Known-Key Distinguishers on 11-Round Feistel and Collision Attacks on Its Hashing Modes

Yu Sasaki and Kan Yasuda

NTT Information Sharing Platform Laboratories, NTT Corporation, Japan
{sasaki.yu,yasuda.kan}@lab.ntt.co.jp

Abstract. We present new attacks on the Feistel network, where each round function consists of a subkey XOR, S-boxes, and then a linear transformation (*i.e.*, an SP round function). Our techniques are based largely on what they call the rebound attacks. As a result, our attacks work most effectively when the S-boxes have a "good" differential property (like the inverse function $x \mapsto x^{-1}$ in the finite field) and when the linear transformation has an "optimal" branch number (*i.e.*, a maximum distance separable matrix). We first describe known-key distinguishers on such Feistel block ciphers of up to 11 rounds, increasing significantly the number of rounds from previous work. We then apply our distinguishers to the Matyas-Meyer-Oseas and Miyaguchi-Preneel modes in which the Feistel ciphers are used, obtaining collision and half-collision attacks on these hash functions.

Keywords: known-key, block cipher, Feistel-SP, rebound attack, MDS, collision attack, hash function, MMO, Miyaguchi-Preneel.

1 Introduction

The security of block ciphers usually relies on the fact that the key value is kept secret in its encryption and decryption process. However, recently cryptographers have become interested in the security of block ciphers when the key value is known to the attackers. That is, in the traditional secret-key setting, it is the randomness and secrecy of a key that guarantees security of block ciphers. On the other hand, in the known-key setting, the value of the key is revealed to attackers; it becomes only the randomness of the key that retains (some) security of the cipher.

In practical applications, block ciphers are indeed used in such a way as their key values are known publicly. A typical example is the hash function constructed from block ciphers, including the Davies-Meyer (DM), Matyas-Meyer-Oseas (MMO) and Miyaguchi-Preneel hashing modes. In particular, the latter two modes make the known-key setting potentially relevant, because a known and essentially random (at least not under the attacker's full control) value is fed into the key input of the block cipher.

Recently, many papers have been published in the context of known-key attacks. In particular, there have been quite a few results on AES and Rijndael

A. Joux (Ed.): FSE 2011, LNCS 6733, pp. 397–415, 2011.

[13,19,17,12,24]. Among them is the work by Gilbert and Peyrin [12] of a known-key distinguisher on AES, which is one of the few attacks that can break 8 rounds of AES-128.[1]

On the other hand, there appear to be only a couple of results that analyzed the known-key security of the Feistel network [13,8]. Both pieces of work assume that a round function consists of a key XOR followed by a mixing function. In practice, the "mixing function" part is frequently realized by a combination of S-boxes (Substitution boxes) and a linear transformation.

In this paper, we show new attacks on the Feistel network. We consider the Feistel network which has SP round functions. That is, we assume that a round function consists of an XOR with a subkey, S-boxes, and then a linear matrix P. Our attacks work effectively up to 11 rounds, improving over the previous best 7 rounds.[2] We first present known-key distinguishers on such Feistel block ciphers and then translate them to collision and half-collision attacks on the MMO and Miyaguchi-Preneel hashing modes using these ciphers.

Our techniques are based on the so-called rebound attacks. Consequently, in our attacks we assume "attacker-unfriendly" properties of S-boxes and of P. Though not mandatory, these properties make our attacks most effective to work and simplest to describe. More specifically, we assume that the S-boxes have low maximum differential probabilities (like the multiplicative inverse function $x \mapsto x^{-1}$ in the finite field) and that the linear transformation P is an MDS (maximum distance separable) matrix. These are believed to be "good" choices of S-boxes and of P for designing a secure symmetric-key primitive.

Organization of the Paper. In Sect. 2 we review previous work, basic notions, and the rebound attacks. We present our new distinguishers on the Feistel network in Sect. 3 and apply them to the MMO and Miyaguchi-Preneel hashing modes in Sect. 4. In Sect. 5 we discuss the generality and applicability of our attack techniques. We conclude the paper in Sect. 6.

2 Preliminaries

2.1 Previous and Related Work

The concept of known-key distinguishers was introduced by Knudsen and Rijmen [13]. They showed known-key attacks on AES reduced to 7-rounds and on 7-round Feistel ciphers with a round function consisting of a round-key XOR followed by an arbitrary key-independent transformation. Their attack on AES(-128) leads to non-ideal behaviors of its MMO hashing mode. Their attack on Feistel ciphers has an even stronger impact, as a pair of blocks colliding in half of the output state can be found only with a complexity of two encryptions.

[1] Another attack on the 8-round AES-128 is [5], which is a "chosen-key" distinguisher.

[2] The 7-round attack [13] works for the Feistel network which has a round function consisting of an XOR with a subkey followed by any function f rather than SP.

Subsequently, Minier *et al.* [19] suggested a formalization of known-key attacks. They also presented known-key distinguishers on Rijndael having large blocks, which was further studied by [24].

As for AES, Mendel *et al.* [17] presented a new known-key distinguisher on 7-round AES-128 with a less attack complexity. Gilbert and Peyrin [12] attacked AES-128 and increased the number of rounds to eight.

The work [8] by Bouillaguet *et al.* presented new attacks on the Feistel network. This work is actually about analyzing certain hash functions (Lesamnta and SHAvite-3) but can be regarded as known-key attacks on two types of (generalized) Feistel ciphers. See also [11,20] for other attacks on SHAvite-3.

There is another type of attack in the open key scenario, namely, the "chosen-key" distinguishers. In the chosen-key setting, attackers can control, rather than know, the key value of block ciphers. For example, Biryukov *et al.* showed chosen-key distinguishers on the full-round AES-256 in the "related-key" scenario [4]. Biryukov and Nikolić showed chosen-key distinguishers on the 8-round AES-128 [5] and on several other block ciphers [6]. Lamberger *et al.* [14] presented a chosen-key distinguisher on the full Whirlpool compression function. See also [21] for other known-key and chosen-key distinguishers on several block ciphers.

2.2 Basic Notions

S-Boxes. An S-box

$$S : \{0,1\}^c \to \{0,1\}^c$$

is a substitution table, being most of the time a non-compressing fixed function. Popular choices are $c = 4, 8$. The main purpose of using S-boxes in block ciphers is to introduce non-linearity over the finite field $\mathbb{F}(2^c)$.

A typical example of an S-box is the multiplicative inverse function

$$S : x \mapsto x^{-1}$$

in the field $\mathbb{F}(2^c)$. This is a popular choice for an S-box in block ciphers, is believed to be a good choice for increasing the cipher's resistance to differential [2] and linear [16] cryptanalyses, and is indeed adopted by AES. With such an S-box the maximum differential and linear probabilities become 2^{-c+2}, being close to the best possible bounds [22,10].

Branch Numbers and MDS Matrices. Recall that the linear diffusion layer $P : \{0,1\}^n \to \{0,1\}^n$ can be written in the form of an $r \times r$ constant matrix over the field $\mathbb{F}(2^c)$, where c is the number of bits in a byte (*i.e.*, the size of an S-box) and r is the number of bytes (*i.e.*, the number of S-boxes) in the input to the round function, so that we have $n = r \cdot c$. Here we treat $\{0,1\}^n$ as $\mathbb{F}(2^c)^r$. For an input $X \in \{0,1\}^n = \mathbb{F}(2^c)^r$ we can write

$$P \cdot X = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1r} \\ p_{21} & p_{22} & \cdots & p_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ p_{r1} & p_{r2} & \cdots & p_{rr} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_r \end{pmatrix},$$

where each of p_{ij} and x_i is an element of the field $\mathbb{F}(2^c)$.

The *weight* $\mathtt{wt}(X)$ (over the field $\mathbb{F}(2^c)$) of a vector $X \in \mathbb{F}(2^c)^r$ is the number of non-zero components in X, so that we have $0 \leq \mathtt{wt}(X) \leq r$. The *branch number* $\mathtt{br}(P)$ of an $r \times r$ matrix P over the field $\mathbb{F}(2^c)$ is defined as

$$\mathtt{br}(P) := \min_X \{\mathtt{wt}(X) + \mathtt{wt}(P \cdot X)\},$$

where the minimum runs over all $X \in \mathbb{F}(2^c)^r$ such that $X \neq 0$. For block ciphers in the secret-key setting, it is desirable to use a linear transformation P having a high branch number in order for the cipher to be resistant to differential and linear cryptanalyses.

The Singleton bound [25] in coding theory implies that the highest branch number possible is $r + 1$. An $r \times r$ matrix P having such a branch number is called an *MDS* (*Maximum Distance Separable*) *matrix* and is frequently utilized in cryptography. For example, the block cipher AES [26] uses a 4×4 MDS matrix, whereas the hash function Whirlpool [1] uses an 8×8 MDS matrix.

Hashing Modes Using Block Ciphers. Matyas-Meyer-Oseas (MMO) and Miyaguchi-Preneel modes provide efficient ways to construct a compression function from a block cipher. They are among the 12 secure schemes [7] of PGV style [23].

Let E be a block cipher, and let E_K denote its encryption algorithm with a key K. The MMO compression function outputs H_i by computing

$$H_i = E_{H_{i-1}}(M_{i-1}) \oplus M_{i-1}$$

for a message block M_{i-1} and a previous chaining value H_{i-1}. Similarly, the Miyaguchi-Preneel mode computes H_i by

$$H_i = E_{H_{i-1}}(M_{i-1}) \oplus M_{i-1} \oplus H_{i-1},$$

given M_{i-1} and H_{i-1}.

In the above definitions, the Merkle-Damgård construction is implicitly assumed (by setting the initial chaining value to be a fixed constant IV and by letting the final output be the hash value). We keep it implicit, as all attacks presented in the current paper deal only with one-block input messages.

2.3 Rebound-Attack Technique

Here we briefly review a technique called rebound attacks, which was first introduced by Mendel *et al.* in analyzing AES-based hash functions [18]. Mendel *et al.* later applied it to a known-key attack on reduced-round AES [17], as already mentioned in Section 2.1. The attacks presented in the current paper mainly depend on this technique as well.

In rebound attacks, an attacker first builds a truncated differential path, fixing byte positions which have a difference. At this stage, the attacker is not interested

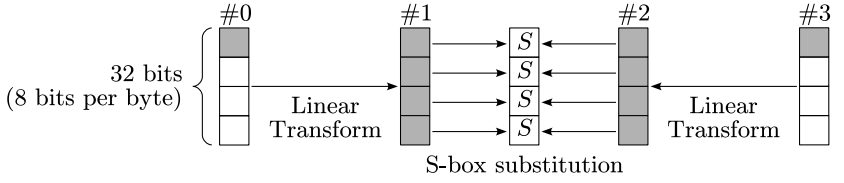

Fig. 1. Example of the inbound phase with 4-byte state (Gray bytes are active.)

in the exact differential value inside each byte. We call a byte position with a difference an *active byte*. Then the attacker tries to find efficiently a pair of values that follows the path.

The truncated differential path is divided into two phases: The *inbound phase* and the *outbound phase*. The inbound phase consists of a linear layer, a non-linear layer, and another linear layer in this order. The inbound phase corresponds to a low probability part of the truncated differential path. The attacker generates sufficiently many paired values that satisfy the truncated differential path of the inbound phase. This can be done with a small complexity on average. These paired values are called *starting points*. Then, the attacker computes the outbound phase with the generated starting points and checks whether or not any of the starting points satisfies the truncated differential path of the outbound phase. The attacker succeeds if he finds a starting point conforming to the truncated differential path of the outbound phase.

Let us explain the basic procedure of the inbound phase by using a 4-byte (1 byte = 8 bits) state with an 8-bit S-box as an example. In this example, the goal of an attacker is to find a pair of values (M, M') that satisfies the truncated differential path $\mathbf{1} \rightarrow \mathbf{4} \rightarrow \mathbf{1}$ (one active byte diverging to four active bytes and then converging to one active byte again) which is illustrated in Fig. 1.

A naive method to obtain such a pair is to generate many pairs with a 1-byte difference at state #0 and then to compute the corresponding difference at state #3. After trying 2^{24} pairs, the attacker should be able to find a desired pair.

The rebound attack can generate such pairs more efficiently. It generates approximately 2^8 pairs satisfying the differential path with a complexity of approximately 2^8—in other words, each pair is generated with a complexity of 1 on average. The detailed attack procedure is as follows:

0. Prepare the differential distribution table (DDT) for the 8-bit S-box.
1. For all 2^8 (more precisely, $2^8 - 1$) possible differences of state #0, compute the corresponding 4-byte differences of state #1 and store them in a table T. Note that we can determine differential propagation irrespective of the byte values owing to the linearity of computation.
2. Choose a difference of state #3 and compute the corresponding 4-byte difference of state #2. For each 4-byte difference in T, check whether or not the computed 4-byte difference of state #2 can be output through the S-boxes by looking up the DDT. If the differences match, output such paired values.

Table 1. Our attack results for practical parameters of Feistel network

Block	Word	Byte	#Bytes	$2c \stackrel{?}{\geq} r$	Cipher	MMO Hash	
N	n	c	r		Known-key	Collision	Half-collision
128	64	8	8	✓	11R	9R	11R
		4	16		11R	9R	11R
64	32	8	4	✓	9R	7R	9R[†]
		4	8	✓	11R	9R	11R

[†] This 9R attack generates $(N-c)$-bit collisions rather than half-collisions.

In the case of the AES S-box (*i.e.*, the multiplicative inverse function in the finite field), for a pair of randomly determined input difference ΔS_{in} and output difference ΔS_{out}, an equation $S(x) \oplus S(x \oplus \Delta S_{\text{in}}) = \Delta S_{\text{out}}$ has approximately one solution with a probability of approximately 2^{-1}. If we find a solution x, then we will automatically obtain *two* paired values $(x, x \oplus \Delta S_{\text{in}})$ and $(x \oplus \Delta S_{\text{in}}, x)$ that satisfy the differential propagation through the S-box. In this example there are four S-boxes between state #1 and state #2. Therefore, if we have 2^4 pairs of Δ#1 and Δ#2, then one of these pairs can be expected to have a solution for each of the four S-boxes. Hence we obtain 2^4 paired values that satisfy the truncated differential path of the inbound phase.

Consequently, for a difference of state #3 in the above procedure, there are $2^8 \cdot 2^{-4} = 2^4$ differences in T that have approximately one solution for each of the four S-boxes. So we obtain $2^4 \cdot 2^4 = 2^8$ paired values that satisfy the truncated differential path (starting points). In other words, we can obtain one starting point with a complexity of 1 on average (Note that the average complexity to obtain one starting point is always 1 for any S-box).

We also need to count the number of starting points obtained. The above procedure can be iterated for all 2^8 differences of state #3 in Step 2. As a result, we obtain $2^8 \cdot 2^8 = 2^{16}$ starting points at maximum.

3 New Known-Key Attacks on Feistel Ciphers

Now we present known-key attacks on Feistel ciphers of the SP structure. Table 1 summarizes our attack results (R stands for rounds), where we use the following variables (which are fixed henceforward):

N: The block length of the cipher (in bits),
n: The word size in bits, equal to the size of the input and output of the round function, so that $n = N/2$,
c: The byte size in bits (In this paper the byte size is not fixed), equal to the size of an S-box,
r: The number of bytes in a word, so that $r = n/c$.

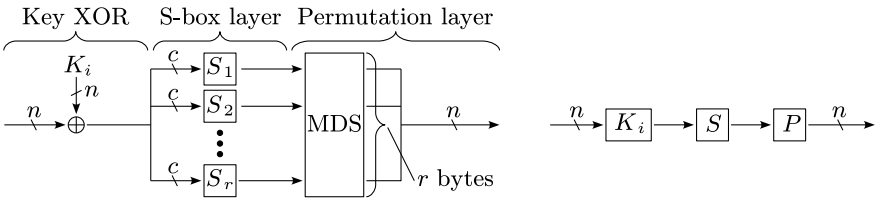

Fig. 2. Left: Detailed description of the SP round function **Right:** Simplified one

Recall that many of the block ciphers designed in practice are equipped with 128-bit or 64-bit blocks and use 8-bit or 4-bit S-boxes. Table 1 includes these practical parameters.

For the sake of simplicity, we assume $2c \geq r$ at the moment. Even with this restriction, our attacks cover most of the block ciphers that are used today: 128-bit block with 8-bit S-box, 64-bit with 8-bit, and 64-bit with 4-bit.

For the sake of completeness, in Sect. 5.3 we shall explain a simple extension of our attacks to treat the case $2c < r$. This case includes the remaining parameter of 128-bit block with 4-bit S-box, which is probably most unlikely to be used in practice among the 4 parameters listed in Table 1. This is because this case requires an MDS matrix acting on 16-byte state, where today such a large MDS matrix is still considered to be too costly to be implemented in systems.

Description of the SP Round Function. Before we proceed to describing our attacks, we specify the SP round function of the Feistel network to be analyzed. The round function is depicted in Fig. 2, consisting of the following three operations:

Key XOR: This layer computes the XOR of a round-function input and a round key K_i.

S-box layer: This layer substitutes each byte value by using one or several S-boxes; the S-boxes $S_1, S_2, \ldots, S_r$ may differ from each other. We assume that all S-boxes are designed to be resistant to differential and linear cryptanalyses, like the ones used in AES [9,26]. Hence, given a pair of randomly chosen input and output differences, there exist paired values following the given input/output differences with a probability of approximately 2^{-1}. If exist, then the number of such paired values is approximately two.

Permutation layer: This layer mixes values by multiplying the word value and an $r \times r$ matrix P over $\mathbb{F}(2^c)$ together. We make the assumption that P is an MDS matrix, so that the total number of active bytes in the input and output of P is always greater than or equal to $r + 1$, as long as there is at least one active byte.

The assumptions that we make about S and P are not quite mandatory. In Sect. 5 we discuss the feasibility of our attacks when S or P does not have these properties.

Table 2. Attack strategy

Input:[†]

- Subkeys generated from a random master key via a key schedule function
- S-boxes secure against differential and linear cryptanalyses
- A linear transformation with a branch number $r + 1$ (*i.e.*, an MDS)

Procedure:

1. Prepare DDTs for all S-boxes in use.
2. Find a pair of values that satisfies the truncated differential path of the inbound phase.
3. Verify that the pair found in Step 2 also follows the truncated differential path of the outbound phase.
4. Confirm that the above procedure requires a less complexity than finding such a pair for a random permutation.

[†] In Sect. 5 we discuss the case where S or P does not satisfy these conditions.

Overview of Our Attacks. Our attack procedure is summarized in Table 2. We shall explain the details of Steps 2 and 3 in the following sections; in Sect. 3.1 we first explain a basic version of our attacks which is effective up to 9 rounds, and then extend it to 11 rounds in Sect. 3.2.

Hereafter, we fix a system of notations as follows:

X^j: The j-th byte of a word X, where $1 \leq j \leq r$ and the size of X^j is c bits,
0: A word where all bytes are non-active,
1: A word where only one byte of the predetermined (j-th) position is active,
F: A word where all bytes are active.

Our attacks amount to finding a pair of values whose input difference is of the form $\big(P(\mathbf{1}), \mathbf{F}\big)$ and whose output difference is also $\big(P(\mathbf{1}), \mathbf{F}\big)$. The point is that our attacks can find such a pair for the Feistel ciphers more efficiently than one could for a random permutation.

So let us mention in advance the time and memory complexities of the above procedure. The cost of Step 1 is $r \cdot 2^{2c}$ in time and $r \cdot 2^{2c}$ in memory for both 9R and 11R attacks. The cost of Step 2 is $r \cdot 2^c$ in time and $r \cdot 2^2$ in memory for 9R attacks and $r \cdot 2^{2c}$ in time and $r \cdot 2^{2c}$ in memory for 11R attacks. Step 3 can be done at a cost of 1. Overall, our attacks can find a pair of values following the truncated differential path with a complexity of $r \cdot 2^{2c}$ in time and $r \cdot 2^{2c}$ in memory.

This means that our known-key attacks work effectively, because for a random permutation such a pair cannot be found with that complexity. Namely, let us consider the complexity to find a pair of values that has the differential form of $\big(P(\mathbf{1}), \mathbf{F}\big)$ for both of the input and output states in a random permutation. Attackers have an access to both encryption and decryption oracles.

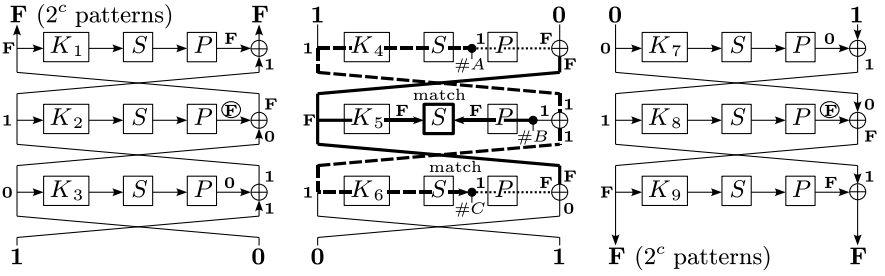

Fig. 3. Basic 9R attack

For such attackers, this problem is regarded as finding a 2^{n-c}-bit collision. Because enough freedom degrees are available to mount the birthday attack, this requires a complexity of $2^{(n-c)/2}$.

Note that we make use of the property of MDS and DDTs generated at Step 1 in performing Step 2 (the inbound phase) efficiently. Also, the outbound phase works for any S-box and any linear transformation.

3.1 Basic 9R Attack on the Feistel Ciphers

Entire Truncated Differential Path. Recall that in rebound attacks, an attacker needs to construct a differential path and divide it into inbound and outbound phases. Here the truncated differential path that we use is

$$(\mathbf{F},\mathbf{F}) \xrightarrow{1^{\text{st}}\text{R}} (\mathbf{1},\mathbf{F}) \xrightarrow{2^{\text{nd}}\text{R}} (\mathbf{0},\mathbf{1}) \xrightarrow{3^{\text{rd}}\text{R}} (\mathbf{1},\mathbf{0})$$
$$(\mathbf{1},\mathbf{0}) \xrightarrow{4^{\text{th}}\text{R}} (\mathbf{F},\mathbf{1}) \xrightarrow{5^{\text{th}}\text{R}} (\mathbf{1},\mathbf{F}) \xrightarrow{6^{\text{th}}\text{R}} (\mathbf{0},\mathbf{1})$$
$$(\mathbf{0},\mathbf{1}) \xrightarrow{7^{\text{th}}\text{R}} (\mathbf{1},\mathbf{0}) \xrightarrow{8^{\text{th}}\text{R}} (\mathbf{F},\mathbf{1}) \xrightarrow{9^{\text{th}}\text{R}} (\mathbf{F},\mathbf{F}),$$

which is shown in Fig. 3.

In the basic 9R attack, we use a 3-round differential path for the inbound phase. The difference propagates from $(\mathbf{1},\mathbf{0})$ to $(\mathbf{0},\mathbf{1})$ through the three rounds (4^{th}R – 6^{th}R).

The outbound phase consists of three rounds in backward direction (3^{rd}R – 1^{st}R) and three rounds in forward direction (7^{th}R – 9^{th}R), in total six rounds. In both directions, the differences propagate to $(\mathbf{F},\mathbf{F})$. Here the patterns in $\mathbf{F}$ are limited in a certain way (Not all patterns are formed).

Given any paired value satisfying the truncated differential path of the inbound phase, the truncated differential path of the outbound phase is satisfied with a probability of 1. Hence, we need only one starting point from the inbound phase.

Three-Round Inbound Phase. In this phase, our goal is to find a pair of values whose difference will propagate as $(\mathbf{1},\mathbf{0}) \xrightarrow{4^{\text{th}}\text{R}} (\mathbf{F},\mathbf{1}) \xrightarrow{5^{\text{th}}\text{R}} (\mathbf{1},\mathbf{F}) \xrightarrow{6^{\text{th}}\text{R}} (\mathbf{0},\mathbf{1})$. We use the differential path depicted in the middle of Fig. 3.

We start by choosing differences of the form **1** in words $\#A$ and $\#B$ in Fig. 3. We propagate these differences through the linear operations with actual word values undetermined. We then search for a matched set of differences at the S-box operation in the 5th round and finally find word values that follow the desired differential path of the 3R inbound phase. More specifically, our attack procedure is as follows:

0. Prepare DDTs for all S-boxes. Choose a byte-position j $(1 \le j \le r)$ in a word to be activated in the differential form of **1**.
1. For all 2^c possible differences in word $\#A$, compute the corresponding full-byte differences after applying the permutation layer and store the results in a table T. Note that the difference in word $\#C$ must be the same as that in $\#A$, which of course takes the differential form of **1**.
2. For each of the 2^c possible differences in word $\#B$, compute the corresponding full-byte difference after applying the inverse permutation. For S-boxes in the 5th round, check whether or not we can match the full-byte difference evolved from $\Delta\#B$ with any of the 2^c differences stored in T. This can be done by looking up the DDTs.
3. Assuming that we find such a matched set of differences, choose one of them and fix word values in accordance with the chosen differences. Now the difference in word $\#A$ is fixed. Also, the word values on the bold lines drawn in Fig. 3 are all fixed.
4. For each of the 2^c possible values of $\#A^j$, compute the corresponding difference in $\#C^j$ and check whether or not the difference becomes equal to $\Delta\#A^j$. This computation is denoted by the broken lines in Fig 3. Namely, compute and check the following:

$$\Delta\Big[S_j\Big(S_j^{-1}(\#A^j) \oplus K_4^j \oplus \#B^j \oplus K_6^j\Big)\Big] \stackrel{?}{=} \Delta\#A^j. \tag{1}$$

5. A solution for (1) is the value we want for the active byte. Make an arbitrary choice for values of the remaining non-active bytes. Now all the values within the inbound phase are determined. In other words, we obtain a starting point.

Let us estimate the time and memory complexities necessary for each of the above steps. We also verify that the success probability of the inbound phase is sufficiently high.

- In Step 0 we need 2^{2c} computations and 2^{2c} memory to prepare the DDT for each c-bit S-box. When S-boxes are all different, Step 0 requires $r \cdot 2^{2c}$ computations and $r \cdot 2^{2c}$ memory.
- Step 1 requires 2^c computations and 2^c memory.
- In Step 2, with a complexity of 2^c, we can check the match of 2^{2c} pairs at maximum. Because each match succeeds with a probability of 2^{-r}, we can expect to find one or more matched pairs as long as $2c \ge r$.

- Step 3 requires negligible computations and memory.
- Step 4 requires 2^c 3-round computations and negligible memory. Note that this step needs to examine only one byte, and thus the actual complexity is lower than 2^c 3-round computations.
- In Step 5 we expect to find a solution for (1), because we can carry out a match check as many times as the number of possible patterns of $\Delta\#C^j$, which is 2^c.
- Therefore, the total complexity for Step 0 through Step 5 is $r \cdot 2^{2c}$ computations and $r \cdot 2^{2c}$ memory.

Outbound Phase. We explain the outbound phase for the last 3 rounds, which is shown in the right part of Fig. 3. As a result of the inbound phase, we obtain the difference $(\mathbf{0}, \mathbf{1})$ as an input to the 7th round. This propagates as $(\mathbf{0}, \mathbf{1}) \xrightarrow{7^{\text{th}}\text{R}} (\mathbf{1}, \mathbf{0}) \xrightarrow{8^{\text{th}}\text{R}} (\mathbf{F}, \mathbf{1}) \xrightarrow{9^{\text{th}}\text{R}} (\mathbf{F}, \mathbf{F})$ with a probability of 1. Because the input to the 8th round has a difference of the form $\mathbf{1}$, the corresponding output differences are of the form $P(\mathbf{1})$, which is emphasized by a circled $\mathbf{F}$ in Fig. 3. Hence, the number of possible differences in the output word is limited to 2^c. Therefore, although all bytes are active in ciphertexts (Recall that P is an MDS), only $2^c \cdot 2^n = 2^{c+n}$ differential patterns are formed in the ciphertexts.

The same applies to the differential propagation in backward direction for the first 3 rounds. The differential path reaches $(\mathbf{F}, \mathbf{F})$ after the 3 rounds with a probability of 1. Since differential patterns in the left half of the plaintext are limited to 2^c variations, only 2^{c+n} differential patterns appear in the plaintext pairs.

Comparison with a Random Permutation. The inbound phase finds a starting point with a time complexity of $r \cdot 2^{2c}$ using $r \cdot 2^{2c}$ memory. Given any solution of the inbound phase, the outbound phase, with a probability of 1, generates a pair of values that has a differential form of $\big(P(\mathbf{1}), \mathbf{F}\big)$ for both plaintext and ciphertext.

By comparing the above complexity with the generic birthday bound, we can derive a condition of parameters with which our attacks work effectively. That is, an inequality $r \cdot 2^{2c} < 2^{(n-c)/2}$ gives us a condition

$$c < \frac{1}{5}(n - 2 \cdot \log_2 r), \tag{2}$$

which needs to be satisfied in order for the above attack to be successful.

Let us consider parameters in practical use with which the condition is satisfied. We see that the condition is met by 128-bit block ciphers having 4- or 8-bit S-boxes and by 64-bit block ciphers having 4-bit S-boxes. See Table 1. Unfortunately, the condition is not met by 64-bit block ciphers having 8-bit S-boxes. This case will be treated separately in Sect. 3.3.

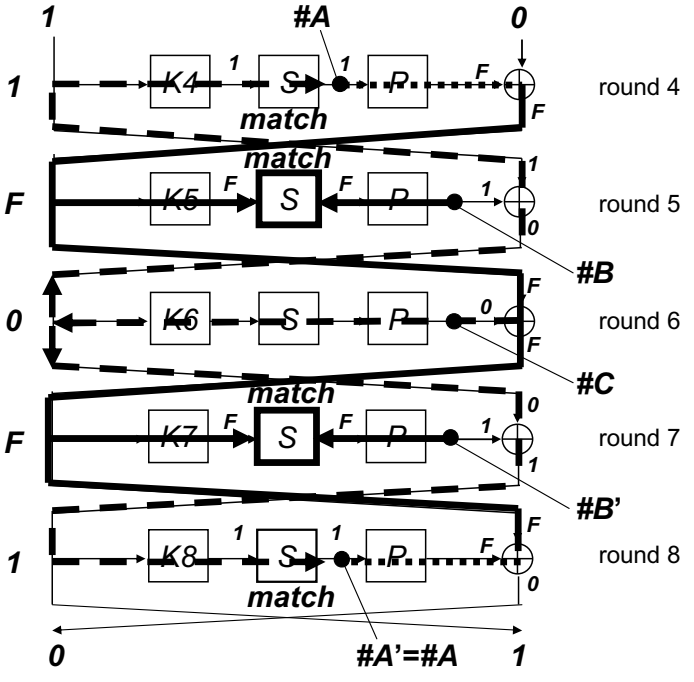

Fig. 4. 5-round inbound phase

3.2 Extended 11R Attack on the Feistel Ciphers

We extend the previous 9R attack to 11 rounds. We change the number of rounds in the inbound phase from three to five. The outbound phase is exactly the same as the basic 9R attack.

Five-Round Inbound Phase. The differential path of the new 5R inbound phase is

$$(\mathbf{1},\mathbf{0}) \xrightarrow{4^{\text{th}}\text{R}} (\mathbf{F},1) \xrightarrow{5^{\text{th}}\text{R}} (\mathbf{0},\mathbf{F}) \xrightarrow{6^{\text{th}}\text{R}} (\mathbf{F},\mathbf{0}) \xrightarrow{7^{\text{th}}\text{R}} (\mathbf{1},\mathbf{F}) \xrightarrow{8^{\text{th}}\text{R}} (\mathbf{0},\mathbf{1}), \quad (3)$$

which is depicted in Fig. 4. We start with the same difference of the form **1** in words $\#A$ and $\#A'$ and then try to find matched sets of differences at the S-boxes in the 5th and 7th rounds, with the differences being expanded from words $\#B$ and $\#B'$, respectively. After finding matched sets, we determine the value of $\#C$ and compute values indicated by broken lines. We finally check the consistency of differences in words $\#A$ and that in $\#A'$. A detailed attack procedure is as follows:

0. Prepare DDTs for all S-boxes. Choose an active-byte position j for differential **1**.

1. For all 2^c differences of the active byte in word $\#A$, compute the corresponding full-byte differences after applying the (forward) permutation layer and store them in a table T. Set the difference in word $\#A'$ to be the same as that in $\#A$. This guarantees that the difference in word $\#C$ is $\mathbf{0}$.
2. For each of the 2^c differences in word $\#B$, compute the corresponding full-byte difference after applying the inverse permutation. For each difference stored in T, check whether we can match it with the above difference by looking up the DDTs. If a matched set of differences for $\#A$ and $\#B$ is found, we can instantly obtain a matched set for $\#A'$ and $\#B'$ by setting $\#B' = \#B$.
3. Now that a matched set of differences is found, we can fix word values and compute the value of word $\#C$. Here the values drawn in broken lines in Fig. 4 are fixed.
   (a) Check whether or not the computed differences in $\#A$ and $\#A'$ in Step 3 and the chosen difference in $\#A = \#A'$ at Step 1 are consistent. Namely, compute and check the following:

$$\Delta\left[S_j\left(S_j^{-1}\left(\left(P^{-1}(\#C)\right)^j\right) \oplus K_6^j \oplus \#B^j \oplus K_4^j\right)\right] \stackrel{?}{=} \Delta\#A^j, \tag{4}$$

$$\Delta\left[S_j\left(S_j^{-1}\left(\left(P^{-1}(\#C)\right)^j\right) \oplus K_6^j \oplus \#B'^j \oplus K_8^j\right)\right] \stackrel{?}{=} \Delta\#A'^j. \tag{5}$$

   (b) If we find a solution for the above two equations, then it means that we have found a starting point of the inbound phase.

Let us evaluate the time and memory complexities necessary for the above procedure:

- Step 0 requires $r \cdot 2^{2c}$ computations and $r \cdot 2^{2c}$ memory to prepare r-many DDTs.
- Step 1 requires 2^c computations and 2^c memory.
- In Step 2, we are expected to find 2^{2c-r} matches between $\#A$ and $\#B$ after trying 2^c differences in $\#B$. Since 2^r solutions are obtained from a match, we obtain 2^{2c} solutions for $\#A$ and $\#B$.
- Similarly, we obtain 2^{2c} solutions for $\#A'$ and $\#B'$.
- Step 3 requires just one computation for each solution and can be iterated $2^{2c} \cdot 2^{2c} = 2^{4c}$ times at maximum.
- In Step 3a, each match succeeds with a probability of 2^{-c}. Therefore, by iterating Step 3a 2^{2c} times, we will find a match.

To sum up, we can find a starting point for the 5R inbound phase with a complexity of $r \cdot 2^{2c}$ time and $r \cdot 2^{2c}$ memory.

Outbound Phase, Ideal Case, and Attackable Parameters. The outbound phase is exactly the same as the one used in the basic 9R attack. Any

pair of values satisfying the differential path of the inbound phase, with a probability of 1, generates a pair of values that has a differential form of $\big(P(\mathbf{1}), \mathbf{F}\big)$ for both of the input and output states.

The attack complexity for the ideal case is also the same as before, which requires $2^{(n-c)/2}$. Therefore, (2) is also the condition that needs to be satisfied for our 11R attack to be applicable.

Since the condition is the same as the one for the basic 9R attack, the same parameters apply: $(N, c) = (128, 8)$, $(128, 4)$ and $(64, 4)$. See Table 1.

3.3 "Shrunken" 9R Attack: Case $(N, c) = (64, 8)$

Our attacks in previous sections cannot be applied to 64-bit block ciphers having 8-bit S-boxes. However, by reducing the number of outbound rounds of the 11R attack, we can attack up to 9 rounds of such ciphers. Namely, we let the differential path consist of

- 2-round backward outbound phase,
- 5-round inbound phase, and
- 2-round forward outbound phase.

Now the differences in plaintext and ciphertext are both $\big(\mathbf{1}, P(\mathbf{1})\big)$. The cost for finding a pair of values satisfying this differential form with a random permutation is equal to the one for finding a collision of $2(n-c)$ bits, which is $2^{2(n-c)/2} = 2^{n-c}$. Hence, the condition on the parameters becomes $r \cdot 2^{2c} < 2^{n-c}$, which is converted as

$$c < \frac{1}{3}(n - \log_2 r),$$

and thus, 64-bit block ciphers with 8-bit S-boxes can be attacked up to 9 rounds.

4 Application to MMO and Miyaguchi-Preneel Modes

We apply the known-key distinguishers to attacking the MMO and Miyaguchi-Preneel hashing modes using these Feistel ciphers. The attacks we present in this section can generate either full N-bit collisions or half n-bit collisions depending on the parameters.

We consider this aspect of our known-key attacks quite important, as it shows that the attacks endangers real-world security of these hash functions. Sometimes known-key attacks tend to become fairly complex. People may wonder what practical implications such results have, which is certainly not the case for our attacks.

Here we only describe our attacks on the MMO mode, but all the attacks can be trivially extended to the Miyaguchi-Preneel mode. This is because the key (randomly given constant) addition to the hash output state used by the Miyaguchi-Preneel mode does not make any impact upon the output value differences.

When we apply the distinguishers to collision attacks on the MMO mode, we iterate Step 2 of Table 2 in order to generate more paired values satisfying the

inbound phase. We perform Step 3 of Table 2 and then check whether ciphertext differences cancel out plaintext differences. If they do, then we obtain a collision of the compression function (due to the feedforward used in the MMO mode).

We also present half-collision attacks. Generally, half-state collisions possibly become full collisions if more than half of the bits are truncated; systems in practice often truncate hash values to their desired length (*e.g.*, SHA-224 and SHA-384). Due to the nature of the Feistel network, in our half-collision attacks it is important which (left or right) half is chopped—our attacks work effectively on odd numbers (9 and 11) of rounds, whereas Feistel ciphers in practice usually have even number of rounds.

4.1 Eleven-Round Half-Collision Attack

We start with the parameters where the extended 11R attack applies. Recall that in our 11-round attack the differential forms of input and of output are both $\big(P(\mathbf{1}), \mathbf{F}\big)$. Therefore, a feed-forward operation (due to the MMO mode) makes the difference in the left half of the output state zero (cancellation) with a probability of 2^{-c}, which yields a half-state collision. Recall that our 11-round attack can generate up to x starting points with a complexity of $x \cdot r \cdot 2^{2c}$ and with $r \cdot 2^{2c}$ memory, where $x \leq 2^{2c}$. So by choosing $x := 2^c$, we can generate a half-state collision with a total complexity of $r \cdot 2^{3c}$. Note that it requires a complexity of $2^{N/4}$ to find a left-half-state collision of an N-bit ideal hash function due to the birthday attack. We can verify that for all the parameters in consideration the complexity $r \cdot 2^{3c}$ is faster than the birthday bound $2^{N/4}$.

4.2 Nine-Round Full-Collision Attack

Our attack can generate 9-round full collisions by using the outbound phase reduced to 2-rounds and using the 5-round inbound phase as described in Section 3.3 (but here we are treating the cases $(N, c) = (128, 8)$, $(128, 4)$ and $(64, 4)$, not the case $(N, c) = (64, 8)$). Now the differential forms of input and output are both $\big(\mathbf{1}, (P(\mathbf{1})\big)$, and after a feed-forward operation, the xored values cancel each other with a probability of 2^{-2c}. Hence, setting $x := 2^{2c}$ allows us to generate a full collision. Of course to find a full collision of an N-bit ideal hash function should require a $2^{N/2}$ complexity, so our attack beats this birthday bound for all the parameters in consideration.

One may doubt that there is an adequate degree of freedom available. However, we can vary the degree of freedom as follows. In Step 0 of the attack procedure described in Section 3.2, we choose an active-byte position for $\mathbf{1}$. Because this can be chosen from r options, the degree of freedom becomes r times, and this makes the attack success probability almost 1.

4.3 Nine-Round Near-Collision Attack: Case $(N, c) = (64, 8)$

Now we discuss the remaining case of 64-bit blocks with 8-bit S-boxes. The shrunken 9-round attack can generate up to x starting points with a complexity

of $x \cdot r \cdot 2^{2c}$ and with $r \cdot 2^{2c}$ memory, where $x \leq 2^{2c}$. The differential forms of input and output are both $\big(\mathbf{1}, (P(\mathbf{1}))\big)$. Hence, by choosing $x := 2^{2c}$ and by spending a running time of $r \cdot 2^{4c}$, a full collision could be generated. However, for this specific parameter, the cost of finding a collision for a random permutation is 2^{4c}, which is obviously faster than $r \cdot 2^{4c}$.

Therefore, instead of generating a full collision, we consider only cancelling the differences of the right half state. Such a cancellation occurs with a probability of 2^{-c}, and if this occurs, then we immediately obtain a collision of $(2r-1) = 7$ bytes out of 8 bytes because the left half state only has the difference in one byte. Thus we can generate 7-byte collision with a complexity of $r \cdot 2^{3c}$, which is faster than the complexity for a random function, namely $2^{3.5c}$.

4.4 Seven-Round Full-Collision Attack: Case $(N, c) = (64, 8)$

If one wants to generate a full collision with the parameter $(N, c) = (64, 8)$, then one would need to use the previous 3R inbound phase and two sets of the 2R outbound phases. This can generate a full collision with a complexity of $r \cdot 2^{3c}$, which is faster than the birthday bound 2^{4c}.

5 Generality of Our Known-Key Distinguishers

So far our known-key distinguishers make the assumptions that a) S-boxes have balanced differential probabilities, b) the linear transformation has the branch number of $r + 1$, and c) the inequality $2c \geq r$ holds. In this section, we discuss possibilities of handling the situations where these conditions are not met.

5.1 When S-Boxes Are Biased

We have made the assumption that the S-boxes have the least maximum differential probabilities like the inverse function $x \mapsto x^{-1}$, so that we can estimate the matching probability roughly at $1/2$. This is not an essential requirement, and in fact S-boxes can be "suboptimal" in order for our attacks to work; one just needs more refined estimate of success probabilities in that case.

It is true that our attacks become infeasible when S-boxes have "very" biased differential distributions. However, block ciphers having such S-boxes are usually vulnerable to differential cryptanalysis, and the attacks should let us construct other types of distinguishers any way.

5.2 When P Is Not an MDS Matrix

The attacks described in Section 3 are based on the assumption that an MDS matrix is used in the P-layer. Again, this is not an absolute requirement. First note that the MDS property is used only in the inbound phase; it is completely irrelevant to the outbound phase. For example, in the inbound phase of the 9-round attack described in Section 3.1, we have used the fact that the branch number is $r + 1$ for satisfying the following three conditions:

1. Active-byte positions in $\Delta\#A$ and $\Delta\#B$ must be identical and should be minimized as much as possible,
2. Active-byte positions in $P(\Delta\#A)$ and $P^{-1}(\Delta\#B)$ must be identical, and
3. The degree of freedom for varying the differences in $\Delta\#A$ and in $\Delta\#B$ must be sufficient to find a match over the substitution (S-box) layer.

We have used the fact that the branch number is $r+1$ to simplify the description of our attacks and to minimize the attack complexity.

However, even if the branch number is smaller than $r+1$, there is a fair chance that the attack would work. If P is not an MDS matrix, then we search for byte positions that satisfy the above three conditions. This can be done in the pre-computation stage, because properties of a linear transformation P can be analyzed independently of a key value (Recall that P is a constant matrix).

5.3 When $2c < r$

The attacks described in Section 3 do not work if $2c < r$, because the number of pairs is insufficient to find a match of differences over the S-box layer (with a probability about 1). This problem can be solved by increasing the number of active bytes in a word. The essence of this generalization is to activate $d \geq 1$ bytes in a word so that $2cd \geq r$. The minimum value of such a d is 1 for $(c, r) = (8, 8), (8, 4), (4, 8)$ and 2 for $(c, r) = (4, 16)$. Hence, by activating two bytes instead of using the differential form **1**, we can attack, for example, Feistel ciphers having a 128-bit block size and 4-bit S-boxes.

With this generalization the total complexity becomes $r \cdot 2^{2cd}$ computations plus $r \cdot 2^{cd}$ memory for both 9-round and 11-round attacks. So for example, when $N = 128$, $c = 4$ and $d = 2$, these figures beat the birthday bound $2^{(n-cd)/2}$ for an ideal permutation.

Here we need to be careful about the fact that activating more than one byte may yield non-active bytes in the output of P or of P^{-1}. If that occurs, then most likely the match over S-boxes would fail (similarly to the case when P is not an MDS matrix). If we activate more than one byte, then such a case is inevitable even if P is an MDS matrix. However, the probability of output with non-active byte should not be so high, and we presume that a match can be found without a considerable increase in complexity even when we activate two bytes in a word.

6 Conclusion

We have presented new known-key distinguishers on block ciphers using the Feistel network whose round function consists of a key XOR, byte-oriented S-boxes, and an MDS matrix. We have considered most of the parameters used in practice and shown that "weak" parameters include 128-bit block with 8-bit S-box, where we can attack the cipher up to 11 rounds.

Our distinguishers can be extended to the MMO and Miyaguchi-Preneel hashing modes. With the weak parameters attackers can find full collisions up to 9

rounds and half-state collisions up to 11 rounds faster than the birthday bounds. To the best of our knowledge, as a generic attack on Feistel-SP ciphers, our attacks significantly increase the number of analyzable rounds from previous results.

References

1. Barreto, P.S.L.M., Rijmen, V.: The WHIRLPOOL hashing function. Submission to NESSIE (2003)
2. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 2–21. Springer, Heidelberg (1991)
3. Biryukov, A., Khovratovich, D.: Related-key cryptanalysis of the full AES-192 and AES-256. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 1–18. Springer, Heidelberg (2009)
4. Biryukov, A., Khovratovich, D., Nikolić, I.: Distinguisher and related-key attack on the full AES-256. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 231–249. Springer, Heidelberg (2009)
5. Biryukov, A., Nikolić, I.: A New Security Analysis of AES-128. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677. Springer, Heidelberg (2009), http://rump2009.cr.yp.to/
6. Biryukov, A., Nikolić, I.: Automatic search for related-key differential characteristics in byte-oriented block ciphers: Application to AES, camellia, khazad and others. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 322–344. Springer, Heidelberg (2010)
7. Black, J.A., Rogaway, P., Shrimpton, T.: Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions from PGV. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 320–335. Springer, Heidelberg (2002)
8. Bouillaguet, C., Dunkelman, O., Leurent, G., Fouque, P.A.: Attacks on hash functions based on generalized Feistel—application to reduced-round Lesamnta and SHAvite-3_512. In: Biryukov, A., Gong, G., Stinson, D. (eds.) SAC 2010. LNCS, vol. 6544, pp. 18–35. Springer, Heidelberg (2011)
9. Daemen, J., Rijmen, V.: AES Proposal: Rijndael. Submission to NIST (1998)
10. Dillon, J.F.: APN polynomials: An update. In: Fq9 Conference (2009)
11. 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 2010. LNCS, vol. 6055, pp. 419–436. Springer, Heidelberg (2010)
12. Gilbert, H., Peyrin, T.: Super-sbox cryptanalysis: Improved attacks for AES-like permutations. In: Hong, S., Iwata, T. (eds.) FSE 2010. LNCS, vol. 6147, pp. 365–383. Springer, Heidelberg (2010)
13. Knudsen, L.R., Rijmen, V.: Known-key distinguishers for some block ciphers. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 315–324. Springer, Heidelberg (2007)
14. Lamberger, M., Mendel, F., Rechberger, C., Rijmen, V., Schläffer, M.: Rebound distinguishers: Results on the full whirlpool compression function. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 126–143. Springer, Heidelberg (2009)
15. Lu, J., Dunkelman, O., Keller, N., Kim, J.-S.: New impossible differential attacks on AES. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 279–293. Springer, Heidelberg (2008)

16. Matsui, M., Yamagishi, A.: A new method for known plaintext attack of FEAL cipher. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 81–91. Springer, Heidelberg (1993)
17. Mendel, F., Peyrin, T., Rechberger, C., Schläffer, M.: Improved cryptanalysis of the reduced Grøstl compression function, ECHO permutation and AES block cipher. In: Jacobson Jr., M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 16–35. Springer, Heidelberg (2009)
18. Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: The rebound attack: Cryptanalysis of reduced whirlpool and `grøstl`. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 260–276. Springer, Heidelberg (2009)
19. Minier, M., Phan, R.C.-W., Pousse, B.: Distinguishers for ciphers and known key attack against rijndael with large blocks size. In: Preneel, B. (ed.) AFRICACRYPT 2009. LNCS, vol. 5580, pp. 60–76. Springer, Heidelberg (2009)
20. Minier, M., Naya-Plasencia, M., Peyrin, T.: Analysis of reduced-SHAvite-3-256 v2. In: Joux, A. (ed.) FSE 2011 Preproceedings. LNCS, vol. 6733, pp. 68–87 (2011)
21. Nikolić, I., Pieprzyk, J., Sokołowski, P., Steinfeld, R.: Known and chosen key differential distinguishers for block ciphers. In: Rhee, K., Nyang, D. (eds.) Preproceedings of ICISC 2010, 1A-3 (2010)
22. Nyberg, K.: Differentially uniform mappings for cryptography. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 55–64. Springer, Heidelberg (1994)
23. Preneel, B., Govaerts, R., Vandewalle, J.: Hash functions based on block ciphers: A synthetic approach. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 368–378. Springer, Heidelberg (1994)
24. Sasaki, Y.: Known-key attacks on Rijndael with large blocks and strengthening ShiftRow parameter. In: Echizen, I., Kunihiro, N., Sasaki, R. (eds.) IWSEC 2010. LNCS, vol. 6434, pp. 301–315. Springer, Heidelberg (2010)
25. Singleton, R.C.: Maximum distance q-nary codes. IEEE Trans. Inf. Theory 10, 116–118 (1964)
26. U.S. Department of Commerce, National Institute of Standards and Technology: Specification for the ADVANCED ENCRYPTION STANDARD (AES) (Federal Information Processing Standards Publication 197) (2001)