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

Charles Bouillaguet[1], Orr Dunkelman[1,2],
Gäetan 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

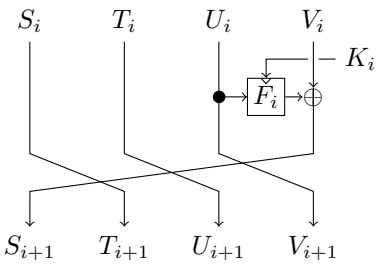
Abstract. In this paper we study the strength of two hash functions which are based on Generalized Feistels. Our proposed attacks themselves are mostly independent of the round function in use, and can be applied to similar hash functions which share the same structure but have different round functions. 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*. We then show a generic integral attack on 20-round *Lesamnta* (which can be used against the block cipher itself). We follow with an attack on 9-round *SHAvite-3*$_{512}$ which is the first cryptanalytic result on the hash function (which also works for the tweaked version of *SHAvite-3*$_{512}$).

1 Introduction

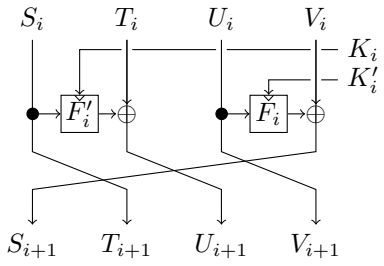
Cryptanalysis of block ciphers and hash functions is mostly based on statistical properties, such as differential cryptanalysis. Another type of attacks is the self-similarity attacks such as the slide and related-key attacks. Recently, algebraic attacks which treat the problem of cryptanalysis as a set of equations to be solved, were also discussed as a cryptographic tool.

In this paper we try a slightly different approach for the cryptanalysis of hash functions, an analysis based on cancellation properties. The main idea behind this approach, which is extremely useful in the cryptanalysis of hash functions, is to cancel the affects of the nonlinear operations using specially selected message words. To some extent, one may consider the local collisions of [2] as such type of cryptanalysis.

In this paper we study two possible ways to build a $4n$-bit Feistel scheme out of a n-bit round function:



Lesamnta structure



SHAvite-3$_{512}$ structure

The keyed function $F(k, x)$ is usually defined as $P(k \oplus x)$, where P is a fixed permutation (or a fixed function). We consider concrete ciphers based on these structures, and we show generic attacks as well as specific attacks using the properties of the fixed function.

The first candidate for the technique is the *Lesamnta* hash function. In the submission document of *Lesamnta*, an attack on 16-round reduced variant is described which is independent of the actual round function. We start with extending this attack to a generic attack on 22-round *Lesamnta* (hereandafter, generic stands for independent of the actual round function). We then extend the attack to 24-round attack (which can be used to find pseudo preimages for the compression function). We follow with an adaptation of this attack to 24-round *Lesamnta*-256 by taking the structure of the true round function into account. We also discuss the 20-round SQUARE attack proposed in [5], and propose improvements of it based on the *cancellation property* of *Lesamnta*.

Finally, we present an attack on 9-round *SHAvite-3*$_{512}$, suggesting the first attack on 9 rounds of *SHAvite-3*$_{512}$. The attack also works for the tweaked version of *SHAvite-3*$_{512}$. The attack makes use of the generalized Feistel structure of *SHAvite-3*$_{512}$, and allows fixing one out of the four output words with two compression function calls. This allows a second preimage attack on 9-round *SHAvite-3*$_{512}$ that takes about 2^{496} time.

The paper is organized as follows. Section 2 explains the basic idea of our cancellation attacks. Section 3 gives a short description of the *Lesamnta* compression function. We briefly describe the previous results on *Lesamnta* in Section 4, and introduce our new attacks in Section 5. SQUARE properties of *Lesamnta* are analyzed in Section 6. We describe a similar result on 9-round *SHAvite-3*$_{512}$ in Section 7. Finally, Section 8 summarizes this paper.

2 The Cancellation Property

In this paper we apply cancellation cryptanalysis to generalized Feistel schemes. In the ideal case, these schemes are used with independent randomly selected round functions. However, in practice, the round functions are usually all derived from a single fixed permutation (or function). This is the basis of the cancellation property.

Our basic attacks are independent of the round function, as long as all the round functions are derived from a single function in the following way: $F_i(X_i) \triangleq F(X_i \oplus K_i)$ for some $F(\cdot)$.

Generic Properties of $F_i(X_i) = F(X_i \oplus K_i)$. Let us 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 e.g., DES and *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*) $\forall \alpha,\ \#\{x : F_i(x) \oplus F_j(x) = \alpha\}$ is even
(*iii*) $\bigoplus_x F_k\big(F_i(x) \oplus F_j(x)\big) = 0$

Property (i) is the basis of the cancellation attack. We will refer to it as the *cancellation property*. It states that if the inputs of two round functions are related by a fixed (specific) difference, then the outputs of both rounds are the same. In a differential attack, this means that if the same difference enters two round functions, and the adversary has some control over the input *values*, he can force the input to the F to be the same, and therefore the output values will be the same. This allows countering the non-linear effects of the round function. The reminder of the paper is exploring this property.

Properties (ii) and (iii) will be used in an integral attack. 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 $\forall x,\ F_i(x) = F_j(x)$. 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 iff $x \oplus K_i \oplus K_j$ 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).

Our attacks use a second 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 take several rounds to affect the full state. Note that the slow diffusion of *Lesamnta* is the basis of a 16-round attack [5], and the slow diffusion of *SHAvite-3*$_{512}$ gives a similar 8-round attack [11].

Let us assume that the input differences of the round function at rounds i and j is δ (in both rounds), and let us denote the corresponding input values by x_i and x_j. The output differences of rounds i and j is $F_i(x_i) \oplus F_i(x_i \oplus \delta)$ and $F_j(x_j) \oplus F_j(x_j \oplus \delta)$, respectively. Thanks to the Cancellation Property, if we set the value $x_i \oplus x_j$ to a particular value $c_{ij} = K_i \oplus K_j$, then the output differences of rounds i and j are the same.

Table 1. Computing Five Rounds of *Lesamnta*.

Round	S_i	T_i	U_i	V_i
0	a	b	c	d
1	$F_0(c) \oplus d$	a	b	c
2	$F_1(b) \oplus c$	$F_0(c) \oplus d$	a	b
3	$F_2(a) \oplus b$	$F_1(b) \oplus c$	$F_0(c) \oplus d$	a
4	$F_3(F_0(c) \oplus d) \oplus a$	$F_2(a) \oplus b$	$F_1(b) \oplus c$	$F_0(c) \oplus d$
5	$F_4(F_1(b) \oplus c) \oplus F_0(c) \oplus d$	$F_3(F_0(c) \oplus d) \oplus a$	$F_2(a) \oplus b$	$F_1(b) \oplus c$

More precisely, let us describe how to use the cancellation property against *Lesamnta*. In Table 1, we show the computation of five rounds of *Lesamnta*. For the description of *Lesamnta*, see Section 3. Consider $S_5 = F_4(F_1(b) \oplus c) \oplus F_0(c) \oplus d$. Using the cancellation property, we can remove the dependency of S_5 on c if $F_1(b) = K_0 \oplus K_4$:

$$\begin{aligned} S_5 &= F_4(F_1(b) \oplus c) \oplus F_0(c) \oplus d \\ &= F(K_4 \oplus F_1(b) \oplus c) \oplus F(K_0 \oplus c) \oplus d \\ &= F(K_0 \oplus c) \oplus F(K_0 \oplus c) \oplus d = d. \end{aligned}$$

Therefore, we can put any difference in U_0, since it will not affect S_5 as long as we fix the value of $T_0 = F^{-1}(K_0 \oplus K_4) \oplus K_1$. Note that in a hash function setting those keys are known to the adversary (or controlled by him).

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 more than once as the input difference to a round function.
- The round functions are built from a fixed permutation (or a function), using a small round key. This differs from the ideal Feistel 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 requirement are not strictly necessary. In Section 6 we show an integral attack on 20-rounds of the inner block cipher of *Lesamnta* which does not assume knowledge of the keys. Moreover, in Section 7 we show an attack on *SHAvite-3*$_{512}$, where the round function uses more keying material.

3 A Short Description of *Lesamnta*

Lesamnta is a hash function proposal by Hirose, Kuwakado, and Yoshida as a candidate in the SHA-3 competition [5]. It makes an extensive use of the building blocks of the AES. Four algorithms are in fact described, *Lesamnta*-224, *Lesamnta*-256, *Lesamnta*-384, and *Lesamnta*-512. While each algorithm has its own digest size, *Lesamnta*-224 and *Lesamnta*-256 share many of their building blocks, just like *Lesamnta*-384 and *Lesamnta*-512. All the functions are built in a standard Merkle-Damgård iteration of a compression function. The compression functions themselves are built using the Matyas-Meyer-Oseas (MMO) transformation of a block cipher:

$$CF(h, m) = E_h(m) \oplus m$$

We note that *Lesamnta* is composed of two block ciphers. The first one is used in the 224- and 256-bit versions, while the second one is used in the 384- and 512-bit versions. Both share many of the design and the main differences are mostly related to the word size. We mostly describe the 256-bit block cipher, and we pinpoint the differences with the 512-bit one when necessary. Also, note that a different block cipher is used in the last invocation of the compression function. As this has no relevance to our attacks, we do not describe the differences in this paper.

The block cipher has a key schedule algorithm which expands the chaining value (key) into 32 words (this is the "chaining value expansion" of the compression function). As our results are independent of the key "chaining value expansion", we omit its description, and refer the interested reader to [5]. The "compressing part" encrypts the message block in 32 successive rounds, using 32 subkeys. Both the round function and the key expansion are based on a 4-thread unbalanced Feistel networks. Each thread is 64-bit wide (128-bit wide for *Lesamnta*-512).

Round Function. Four other registers S_0, T_0, U_0, and V_0 are initialized with the message block (plaintext) and then updated according to the following process:

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

The F function resembles four AES rounds where the AddRoundKey operation is omitted. An alternative description is possible (where we only keep the S register, and use the name X to avoid confusion with the "standard' values of S):

$$X_i = X_{i-4} \oplus F\left(X_{i-3} \oplus K_i\right)$$

where the message block is initially loaded into X_{-3}, X_{-2}, X_{-1} and X_0. The last four values of X are the output of the block cipher (before the feed forward operation). The F_{256} function used in *Lesamnta*-256 is a permutation over 64 bits (F_{512} is the permutation over 128 bits used in *Lesamnta*-512). It is similar to four AES rounds up to the omission of the AddRoundKey after each round (whereas F_{512} is precisely four AES rounds, without the AddRoundKey operation). F_{256} operates over a 2×4 byte matrix and iterates the following operations four times:

1. The AES S-box is applied to all bytes.
2. The second row is cyclically shifted: $(a, b, c, d) \rightarrow (b, c, d, a)$. In this paper, we denote this operation by "`ShiftRows`".
3. A 2×2 MDS matrix over $GF(2^8)$ is applied in parallel to the four columns. We denote this operation by "`MixColumns`". The matrix is defined by:

$$M_F = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$$

4 Previous Results on *Lesamnta*

Several attacks on reduced-round *Lesamnta* are presented in [5]. 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:

Round	S_i	T_i	U_i	V_i
Input	Δ_0	Δ_1	Δ_2	$\Delta_3 \oplus \delta$
0	Δ_3	Δ_0	Δ_1	Δ_2
1	0	Δ_3	Δ_0	Δ_1
2	0	0	Δ_3	Δ_0
3	0	0	0	Δ_3
4	Δ_3	0	0	0
5	0	Δ_3	0	0
6	0	0	Δ_3	0
7	?	0	0	Δ_3
8	Δ_3	?	0	0
9	0	Δ_3	?	0
10	?	0	Δ_3	?
11	?	?	0	Δ_3
12	Δ_3	?	?	0
13	?	Δ_3	?	?
14	?	?	Δ_3	?
15	?	?	?	Δ_3
Feedforward	?	?	?	δ

where

$$\begin{aligned}
\Delta_2 &= M_2 \oplus F^{-1}(F(M_2 \oplus K_0) \oplus \delta) \oplus K_0,\\
\Delta_1 &= M_1 \oplus F^{-1}(F(M_1 \oplus K_1) \oplus \Delta_2) \oplus K_1,\\
\Delta_0 &= M_0 \oplus F^{-1}(F(M_0 \oplus K_2) \oplus \Delta_1) \oplus K_2,\\
\Delta_3 &= (M_3 \oplus \delta) \oplus F^{-1}(F(M_0 \oplus K_3 \oplus \delta) \oplus \Delta_0) \oplus K_3,
\end{aligned}$$

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 at most two queries to the compression function (independent of the chaining value). This is done by picking a random message m, and checking whether the fourth output word has the desired value, d. If not, let the value of the fourth word be d'. It is possible to pick $m' = m \oplus (\Delta_0, \Delta_1, \Delta_2, \Delta_3 \oplus \delta)$ (where $\delta = d \oplus d'$), for which it is assured that the fourth output word is d.

This allows a more efficient 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 function is an ideal permutation. We also note that to some extent the 16-round attack may be considered "more" generic, as it allows for *different and completely* independent round functions all together.

5 New Collision and Preimage Attacks on Reduced-Round *Lesamnta*

In this section we show some improvements to the 16-round attacks of the submission document. We first show some attacks that are generic in F, as long as the round functions are defined as $F_i(X_i) = F(X_i \oplus K_i)$. After showing the new generic attacks (on 22, 23, and 24 rounds), we use a specific property of the round function of *Lesamnta*-256 to obtain a better 24-round attack.

5.1 Generic Attacks

The cancellation property of Section 2 can be used three times, thus, controlling a specific word up to 17 rounds, as shown in Table 2. We end up with $X_{17} = F(c \oplus \alpha) \oplus \beta$, where

$$\alpha = K_9 \oplus F_6(F_3(a) \oplus b) \oplus F_2(b) \qquad \text{and} \qquad \beta = d$$

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

round 5 $F_2(b) = c_{1,5} = K_1 \oplus K_5$ $(\Rightarrow b \triangleq F_2^{-1}(K_1 \oplus K_5))$
round 11 $F_8(d) = c_{7,11} = K_7 \oplus K_{11}$ $(\Rightarrow d \triangleq F_8^{-1}(K_7 \oplus K_{11})$
round 17 $F_{14}(F_3(a) \oplus b) = K_{13} \oplus K_{17}$ $(\Rightarrow a \triangleq F_3^{-1}(F_{14}^{-1}(K_{13} \oplus K_{17}) \oplus b))$

Hence, one can set X_{17} to any desired value by setting:

$$c = F_9^{-1}(X_{17} \oplus d) \oplus F_6(F_3(a) \oplus b) \oplus F_2(b).$$

Table 2. Repeating the Cancellation Property

Round	X_i
-3	d
-2	c
-1	b
0	a
1	$F_1(c) \oplus d$
2	$F_2(b) \oplus c$
3	$F_3(a) \oplus b$
4	$F_4(F_1(c) \oplus d) \oplus a$
5	$\cancel{F_5(F_2(b) \oplus c)} \oplus \cancel{F_1(c)} \oplus d$
6	$F_6(F_3(a) \oplus b) \oplus F_2(b) \oplus c$
7	$F_7(F_4(F_1(c) \oplus d) \oplus a) \oplus F_3(a) \oplus b$
8	$F_8(d) \oplus F_4(F_1(c) \oplus d) \oplus a$
9	$F_9(F_6(F_3(a) \oplus b) \oplus F_2(b) \oplus c) \oplus d$
10	$F_{10}(F_7(F_4(F_1(c) \oplus d) \oplus a) \oplus F_3(a) \oplus b) \oplus F_6(F_3(a) \oplus b) \oplus F_2(b) \oplus c$
11	$\cancel{F_{11}(F_8(d) \oplus F_4(F_1(c) \oplus d) \oplus a)} \oplus \cancel{F_7(F_4(F_1(c) \oplus d) \oplus a)} \oplus F_3(a) \oplus b$
12	?
13	$F_{13}(X_{10}) \oplus F_9(F_6(F_3(a) \oplus b) \oplus F_2(b) \oplus c) \oplus d$
14	$F_{14}(F_3(a) \oplus b) \oplus X_{10}$
15	?
16	?
17	$\cancel{F_{17}(F_{14}(F_3(a) \oplus b) \oplus X_{10})} \oplus \cancel{F_{13}(X_{10})} \oplus F_9(F_6(F_3(a) \oplus b) \oplus F_2(b) \oplus c) \oplus d$

Table 3. Collision and Preimage Characteristic for the 22-Round Attack.
α and β can be computed from a, b, d and the key:
$\alpha = K_{11} \oplus F_8(F_5(a) \oplus b) \oplus F_4(b)$, $\beta = d$

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

22-Round Attacks. As a straightforward application of the cancellation path in Table 2, we can build attacks on 22-round *Lesamnta*. We just add two rounds at the beginning, and set the values of the state variables as given in Table 3. A dash (-) is used to denote a value that is independent of c. A different interpretation of this property as a differential characteristic is described in Table 4.

This characteristics allows to choose the last output word of the compression function. The attack is similar to the generic attack on 16 rounds described in [5]. Let H be the target value of the fourth output word, then the attack proceeds as follows:

1. Set a, b, and d to the values that allow the cancellation property (as described in Section 5.1) and pick a random value for c. This sets the state at round 2: S_2, T_2, U_2, V_2.
2. Compute the round function backwards up to round 0, and downwards to round 22.

Table 4. Differential Characteristic for the 22-Round Attacks

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

3. If V_{22} is the desired value (i.e., $V_{22} = H \oplus V_0$), stop. Otherwise, compute the desired value V_{22}^* for V_{22} as $V_{22}^* = H \oplus V_0$.
4. Compute $U_2^* = F^{-1}(V_{22}^* \oplus \beta) \oplus \alpha$ (for the required α and β described above).
5. Starting from the state S_2, T_2, U_2^*, V_2, computing two rounds backwards to obtain S_0^*, T_0^*, U_0^*, V_0 we are assured to generate the desired output.

This costs at most two compression function calls, and does not require any memory.

For a given chaining value (i.e., a set of subkeys), there is only one message which this algorithm can output. To make a full preimage or 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^{97} (2^{193} for *Lesamnta*-512), and a second-preimage attack with complexity 2^{193} (2^{385} for *Lesamnta*-512).

23-Round Attacks Table 5 shows what happens when add round at the beginning. If we add one extra round, we end up with steps 1-24 of Table 6, without the first step. The output word we try to control is equal to $c \oplus F(c \oplus \alpha) \oplus \beta \oplus \gamma$. Define $h_\alpha(x) = x \oplus F(x \oplus \alpha)$, and build a big inversion table for all h_α. Using this inversion table, we can choose one output word just like in the 22-round attack.

We note that there are $2^{n/4}$ tables, each of size $2^{n/4}$. Once a, b, d are fixed, the adversary just picks the corresponding table. Hence, if we start with a, b, d satisfying the cancellation conditions, and a random c, the last output word of the compression function is $V_0 \oplus V_{23} = h_\alpha(c) \oplus \beta \oplus \gamma$. If the

Table 5. Going up from the state (a, b, c, d).
$X_1 = c \oplus \gamma$, with $\gamma = F_1(b \oplus F_2(a \oplus F_3(d)))$
$X_0 = \lambda \oplus F_0(c \oplus \gamma)$, with $\lambda = d$

Round	X_i
0	$d \oplus F_0(c \oplus F_1(b \oplus F_2(a \oplus F_3(d))))$
1	$c \oplus F_1(b \oplus F_2(a \oplus F_3(d)))$
2	$b \oplus F_2(a \oplus F_3(d))$
3	$a \oplus F_3(d)$
4	d
5	c
6	b
7	a

target value is H, we set $\delta = H \oplus V_0 \oplus V_{23}$ and we just have to use the value $c^* = h_\alpha^{-1}(h_\alpha(c) \oplus \delta)$, so that $h_\alpha(c^*) = h_\alpha(c) \oplus \delta$.

Table 6. Collision and preimage path for the 24-round attack.
α, β, γ and λ can be computed from a, b, d and the key:
$\alpha = K_{13} \oplus F_{10}(F_7(a) \oplus b) \oplus F_6(b)$, $\beta = d$
$\gamma = F_1(b \oplus F_2(a \oplus F_3(d)))$, $\lambda = d$

round	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 2			
21	$F(c \oplus \alpha) \oplus \beta$	?	?	?
22	?	$F(c \oplus \alpha) \oplus \beta$	?	?
23	?	?	$F(c \oplus \alpha) \oplus \beta$	?
24	?	?	?	$F(c \oplus \alpha) \oplus \beta$

24-Round Attacks Similarly, we can still add one extra round at the beginning of the path. The resulting 24-round path is given in Table 6. The output word we try to control is equal to $F(c \oplus \gamma) \oplus F(c \oplus \alpha) \oplus \beta \oplus \lambda$, for some constants α, β, γ and λ that depend on the chaining value. We will have to invert the family of functions $h_{\alpha,\gamma}(x) = F(x \oplus \alpha) \oplus F(x \oplus \gamma)$.

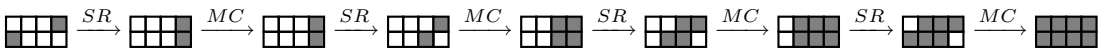
5.2 Attacks Using Specific Properties of the Round Function

We now use a specific property of the round function of *Lesamnta*-256 to improve our attacks. We will show a preimage and collision attack on 24-round without the need of the huge table of the previous attacks.

Neutral Subspaces in F_{256}. We first describe a property of the F function of *Lesamnta*-256 which limits the difference to some subspace of this function. If the input difference to the round

function is in a space Γ, then the output difference is in a space Λ. For some input differences of this function, the output is in another subspace. This property can be used with a new differential characteristic, similarly to those used in [5].

The round function of *Lesamnta*-256 achieves full diffusion of the values, but some linear combinations of the output are not modified after 4 rounds. Starting from a single active column, we have:



All the output bytes are active, but there are some linear relations between them, since the MixColumns operation is linear. Indeed, the last MixColumns is a linear mapping and a difference on 16 bits in the input of this operation is changed into another difference on the 64 bits of output. However, as can be easily seen, all these differences must be of a special form (i.e., that 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.

In terms of difference, another description of this property is that there exists linear subspaces Γ and Λ, such that

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

Γ has dimension 16, and Λ has dimension 48 (out of 64 bits). This property is used to build specific attacks on *Lesamnta*-256.

Improved Collision and Preimage Attacks. This property can be used in an attack following the path of Table 6.

This attack allows to choose 16 linear relations of the output of the compression function, which correspond to the the projection of the output on the subspace Λ (cf. 5.2). The adversary first selects a state at round 4 satisfying the conditions of the extended cancellation property. a, b, and d will be fixed, while c can be chosen at random. Then he computes 20 rounds forwards and 4 rounds backwards, and gets the output through the feed-forward.

He checks that the output value is in Λ and if it gives the desired 16 bits of output, the adversary can modify c with a difference in Γ and get 2^{16} messages for which the same linear relations still holds. This gives an amortized cost of 1 compression function call per message with 16 chosen bits.

This allows to make a second-preimage attack on *Lesamnta*-256 (a preimage attack on the compression function) reduced to 24 rounds with complexity 2^{240}, and a collision attack on the compression function with complexity 2^{120}.

6 A Integral Distinguisher for 20-Round *Lesamnta*

In this section, we show the applicability of the cancellation technique to block ciphers. We will show 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 can not select the message that will ensure that the cancellation property will happen. Therefore, we use an integral cryptanalysis to exploit the cancellation property. The basic idea is to use a set of messages, knowing that a subset of the messages in the set follow the cancellation path.

In the original submission document of *Lesamnta* [5] a 19-round SQUARE distinguisher is described. This SQUARE is very straightforward, and suggests an efficient distinguisher for *Lesamnta*. However, the original SQUARE is faulty. This was found by experimenting with

reduced version, and we will give an explanation of why the SQUARE attack does not work. Then we suggest an improved and corrected 20-round integral attack which relies on the cancellation property. We use the term integral cryptanalysis rather than SQUARE to describe our new attack, because we use a higher order property.

In Table 7, 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.

However, 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. But this leads to a contradiction later on because when we reuse symbols we are performing a change of variable, but it 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.

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

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

Round	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 [5]

Round	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

6.1 The New Attack

In Table 8, we describe the integral property used in our new attack.

At round 8, we use the value $F_8(C \oplus b) \oplus F_4(b)$. Using Property (i), we know that the sum will vanish for a particular C. For the rest of the path, we assume that C takes this particular value.

At round 14, we have $F_{14}(F_7(F_4(b))) \oplus F_{10}(F_7(F_4(b)))$. This has the special property that each value is taken an even number of times, according to Property (ii).

At round 17, we have $F_{17}(F_{14}(F_7(F_4(b))) \oplus F_{10}(F_7(F_4(b)))) \oplus F_{13}(F_{10}(F_7(F_4(b)))) \oplus b$. When we sum this over all b's, we have:

$$\bigoplus_b F_{17}\Big(F_{14}\big(F_7(F_4(b))\big) \oplus F_{10}\big(F_7(F_4(b))\big)\Big) \oplus \bigoplus_b F_{13}(F_{10}(F_7(F_4(b)))) \oplus \bigoplus_b b$$

The first term sums to zero because each input to F_{17} is taken an even number of times (cf. Property (iii)), and the two last terms cancel each other because both are permutations of b. This value is the fourth output word after 20 rounds ($S_{17} = V_{20}$), so our distinguisher works on 20 rounds.

This property can be used to attack the block cipher of *Lesamnta*. The adversary takes $2^{n/2}$ plaintexts, iterating over all possible C and b. Then, for each C, he computes the sum of V_{20} over all b, and checks whether this sum is zero. If the data was generated using *Lesamnta*'s compression functions, he knows that at least one value of C gives a zero sum over b. Otherwise, there is a $1/e$ probability that all the sums are non-zero. By querying a new set of $2^{n/4}$ plaintexts with the same C, one can easily identify that the right C value was used, and even retrieve some keying material (specifically, $K_4 \oplus K_8$). This has been experimentally verified on reduced versions.

Table 8. Integral Attack.
The second part of the path assumes that C is fixed to be $K_4 \oplus K_8$, and only gives the dependencies in b.

-3	b
-2	C
-1	-
0	-
1	b
2	C
3	-
4	$F_4(b)$
5	$C + b$
6	C
7	$F_7(F_4(b))$
8	~~$F_8(C \oplus b)$~~ $\oplus$ ~~$F_4(b)$~~
9	b
10	$F_{10}(F_7(F_4(b)))$
11	$F_7(F_4(b))$
12	$F_{12}(b)$
13	$F_{13}(F_{10}(F_7(F_4(b)))) \oplus b$
14	$F_{14}(F_7(F_4(b))) \oplus F_{10}(F_7(F_4(b)))$
15	$F_{15}(F_{12}(b)) \oplus F_7(F_4(b))$
16	$F_{16}(F_{13}(F_{10}(F_7(F_4(b)))) \oplus b) \oplus F_{12}(b)$
17	$F_{17}(F_{14}(F_7(F_4(b))) \oplus F_{10}(F_7(F_4(b)))) \oplus F_{13}(F_{10}(F_7(F_4(b)))) \oplus b$

7 9-Round Attack on *SHAvite-3*$_{512}$

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

SHAvite-3$_{512}$ is part of the SHA-3 candidate *SHAvite-3*, designed to produce outputs of 257–512 bits [1]. *SHAvite-3*$_{512}$ is based on a 4-thread Feistel construction with two nonlinear functions in each round, where the nonlinear functions themselves are composed of four full AES rounds (with a whitening key before the first round, and where the last AddRoundKey operation is omitted). The key schedule algorithm accepts a message (*SHAvite-3*$_{512}$ employs the Davies-Meyer mode) of 1024 bits, a salt of 512 bits, and a counter of 128 bits, which are then transformed into 112

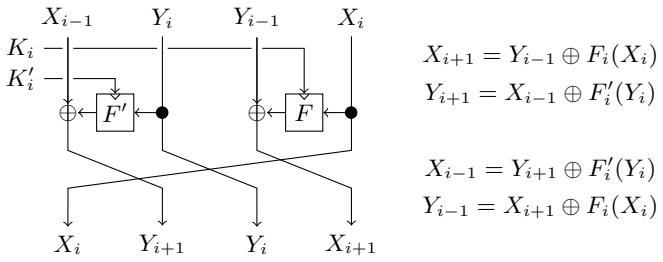


Fig. 1. The Underlying Block Cipher of C_{512}

subkeys of 128 bits each. We note that the original key schedule allowed for a specific set of message, salt, and counter to lead to all the subkeys being zero, and thus was tweaked. We note that our attack applies both for the original message expansion as well as the tweaked version.

SHAvite-3$_{512}$ is based on the compression function C_{512}, which 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 E^{512}. The chaining value is divided into four 128-bit words, and each round two of these 128-bit words enter the nonlinear round function and affect the other two (each word enters one nonlinear function and affect one word). The nonlinear function $F^4(\cdot)$ is composed of four full rounds of AES. The compression function is depicted in Figure 1 and the message expansion is given in Appendix A.

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 as it is defined as

$$F(k_0, k_1, k_2, k_3, x) = P(k_3 \oplus P(k_2 \oplus P(k_1 \oplus P(k_0 \oplus x)))).$$

To apply the cancellation property to *SHAvite-3*$_{512}$, we will need that the inner keys of the two cancelling functions are the same, so that both F functions collapses to $P(k_0 \oplus x)$. In Appendix B we discuss an algorithm to find such a key through the message expansion.

The cancellation path is described in Table 9. We have a cancellation property on 7 rounds under the following condition:

Round 7 $\forall x, F'_4(x) = F_6(F_5(c \oplus F_4(d)) \oplus x)$.
This happens if $k^0_{1,4} = k^0_{0,6} \oplus F_5(c \oplus F_4(d))$ and $(k^1_{1,4}, k^2_{1,4}, k^3_{1,4}) = (k^1_{0,6}, k^2_{0,6}, k^3_{0,6})$.

Moreover, if we can satisfy an extra condition at round 9, we have a cancellation property on 9 round:

Round 9 $\forall x, F'_6(x) = F_8(F_7(a) \oplus x)$.
This happens if $k^0_{1,6} = k^0_{0,8} \oplus F_7(a)$ and $(k^1_{1,6}, k^2_{1,6}, k^3_{1,6}) = (k^1_{0,8}, k^2_{0,8}, k^3_{0,8})$.

The constraints for the 7-round property are easy to satisfy and allow a 7-round attack on *SHAvite-3*$_{512}$. If we can additionally satisfy the constraints at round 9, then we can have an attack on 9-round *SHAvite-3*$_{512}$.

7.2 Dealing with the key expansion

To make the attack on 9-rounds of *SHAvite-3*$_{512}$, we need to satisfy a 768-bit condition on the subkeys:

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

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

Round	X_{i-1}	Y_i	Y_{i-1}	X_i
0	X_{-1}	Y_0	Y_{-1}	X_0
		(4 rounds)		
4	a	b	c	d
		(5 rounds)		
9	X_8	Y_9	Y_8	X_9
FF	$X_8 \oplus X_{-1}$	$Y_9 \oplus Y_0$	$Y_8 \oplus Y_{-1}$	$X_9 \oplus X_0$

Round	X_i
X_0	$d \oplus F_3(a) \oplus F_1'(a \oplus F_2(b \oplus F_3'(c)))$
Y_0	$b \oplus F_3'(c) \oplus F_1(c \oplus F_2'(d \oplus F_3(a)))$
X_1	$c \oplus F_2'(d \oplus F_3(a))$
Y_1	$a \oplus F_2(b \oplus F_3'(c))$
X_2	$b \oplus F_3'(c)$
Y_2	$d \oplus F_3(a)$
X_3	a
Y_3	c
X_4	d
Y_4	b
X_5	$c \oplus F_4(d)$
Y_5	$a \oplus F_4'(b)$
X_6	$b \oplus F_5(c \oplus F_4(d))$
Y_6	$d \oplus F_5'(a \oplus F_4'(b))$
X_7	$a \oplus \cancel{F_4'(b)} \oplus \cancel{F_6(b \oplus F_5(c \oplus F_4(d)))}$
Y_7	$c \oplus F_4(d) \oplus F_6'(d \oplus F_5'(a \oplus F_4'(b)))$
X_8	$d \oplus F_5'(a \oplus F_4'(b)) \oplus F_7(a)$
X_9	$c \oplus F_4(d) \oplus \cancel{F_6'(d \oplus F_5'(a \oplus F_4'(b)))} \oplus \cancel{F_8(d \oplus F_5'(a \oplus F_4'(b)) \oplus F_7(a))}$

In Appendix B we outline a technique to find a suitable message (recall that *SHAvite-3*$_{512}$ is used in a Davies-Meyer mode). 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 Nikolic [10] to find a good message efficiently.

7.3 Applying Cancellation Attack to *SHAvite-3*$_{512}$

The cancellation property allows to find a key/message pair with a given value on the last 128-bit for an amortized cost of one hash function evaluation. The attack is the following: first find a message that fulfills the conditions on the subkeys. Then pick a state a, c, d at round 4 satisfying the cancellation conditions, and compute 5 rounds forwards and 4 rounds backwards as shown in Table 9. Then, X_9 is fixed, and it is easy to compute the b that gives the desired X_0. Each key (message) can be used with 2^{128} different a,c,d, and the cost of finding a suitable key is 2^{224} (see Appendix B). Hence, the amortized cost for finding one value is 2^{96}. Hence, the cost of finding a pseudo-preimage for the compression function is 2^{480}. The full attack is described in Appendix B.

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

Input: Target value $\tilde{H}$
Output: A message M and a chaining value X s.t. $CF(X, M) = \tilde{H}$
Running Time: 2^{480}

loop
  Find a message M s.t. $(k^1_{1,4}, k^2_{1,4}, k^3_{1,4}) = (k^1_{0,6}, k^2_{0,6}, k^3_{0,6})$ and $(k^1_{1,6}, k^2_{1,6}, k^3_{1,6}) = (k^1_{0,8}, k^2_{0,8}, k^3_{0,8})$ (see Appendix B for the exact algorithm).
  $a \leftarrow F_7^{-1}(k^0_{1,6} \oplus k^0_{0,8})$
  for all c **do**
    $d \leftarrow F_4^{-1}(F_5^{-1}(k^0_{1,4} \oplus k^0_{0,6}) \oplus c)$
    Compute b as $F_2^{-1}(F_1'^{-1}(\tilde{H}_4 \oplus c \oplus F_4(d) \oplus d \oplus F_3(a)) \oplus a) \oplus F_3'(c)$
    Compute 4 rounds backwards and 5 rounds forwards from a, b, c, d
    Then $H_4 = X_0 \oplus X_9 = \tilde{H}_4$
    if $H = \tilde{H}$ **then**
      return X, M
    end if
  end for
end loop

8 Conclusion

In this paper, we presented attacks on the *Lesamnta* hash function and on the *SHAvite-3*$_{512}$ hash function. We summarize the obtained attacks in Tables 10 and 11.

These attacks are based on the cancellation property which we show to be an efficient technique to study generalized Feistel structures.

Table 10. Summary of the Attacks on *Lesamnta*

Version	Attack	Rounds	Complexity		
			Data	Time	Memory
Lesamnta-256	Collision [5]	16	2^{97}	2^{97}	-
Lesamnta-256	Second Preimage [5]	16	2^{193}	2^{193}	-
Lesamnta-512	Collision [5]	16	2^{193}	2^{193}	-
Lesamnta-512	Second Preimage [5]	16	2^{385}	2^{385}	-
Lesamnta-256	Collision (Sect. 5.1)	22	2^{97}	2^{97}	-
Lesamnta-256	Second Preimage (Sect. 5.1)	22	2^{193}	2^{193}	-
Lesamnta-512	Collision (Sect. 5.1)	22	2^{193}	2^{193}	-
Lesamnta-512	Second Preimage (Sect. 5.1)	22	2^{385}	2^{385}	-
Lesamnta-256	Second Preimage (Sect. 5.1)	23	2^{193}	2^{193}	2^{128}
Lesamnta-512	Second Preimage (Sect. 5.1)	23	2^{385}	2^{385}	2^{256}
Lesamnta-256	Second Preimage (Sect. 5.1)	24	2^{193}	2^{193}	2^{192}
Lesamnta-512	Second Preimage (Sect. 5.1)	24	2^{385}	2^{385}	2^{384}
Lesamnta-256	Collision (Sect. 5.2)	24	2^{120}	2^{120}	-
Lesamnta-256	Second Preimage (Sect. 5.2)	24	2^{240}	2^{240}	-

Table 11. Summary of the Attacks on *SHAvite-3*$_{512}$

Attack	Rounds	Complexity	
		Data	Time
Second Preimage [11]	8	2^{448}	2^{448}
Second Preimage (Sect. 7)	9	2^{496}	2^{496}

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.

References

1. Eli Biham and Orr Dunkelman, *The SHAvite-3 Hash Function*, submission for SHA-3 competition, 2008.
2. Florent Chabaud, Antoine Joux, *Differential Collisions in SHA-0*, Advances in Cryptology, proceedings of CRYPTO 1998, Lecture Notes in Computer Science 1462, pp. 56–71, Springer, 1998.
3. Christophe De Cannière and Christian Rechberger, *Finding SHA-1 Characteristics: General Results and Applications*, Advances in Cryptology, proceedings of ASIACRYPT 2006, Lecture Notes in Computer Science 4284, pp. 1–20, Springer, 2006
4. Joan Daemen and Vincent Rijmen, *Plateau characteristics*, IET Information Security, vol. 1, issue 1, pp. 11–17, March 2007.
5. Shoichi Hirose, Hidenori Kuwakado, and Hirotaka Yoshida, *SHA-3 Proposal: Lesamnta*, submission for SHA-3 competition, 2008.
6. National Institute of Standard and Technology, *NIST's Plan for New Cryptographic Hash Functions*, available on-line at `http://www.csrc.nist.gov/pki/HashWorkshop/index.html`, 2007.
7. Tri Van Le, Rüdiger Sparr, Ralph Wernsdorf, and Yvo Desmedt, *Complementation-Like and Cyclic Properties of AES Round Functions*, proceedings of AES Conference 2004, Lecture Notes in Computer Science 3373, pp. 128–141, Springer, 2004.
8. Xiaoyun Wang and Hongbo Yu, *How to Break MD5 and Other Hash Functions*, Advances in Cryptology, proceedings EUROCRYPT 2005, Lecture Notes in Computer Science 3494, pp. 19–35, Springer, 2005.
9. Xiaoyun Wang, Hongbo Yu, and Yiqun Lisa Yin, *Finding Collisions in the Full SHA-1*, Advances in Cryptology, proceedings of CRYPTO 2005, Lecture Notes in Computer Science 3621, pp. 17–36, Springer, 2005.
10. Dmitry Khovratovich, Alex Biryukov and Ivica Nikolic, *Speeding up Collision Search for Byte-Oriented Hash Functions*, proceeding of CT-RSA 2009, Lecture Notes in Computer Science 5473, pp. 164–181, Springer, 2009.
11. Private communication *A preimage attack on 8-round SHAvite-3-512*, Graz ECRYPT meeting 2009.

A *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.

Let $rk[\cdot]$ be an array of 448 32-bit words whose first 32 words are initialized with $msg[0,\ldots,31]$ (the message). Besides the message, the key schedule algorithm accepts a counter $cnt[0,\ldots,3]$

and a salt $salt[0,\ldots,15]$, which in our attacks we assume to be fixed to some pre-determined value.

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.

As the counter has no effect on the results of the paper, we omit its use from the description. The computation of $rk[\cdot]$ is done as follows (up to the counter):

- For $i = 0,\ldots,31$ set $rk[i] \leftarrow msg[i]$.
- Set $i \leftarrow 32$
- Repeat six times:
  1. **Nonlinear Expansion Step:** Repeat twice:
     (a) Let
     $$t[0..3] = AESR\Big((rk[i-31]||rk[i-30]||rk[i-29]||rk[i-32]) \oplus (salt[0]||salt[1]||salt[2]||salt[3])\Big).$$
     (b) For $j = 0,\ldots,3$: $rk[i+j] \leftarrow t[j] \oplus rk[i-4+j]$.
     (c) $i \leftarrow i+4$.
     (d) Let
     $$t[0..3] = AESR\Big((rk[i-31]||rk[i-30]||rk[i-29]||rk[i-32]) \oplus (salt[4]||salt[5]||salt[6]||salt[7])\Big).$$
     (e) For $j = 0,\ldots,3$: $rk[i+j] \leftarrow t[j] \oplus rk[i-4+j]$.
     (f) $i \leftarrow i+4$.
     (g) Let
     $$t[0..3] = AESR\Big((rk[i-31]||rk[i-30]||rk[i-29]||rk[i-32]) \oplus (salt[8]||salt[9]||salt[10]||salt[11])\Big).$$
     (h) For $j = 0,\ldots,3$: $rk[i+j] \leftarrow t[j] \oplus rk[i-4+j]$.
     (i) $i \leftarrow i+4$.
     (j) Let
     $$t[0..3] = AESR\Big((rk[i-31]||rk[i-30]||rk[i-29]||rk[i-32]) \oplus (salt[12]||salt[13]||salt[14]||salt[15])\Big).$$
     (k) For $j = 0,\ldots,3$: $rk[i+j] \leftarrow t[j] \oplus rk[i-4+j]$.
     (l) $i \leftarrow i+4$.
  2. **Linear Expansion Step:** Repeat 32 times:
     (a) $rk[i] \leftarrow rk[i-32] \oplus rk[i-7]$.
     (b) $i \leftarrow i+1$.
- Repeat the **Nonlinear Expansion Step** an additional time.

where $AESR(x)$ is an AES round (without the AddRoundKey operation).

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.,

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

B Dealing with the key expansion of *SHAvite-3*$_{512}$

For the cancellation attack on 9-round *SHAvite-3*$_{512}$, we need to solve a set of equations on the subkeys:

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

Or in $rk[\cdot]$ terms:

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

For the description of the algorithm we define $tk[\cdot]$ to be the values after applying $AESR(\cdot)$ to $rk[\cdot]$, i.e.,

$$tk[156, \ldots, 159] = AESR\Big((rk[157]||rk[158]||rk[159]||rk[156]) \oplus (salt[12]||salt[13]||salt[14]||salt[15])\Big)$$

We are actually trying 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.

B.1 Propagation of the constraints

First, we will propagate the constraints and deduce new equalities between the variables.

The non-linear equation of the key-schedule give:

$$tk[156,\ldots,159] = AESR\Big((rk[157]||rk[158]||rk[159]||rk[156]) \oplus (salt[12]||salt[13]||salt[14]||salt[15])\Big)$$
$$tk[204,\ldots,207] = AESR\Big((rk[205]||rk[206]||rk[207]||rk[204]) \oplus (salt[12]||salt[13]||salt[14]||salt[15])\Big)$$

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

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 that $rk[187..191] = rk[235..239]$.

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 shown that $rk[190] = rk[238]$ and $tk[158] = tk[206]$, therefore $rk[186] = rk[234]$. Similarly, we get $rk[176..186] = rk[224..234]$.

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 $rk[186] = rk[234]$ and $rk[218] = rk[266]$, 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 $rk[249..255] = rk[153..159]$.

B.2 Finding a solution

To find a solution to the system, we will use a guess and determine technique. We guess 15 state variables, and we will 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 one time out of $2^{32\times 7} = 2^{224}$. This gives a complexity of 2^{224} to find a good message.

- Choose a random value from $rk[209..223]$
- Compute $rk[184..191]$ from $rk[209..223]$
- Compute $tk[156..159]$ from $rk[184..191]$
- Compute $rk[156..159]$ from $tk[156..159]$. Note that $rk[252..255] = rk[156..159]$.
- Compute $tk[212..223]$ from $rk[212..223]$.
- Compute $rk[240..251]$ from $tk[212..223]$ and $rk[252..255]$.
- Compute $tk[208..211]$ from $rk[240..243]$ and $rk[236..239]$ $(= rk[188..191])$.
- Compute $rk[208..211]$ from $tk[208..211]$ and check consistency with the initial values of $rk[209..211]$.
- Get $rk[201..207]$ from $rk[249..255]$.
- Compute $rk[176..183]$ from $rk[201..215]$.
- Get $rk[224..239]$ from $rk[176..191]$. One full state $rk[224..255]$ has been recovered, and it possible to check the consistency of the solution.