(c)))$
X_1	$a \oplus F'_2(b \oplus F_3(c))$
Y_1	$c \oplus F_2(d \oplus F'_3(a))$
X_2	$d \oplus F'_3(a)$
Y_2	$b \oplus F_3(c)$
X_3	c
Y_3	a
X_4	b
Y_4	d
X_5	$a \oplus F_4(b)$
Y_5	$c \oplus F'_4(d)$
X_6	$d \oplus F_5(a \oplus F_4(b))$
Y_6	$b \oplus F'_5(c \oplus F'_4(d))$
X_7	$c \oplus \cancel{F'_4(\underline{d})} \oplus \cancel{F_6(\underline{d} \oplus F_5(a \oplus F_4(b)))}$
Y_7	$a \oplus F_4(b) \oplus F'_6(b \oplus F'_5(c \oplus F'_4(d)))$
X_8	$b \oplus F'_5(c \oplus F'_4(d)) \oplus F_7(c)$
Y_8	$d \oplus F_5(a \oplus F_4(b)) \oplus F'_7(a \oplus F_4(b) \oplus F'_6(b \oplus F'_5(c \oplus F'_4(d))))$
X_9	$a \oplus F_4(b) \oplus \cancel{F'_6(\underline{b \oplus F'_5(c \oplus F'_4(d))})} \oplus \cancel{F_8(\underline{b \oplus F'_5(c \oplus F'_4(d))} \oplus F_7(c))}$

In order to express the constraints needed for the cancellation properties, we look at the *values* of the registers for this truncated differential. In Table 10, we begin at round 4 with $(S_4, T_4, U_4, V_4) = (Y_3, X_4, X_3, Y_4) = (a, b, c, d)$, and we compute up to round 9.

We have a cancellation property on 9 rounds under the following conditions:

Round 7 We have $F'_4(\underline{d}) \oplus F_6(\underline{d} \oplus F_5(a \oplus F_4(b)))$. They cancel if:
$F_5(a \oplus F_4(b)) = k^0_{1,4} \oplus k^0_{0,6}$ and $(k^1_{1,4}, k^2_{1,4}, k^3_{1,4}) = (k^1_{0,6}, k^2_{0,6}, k^3_{0,6})$.
Round 9 We have $F'_6(\underline{b \oplus F'_5(c \oplus F'_4(d))}) \oplus F_8(\underline{b \oplus F'_5(c \oplus F'_4(d))} \oplus F_7(c))$. They cancel if:
$F_7(c) = k^0_{1,6} \oplus k^0_{0,8}$ and $(k^1_{1,6}, k^2_{1,6}, k^3_{1,6}) = (k^1_{0,8}, k^2_{0,8}, k^3_{0,8})$.

Therefore, the truncated differential is followed if:

$$F_5(a \oplus F_4(b)) = k^0_{1,4} \oplus k^0_{0,6} \qquad F_7(c) = k^0_{1,6} \oplus k^0_{0,8} \tag{C0}$$

$$(k^1_{1,4}, k^2_{1,4}, k^3_{1,4}) = (k^1_{0,6}, k^2_{0,6}, k^3_{0,6}) \qquad (k^1_{1,6}, k^2_{1,6}, k^3_{1,6}) = (k^1_{0,8}, k^2_{0,8}, k^3_{0,8}) \tag{C1}$$

The constraints for the cancellation at round 7 are easy to satisfy and allow a 7-round attack on *SHAvite-3*$_{512}$. However, for a 9-round attack we have more constraints on the subkeys, and this requires special attention.

4.3 Dealing with the Key Expansion

Let us outline an algorithm to find a suitable message (recall that *SHAvite-3*$_{512}$ is used in a Davies-Meyer mode) for a given salt and counter value. We have to solve a system involving linear and non-linear equations, and we use the fact that the system is almost triangular. We note that it might be possible to improve our results using the technique of Khovratovich, Biryukov and Nikolić [7] to find a good message efficiently.

For the cancellation attack on 9-round *SHAvite-3*$_{512}$, we need to satisfy a 768-bit condition on the subkeys, *i.e.*:

$$(k^1_{1,4}, k^2_{1,4}, k^3_{1,4}) = (k^1_{0,6}, k^2_{0,6}, k^3_{0,6}) \qquad (k^1_{1,6}, k^2_{1,6}, k^3_{1,6}) = (k^1_{0,8}, k^2_{0,8}, k^3_{0,8}) \tag{C1}$$

Or in $rk[\cdot]$ terms:

$$rk[148, \dots, 159] = rk[196, \dots, 207] \qquad rk[212, \dots, 223] = rk[260, \dots, 271]$$

We are actually trying to solve a system of equation with:

- 224 variables: $tk[128..159]$, $tk[192..223]$ and $rk[128..287]$
- 192 equations from the key schedule (64 non-linear and 128 linear)
- 24 constraints

Therefore we have 8 degrees of freedom. The relations between the variables are shown in Figure 3, while the full key expansion of *SHAvite-3*$_{512}$ is described in Appendix D.

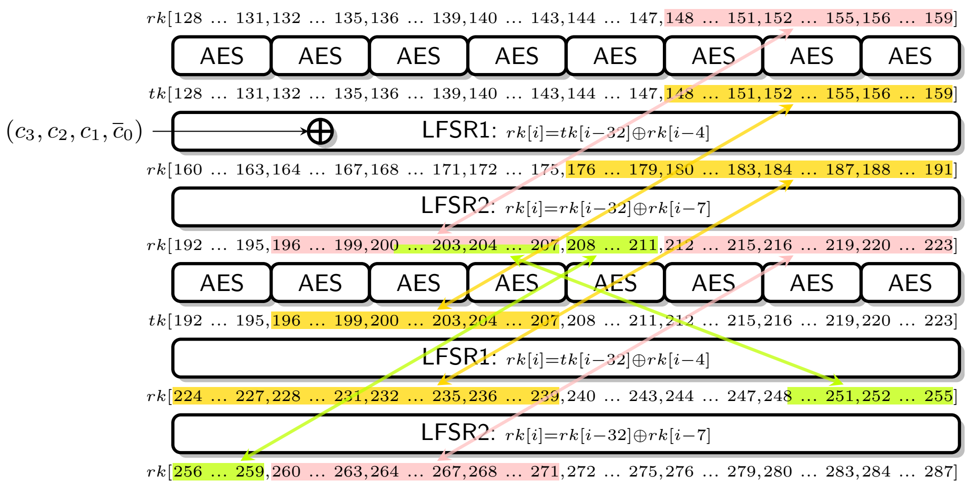

rk[128 … 131,132 … 135,136 … 139,140 … 143,144 … 147,148 … 151,152 … 155,156 … 159]
AES AES AES AES AES AES AES AES
tk[128 … 131,132 … 135,136 … 139,140 … 143,144 … 147,148 … 151,152 … 155,156 … 159]
(c3, c2, c1, c̄0)
LFSR1: rk[i]=tk[i−32]⊕rk[i−4]
rk[160 … 163,164 … 167,168 … 171,172 … 175,176 … 179,180 … 183,184 … 187,188 … 191]
LFSR2: rk[i]=rk[i−32]⊕rk[i−7]
rk[192 … 195,196 … 199,200 … 203,204 … 207,208 … 211,212 … 215,216 … 219,220 … 223]
AES AES AES AES AES AES AES AES
tk[192 … 195,196 … 199,200 … 203,204 … 207,208 … 211,212 … 215,216 … 219,220 … 223]
LFSR1: rk[i]=tk[i−32]⊕rk[i−4]
rk[224 … 227,228 … 231,232 … 235,236 … 239,240 … 243,244 … 247,248 … 251,252 … 255]
LFSR2: rk[i]=rk[i−32]⊕rk[i−7]
rk[256 … 259,260 … 263,264 … 267,268 … 271,272 … 275,276 … 279,280 … 283,284 … 287]

Fig. 3. Constraints in the Key Expansion of *SHAvite-3*$_{512}$
Initial constraints in pink, constraints from steps 1 to 3 in yellow, constraints from step 4 in green

Propagation of the Constraints First, we propagate the constraints and deduce new equalities between the variables. Figure 3 shows the initial constraints and the propagated constraints.

1. The non-linear equations of the key-schedule give:

$$tk[156..159] = AESR\Big((rk[157, 158, 159, 156]) \oplus (salt[12..15])\Big)$$
$$tk[204..207] = AESR\Big((rk[205, 206, 207, 204]) \oplus (salt[12..15])\Big)$$

   since $rk[156..159] = rk[204..207]$, we know that $tk[156..159] = tk[204..207]$. Similarly, we get $tk[148..159] = tk[196..207]$

2. From the key expansion, we have $rk[191] = rk[223] \oplus rk[216]$, and $rk[239] = rk[271] \oplus rk[264]$. Since we have the constraints $rk[223] = rk[271]$ and $rk[216] = rk[264]$, we can deduce that $rk[191] = rk[239]$ Similarly, we get $rk[187..191] = rk[235..239]$.
3. From the linear part of the expansion, we have $rk[186] = rk[190] \oplus tk[158]$ and $rk[234] = rk[238] \oplus tk[206]$. We have seen that $rk[190] = rk[238]$ at step 2 and $tk[158] = tk[206]$ at step 1, therefore $rk[186] = rk[234]$ Similarly, we get $rk[176..186] = rk[224..234]$.
4. Again, from the linear part of the key expansion, we have $rk[211] = rk[218] \oplus rk[186]$ and $rk[259] = rk[266] \oplus rk[234]$. We have seen that $rk[186] = rk[234]$ at step 3 and we have $rk[218] = rk[266]$ as a constraint, thus $rk[211] = rk[259]$ Similarly, we obtain $rk[201..211] = rk[249..259]$ Note that we have $rk[201..207] = rk[153..159]$ as a constraint, so we must have $rk[249..255] = rk[153..159]$.

Finding a Solution To find a solution to the system, we use a guess and determine technique. We guess 11 state variables, and we show how to compute the rest of the state and check for consistency. Since we have only 8 degrees of freedom, we expect the random initial choice to be valid once out of $2^{32\times 3} = 2^{96}$ times. This gives a complexity of 2^{96} to find a good message.

- Choose random values for $rk[200], rk[204..207], rk[215..216], rk[220..223]$
- Compute $tk[220..223]$ from $rk[220..223]$
- Compute $rk[248..251]$ from $tk[220..223]$ and $rk[252..255]$ $(= rk[204..207])$
- Deduce $rk[201..203] = rk[249..251]$, so $rk[200..207]$ is known
- Compute $tk[152..159]$ from $rk[152..159]$ $(= rk[200..207])$
- Compute $rk[190..191]$ from $rk[215..216]$ and $rk[222..223]$
- Compute $rk[186..187]$ from $rk[190..191]$ and $rk[158..159]$
- Compute $rk[182..183]$ from $rk[186..187]$ and $rk[154..155]$
- Compute $rk[214]$ from $rk[207]$ and $rk[182]$
- Compute $rk[189]$ from $rk[214]$ and $rk[219]$; then $rk[185]$ and $rk[181]$
- Compute $rk[213]$ from $rk[206]$ and $rk[181]$
- Compute $rk[188]$ from $rk[213]$ and $rk[220]$, then $rk[184]$ and $rk[180]$
- Compute $rk[212]$ from $rk[205]$ and $rk[180]$
- Compute $rk[219]$ from $rk[212]$ and $rk[187]$
- Compute $rk[208, 209]$ from $rk[215, 216]$ and $rk[183, 184]$
- We have $tk[216..219] = AESR\Big((rk[216..219])\Big)$ with a known key. Since $rk[216]$ and $rk[219]$ are known, we know that $tk[216..219]$ is a linear subspace of dimension 64 over $\mathbb{F}_2$.
- Similarly, $tk[208..211]$ is in a linear subspace of dimension 64 ($rk[208]$ and $rk[209]$ are known).
- Moreover, there are linear relations between $tk[216..219]$ and $tk[208..211]$: we can compute $rk[240..243]$ from $tk[208..211]$ and $rk[236..239]$; $rk[244..247]$ from $rk[240..243]$ and $tk[212..215]$; $tk[216..219]$ from $rk[244..247]$ and $rk[248..251]$.
- On average, we expect one solution for $tk[216..219]$ and $tk[208..211]$.
- At this point we have computed the values of $rk[200..223]$. We can compute $tk[200.223]$ and $rk[228..255]$.
- Compute $rk[176..179]$ from $rk[201..204]$ and $rk[208..211]$
- Since $rk[224..227] = rk[176..179]$, we have a full state $rk[224..255]$. We can check consistency of the initial guess.

4.4 9-round Attacks

The cancellation property allows to find a key/message pair with a given value on the last 128 bits. The attack is the following: first find a message that fulfills the conditions on the subkeys, and set a, b and c at round 4 satisfying the cancellation conditions (C0). Then the second output word is:

$$T_9 \oplus T_0 = X_9 \oplus X_0 = a \oplus F_4(b) \oplus b \oplus F_3(c) \oplus F_1'\big(c \oplus F_2(d \oplus F_3'(a))\big)$$

If we set

$$d = F_2^{-1}\Big(F_1'^{-1}\big(\overline{H} \oplus a \oplus F_4(b) \oplus b \oplus F_3(c)\big) \oplus c\Big) \oplus F_3'(a)$$

we have $X_9 \oplus X_0 = \overline{H}$. Each key (message) can be used with 2^{128} different a,b,c, and the cost of finding a suitable key is 2^{96}. Hence, the amortized cost for finding a 128-bit partial preimage is one compression function evaluation. The cost of finding a full preimage for the compression function is 2^{384}, as described in Algorithm 1.

Algorithm 1 *SHAvite-3*$_{512}$ cancellation attack on 9 rounds

```
Input: Target value $\overline{H}$
Output: A message $M$ and a chaining value $X$ s.t. $F(X,M) = \overline{H}$
Running Time: $2^{384}$
 1: loop
 2:     Find $M$ such that $(k^1_{1,4}, k^2_{1,4}, k^3_{1,4}) = (k^1_{0,6}, k^2_{0,6}, k^3_{0,6})$ and $(k^1_{1,6}, k^2_{1,6}, k^3_{1,6}) = (k^1_{0,8}, k^2_{0,8}, k^3_{0,8})$
 3:     $c \leftarrow F_7^{-1}(k^0_{1,6} \oplus k^0_{0,8})$
 4:     for all $a$ do
 5:         $b \leftarrow F_4^{-1}(F_5^{-1}(k^0_{1,4} \oplus k^0_{0,6}) \oplus a)$
 6:         Compute $d$ as $F_2^{-1}\Big(F_1'^{-1}\big(\overline{H}_4 \oplus a \oplus F_4(b) \oplus b \oplus F_3(c)\big) \oplus c\Big) \oplus F_3'(a)$
 7:         Compute 4 rounds backwards and 5 rounds forwards from $a$, $b$, $c$, $d$
 8:         Then $H_4 = X_0 \oplus X_9 = \overline{H}_4$
 9:         if $H = \overline{H}$ then
10:             return $X, M$
11:         end if
12:     end for
13: end loop
```

Second Preimage Attack on the Hash Function We can use the preimage attack on the compression function to build a second preimage attack on the hash function reduced to 9 rounds. Using a generic unbalanced meet-in-the-middle attack the complexity is about 2^{448} compression function evaluations and 2^{64} memory. Note that we cannot find preimages for the hash function because we cannot find correctly padded message blocks.

5 Conclusion

In this paper we explore new ways to use efficiently degree of freedom in generalized Feistel structures. In addition to the attacks on *Lesamnta* and *SHAvite-3*$_{512}$, we describe an attack against *SMS4* in Appendix B. We summarize the obtained attacks in Tables 11, 12, and 13.

Table 11. Summary of the Attacks on the *Lesamnta* Hash Function

				Lesamnta-256		*Lesamnta*-512	
	Attack		Rounds	Time	Memory	Time	Memory
Generic	Collision	[6]	16	2^{97}	-	2^{193}	-
	Second Preimage	[6]	16	2^{193}	-	2^{385}	-
	Collision	(Sect. 3.3)	22	2^{96}	-	2^{192}	-
	Second Preimage	(Sect. 3.3)	22	2^{192}	-	2^{384}	-
	Collision	(Sect. 3.3)	24	2^{96}	2^{64}	2^{192}	2^{128}
	Second Preimage	(Sect. 3.3)	24	2^{192}	2^{64}	2^{384}	2^{128}
Specific	Collision	(Sect. 3.4)	24	2^{112}	-	2^{224}	-
	Second Preimage	(Sect. 3.4)	24	2^{240}	-	*N/A*	

Table 12. Summary of the Attacks on *SHAvite-3*$_{512}$

Attack		Rounds	Comp. Fun.		Hash Fun.	
			Time	Memory	Time	Memory
Second Preimage	[4]	8	2^{384}	-	2^{448}	2^{64}
Second Preimage	(Sect. 4.4)	9	2^{384}	-	2^{448}	2^{64}
Second Preimage	(extension of this work) [5]	10	2^{480}	-	2^{496}	2^{16}
Second Preimage	(improving [5] with Sect. 4.3)	10	2^{448}	-	2^{480}	2^{32}
Second Preimage	(improving [5] with Sect. 4.3)	10	2^{416}	2^{64}	2^{464}	2^{64}
Second Preimage	(improving [5] with Sect. 4.3)	10	2^{384}	2^{128}	2^{448}	2^{128}
Collision[1]	(extension of this work) [5]	14	2^{192}	2^{128}	*N/A*	
Preimage[1]	(extension of this work) [5]	14	2^{384}	2^{128}	*N/A*	
Preimage[1]	(extension of this work) [5]	14	2^{448}	-	*N/A*	

[1] Chosen salt attacks

Table 13. Summary of the Attacks on *SMS4*

Attack		Rounds	Data	Time
Boomerang	[13]	18	2^{120} ACPC	2^{120}
Rectangle	[13]	18	2^{124} CP	2^{124}
Differential	[14]	21	2^{118} CP	$2^{126.6}$
Differential	[13]	22	2^{118} CP	$2^{125.7}$
Linear	[13]	22	2^{117} KP	2^{117}
Truncated Differential	(Sect. B.1)	19	2^{104} CP	2^{104}

Acknowledgements

We would like to thank the members of the Graz ECRYPT meeting. Especially, we would like to express our gratitude to Emilia Käsper, Christian Rechberger, Søren S. Thomsen, and Ralf-Philipp Weinmann for the inspiring discussions. We are grateful to the *Lesamnta* team, and especially to Hirotaka Yoshida, for helping us with this research. We would also like to thank the anonymous referees for their comments.

References

1. Biham, E., Dunkelman, O.: The SHAvite-3 Hash Function. Submission to NIST (2008)
2. Bouillaguet, C., Dunkelman, O., Fouque, P.A., Leurent, G.: Another Look at the Complementation Property. In Hong, S., Iwata, T., eds.: FSE '10. Lecture Notes in Computer Science, Springer (2010)
3. Diffie, W., (translators), G.L.: SMS4 Encryption Algorithm for Wireless Networks. Cryptology ePrint Archive, Report 2008/329 (2008) `http://eprint.iacr.org/`.
4. *et al.*, F.M.: A preimage attack on 8-round SHAvite-3-512. Graz ECRYPT meeting 2009 (May 2009)
5. Gauravaram, P., Leurent, G., Mendel, F., Naya-Plasencia, M., Peyrin, T., Rechberger, C., Schläffer, M.: Cryptanalysis of the 10-Round Hash and Full Compression Function of SHAvite-3-512. In Bernstein, D.J., Lange, T., eds.: AFRICACRYPT. Volume 6055 of Lecture Notes in Computer Science., Springer (2010) 419–436
6. Hirose, S., Kuwakado, H., Yoshida, H.: SHA-3 Proposal: Lesamnta. Submission to NIST (2008)
7. Khovratovich, D., Biryukov, A., Nikolic, I.: Speeding up Collision Search for Byte-Oriented Hash Functions. In Fischlin, M., ed.: CT-RSA. Volume 5473 of Lecture Notes in Computer Science., Springer (2009) 164–181
8. Lai, X., Massey, J.L.: Hash Function Based on Block Ciphers. In Rueppel, R.A., ed.: EUROCRYPT. Volume 658 of Lecture Notes in Computer Science., Springer (1992) 55–70
9. Le, T.V., Sparr, R., Wernsdorf, R., Desmedt, Y.: Complementation-Like and Cyclic Properties of AES Round Functions. In Dobbertin, H., Rijmen, V., Sowa, A., eds.: AES Conference. Volume 3373 of Lecture Notes in Computer Science., Springer (2004) 128–141
10. Nandi, M., Paul, S.: OFFICIAL COMMENT: SHAvite-3. Available online (2009)
11. Peyrin, T.: Chosen-salt, chosen-counter, pseudo-collision on SHAvite-3 compression function. Available online (2009)

12. Shoichi Hirose, Hidenori Kuwakado, H.Y.: Security Analysis of the Compression Function of Lesamnta and its Impact. Available online (2009)
13. Taehyun Kim, Jongsung Kim, S.H., Sung, J.: Linear and Differential Cryptanalysis of Reduced SMS4 Block Cipher. Cryptology ePrint Archive, Report 2008/281 (2008) `http://eprint.iacr.org/`.
14. Zhang, L., Zhang, W., Wu, W.: Cryptanalysis of Reduced-Round SMS4 Block Cipher. In Mu, Y., Susilo, W., Seberry, J., eds.: ACISP. Volume 5107 of Lecture Notes in Computer Science., Springer (2008) 216–229

A An Integral Attack against 21-Round *Lesamnta*

In this section, we show the applicability of the cancellation technique to block ciphers analysis by giving an application to the implicit block cipher of *Lesamnta*. The main difference with the hash function attacks is that in a block cipher the attacker does not know the key, and cannot build a message satisfying the cancellation conditions. Therefore, we use an attack based on integral cryptanalysis. The basic idea is to use several sets of messages, so that the messages in one of the sets follow the cancellation path.

In the original submission document of *Lesamnta* [6] a 19-round SQUARE distinguisher is described. This SQUARE is very straightforward, and suggests an efficient distinguisher for *Lesamnta*. However, by experimenting with reduced versions, we found that the original SQUARE is faulty. We first give an explanation of why the SQUARE attack does not work. Then we suggest an improved and corrected 20-round integral distinguisher attack which relies on the cancellation property. This distinguisher gives a 21-round key-recovery attack using partial decryption of the last round. We use the term integral cryptanalysis rather than SQUARE to describe our new attack, because we use a higher order property.

In Table 14, the symbols b_1, b_2, b_3 are used to denote three variables that independently take all possible values. So, in the first round, T_0, U_0, V_0 take all the $2^{3n/4}$ possible values. At round 1, we have $S_1 = F_1(U_0) \oplus V_0$. We see that $F_1(U_0) \oplus V_0, T_0, U_0$ take all possible values, so we can reuse the symbol b_3 for S_1. This can be seen as a change of variables.

Starting from round 4, we have two values denoted by b_3 in the original SQUARE. This is used to denote that R_4, S_4, T_4 take all possible values, while S_4, T_4, U_4 also take all possible values. However, this leads to a contradiction later on because there is an implicit change of variables when we reuse symbols and this cannot be done for a variable that appears twice. The first problem appears at round 7. We have that S_6, T_6, V_6 and S_6, U_6, V_6 take all possible values. The original SQUARE suggests that this implies that S_7, S_6, T_6 take all possible values, where $S_7 = F_7(U_6) \oplus V_6$. However this is not true in general. For instance, we could have $U_6 = F_7^{-1}(T_6 \oplus V_6)$. This is compatible with the assumptions of independence but in this case we have $S_7 = T_6$ and S_7, S_6, T_6 do not take all possible values.

Actually the SQUARE described in this attack can be detected after 18 rounds, but not after 19 rounds.

Table 14. The originally suggested SQUARE, and its actual development. We see that the independence assumptions of round 7 do not hold.

i	S_i	T_i	U_i	V_i
0	-	b_1	b_2	b_3
1	b_3	-	b_1	b_2
2	b_2	b_3	-	b_1
3	b_1	b_2	b_3	-
4	b_3	b_1	b_2	b_3
5	b_3	b_3	b_1	b_2
6	b_2	b_3	b_3	b_1
7	b_1	b_2	b_3	b_3

Suggested in [6]

i	S_i	T_i	U_i	V_i
0	-	b_1	b_2	b_3
1	b_3	-	b_1	b_2
2	b_2	b_3	-	b_1
3	b_1	b_2	b_3	-
4	$F(b_3)$	b_1	b_2	b_3
5	$F(b_2) \oplus b_3$	$F(b_3)$	b_1	b_2
6	$F(b_1) \oplus b_2$	$F(b_2) \oplus b_3$	$F(b_3)$	b_1
7	$F(F(b_3)) \oplus b_1$	$F(b_1) \oplus b_2$	$F(b_2) \oplus b_3$	$F(b_3)$

Actual SQUARE

A.1 The New Attack

Our new attack is based on the cancellation path of Table 15. Starting with $(S_0, T_0, U_0, V_0) = (a, b, c, d)$, we have the following condition:

Round 8: we have $F_7(F_4(F_1(b) \oplus c) \oplus \underline{F_0(c) \oplus d}) \oplus F_3(\underline{F_0(c) \oplus d})$. They cancel if:
$F_4(F_1(b) \oplus c) = K_3 \oplus K_7$

Table 15. Cancellation path for the integral attack on *Lesamnta*

i	$X_i (= S_i)$
-3	d
-2	c
-1	b
0	a
1	$F_0(c) \oplus d$
2	$F_1(b) \oplus c$
3	$F_2(a) \oplus b$
4	$F_3(F_0(c) \oplus d) \oplus a$
5	$F_4(F_1(b) \oplus c) \oplus F_0(c) \oplus d$
6	$F_5(F_2(a) \oplus b) \oplus F_1(b) \oplus c$
7	$F_6(F_3(F_0(c) \oplus d) \oplus a) \oplus F_2(a) \oplus b$
8	$\cancel{F_7(F_4(F_1(b) \oplus c) \oplus \underline{F_0(c) \oplus d})} \oplus \cancel{F_3(\underline{F_0(c) \oplus d})} \oplus a$

Since we do not know the value of $K_3 \oplus K_7$, we cannot build a message that would satisfy the cancellation condition with probability one. However, in an integral attack, if we iterate over all values of c while b if fixed, we know that for one value there is a cancellation. Moreover, for each c, we can iterate over d and study properties of the set of ciphertexts generated in this way.

More precisely, for a random choice of A and B, we define the following sets of messages:

$$\mathcal{S}_C = \{(A, B, C, d) : d \in \mathbb{F}_{2^{64}}\} \qquad (d \in \mathbb{F}_{2^{128}} \text{ for } \textit{Lesamnta}\text{-512})$$

We consider 2^{64} sets (2^{128} for *Lesamnta*-512) with all possible values of C, while A and B are fixed. We know that one particular set satisfies $F_4(F_1(B) \oplus C) = K_3 \oplus K_7$. For this set, we show the dependencies of the state on d in Table 16. Note that:

- At round 8, the values $F_7(d) \oplus F_3(d)$ cancels out for this particular set.
- At round 14 we have $F_{13}(\underline{F_6(F_3(d))}) \oplus F_9(\underline{F_6(F_3(d))})$. This has the special property that each value is taken an even number of times, according to Property (*ii*).
- At round 17, we have $F_{16}\Big(F_{13}\big(\underline{F_6(F_3(d))}\big) \oplus F_9\big(\underline{F_6(F_3(d))}\big)\Big) \oplus F_{12}\big(F_9(F_6(F_3(d)))\big) \oplus d$. When we sum this over all d's, this gives:

$$\bigoplus_d F_{16}\Big(F_{13}\big(\underline{F_6(F_3(d))}\big) \oplus F_9\big(\underline{F_6(F_3(d))}\big)\Big) \oplus \bigoplus_d F_{12}(F_9(F_6(F_3(d)))) \oplus \bigoplus_d d$$

The first term sums to zero because each input to F_{16} is taken an even number of times (cf. Property (*iii*)), and the two last terms sum to zero because they are permutations of d.

Since X_{17} is the fourth output word after 20 rounds ($X_{17} = V_{20}$), this gives an integral property on 20 rounds. This has been experimentally verified on reduced versions.

Table 16. Integral Attack. We only gives the dependencies in d.

i	X_i
-3	d
-2	-
-1	-
0	-
1	d
2	-
3	-
4	$F_3(d)$
5	d
6	-
7	$F_6(F_3(d))$
8	$\cancel{F_7(d)} \oplus \cancel{F_3(d)}$
9	d
10	$F_9(F_6(F_3(d)))$
11	$F_6(F_3(d))$
12	$F_{11}(d)$
13	$F_{12}(F_9(F_6(F_3(d)))) \oplus d$
14	$F_{13}\underline{(F_6(F_3(d)))} \oplus F_9\underline{(F_6(F_3(d)))}$
15	$F_{14}(F_{11}(d)) \oplus F_6(F_3(d))$
16	$F_{15}(F_{12}(F_9(F_6(F_3(d)))) \oplus d) \oplus F_{11}(d)$
17	$F_{16}(F_{13}\underline{(F_6(F_3(d)))} \oplus F_9\underline{(F_6(F_3(d)))}) \oplus F_{12}(F_9(F_6(F_3(d)))) \oplus d$

This property can be used to attack the block cipher of *Lesamnta*. One has to encipher the $2^{n/4}$ plaintexts in the sets $\mathcal{S}_C$ for each C, and to compute the sum of V_{20} over each set. If the data was generated using the compression function of *Lesamnta*, then at least one of the sets $\mathcal{S}_C$ give a zero sum. Otherwise, there is only a probability $1/e$ that all the sums are non-zero. Moreover, when a set with a zero sum is found, it is possible to verify that this is due to the cancellation property, by building a new set $\mathcal{S}_C^*$ with the same C and B, but a different value A. As seen earlier, the cancellation condition does not depend on A, so this new set also gives a zero sum if C is the correct value for the cancellation condition.

This gives a distinguisher on 20-round *Lesamnta* with complexity $2^{n/2}$. Moreover, using partial decryption of the last round, it can be extended to a key recovery attack on 21 rounds with complexity $2^{3n/4}$.

B An Attack on 19-Round *SMS4*

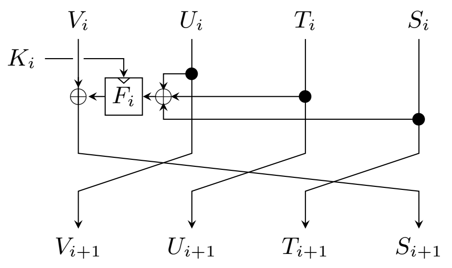
SMS4 is a block cipher used in WAPI (the Chinese national standard for wireless networks) [3], based on a generalized Feistel network. *SMS4* accepts a 128-bit plaintext and a 128-bit user key as inputs, and is composed of 32 rounds. In each round, the least significant three words of the state are XORed with the round key and the result passes the F transformation. The F transformation uses an 8-bit to 8-bit bijective SBox four times in parallel to process each byte, then the concatenated bytes are processed using a linear transformation L. The general structure is described by Figure 4, and can be written as:

$$S_{i+1} = V_i \oplus F(S_i \oplus T_i \oplus U_i \oplus K_i) \qquad T_{i+1} = S_i \qquad U_{i+1} = T_i \qquad V_{i+1} = U_i$$

where the K_i is the ith round subkey.

B.1 New Attack on *SMS4*

Our attack on *SMS4* is not based on the same cancellation property as used to attack *Lesamnta* and *SHAvite-3*$_{512}$. However, it shares the same core idea: we describe a generic attack on the



SMS4 structure

Fig. 4. The Generalized Feistel structure of *SMS4*

Feistel structure based on a truncated differential, using available degrees of freedom to control the non-linearity. While this attack is not as efficient as the best attacks on *SMS4*, we believe it is interesting because it is generic in the round function.

Our attack is based on the following truncated differential on four rounds:

i	S_i	T_i	U_i	V_i		
0	x	x	a	b		$x = a \oplus b$
1	x	a	b	x		
2	a	b	x	x	$\underline{b \to u}$	let $c = a \oplus u$
3	b	x	x	c	$\underline{c \to u}$	let $d = b \oplus u$
4	x	x	c	d		$x = c \oplus d$

In this truncated differential, we do not specify the values of a, b, c, d and u. We start with any difference (x, x, a, b) with $x = a \oplus b$, and we end up with a difference of the same form: (x, x, c, d) with $x = c \oplus d$. The only condition for this truncated differential to be followed is that the transitions at step 2 and 3, $b \to u$ and $c \to u$ have to go to the same u. Since we do not care about the particular value of u, this happens with probability 2^{-32} (the words are 32-bit wide).

We can iterate this truncated differential four time, and add two extra rounds at the end. This gives the following characteristic:

$$(x, x, a, b)_{|a \oplus b = x} \xrightarrow{18 rounds} (e, f, x, x)_{|e \oplus f = x} \quad \text{with probability } 2^{-128}$$

For a random mapping, this characteristic has probability 2^{-96}. Therefore, to detect the bias, we need a few times $2^{64+96} = 2^{160}$ input pairs.

We can build the input pairs using structures: for a random A and B, we generate 2^{64} messages $M_{i,j} = (A \oplus i \oplus j, B \oplus i \oplus j, i, j)$, by varying i and j. This gives 2^{128} input pairs for the differential: each pair $M_{i,j}, M_{i',j'}$ has a difference $(i \oplus j \oplus i' \oplus j', i \oplus j \oplus i' \oplus j', i \oplus i', j \oplus j')$. If we repeat this with a few times 2^{32} choices of A and B, we have enough pairs to detect the bias. This gives a distinguisher on 18 rounds of *SMS4* with a few times 2^{96} chosen plaintexts. Experiments on reduced versions confirm this analysis.

This distinguisher can be used for a 19-round attack, using partial decryption of the last round. The corresponding differential is:

i	S_i	T_i	U_i	V_i	
0	x	x	a	b	$x = a \oplus b$
$\vdots$					
4	x	x	c	d	$x = c \oplus d$
$\vdots$					
16	x	x	e	f	$x = e \oplus f$
17	x	e	f	x	
18	e	f	x	x	
19	f	x	x	?	

We can recover the subkey of rounds 19 with the following algorithm:

1. Repeat 2^{40} times the following:
2. Choose a random A and B
3. Query the block cipher on the 2^{64} messages $M_{i,j} = (A \oplus i \oplus j, B \oplus i \oplus j, i, j)$
4. Let the plaintext/ciphertext be $(q_{i,j}, r_{i,j}, s_{i,j}, t_{i,j}) \rightarrow (\alpha_{i,j}, \beta_{i,j}, \gamma_{i,j}, \delta_{i,j})$, respectively. Store the plaintext/ciphertext pair of $((q_{i,j}, r_{i,j}, s_{i,j}, t_{i,j}), (\alpha_{i,j}, \beta_{i,j}, \gamma_{i,j}, \delta_{i,j}))$ in a hash table indexed by $q_{i,j} \oplus \alpha_{i,j}, q_{i,j} \oplus \beta_{i,j}$.
5. We search for collisions in the table, each offering a pair of plaintexts and ciphertexts $(q_{i,j}, r_{i,j}, s_{i,j}, t_{i,j}) \rightarrow (\alpha_{i,j}, \beta_{i,j}, \gamma_{i,j}, \delta_{i,j})$ and $(q_{i',j'}, r_{i',j'}, s_{i',j'}, t_{i',j'}) \rightarrow (\alpha_{i',j'}, \beta_{i',j'}, \gamma_{i',j'}, \delta_{i',j'})$ for which $q_{i,j} \oplus q_{i',j'} = \alpha_{i,j} \oplus \alpha_{i',j'} = \beta_{i,j} \oplus \beta_{i',j'} (= r_{i,j} \oplus r_{i',j'})$. This defines the difference x. There should be 2^{63} collisions on average.
6. For each of these pairs, obtain the input difference (f) to round 19, and the expected output difference $(e \oplus \delta_{i,j} \oplus \delta_{i',j'})$, and retrieve the one subkey suggestion (on average) for the subkey of round 19 which satisfies the differential transition[3].

The right subkey is expected to be suggested $2^{71} + 2^{39}$ times, while the wrong ones are expected to be suggested 2^{71} times. With very high probability (of more than 85%), the most suggested subkey is the correct one.

This gives a key-recovery attack on 19-round *SMS4* with 2^{104} chosen plaintexts, and a complexity of 2^{104} time.

C Implementation of the 24-round *Lesamnta* Attack

To verify our attacks, we implemented the attack on 24-round Lesamnta based on symmetry properties of the round function. We cannot find a full preimage because the complexity is too high, but we can show a partial preimage to prove the validity of our attack:

Chaining Value
`33212102 5c23803f 00957df0 94a1d777`
`4953d309 0b3b6624 8c8d523c b14eec82`

Message
`904fe6d0 cdb99073 1949261e de5b3575`
`70e209ed 1fe0a8d0 e7bc6031 6a88ceef`

Output
`03874543 a3a0eef7 3665a8bd 163bdaea`
`8a227d57 a6b6210f ffffffff ffffffff`

D *SHAvite-3*$_{512}$ Message Expansion

The message expansion of *SHAvite-3*$_{512}$ accepts a 1024-bit message block, a 128-bit counter, and a 512-bit salt. All are treated as arrays of 32-bit words (of 32, 4, and 16 words, respectively), which are used to generate 112 subkeys of 128 bits each, or a total of 448 32-bit words.

[3] Note that we only need the differential table of the 8-bit S-Box for this step.

Let $rk[\cdot]$ be an array of 448 32-bit words whose first 32 words are initialized with the message. After the initialization of $rk[0,\ldots,31]$, two processes are repeated, a nonlinear one (which generates 32 new words using the AES round function) and a linear one (which generates the next 32 words in a linear manner). These processes are repeated 6 times, and then the nonlinear process is repeated once more. The computation of $rk[\cdot]$ is done as follows:

Using the counter: the counter is used at 4 specific positions.

In order to simplify the description, we define a new table holding the preprocessed counter:

$$\begin{array}{llll}
ck[\ \ 32] = cnt[0], & ck[\ \ 33] = cnt[1], & ck[\ \ 34] = cnt[2], & ck[\ \ 35] = \overline{cnt[3]} \\
ck[164] = cnt[3], & ck[165] = cnt[2], & ck[166] = cnt[1], & ck[167] = \overline{cnt[0]} \\
ck[440] = cnt[1], & ck[441] = cnt[0], & ck[442] = cnt[3], & ck[443] = \overline{cnt[2]} \\
ck[316] = cnt[2], & ck[317] = cnt[3], & ck[318] = cnt[0], & ck[319] = \overline{cnt[1]}
\end{array}$$

For all the other values, $ck[i] = 0$.

AES rounds: for $i \in \{0, 64, 128, 192, 256, 320, 384\} + \{0, 4, 8, 12, 16, 20, 24, 28\}$:

$tk[(i, i+1, i+2, i+3)] = \mathrm{AESR}(rk[(i+1, i+2, i+3, i)] \oplus salt[(i, i+1, i+2, i+3) \bmod 16])$

Linear Step 1: for $i \in \{32, 96, 160, 224, 288, 352, 416\} + \{0,\ldots,31\}$:

$rk[i] = tk[i-32] \oplus rk[i-4] \oplus ck[i]$

Linear Step 2: for $i \in \{64, 128, 192, 256, 320, 384\} + \{0,\ldots,31\}$:

$rk[i] = rk[i-32] \oplus rk[i-7]$

Once $rk[\cdot]$ is initialized, its 448 words are parsed as 112 words of 128-bit each, which are the subkeys (14 double quartets of 128-bit words each), *i.e.*:

$$\begin{aligned}
RK_{0,i} = (k^0_{0,i}, k^1_{0,i}, k^2_{0,i}, k^3_{0,i}) = \Big(& (rk[32\cdot i], rk[32\cdot i+1], rk[32\cdot i+2], rk[32\cdot i+3]), \\
& (rk[32\cdot i+4], rk[32\cdot i+5], rk[32\cdot i+6], rk[32\cdot i+7]), \\
& (rk[32\cdot i+8], rk[32\cdot i+9], rk[32\cdot i+10], rk[32\cdot i+11]), \\
& (rk[32\cdot i+12], rk[32\cdot i+13], rk[32\cdot i+14], rk[32\cdot i+15])\Big) \\
RK_{1,i} = (k^0_{1,i}, k^1_{1,i}, k^2_{1,i}, k^3_{1,i}) = \Big(& (rk[32\cdot i+16], rk[32\cdot i+17], rk[32\cdot i+18], rk[32\cdot i+19]), \\
& (rk[32\cdot i+20], rk[32\cdot i+21], rk[32\cdot i+22], rk[32\cdot i+23]), \\
& (rk[32\cdot i+24], rk[32\cdot i+25], rk[32\cdot i+26], rk[32\cdot i+27]), \\
& (rk[32\cdot i+28], rk[32\cdot i+29], rk[32\cdot i+30], rk[32\cdot i+31])\Big)
\end{aligned}$$