Cryptanalysis of Feistel Networks with Secret Round Functions *

Alex Biryukov[1], Gaëtan Leurent[2], and Léo Perrin[3]

[1] alex.biryukov@uni.lu, University of Luxembourg
[2] gaetan.leurent@inria.fr, Inria, France
[3] leo.perrin@uni.lu, SnT,University of Luxembourg

Abstract. Generic distinguishers against Feistel Network with up to 5 rounds exist in the regular setting and up to 6 rounds in a multi-key setting. We present new cryptanalyses against Feistel Networks with 5, 6 and 7 rounds which are not simply distinguishers but actually recover completely the unknown Feistel functions.
When an exclusive-or is used to combine the output of the round function with the other branch, we use the so-called *yoyo game* which we improved using a heuristic based on particular cycle structures. The complexity of a complete recovery is equivalent to $O(2^{2^n})$ encryptions where n is the branch size. This attack can be used against 6- and 7-round Feistel Networks in time respectively $O(2^{n2^{n-1}+2n})$ and $O(2^{n2^n+2n})$. However when modular addition is used, this attack does not work. In this case, we use an optimized guess-and-determine strategy to attack 5 rounds with complexity $O(2^{n2^{3n/4}})$.
Our results are, to the best of our knowledge, the first recovery attacks against generic 5-, 6- and 7-round Feistel Networks.
Keywords: Feistel Network, Yoyo, Generic Attack, Guess-and-determine

1 Introduction

The design of block ciphers is a well researched area. An overwhelming majority of modern block ciphers fall in one of two categories: Substitution-Permutation Networks (SPN) and Feistel Networks (FN). Examples of those two structures are the block ciphers standardized by the American National Institute for Standards and Technology, respectively the AES [1] and the DES [2]. However, since block ciphers are simply keyed permutations, the same design strategies can be applied to the design of so-called S-Boxes.

S-Boxes are "small" functions operating usually on at most 8-bits of data which are used to ensure a high non-linearity. For instance, both the DES and the AES use S-Boxes, respectively mapping 6 bits to 4 and permuting 8 bits. If a bijective S-Box is needed, it can be built like a small unkeyed block cipher. For example, the S-Box of Khazad [3] is a 3-round SPN and the S-Box of Zorro [4] is a 3-round FN. These 8×8 bits S-Boxes are built from smaller 4×4 ones to diminish memory requirements.

* Full version of the paper published in the proceedings of SAC 2015.

Keeping the design process of an S-Box secret might be necessary in the context of white-box cryptography, as described e.g. in [5]. In this paper, Biryukov *et al.* describe a memory-hard white-box encryption scheme relying on a SPN with large S-Boxes built like smaller SPN. Their security claim needs the fact that an adversary cannot decompose these S-Boxes into their different linear and non-linear layers. Such memory-hard white-box implementation may also be used to design proofs-of-work such that one party has an advantage over the others. Knowing the decomposition of a memory-hard function would effectively allow a party to bypass this memory-hardness. Such functions can have many use cases including password hashing and crypto-currency design.

Decomposing SPNs into their components is possible for up to 3 S-Box layers when the S-Boxes are small using the multi-set attack on SASAS described in [6]. A more general strategy for reverse-engineering of unknown S-Boxes was proposed recently in [7]. Our work pursues the same line of research but targets FN specifically: what is the complexity of recovering all Feistel functions of a R-round FN? Our results are different depending on whether the Feistel Network attacked uses an exclusive-or ($\oplus$) or a modular addition ($\boxplus$). Thus, we refer to a Feistel Network using XOR as a $\oplus$-Feistel and to one based on modular addition as a $\boxplus$-Feistel.

This work also has implications for the analysis of format-preserving encryption schemes, which are designed to encrypt data with a small plaintext set, for instance credit-card numbers (a 16 decimal digits number). In particular, the BPS [8] and FFX [9] constructions are Feistel schemes with small blocks; BPS uses 8 rounds, while FFX uses between 12 and 36 rounds (with more rounds for smaller domains). When these schemes are instantiated with small block sizes, recovering the full round functions might be easier than recovering the master key and provides an equivalent key.

Previous Work Lampe *et al.* [10], followed by Dinur *et al.* [11], studied Feistel Networks where the Feistel function at round i consists in $x \mapsto F_i(x \oplus k_i)$, with F_i being public but k_i being kept secret. If the subkeys are independent then it is possible to recover all of them for a 5-round (respectively 7-round) FN in time $O(2^{2n})$ (resp. $O(2^{3n})$) using only 4 known plaintexts with the optimised Meet-in-the-Middle attack described in [11]. However, we consider the much more complex case where the Feistel functions are completely unknown.

A first theoretical analysis of the Feistel structure and the first generic attacks were proposed in the seminal paper by Luby and Rackoff [12]. Since then, several cryptanalyses have been identified with the aim to either distinguish a Feistel Network from a random permutation or to recover the Feistel functions. Differential distinguishers against up to 5 rounds in the usual setting and 6 rounds in a multi-key setting are presented in [13], although they assume that the Feistel functions are random functions and thus have inner-collisions. Conversely, an impossible differential covering 5 rounds in the case where the Feistel functions are permutations is described in [14] and used to attack DEAL, a block cipher based on a 6-round Feistel Network. Finally, a method relying on a SAT-solver was recently shown in [7]. It is capable of decomposing Feistel Networks with

up to $n = 7$ in at most a couple of hours. How this time scales for larger n is unclear. These attacks, their limitations and their efficiency are summarized in Table 1. A short description of some of them is given in Appendix A for the sake of completeness.

Table 1: Generic Attacks against Feistel Networks.

R	Type	Power	Restrictions	Time	Data	Ref.
4	Differential	Distinguisher	Non bij. round func.	$2^{n/2}$	$2^{n/2}$	[13]
	Guess & Det.	Full recovery	–	$2^{3n/2}$	$2^{3n/2}$	Sec. 4.2
5	Differential	Distinguisher	Non bij. round func.	2^{n}	2^{n}	[13]
	Imp. diff.	Distinguisher	Bij. round func.	2^{2n}	2^{n}	[14]
	SAT-based	Full recovery	$n \leq 7$	Practical	2^{2n}	[7]
	Yoyo	Full Recovery	Only for $\oplus$-Feistel	2^{2n}	2^{2n}	Sec. 3.3
	Integral	Full recovery	S_1 or S_3 bij.	$2^{2.81n}$	2^{2n}	Sec. 5
	Guess & Det.	Full recovery	–	$2^{n2^{3n/4}}$	2^{2n}	Sec. 4.3
6	Differential	Distinguisher	Multi-key setting	2^{2n}	2^{2n}	[13]
	Yoyo	Full recovery	Only for $\oplus$-Feistel	$2^{n2^{n-1}+2n}$	2^{2n}	Sec. 3.5
7	Yoyo	Full recovery	Only for $\oplus$-Feistel	$2^{n2^{n}+2n}$	2^{2n}	Sec. 3.5

Our Contribution We present attacks against generic 5-round Feistel Networks which recover all Feistel functions efficiently instead of only distinguishing them from random. Furthermore, unlike distinguishers from the litterature, our attacks do not make any assumptions about whether the Feistel functions are bijective or not. Our attack against $\oplus$-Feistel uses the *yoyo game*, a tool introduced in [15] which we improve by providing a more general theoretical framework for it and leveraging particular cycle structures to diminish its cost. The principle of the yoyo game is introduced in Section 2 and how to use cycles to improve it is described in Section 3. We also present an optimized guess-and-determine attack which, unlike yoyo cryptanalysis, works against $\boxplus$-Feistel. It exploits a boomerang-like property related to the one used in our yoyo game to quickly explore the implications of the guess of an S-Box entry, see Section 4. Finally, an integral attack is given in Section 5.

We note that several of our attack have a double exponential complexity, and can only be used in practice for small values of n, as used for 8-bit S-Boxes ($n = 4$) or format-preserving encryption with a small domain.

Notation We introduce some notation for the different states during encryption (see Figure 1). Each of the values is assigned a letter, e.g. the left side of the

input is in position "A". When we look at 5-round Feistel Networks, the input is fed in positions A and B and the output is read in G, F. For 6 rounds, the input is the same but the output is read in H, G with $H = S_5(G) + F$. If we study a $\boxplus$-Feistel then "+" denotes modular addition ($\boxplus$); it denotes exclusive-or ($\oplus$) if we attack a $\oplus$-Feistel. Concatenation is denoted "||" and encryption is denoted $\mathcal{E}$ (the number of rounds being clear from the context). For example, $\mathcal{E}(a||b) = g||f$ for a 5-round Feistel Network. The bit-length of a branch of the Feistel Network is equal to n.

In addition we remark that, for an R-round Feistel, we can fix one entry of the last $R-2$ Feistel functions (or the first $R-2$ ones) arbitrarily. For example, the output of the 5-round Feistel Network described in Figure 2 does not depend on α_0, α_1 or α_2.

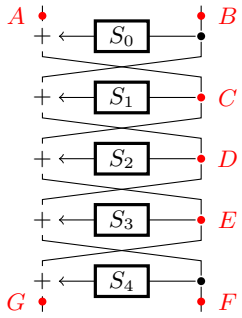


Fig. 1: Notation for the internal states.

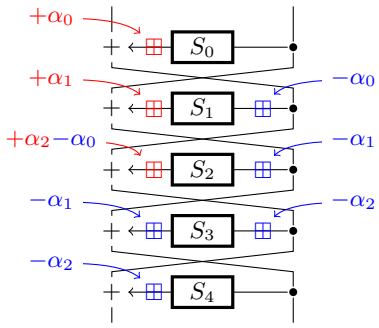


Fig. 2: Equivalent Feistel Networks.

2 Yoyo Game and Cryptanalysis

2.1 The Original Yoyo Game

Several cryptanalyses have been proposed in the literature that rely on encrypting a plaintext, performing an operation on the ciphertext and then decrypting the result. For example, the "double-swiping" used against newDES [16] in the related-key setting relies on encrypting a pair of plaintexts using two related-keys and decrypting the result using two different related-keys. Another example is the boomerang attack introduced by Wagner [17] in the single-key setting. A pair with input difference δ is encrypted. Then, a difference Δ is added to the ciphertexts and the results are decrypted, hopefully yielding two plaintexts with a difference of δ.

The yoyo game was introduced by Biham *et al.* in [15] where it was used to attack the 16 center rounds of Skipjack [18], a block cipher operating on

four 16-bits words. We describe this attack using slightly different notation and terminology to be coherent with the rest of our paper. In this paragraph, $\mathcal{E}_k$ denotes an encryption using round-reduced Skipjack under key k.

It was noticed that if the difference between two encryptions at round 5 is $(0, \Delta, 0, 0)$ where $\Delta \neq 0$ then the other three words have difference 0 between rounds 5 and 12. Two encryptions satisfying this truncated differential are said to be *connected*. The key observation is the following. Consider two plaintexts $x = (x_0, x_1, x_2, x_3)$ and $x' = (x'_0, x'_1, x_2, x'_3)$ where x_2 is constant. If they are connected, then the pair $\phi(x, x') = \big((x_0, x'_1, x_2, x_3), (x'_0, x_1, x_2, x'_3)\big)$ is connected as well (see [15] for a detailed explanation on why it is the case). Furthermore, let $y = (y_0, y_1, y_2, y_3) = \mathcal{E}_k(x)$ and $y' = (y'_0, y'_1, y'_2, y'_3) = \mathcal{E}_k(x')$. We can form two new ciphertexts by swapping their first words to obtain $z = (y'_0, y_1, y_2, y_3)$ and $z' = (y_0, y'_1, y'_2, y'_3)$. If we decrypt them to obtain $(u, u') = (\mathcal{E}_k^{-1}(z), \mathcal{E}_k^{-1}(z'))$, then u and u' are connected. If we denote $\psi(x, x')$ the function which encrypts x and x', swaps the first words of the ciphertexts obtained and decrypts the result then ψ preserves connection, just like ϕ. It is thus possible to iterate ϕ and ψ to obtain many connected pairs, this process being called the *yoyo game*.

In this section, we present other definitions of the connection and of the functions ϕ and ψ which allow us to play a similar yoyo game on 5-round Feistel Networks.

2.2 Theoretical Framework for the Yoyo Game

Consider two plaintexts $a||b$ and $a'||b'$ such that the difference between their encryptions in positions (C, D) is equal to $(\gamma, 0)$ with $\gamma \neq 0$. Then the difference in position E is equal to γ. Conversely, the difference in (E, D) being $(\gamma, 0)$ implies that the difference in C is γ. When this is the case, the two encryptions satisfy the systems of equations and the trail described in Figure 3.

Top equations

$$\begin{cases} S_0(b) \oplus S_0(b') = a \oplus a' \oplus \gamma \\ S_1(a \oplus S_0(b)) \oplus S_1(a' \oplus S_0(b')) = b \oplus b' \end{cases}$$

Bottom equations

$$\begin{cases} S_4(f) \oplus S_4(f') = g \oplus g' \oplus \gamma \\ S_3(g \oplus S_4(f)) \oplus S_3(g' \oplus S_4(f')) = g \oplus g' \end{cases}$$

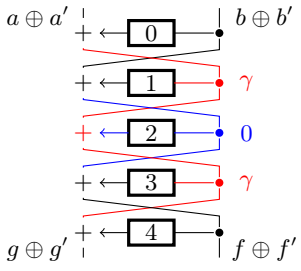


Fig. 3: The equations defining connection in γ and the corresponding differential trail.

Definition 1. *If the encryptions of $a||b$ and $a'||b'$ follow the trail in Figure 3 then they are said to be* connected in γ.

This *connection* is an "exclusive" relation: if $(a||b)$ and $(a'||b')$ are connected, then neither $(a||b)$ nor $(a'||b')$ can be connected to anything else. Furthermore, we can replace (a, a') by $(a \oplus \gamma, a' \oplus \gamma)$ in the top equations and still have them being true. Indeed, the two γ cancel each other in the first one. In the second, the values input to each call to S_1 are simply swapped as a consequence of the first equation. Similarly, we can replace (g, g') by $(g \oplus \gamma, g' \oplus \gamma)$ in the bottom equations.[4] As consequence of these observation, we state the following lemma.

Lemma 1. *We define the following two involutions*

$$\phi_\gamma(a||b) = (a \oplus \gamma)||b, \;\; \psi_\gamma = \mathcal{E}^{-1} \circ \phi_\gamma \circ \mathcal{E}.$$

If $a||b$ and $a'||b'$ are connected then, with probability 1:

- *$\phi_\gamma(a||b)$ and $\phi_\gamma(a'||b')$ are connected,*
- *$\psi_\gamma(a||b)$ and $\psi_\gamma(a'||b')$ are connected.*

By repeatedly applying ϕ_γ and ψ_γ component-wise on a pair of plaintexts (x, x'), we can play a yoyo game which preserves connection in γ. This process is defined formally below.

Definition 2. *Let $\big(x_0 = (a_0||b_0), x'_0 = (a'_0||b'_0)\big)$ be a pair of inputs. The* yoyo game in γ *starting in (x_0, x'_0) is defined recursively as follows:*

$$\big(x_{i+1}, x'_{i+1}\big) = \begin{cases} \big(\phi_\gamma(x_i), \phi_\gamma(x'_i)\big) & \text{if } i \text{ is even,} \\ \big(\psi_\gamma(x_i), \psi_\gamma(x'_i)\big) & \text{if } i \text{ is odd} \end{cases}$$

Lemma 2. *If (x_0, x'_0) is connected in γ then all pairs in the game starting in (x_0, x'_0) are connected in γ. In other words, either all pairs within the game played using ϕ_γ and ψ_γ are connected in γ or none of them are.*

2.3 The Yoyo Cryptanalysis Against 5-Round $\oplus$-Feistel Networks

Given a yoyo game connected in γ, it is easy to recover Feistel functions S_0 and S_4 provided that the yoyo game is long enough, i.e. that it contains enough connected pairs to be able to recover all 2^n entries of both S-Boxes. If the yoyo game is not connected in γ then *yoyo cryptanalysis* (Algorithm 1) identifies it as such very efficiently.

It is a differential cryptanalysis using that all pairs in the game are (supposed to be) right pairs for the differential trail defining connection in γ. If it is not the case, S_0 or S_4 will end up requiring contradictory entries, e.g. $S_0(0) = 0$ and $S_0(0) = 1$. In this case, the game is not connected in γ and must be discarded. Yoyo cryptanalysis is described in Algorithm 1.[5]. It only takes as inputs a (possible) yoyo game and the value of γ. Algorithm 2 describes AddEntry, a subroutine

[4] However, such a yoyo game cannot be played against a $\boxplus$-Feistel, as explained in Appendix B. It only works in characteristic 2.

[5] It can also recover S_4 in an identical fashion but this part is omitted for the sake of clarity

handling some linear equations. Note that one entry can be set arbitrarily (here, $S_0(0) = 0$) as summarized in Figure 2.

Let $\mathcal{Y}$ be a (supposed) yoyo game of size $|\mathcal{Y}|$. For each pair in it, either an equation is added to the list, `FAIL` is returned or AddEntry is called. While the recursive calls to AddEntry may lead to a worst time complexity quadratic in $|\mathcal{Y}|$ if naively implemented, this problem can be mitigated by using a hashtable indexed by the Feistel functions inputs instead of a list. Furthermore, since already solved equations are removed, the total time complexity is $O(|\mathcal{Y}|)$.

Algorithm 1 Yoyo cryptanalysis against a 5-round $\oplus$-Feistel Network
Inputs supposed yoyo game $\left(a_i||b_i, a'_i||b'_i\right)$; difference γ | **Output** S_0 or `FAIL`

$L_e \leftarrow []$ ▷ List of equations
$S_0 \leftarrow$ empty S-Box
$\delta_0 \leftarrow a_0 \oplus a'_0 \oplus \gamma$
$S_0(b_0) \leftarrow 0$, $S_0(b'_0) \leftarrow \delta_0$
for all $i \geq 1$ **do**
  $\delta_i \leftarrow a_i \oplus a'_i \oplus \gamma$
  if $S_0(b_i)$ and $S_0(b'_i)$ are already known and $S_0(b_i) \oplus S_0(b'_i) \neq \gamma$ **then**
    return `FAIL`
  else if $S_0(b_i)$ is known but not $S_0(b'_i)$ **then**
    AddEntry$\left(S_0, b'_i, S_0(b_i) \oplus \delta_i, L_e\right)$; **if** it fails **then return** `FAIL`
  else if $S_0(b'_i)$ is known but not $S_0(b_i)$ **then**
    AddEntry$\left(S_0, b_i, S_0(b'_i) \oplus \delta_i, L_e\right)$; **if** it fails **then return** `FAIL`
  else
    add "$S_0(b'_i) \oplus S_0(b_i) = \delta_i$" to L_e.
  end if
end for
return S_0

3 An Improvement: Using Cycles

3.1 Cycles and Yoyo Cryptanalysis

A yoyo game is a cycle of ψ_γ and ϕ_γ applied iteratively component-wise on a pair of elements. Thus, it can be decomposed into two cycles, one for each "side" of the game: $(x_0, x_1, x_2, ...)$ and $(x'_0, x'_1, x'_2, ...)$. This means that both cycles must have the same length, otherwise the game would imply that x_0 is connected to x'_j for $j \neq 0$, which is impossible. Since both ϕ_γ and ψ_γ are involutions, the cycle can be iterated through in both directions. Therefore, finding one cycle gives us two directed cycles.

In order to exploit yoyo games, we could generate pairs (x_0, x'_0) at random, generate the yoyo game starting at this pair and then try and recover S_0 and S_4 but this endeavour would only work with probability 2^{-2n} (the probability for

Algorithm 2 Adding new entry to S_0 (AddEntry)
Inputs S-Box S_0 ; input x ; output y ; List of equations L_e | **Output** `SUCCESS` or `FAIL`

if $S_0(x)$ already set and $S_0(x) = y$ **then**
  return `SUCCESS` ▷ No new information
else if $S_0(x)$ already set and $S_0(x) \neq y$ **then**
  return `FAIL` ▷ Contradiction identified
else
  $S_0(x) \leftarrow y$
  for all Equation $S_0(x_i) \oplus S_0(x'_i) = \Delta_i$ in L_e **do**
    if $S_0(x_i)$ and $S_0(x'_i)$ are set **then**
      if $S_0(x_i) \oplus S_0(x'_i) \neq \Delta_i$ **then return** `FAIL` ; **else** Remove eq. from L_e ▷ Eq. satisfied
    else if $S_0(x_i)$ is set but not $S_0(x'_i)$ **then**
      AddEntry$\big(S_0, x'_i, S_0(x_i) \oplus \Delta_i, L_e\big)$; **if** it fails **then return** `FAIL` ▷ Eq. gives new entry
    else if $S_0(x'_i)$ is set but not $S_0(x_i)$ **then**
      AddEntry$\big(S_0, x_i, S_0(x'_i) \oplus \Delta_i, L_e\big)$; **if** it fails **then return** `FAIL` ▷ Eq. gives new entry
    end if
  end for
end if
return `SUCCESS`

two random points to be connected). Instead, we can use the link between cycles, yoyo games and connection in γ as is described in this section. Note that the use of cycles in cryptography is not new; in fact it was used in the first cryptanalyses against ENIGMA. More recently, particular distribution of cycle sizes were used to distinguish rounds-reduced PRINCE-core [19] from random [20] and to attack involutional ciphers [21].

3.2 Different Types of Cycles

Let $\mathcal{C} = (x_i)_{i=0}^{\ell-1}$ be a cycle of length ℓ of ψ_γ and ϕ_γ, with $x_{2i} = \psi_\gamma(x_{2i-1})$ and $x_{2i+1} = \phi_\gamma(x_{2i})$. We denote the point connected to x_i as y_i, where all indices are taken modulo ℓ. Since x_i and y_i are connected, and the connection relation is one-to-one, we also have $y_{2i} = \psi_\gamma(y_{2i-1})$ and $y_{2i+1} = \phi_\gamma(y_{2i})$. Therefore, $\mathcal{C}' = (y_i)_{i=0}^{\ell-1}$ is also a cycle of length ℓ.

We now classify the cycles according to the relationship between $\mathcal{C}$ and $\mathcal{C}'$.

- If $\mathcal{C}$ and $\mathcal{C}'$ are **D**istincts, $\mathcal{C}$ is a **Type-D** cycle. A representation is given in Figure 4a. Otherwise, there exists k such that $y_0 = x_k$.
- If k is even, we have $x_{k+1} = \phi_\gamma(x_k)$. Since $x_k = y_0$ is connected to x_0, $x_{k+1} = \phi_\gamma(x_k)$ is connected to $\phi_\gamma(x_0) = x_1$, i.e. $y_1 = x_{k+1}$. Further, $x_{k+2} = \psi_\gamma(x_{k+1})$ is connected to $\psi_\gamma(x_1) = x_2$, i.e. $y_2 = x_{k+2}$. By induction, we have $y_i = x_{k+i}$. Therefore x_0 is connected to x_k and x_k is connected to x_{2k}. Since the connection relation is one-to-one, this implies that $2k = \ell$.

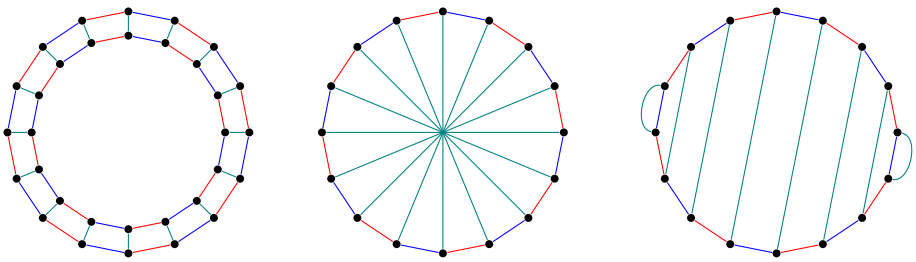
We denote this setting as a **Type-S** cycle. Each element x_i is connected to $x_{i+\ell/2}$. Thus, if we represent the cycle as a circle, the connections between the elements would all cross in its center, just like **S**pokes, as can be seen in Figure 4b.

– If k is odd, we have $x_{k-1} = \phi_\gamma(x_k)$. Since $x_k = y_0$ is connected to x_0, $x_{k-1} = \phi_\gamma(x_k)$ is connected to $\phi_\gamma(x_0) = x_1$, i.e. $y_1 = x_{k-1}$. Further, $x_{k-2} = \psi_\gamma(x_{k-1})$ is connected to $\psi_\gamma(x_1) = x_2$, i.e. $y_2 = x_{k-2}$. By induction, we have $y_i = x_{k-i}$.

  We denote this setting as a **Type-P** cycle. If we represent the cycle as a circle, the connections between the elements would all be **P**arallel to each other as can be seen in Figure 4c.

  In particular, there at exactly two pairs (x_i, x_{i+1}) such that x_i and x_{i+1} are connected. Indeed, we have $x_{i+1} = y_i$ if and only if $i + 1 \equiv k - i \bmod \ell$ i.e. $i \equiv (k-1)/2 \bmod \ell/2$. As a consequence, the existence of w connected pairs (x, x') with $x' = \phi_\gamma(x)$ or $x' = \psi_\gamma(x)$ implies the existence of $w/2$ Type-P cycles.

  In addition, Type-P cycles can only exist if either S_1 or S_3 are not bijections. Indeed, if $(a||b)$ and $(a \oplus \gamma||b)$ are connected then the difference in position D cannot be zero unless S_1 can map a difference of γ to zero. If it is a permutation, this is impossible. The situation is identical for S_3. Furthermore, each value c such that $S_1(c) = S_1(c \oplus \gamma)$ implies the existence of 2^n values $(a||b)$ connected to $\phi_\gamma(a||b)$ as b can be chosen arbitrarily and a computed from b and c. Again, the situation is identical for S_3. Thus, if $S_1(x) = S_1(x \oplus \gamma)$ has w_1 solutions and if $S_3(x) = S_3(x \oplus \gamma)$ has w_3 solutions then there are $(w_1 + w_3) \cdot 2^{n-2}$ Type-P cycles. See Appendix C.1 for actual examples of structures of the functional graphs of ϕ_γ and ψ_γ for small n.



(a) Type-D cycles. (b) A Type-S cycle. (c) A Type-P cycle.

Fig. 4: All the types of cycles that can be encountered. ϕ_γ is a blue line, ψ_γ is a red one and connection is a green one (remember that ϕ_γ and ψ_γ are involutions).

3.3 The Cycle-Based Yoyo Cryptanalysis

Exploiting a Type-S cycle is a lot easier than exploiting a Type-P or a pair of Type-D cycles. Indeed, the connected pairs $(x_i, x_{i+\ell/2})$ can be immediately derived from the length ℓ of the cycle, while we have to guess a shift amount for connected pairs in a Type-P cycle, or between two type D cycles. Thus, it makes sense to target those specifically, for instance by implementing Algorithm 3.

Algorithm 3 The cycle based yoyo cryptanalysis.

for all $\gamma \in \{0,1\}^{2n} \backslash \{0\}$ **do**
  for all $s \in \{0,1\}^{2n}$ **do**
    if s was not encountered before for this γ **then**
      $\mathcal{C} \leftarrow$ empty list
      $x \leftarrow s$
      repeat
        $x \leftarrow \phi(x)$; append x to $\mathcal{C}$
        $x \leftarrow \psi(x)$; append x to $\mathcal{C}$
      until $x = s$
      if $|\mathcal{C}| \geq 2^{n+2}$ **then**
        Build yoyo game $\mathcal{Y} = \big(\mathcal{C}[0,..,\ell-1], \mathcal{C}[\ell,..,2\ell-1]\big)$ with $\ell = \frac{|\mathcal{C}|}{2}$
        Run yoyo cryptanalysis (Alg. 1) against $\mathcal{Y}$
        if yoyo cryptanalaysis is a success **then**
          return S_0, S_4
        end if
      end if
    end if
  end for
end for

Let $q_S(n)$ be the probability that a Type-S cycle exists for the chosen γ for a 5-round Feistel Network built out of bijective Feistel functions. When averaged over all such Feistel Networks, this probability does not depend on γ. A discussion about its value is given in Section 3.4.

This attacks requires $\mathrm{O}(2^{2n}/n)$ blocks of memory to store which plaintexts were visited and $\mathrm{O}(2^{2n})$ time. Indeed, at most all elements of the codebook will be evaluated and inspected a second time when attempting a yoyo cryptanalysis on each cycle large enough. Even though the attack must be repeated about $1/q_S(n)$ times to be able to obtain a large enough Type-S cycle, $q_S(n)$ increases with n so that $1/q_S(n)$ can be upper-bounded by a constant independent of n. Note also that special points can be used to obtain a time-memory tradeoff: instead of storing whether all plaintexts were visited or not, we only do so for those with, say, the first $\mathcal{B}$ bits equal to 0. In this case, the time complexity becomes $\mathrm{O}(\mathcal{B} \cdot 2^{2n})$ and the memory complexity $\mathrm{O}\big(2^{2n}/(n \cdot \mathcal{B})\big)$. Access to the hash table storing whether an element has been visited or not is a bottle-neck in practice so special points actually give a "free" memory improvement in the

sense that memory complexity is decreased without increasing time. In fact, wall clock time may actually decrease. An attack against a $\oplus$-Feistel with $n = 14$ on a regular desktop computer[6] takes about 1 hour to recover both S_0 and S_4.

3.4 Experimental Results on the Cycle Type Distribution

We call γ_{win} the first value of γ such that the attack works, i.e. such that a large enough Type-S cycle was found. Since $q_S(n)$ does not depend on γ, the probability distribution of γ_{win} is:

$$P[\gamma_{\text{win}} = x] = \big(1 - q_S(n)\big)^{x-1} \cdot q_S(n).$$

This is coherent with our experimental results. Indeed, we found that

$$P[\gamma_{\text{win}} = x] \approx Q \cdot \exp(-x/\tau).$$

The values of Q and τ are given for different values of n in the bijective and non-bijective case in Appendix C.2 in Tables 2a and 2b. The final result is Figure 5 which shows our estimation of $q_S(n)$ and its counterpart in the non-bijective case.

As we can see, there is a descrepancy: the probability is on average about 2.65 times smaller in the case of non-bijective Feistel functions. This can be explained by the massive presence of Type-P cycles. Let w_i be the number of solutions of $S_i(x) = S_i(x \oplus \gamma)$. Then, as explained in Section 3.1, there are $(w_1 + w_3) \cdot 2^{n-2}$ Type-P cycles. If $w_1 + w_3 > 0$ then the large number of such cycles we obtain effectively "clogs" the cycle space and prevents the existence of large enough Type-S cycles. In fact, $w_i/2$ follows a Poisson distribution with parameter $1/2$ when S_i is a random function mapping n bits to n [22]. Hence, the probability that $w_i = 0$ is $\exp(-1/2)$ and the probability that $w_1 + w_3 = 0$ is equal to e^{-1}, meaning that the probability is about 2.72 times smaller in the non-bijective case. This result is coherent with the ratio of 2.65 we found experimentally.

3.5 Attacking 6 and 7 Rounds

An Attack on 6 rounds A naive approach could consist in guessing all of the entries of S_5 and, for each guess, try running a cycle-based yoyo cryptanalysis. If it fails then then the guess is discarded. Such an attack would run in time $\mathrm{O}\big(2^{n2^n+2n}\big)$. However, it is possible to run such an attack at a cost similar to that of guessing only half of the entries of S_5, namely $\mathrm{O}(2^{n2^{n-1}+2n})$ which corresponds to a gain of $2^{n2^{n-1}}$.

Instead of guessing all the entries, this attack requires guessing the values of $\Delta_5(x, \gamma) = S_5(x) \oplus S_5(x \oplus \gamma)$. Once these are know, we simply need to replace ψ_γ by ψ'_γ with

$$\big(\mathcal{E} \circ \psi'_\gamma \circ \mathcal{E}^{-1}\big)(g||h) = \big(g \oplus \gamma \;||\; h \oplus \Delta_5(x, \gamma)\big).$$

[6] CPU: Intel core i7-3770 (3.40 GHz); 8 Gb of RAM. The program was compiled with `g++` and optimization flag `-O3`.

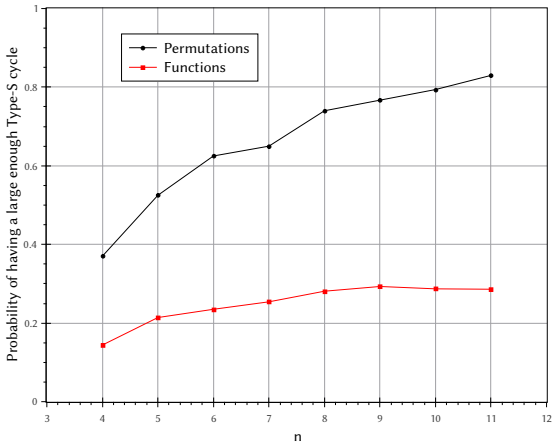


Fig. 5: Probability of having a large enough Type-S cycle when all Feistel functions are permutations (black) and when collisions are allowed (red).

The cycle-based yoyo cryptanalysis can then be run as previously because, again, both ϕ_γ and ψ'_γ preserve connection in γ. Once it succeeds, the top S-Box is known which means that it can be peeled of. The regular attack is then performed on the remaining 5 rounds. Note that if the yoyo cryptanalysis fails because of inner collisions in S_1 or S_3 then we can still validate a correct guess by noticing that there are $\mathrm{O}(2^n)$ cycles instead of $\mathrm{O}(2n)$ as would be expected[7].

In this algorithm, 2^{n-1} values of $[0, 2^n - 1]$ must be guessed and for each of those an attack with running time $\mathrm{O}(2^{2n})$ must be run. Hence, the total running time is $\mathrm{O}\big(2^{n2^{n-1}+2n}\big)$. The time necessary to recover the remainder of the Feistel functions is negligible.

An Attack on 7 rounds A $\oplus$-Feistel with 7 rounds can be attacked in a similar fashion by guessing both $\Delta_0(x, \gamma)$ and $\Delta_6(x, \gamma)$ for all x. These guesses allow the definition of ϕ''_γ and ψ''_γ, as follows:

$$\begin{aligned} \phi''_\gamma(a||b) &= \big(a \oplus \Delta_0(x,\gamma) \;||\; b \oplus \gamma\big) \\ \big(\mathcal{E} \circ \psi''_\gamma \circ \mathcal{E}^{-1}\big)(g||h) &= \big(h \oplus \Delta_6(x,\gamma) \;||\; g \oplus \gamma\big). \end{aligned}$$

For each complete guess $\big((\Delta_0(x,\gamma_0), \forall x), (\Delta_6(x,\gamma_0), \forall x)\big)$, we run a yoyo cryptanalysis. If it succeeds, we repeat the attack for a new difference γ_1. In this second step, we don't need to guess 2^{n-1} values for each $\Delta_0(x,\gamma_1)$ and $\Delta_6(x,\gamma_1)$ but only 2^{n-2} as $\Delta_i(x \oplus \gamma_0, \gamma_1) = \Delta_i(x,\gamma_0) \oplus \Delta(x \oplus \gamma_1, \gamma_0) \oplus \Delta_i(x,\gamma_1)$. We run again a cycle-based yoyo cryptanalysis to validate our guesses. The process is repeated $n-1$ times in total so as to have $\sum_{k=0}^{n-1} 2^k = 2^n$ independent linear

[7] A random permutation of a space of size N is expected to have about $\log_e(N)$ cycles.

equations connecting the entries of S_0 and another 2^n for the entries of S_6. Solving those equations gives the two outer Feistel functions, meaning that they can be peeled off. We then run a regular yoyo-cryptanalysis on the 5 inner rounds to recover the remainder of the structure.

Since $\sum_{k=0}^{n-1} 2^{n2^k+2n} = \mathrm{O}\left(2^{n2^n+2n}\right)$, the total time complexity of this attack is $\mathrm{O}\left(2^{n2^n+2n}\right)$, which is roughly the complexity of a naive 6-round attack based on guessing a complete Feistel function and running a cycle-based yoyo cryptanalysis on the remainder.

4 Guess and Determine Attack

Since the yoyo game is only applicable to an $\oplus$-Feistel (as shown in Appendix B), we now describe a different attack that works for any group operation $+$. This guess and determine attack is based on a well-known boomerang-like distinguisher for 3-round Feistel Networks, initially described by Luby and Rackoff [12]. This is used to attack Feistel Networks with 4 or 5 rounds using a guess and determine approach: we guess entries of S_3 and S_4 in order to perform partial encryption/decryption for some values, and we use the distinguisher on the first three rounds in order to verify the consistency of the guesses, and to recover more values of S_3 and S_4.

4.1 Three-round property

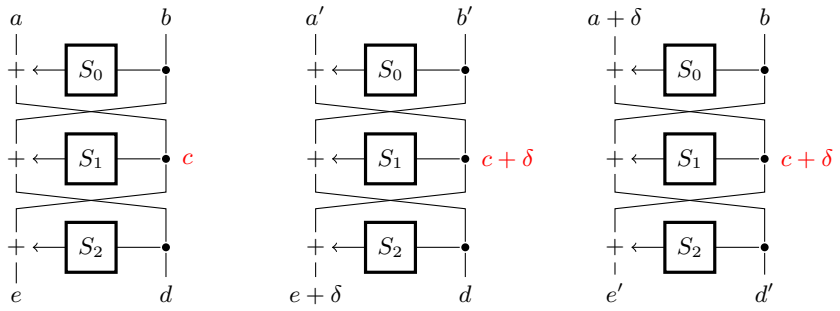


Fig. 6: Distinguisher for a 3-round Feistel

The distinguisher is illustrated by Figure 6, and works as follows:

- Select arbitrary values a, b, δ ($\delta \neq 0$);
- Query $(e, d) = \mathcal{E}(a, b)$ and $(e', d') = \mathcal{E}(a + \delta, b)$;
- Query $(a', b') = \mathcal{E}^{-1}(e + \delta, d)$;
- If $\mathcal{E}$ is a three-round Feistel, then $d - b' = d' - b$.

The final equation is always true for a 3-round Feistel Network because the input to the third Feistel function is $c + \delta$ for both queries $\mathcal{E}(a + \delta, b)$ and $\mathcal{E}^{-1}(e + \delta, d)$. Therefore the output of S_1 is the same in both cases. On the other hand, the relation only holds with probability 2^{-n} for a random permutation.

4.2 Four-round attack

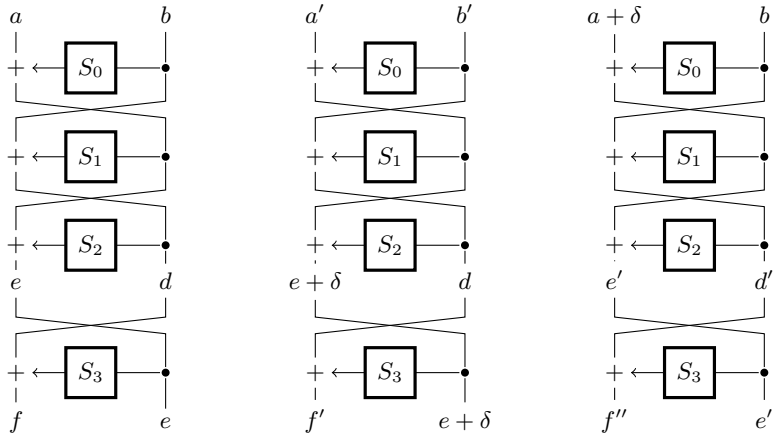


Fig. 7: Attack against 4-round Feistel

We now explain how to use this property to decompose a four-round Feistel network. We first fix $S_3(0) = 0$, and guess the value $S_3(1)$. Then we use known values of S_3 to iteratively learn new values as follows (see Figure 7):

- Select e and δ such that $S_3(e)$ and $S_3(e + \delta)$ are known, with $\delta \neq 0$.
- For every $d \in \mathbb{F}_2^n$, we set $f = d + S_3(e)$ and $f' = d + S_3(e + \delta)$; we query $(a, b) = \mathcal{E}^{-1}(f, e)$ and $(a', b') = \mathcal{E}^{-1}(f', e + \delta)$
- Then we query $(f'', e') = \mathcal{E}(a + \delta, b)$. Using the three-round property, we know that:
$$d - b' = d' - b, \quad \text{where } d' = f'' - S_3(e').$$
This gives the value of $S_3(e')$ as $f'' - d + b' - b$.

We iterate the deduction algorithm until we either detect a contradiction (if the guess of $S_3(1)$ is wrong), or we recover the full S_3. Initially, we select $e = 0, \delta = 1$, or $e = 1, \delta = -1$, with 2^n choices of d: this allows 2^{n+1} deductions. If the guess of $S_3(1)$ is wrong, we expect to find a contradiction after about $2^{n/2}$ deductions. If the guess is correct, almost all entries of S_3 will be deduced with a single choice of e and δ, and we will have many options for further deduction. Therefore, the complexity of this attack is about $2^{3n/2}$.

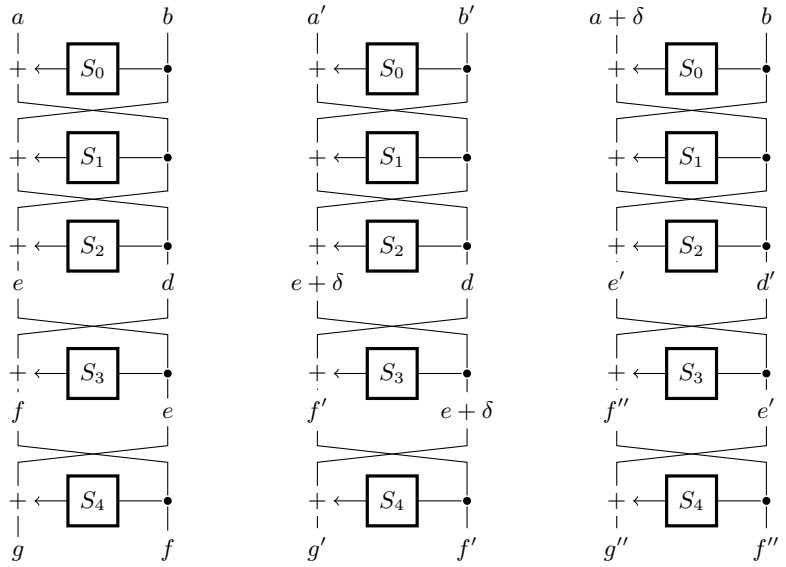


Fig. 8: Attack against 5-round Feistel

4.3 Five-round attack

The extension from 4 rounds to 5 rounds is similar to the extension from a three-round distinguisher to a four-round attack. First, we guess some entries of the last S-Box, so that we can invert the last round for a subset of the outputs. Then, we use those pairs to perform an attack on a reduced version so as to test whether the guess was valid. However, we need to guess a lot more entries in this context. The deductions are performed as follows (see Figure 8):

- Select d, e and δ such that $S_3(e)$, $S_3(e+\delta)$, $S_4(d+S_3(e))$ and $S_4(d+S_3(e+\delta))$ are known.
- Let $(f, g) = (d+S_3(e), e+S_4(f))$ and $(f', g') = (d+S_3(e+\delta), e+\delta+S_4(f'))$, then query $(a, b) = \mathcal{E}^{-1}(g, f)$ and $(a', b') = \mathcal{E}^{-1}(g', f')$
- Finally, query $(g'', f'') = \mathcal{E}(a+\delta, b)$. Assuming that $S_4(f'')$ is known, we can use the three-round property and deduce:

$$d - b' = d' - b, \quad \text{where } d' = f'' - S_3(g'' - S_4(f''))$$

This gives the value of $S_3(g'' - S_4(f''))$ as $f'' - d + b' - b$.

Guessing strategy. The order in which we guess entries of S_3 and S_4 is very important in order to obtain a low complexity attack. We first guess the values of $S_3(i)$ and $S_4(S_3(i))$ for $i < \ell$, with $\ell > 2^{n/2}$. This allows to try deductions with $d = 0$ and any $e, e+\delta \leq \ell$, *i.e.* ℓ^2 attempts. Since ℓ entries of S_4 are known, each attempt succeeds with probability $\ell 2^{-n}$, and we expect to guess about $\ell^3 2^{-n}$

new values of S_3. With $\ell > 2^{n/2}$, this will introduce a contradiction with high probability.

When an intial guess is non-contradictory, we select x such that $S_3(x)$ has been deduced earlier, we guess the corresponding value $S_4(S_3(x))$, and run again the deduction. The new guess allows to make ℓ new deduction attempts with $d = 0$, $e < \ell$ and $e' = x$. We expect about $\ell^2 2^{-n}$ successful new deductions. With $\ell = 2^{3n/4+\varepsilon}$ with a small $\varepsilon > 0$, the probability of finding a contradiction is higher than 2^{-n}, and the size of the search tree decreases.

The attack will also work if we start with $2^{n/4}$ entries in S_3 and $2^{3n/4}$ entries in S_4: the first step will deduce $2^{3n/4}$ values in S_3. Therefore, we have to make only $2^{n/4} + 2^{3n/4} \approx 2^{3n/4}$ guesses, and the total complexity is about $2^{n2^{3n/4}}$.

Application to $n = 4$. We now explain the attack in more detail with $n = 4$. We first set $S_4(0) = S_3(0) = 0$ and we guess the values of $S_3(1)$, $S_3(2)$, $S_4(S_3(1))$, and $S_4(S_3(2))$. In particular this allows to compute the last two rounds for the following (e, d) values:

$$(0,0) \qquad (1,0) \qquad (2,0)$$

This gives 6 candidates $(e, d), \delta$ for the deduction algorithm:

$$(0,0), \delta \in \{1,2\} \qquad (1,0), \delta \in \{1,-1\} \qquad (2,0), \delta \in \{-1,-2\}$$

Each candidate gives a deduction with probability 3/16 because three entries are known in S_4. Therefore there is a good probability to get one deduction $S_4(x)$. In this case, we guess the value $S_3(S_4(x))$, so that we can also compute the last two rounds for $(d, e) = (x, 0)$. We have at least 6 new candidates $(e, d), \delta$ for the deduction algorithm:

$$(x,0), \delta \in \{-x, 1-x, 2-x\} \quad (0,0), \delta = x \quad (1,0), \delta = x-1 \quad (2,0), \delta = x-2$$

In total, we have 12 candidates, and each of them gives a deduction with probability 4/16, including the deduction made in the first step. We expect about 3 deductions in total, which leads to 7 known values in S_3. Since $7 > 2^{n/2}$, there is already a good chance to detect a contradiction. For the remaining cases, we have to make further guesses of S_4 entries, and repeat the deduction procedure.

Since we had to make five guesses for most branches of the guess and determine algorithm, the complexity is about 2^{20}. In practice, this attack takes less than one second on a single core with $n = 4$ (on a 3.4 GHz Haswell CPU).

5 Integral attack

Finally, we present an integral attack against 5-round Feistels, that was shown to us by one of the anonymous reviewers. This attack has a complexity of 2^{3n}, and works for any group operation $+$, but it requires S_1 or S_3 to be a permutation.

The attack is based on an integral property, as introduced in the cryptanalysis of SQUARE [23,24]. In the following, we assume that S_1 is a permutation; if S_3 is a permutation instead, the attack is performed against the decryption oracle. An attacker uses a set of 2^n plaintexts (a_i, b) where a_i takes all possible values, and b is fixed to a constant value. She can then trace the evolution of this set of plaintext through the Feistel structure:

- $c_i = a_i + S_0(b)$ takes all possible values once;
- $d_i = b + S_1(c_i)$ takes all possible values once, since S_1 is a permutation;
- $e_i = c_i + S_2(d_i)$ has a fixed sum:

$$\sum_i e_i = \sum_{x \in \{0 \dots 2^n - 1\}} x + \sum_{x \in \{0 \dots 2^n - 1\}} S_2(x) = S.$$

  The first term is 0 for a $\oplus$-Feistel, and 2^{n-1} for a $\boxplus$-Feistel, while the second term is equal to the first if S_2 is a permutation, but otherwise unknown.

After collecting the 2^n ciphertexts (g_i, f_i) corresponding to the set of plaintexts, she can express $e_i = g_i - S_4(f_i)$. The fixed sum $\sum e_i = S$ gives a linear equation between the values $S_4(f_i)$ and S. This can be repeated with 2^n different sets of plaintexts, in order to build 2^n linear equations. Solving the equations recovers the values $S_4(f_i)$, i.e. the full S_4 box.

When S_2 is a permutation, S is known, and the system has a single solution with high probability. However, when S is unknown, the system has n equations and $n+1$ unknowns; with high probability it has rank n and 2^n solutions. Therefore, an attacker has to explore the set of solutions, and to use a 4-round distinguisher to verify the guess. Using the attack of Section 4.2, this has complexity $2^{5n/2}$.

For a $\oplus$-Feistel, the cost of solving the linear system is 2^{3n} with Gaussian elimination, but can be improved to $O(2^{2.81n})$ with Strassen's Algorithm (the currently best known algorithm [25] has complexity only $O(2^{2.3729n})$ but is probably more expensive for practical values of n.)

For a $\boxplus$-Feistel, solving a linear system over $\mathbb{Z}/2^n\mathbb{Z}$ is harder. However, we can solve the system bit-by-bit using linear algebra over $\mathbb{F}_2$. We first consider the equations modulo 2, and recover the least significant bit of S_4. Next, we consider the equations modulo 4. Since the least significant bits are known, this also turns into linear equations over $\mathbb{F}_2$, with the scond bits as unknowns. We repeat this technique from the least significant bit to the most significant. At each step, we might have to consider a few different candidates if the system is not full rank.

In total, this attack has a time complexity about $2^{2.81n}$.

6 Conclusion

We presented new generic attacks against Feistel Networks capable of recovering the full description of the Feistel functions without making any assumptions

regarding whether they are bijective or not. To achieve this, we have improved the yoyo game proposed in [15] using cycles and found an efficient guessing strategy to be used if modular addition is used instead of exclusive-or. We implemented our attacks to check our claims. We finally described an integral attack suggested by an anonymous reviewer.

Our attacks allow an efficient recovery of the Feistel functions for 5-round Feistel Networks and cycle-based yoyo cryptanalysis can be pushed to attack 6-round and(respectively 7-round) $\oplus$-Feistel at a cost similar to guessing half (resp. all) of the entries of a Feistel function.

Our results differ significantly between $\oplus$-Feistel and $\boxplus$-Feistel. It remains an open problem to find a more efficient attack against $\boxplus$-Feistel or to theoretically explain such a difference.

Acknowledgment

We would like to thank the anonymous reviewers for their valuable comments, and in particular for showing us the attack of Section 5. The work of Léo Perrin is supported by the CORE ACRYPT project (ID C12-15-4009992) funded by the *Fonds National de la Recherche* (Luxembourg).

References

1. Daemen, J., Rijmen, V.: The design of Rijndael: AES-the advanced encryption standard. Springer (2002)
2. U.S. DEPARTMENT OF COMMERCE/National Institute of Standards and Technology: Data encryption standard. Federal Information Processing Standards Publication (1999)
3. Barreto, P., Rijmen, V.: The Khazad legacy-level block cipher. Primitive submitted to NESSIE **97** (2000)
4. Gérard, B., Grosso, V., Naya-Plasencia, M., Standaert, F.X.: Block ciphers that are easier to mask: how far can we go? In: Cryptographic Hardware and Embedded Systems-CHES 2013. Springer (2013) 383–399
5. Biryukov, A., Bouillaguet, C., Khovratovich, D.: Cryptographic schemes based on the ASASA structure: Black-box, white-box, and public-key (extended abstract). In Sarkar, P., Iwata, T., eds.: Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I. Volume 8873 of Lecture Notes in Computer Science., Springer (2014) 63–84
6. Biryukov, A., Shamir, A.: Structural cryptanalysis of SASAS. In Pfitzmann, B., ed.: Advances in Cryptology – EUROCRYPT 2001. Volume 2045 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2001) 395–405
7. Biryukov, A., Perrin, L.: On reverse-engineering S-Boxes with hidden design criteria or structure. In: Advances in Cryptology – CRYPTO 2015. Lecture Notes in Computer Science. Springer Berlin Heidelberg (2015) (to appear)
8. Brier, E., Peyrin, T., Stern, J.: BPS: a format-preserving encryption proposal. Submission to NIST, available from http://csrc.nist.gov/groups/ST/toolkit/BCM/modes_development.html (2010)

9. Bellare, M., Rogaway, P., Spies, T.: The FFX mode of operation for format-preserving encryption. Submission to NIST, available from http://csrc.nist.gov/groups/ST/toolkit/BCM/modes_development.html (2010)
10. Lampe, R., Seurin, Y.: Security analysis of key-alternating Feistel ciphers. In Cid, C., Rechberger, C., eds.: Fast Software Encryption. Volume 8540 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2015) 243–264
11. Dinur, I., Dunkelman, O., Keller, N., Shamir, A.: New attacks on Feistel structures with improved memory complexities. In: Advances in Cryptology – CRYPTO 2015. Lecture Notes in Computer Science. Springer Berlin Heidelberg (2015) (to appear)
12. Luby, M., Rackoff, C.: How to construct pseudorandom permutations from pseudorandom functions. SIAM Journal on Computing **17**(2) (1988) 373–386
13. Patarin, J.: Generic attacks on Feistel schemes. Cryptology ePrint Archive, Report 2008/036 (2008) http://eprint.iacr.org/.
14. Knudsen, L.R.: DEAL – a 128-bit block cipher, aes submission (1998)
15. Biham, E., Biryukov, A., Dunkelman, O., Richardson, E., Shamir, A.: Initial observations on Skipjack: Cryptanalysis of Skipjack-3XOR. In: Selected Areas in Cryptography, Springer (1999) 362–375
16. Kelsey, J., Schneier, B., Wagner, D.: Related-key cryptanalysis of 3-way, Biham-DES, CAST, DES-X, newDES, RC2, and TEA. Information and Communications Security (1997) 233–246
17. Wagner, D.: The boomerang attack. In Knudsen, L., ed.: Fast Software Encryption. Volume 1636 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (1999) 156–170
18. National Security Agency, N.S.A.: SKIPJACK and KEA Algorithm Specifications (1998)
19. Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E.B., Knezevic, M., Knudsen, L.R., Leander, G., Nikov, V., Paar, C., Rechberger, C., et al.: PRINCE – a low-latency block cipher for pervasive computing applications. In: Advances in Cryptology–ASIACRYPT 2012. Springer (2012) 208–225
20. Derbez, P., Perrin, L.: Meet-in-the-middle attacks and structural analysis of round-reduced PRINCE. In: Fast Software Encryption: 22th International Workshop, FSE 2015, Istanbul, 2015., Springer (2015) To appear
21. Biryukov, A.: Analysis of involutional ciphers: Khazad and Anubis. In: Fast Software Encryption, Springer (2003) 45–53
22. Daemen, J., Rijmen, V.: Probability distributions of correlation and differentials in block ciphers. Journal of Mathematical Cryptology JMC **1**(3) (2007) 221–242
23. Daemen, J., Knudsen, L.R., Rijmen, V.: The block cipher square. In Biham, E., ed.: Fast Software Encryption, 4th International Workshop, FSE '97, Haifa, Israel, January 20-22, 1997, Proceedings. Volume 1267 of Lecture Notes in Computer Science., Springer (1997) 149–165
24. Knudsen, L.R., Wagner, D.: Integral cryptanalysis. In Daemen, J., Rijmen, V., eds.: Fast Software Encryption, 9th International Workshop, FSE 2002, Leuven, Belgium, February 4-6, 2002, Revised Papers. Volume 2365 of Lecture Notes in Computer Science., Springer (2002) 112–127
25. Gall, F.L.: Powers of tensors and fast matrix multiplication. In Nabeshima, K., Nagasaka, K., Winkler, F., Szántó, Á., eds.: International Symposium on Symbolic and Algebraic Computation, ISSAC '14, Kobe, Japan, July 23-25, 2014, ACM (2014) 296–303

A Attacks Against 5- and 6-Round Feistel Networks

A.1 Differential Distinguishers

In [13], Patarin shows a differential distinguisher against 5-round Feistel Networks. However, it only works if the Feistel functions have inner-collisions. It is based on the following observation. Let $(g_i||f_i)$ be the image of $(a_i||b_i)$ by a permutation and let b_i be constant. Then for $i \neq j$, such that $f_i = f_j$, count how many times $a_i \oplus a_j = g_i \oplus g_j$. This number is roughly twice as high for a 5-round Feistel Network than for a random permutation.

In the same paper, Patarin suggests two distinguishers against 6-round $\oplus$-Feistel Networks. However, these do not target a permutation but a generator of permutation. This can be interpreted as a multi-key attack: the attacker has a black-box access to several permutations and either none or all of which are 6-round $\oplus$-Feistel Networks. The first attack uses that the signature of a $\oplus$-Feistel Network is always even. The second attack exploits a statiscal bias too weak to be reliably observable using one codebook but usable when several permutations are available. It works by counting all quadruples of encryptions $(a_i||b_i) \rightarrow (g_i||h_i)$, $i = 1..4$ satisfying this system:

$$\begin{cases} b_1 = b_3, \ b_2 = b_4 \\ g_1 = g_2, \ g_3 = g_4 \\ a_1 \oplus a_3 = a_2 \oplus a_4 = g_1 \oplus g_3 \\ h_1 \oplus h_2 = h_3 \oplus h_4 = b_1 \oplus b_2. \end{cases}$$

If there are λ black-boxes to distinguish and if m queries are performed for each then we expect to find about $\lambda m^4 2^{-8n}$ solutions for a random permutation and $2\lambda m^4 2^{-8n}$ for 6-round Feistel Networks, i.e. twice as much.

A.2 Impossible Differential

Knudsen described in [14] an impossible differential attack against his AES proposal, DEAL, a 6-round Feistel Network using the DES [2] as a round function. This attack is made possible by the existence of a 5-round impossible differential caused by the Feistel functions being permutations. In this case, an input difference $(\alpha||0)$ cannot be mapped to a difference of $(\alpha||0)$ after 5 rounds. This would imply that the non-zero difference which has to appear in D as the image of α by S_2 is mapped to 0, which is impossible.

To distinguish such a 5-round FN from a random permutation we need to generate $\lambda \cdot 2^{2n}$ pairs with input difference $(\Delta||0)$. Among those, about λ should have an output difference equal to $(\Delta||0)$ if the permutation is a random permutation while it is impossible to observe if for a 5-round FN with bijective Feistel functions. Note that while the time complexity is $O(2^{2n})$, the data complexity can be brought down to $O(2^n)$ using structures.

An attack on 6 rounds uses this property by identifying pairs of encryptions with difference $(\alpha||0)$ in the input and $(\alpha||\Delta)$ for the output for any $\Delta \neq 0$. A pair

as a correct output difference with probability $2^{-n}(1-2^{-n})$ since α is fixed and Δ can take any value except 0. We repeat this process for the whole codebook and all $\alpha \neq 0$ to obtain $2^{n+(2n-1)} \cdot 2^{-n}(1-2^{-n}) = 2^{2n-1} - 2^{n-1}$ pairs. Each of them gives an impossible equation for S_5: if $\{(a||b) \to (g||h), (a \oplus \alpha||b) \to (g \oplus \alpha||h \oplus \Delta)\}$ is a pair of encryptions then it is impossible that $S_5(g) \oplus S_5(g \oplus \alpha) = \Delta$ as it would imply the impossible differential. In the end, we have a system of about $2^{2n-1} - 2^{n-1}$ impossible equations, a random Feistel function satisfying an impossible equation with probability $(1-2^{-n})$. Thus, this attack filters out all but the following fraction of candidates for S_5:

$$\text{Impossible differential filter} \;=\; \left(1-2^{-n}\right)^{2^{2n-1}-2^{n-1}} \approx 2^{0.72-1.443 \cdot 2^{n-1}}.$$

B On the Infeaseability of Our Yoyo Game Against an $\boxplus$-Feistel

Assume that the following equations holds:

$$\begin{cases} \big(S_0(b)+a\big) - \big(S_0(b')+a'\big) = \gamma \\ \big(S_1(S_0(b)+a)+b\big) - \big(S_1(S_0(b')+a')+b'\big) = 0. \end{cases} \tag{1}$$

In order to be able to play a yoyo game against the corresponding $\boxplus$-Feistel, we need to be able to replace a by $a+\gamma$ and a' by $a'+\gamma$ in System (1) and still have it hold. In other words, we need that Equations (1) holding implies that the following equations hold as well:

$$\begin{cases} \big(S_0(b)+a+\gamma\big) - \big(S_0(b')+a'+\gamma\big) = \gamma \\ \big(S_1(S_0(b)+a+\gamma)+b\big) - \big(S_1(S_0(b')+a'+\gamma)+b'\big) = 0. \end{cases} \tag{2}$$

The first one trivially does. Using it, we note that $S_0(b)+a+\gamma = S_0(b')+a'+2\gamma$. Let $X = S_0(b')+a'$. Then the left-hand side of the second equation in System (2) can be re-written as $S_1(X+2\gamma) - S_1(X+\gamma) + b - b'$. Furthermore, the second equation in System (1), which is assumed to hold, implies that $S_1(X+\gamma) - S_1(X) = b' - b$. Thus, the left-hand side of the second equation in System (2) is equal to

$$S_1(X+2\gamma) - (b'-b+S_1(X)) + b - b' \;=\; S_1(X+2\gamma) - S_1(X) - 2(b'-b).$$

The term $S_1(X+2\gamma) - S_1(X)$ has an unknown value unless $\gamma = 2^{n-1}$. Nevertheless, in this case, we would need $2(b'-b) = 0$ which does not have a probability equal to 1. However both $S_1(X+2\gamma) - S_1(X)$ and $2(b'-b)$ are always equal to 0 in characteristic 2 which is why our yoyo game can always be played against a $\oplus$-Feistel.

C Experimental Results About Cycles

C.1 Examples of Cycle Distribution

To illustrate the different types of cycles and their behaviour, we plotted the functionnal graphs of ϕ_γ and ψ_γ along with connection in γ for three different Feistel Networks with $n = 3$.

Permutations E_0 is built out of permutations, namely:

0. $S_0 =$ `[2, 0, 1, 3, 4, 7, 6, 5]`
1. $S_1 =$ `[5, 6, 2, 1, 7, 0, 3, 4]`
2. $S_2 =$ `[7, 2, 1, 3, 5, 6, 0, 4]`
3. $S_3 =$ `[4, 1, 6, 2, 3, 7, 0, 5]`
4. $S_4 =$ `[6, 3, 0, 5, 2, 1, 4, 7]`

It yields two Type-S cycles (see Figure 9a).

2 Collisions E_1 is built out of functions, namely:

0. $S_0 =$ `[2, 6, 1, 0, 6, 3, 4, 6]`
1. $S_1 =$ `[6, 6, 0, 6, 2, 5, 1, 2]`
2. $S_2 =$ `[7, 5, 1, 5, 1, 2, 5, 4]`
3. $S_3 =$ `[7, 1, 6, 2, 4, 3, 0, 1]`
4. $S_4 =$ `[4, 5, 2, 7, 6, 7, 7, 3]`

Its second Feistel function (S_1) has two collisions for an input difference of $\gamma = 1$ while S_3 has none. Thus, it yields $2 \cdot 2^{n-2} = 4$ Type-P cycles. It also yields two Type-D ones (see Figure 9b).

4 Collisions E_2 is built out of functions, namely:

0. $S_0 =$ `[3, 7, 1, 1, 1, 5, 6, 2]`
1. $S_1 =$ `[2, 2, 1, 5, 3, 2, 6, 0]`
2. $S_2 =$ `[1, 0, 5, 4, 4, 4, 6, 4]`
3. $S_3 =$ `[7, 3, 1, 1, 3, 4, 1, 7]`
4. $S_4 =$ `[0, 2, 1, 0, 5, 1, 3, 4]`

Both S_1 and S_3 have two collisions for an input difference of $\gamma = 1$. Thus, it yields $(2 + 2) \cdot 2^{n-2} = 8$ Type-P cycles (see Figure 9c).

C.2 Experimental Estimation of $q_S(n)$

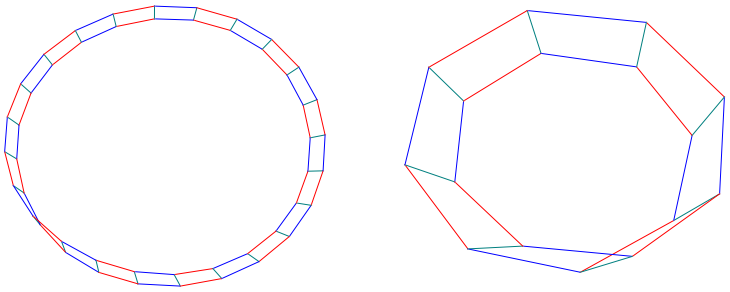
These values where obtained by computing the distribution of γ_{win} experimentally and then fitting the distribution obtained with an exponential decay using `qtiplot`. The value of $q_S(n)$ can then be estimated in two different ways using the following equivalent expression of the probability we experimentally estimated:

$$P[\gamma_{\text{win}} = x] = \frac{q_S(n)}{1 - q_S(n)} \cdot \exp\left(\frac{x}{1/\log(1 - q_S(n))}\right),$$

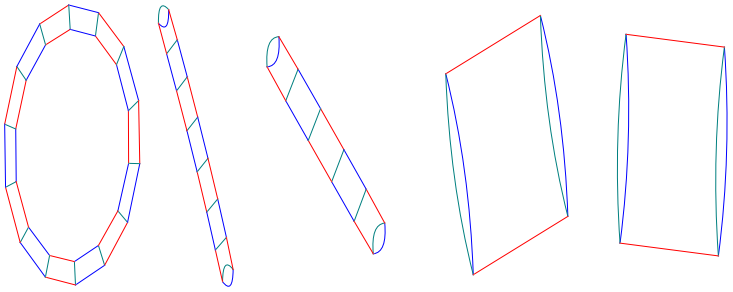
so that:

$$q_S(n) = \frac{Q}{Q+1} = 1 - \exp(-1/\tau).$$

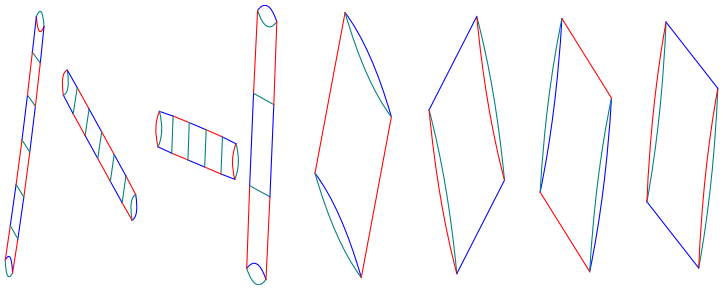
These two distinct estimations provide a sanity check. We used their average when plotting the values of $q_S(n)$ in Figure 5.



(a) E_0.



(b) E_1.



(c) E_2.

Fig. 9: Functionnal graphs of ϕ_γ and ψ_γ for different 5-round Feistel Networks. ϕ_γ is in blue, ψ_γ is red and connection in γ is green.

Table 2: Experimentally found expression of $q_S(n)$.

(a) Feistel functions are bijective.

n	Q	τ	$q_S(n)$ (from Q)	$q_S(n)$ (from τ)	$q_S(n)$ (avg.)
4	0.584	2.16	0.369	0.371	0.370
5	1.110	1.349	0.526	0.524	0.525
6	1.662	1.020	0.624	0.625	0.625
7	1.867	0.9558	0.651	0.649	0.650
8	2.833	0.7427	0.739	0.740	0.740
9	3.254	0.6833	0.765	0.769	0.767
10	3.880	0.6338	0.795	0.794	0.794
11	4.943	0.5679	0.832	0.828	0.830

(b) Feistel functions are not bijective.

n	Q	τ	$q_S(n)$ (from Q)	$q_S(n)$ (from τ)	$q_S(n)$ (avg.)
4	0.173	6.549	0.147	0.142	0.144
5	0.2687	4.094	0.212	0.217	0.214
6	0.310	3.746	0.237	0.234	0.235
7	0.3386	3.411	0.253	0.254	0.254
8	0.3837	2.969	0.277	0.286	0.281
9	0.4096	2.851	0.291	0.296	0.293
10	0.4065	2.964	0.289	0.286	0.287
11	0.4019	2.964	0.287	0.286	0.286