Analyses of Curupira Block Cipher

Jorge Nakahara Jr

[1]jorge_nakahara@yahoo.com.br

Abstract. *Curupira is a 96-bit block cipher, with keys of 96, 144 or 192 bits, and variable number of rounds, an algorithm described at SBRC 2007. This paper[1] presents impossible-differential, boomerang and higher-order multiset attacks on reduced-round versions of Curupira, that were not previously considered in the security analysis by its designers. Also, we provide security analyses based solely on the block size and on the key size of the cipher, such as plaintext leakage and related-key attacks. Our analyses indicate new attacks on up to 6-round Curupira.*

1. Introduction

Curupira is a block cipher designed by Barreto and Simplício and presented at SBRC 2007 [Barreto and Simplício Jr 2007]. This cipher has a Substitution Permutation Network (SPN) structure, like the AES [AES 1997] and is based on the wide-trail design strategy [Daemen 1995]. In [Barreto and Simplício Jr 2007], the designers of Curupira argued about its security under several attack settings. The best reported attack was the square or saturation attack [Daemen et al. 1997] on up to seven rounds of Curupira, with a complexity smaller than that of an exhaustive key search. In this paper, we describe impossible-differential (ID), boomerang and higher-order multiset attacks on reduced-round versions of Curupira. The motivation for the ID and boomerang attacks are the fact that these techniques exploit the cipher from both the encryption and decryption directions, namely, the corresponding distinguishers are based on differentials constructed from both ends of a given cipher. The end results are new attacks on 6-round Curupira.

This paper is organized as follows: Sect. 2. briefly describes the Curupira cipher; Sect. 3. describes higher-order multiset attacks on Curupira; Sect. 4. describes impossible-differential attacks; Sect. 5. describes boomerang attacks. Sect. 6. discusses plaintext leakage properties of Curupira; Sect. 7. related-key aspects of the cipher; Sect. 8. discusses related-cipher attacks; Sect. 9. concludes the paper.

2. The Curupira Block Cipher

Curupira is an iterated block cipher proposed in [Barreto and Simplício Jr 2007], by Barreto and Simplício, for confidentiality purposes in constrained environments, such as sensor networks. Curupira operates on 96-bit text blocks (plaintext and ciphertext), with keys of 96, 144 and 192 bits. The number of rounds is variable. For 96-bit keys, the number of rounds ranges from 10 (minimum) to 11 (maximum). For 144-bit keys, the cipher iterates from 14 up to 17 rounds. For 192-bit keys, the cipher iterates 18 up to 23 rounds.

Plaintext, ciphertext and subkeys are organized in a $3 \times N$ matrix of bytes, called state matrix, where N is the number of 24-bit words. For example, the state matrix for

[1]Research funded by FAPESP under contract 2005/02102-9.

the $3 * N$-byte data $A = (a_0, a_1, \ldots, a_{3N-1})$ is pictorially represented as

$$\text{State} = \begin{pmatrix} a_0 & a_3 & \ldots & a_{3N-3} \\ a_1 & a_4 & \ldots & a_{3N-2} \\ a_2 & a_5 & \ldots & a_{3N-1} \end{pmatrix}, \tag{1}$$

namely, with bytes inserted columnwise, in top-down order. All internal operations of Curupira are bytewise, with bytes represented as elements over GF(2^8) = GF(2)[x]/$p(x)$, where $p(x) = x^8 + x^6 + x^3 + x^2 + 1$ is a primitive polynomial of degree eight over GF(2).

Curupira follows the wide-trail design strategy [Daemen 1995] used in the AES [AES 1997].

One full round of Curupira consists of four transformations over M_n, the set of $3 \times n$ matrices over GF(2^8):

- σ_k: the bitwise exclusive-or of the k-th round subkey matrix with the intermediate cipher state;
- γ: the parallel application of a fixed, 8×8-bit S-box to each byte of the state. This S-box is the same one defined for the Khazad block cipher [Barreto and Rijmen 2000];
- π: a byte permutation represented by row rotations of the state matrix; let $a, b \in M_n$, then $\pi(a) = b \leftrightarrow b_{i,j} = a_{i,i\oplus j}$, where $0 \leq i < 3$ and $0 \leq j < n$;
- θ: it is a diffusion layer consisting of a linear transformation based on the [6,3,4] MDS code with generator matrix $G_d = [ID]$, where

$$D = \begin{pmatrix} 3 & 2 & 2 \\ 4 & 5 & 4 \\ 6 & 6 & 7 \end{pmatrix}. \tag{2}$$

This transformation has branch number four, following [NIST 2001].

Notice that each round transformation corresponds, roughly, to a quarter of a round (a fraction of 0.25). This feature will be used to measure the size of distinguishers in the attacks described further.

One full round of Curupira can be denoted $\eta_k(x) = \theta \circ \pi \circ \gamma \circ \sigma_k(x) = \theta(\pi(\gamma(\sigma_k(x))))$, $x \in M_n$. We assume the subkey numbering starts from k_0.

Notice that σ_k is the only key-dependent round transformation. Moreover, all four round transformations are involutions, that is $\sigma_k \circ \sigma_k(x) = x$; $\pi \circ \pi(x) = x$; $\theta \circ \theta(x) = x$ for all $x \in M_n$; and $\gamma \circ \gamma(y) = y$ for all $y \in$ GF(2^8).

Some of these round transformations commute, such as $\pi \circ \gamma = \gamma \circ \pi$, since both of them operate bytewise, one permuting bytes, while the other performing a non-linear byte transformation. Also, $\pi \circ \sigma_k = \sigma_k \circ \pi$, and $\sigma_k \circ \theta = \theta \circ \sigma_{k'}$, where $k' = \theta(k)$. Other pairs of round transformations do not commute in general. For instance, $\theta \circ \pi \neq \pi \circ \theta$, $\gamma \circ \sigma_k \neq \sigma_k \circ \gamma$, and $\theta \circ \gamma \neq \gamma \circ \theta$.

The attacks in this report do not depend on the particular secret key used, and thus, are independent on the key schedule algorithm.

The key schedule of Curupira allow on-the-fly round subkey generation for both encryption or decryption. Moreover, subkey generation is not one way, that is, from any (recovered) round subkey, one can reconstruct any following or previous subkeys, and even the original user key. This means that the entropy of each round subkey cannot be larger than 96 bits (the block size). Further details about the key schedule of Curupira can be found in [Barreto and Simplício Jr 2007].

3. Multiset Analyses

The technique used in a multiset attack [Biryukov and Shamir 2001] has similarities with the Square attack [Daemen et al. 1997], saturation attack [Lucks 2001] and integral cryptanalysis [Hu et al. 1999, Knudsen and Wagner 2002]. All of these techniques operate in a chosen-plaintext (CP) setting, and the first published one was a dedicated attack on the Square block cipher [Daemen et al. 1997]. A fundamental concept in a multiset attack is the Λ-set [Daemen et al. 1997], which is a *multiset* [Biryukov and Shamir 2001] (a set with multiplicities) containing b *full n-bit text blocks*, where n is the block size and b is typically a power of 2. These n-bit text blocks are analysed by tracing fixed (but not necessarily contiguous) w bits, $w < n$, over all the text blocks. For example,

$$\{(0|1|2|3),(1|2|2|1),(3|1|2|2),(2|2|2|1),(7|5|2|0),(4|5|2|7),(5|5|2|4),(6|5|2|4)\} \quad (3)$$

is a Λ-set with $b = 2^w = 8$ text blocks, each of which is 12 bits wide ($n = 12$). We further consider each of these 2^w text blocks as a concatenation of four w-bit words ($w = 3$), and thus, keep track of particular *patterns* in these w-bit words for the 2^w text blocks. These w bits are often composed of *contiguous* bits that respect word boundaries, hence the use of the term *word*. The patterns of interest are the following:

- if the w-bit word in a Λ-set assumes each of the values 0 to $2^w - 1$, then the word is called a *permutation* or an *active* word [Daemen et al. 1997], and is denoted ’A’. This is the case of the word formed by the first 3-bit word of each element in the multiset (3), which is $\{0, 1, 3, 2, 7, 4, 5, 6\}$;
- if the w-bit word assumes an arbitrary constant value, it is called *passive* [Daemen et al. 1997], and is denoted ’P’. This is the case of the third set of 3 bits in (3), which is $\{2, 2, 2, 2, 2, 2, 2, 2\}$;
- if the w-bit word contains an *even* number of repetitions of some arbitrary values (each element has even multiplicity), the word is called *even*, and is denoted ’E’. This is the case of the second set of 3 bits in (3), which is $\{1, 2, 1, 2, 5, 5, 5, 5\}$;
- if the sum of all w-bit values in a given Λ-set under some operator $\boxdot$, results in a *predictable* amount, then this word is called *balanced*, and is denoted ’B’; in Curupira, we use $\boxdot = \oplus$;
- otherwise, if the $\boxdot$-sum results in an unpredictable value, the word is called *unbalanced*, and is denoted ’?’. This is the case of the fourth set of 3 bits in (3), which is $\{3, 1, 2, 1, 0, 7, 4, 4\}$, with $\boxdot = \oplus$ i.e. exclusive-or.

The rationale behind the multiset technique is to use balanced multisets of w-bit words to attack permutation mappings. Thus, multiset attacks exploit the bijective nature of internal cipher components. In particular, ciphers that operate on neatly partitioned words are the main targets. A typical multiset attack starts with multisets in which all words are balanced (usually only ’A’ and ’P’) and the propagation of balanced words in the multisets across multiple rounds of a cipher is traced up to the point in which the multiset is composed only of unbalanced bits.

In [Barreto and Simplício Jr 2007], multiset attacks were presented based probably on a (1st-order) multiset distinguisher such as (4). To simplify the notation, we adopt σ instead of σ_k, since the behaviour of the distinguisher does not depend on the particular subkey values. Distinguisher (4) reaches 3.25 rounds, and could be used to simply distinguish 3.25-round Curupira from a random permutation. Additionally, it allows key-recovery attacks whose complexities are listed in Table 1.

$$\begin{pmatrix} A & P & P & P \\ P & P & P & P \\ P & P & P & P \end{pmatrix} \overset{\theta\circ\pi\circ\gamma\circ\sigma}{\longrightarrow} \begin{pmatrix} A & P & P & P \\ A & P & P & P \\ A & P & P & P \end{pmatrix} \overset{\pi\circ\gamma\circ\sigma}{\longrightarrow} \begin{pmatrix} A & P & P & P \\ P & A & P & P \\ P & P & A & P \end{pmatrix}$$
$$\overset{\theta}{\longrightarrow} \begin{pmatrix} A & A & A & P \\ A & A & A & P \\ A & A & A & P \end{pmatrix} \overset{\pi\circ\gamma\circ\sigma}{\longrightarrow} \begin{pmatrix} A & A & A & P \\ A & A & P & A \\ A & P & A & A \end{pmatrix} \overset{\sigma\circ\theta}{\longrightarrow} \begin{pmatrix} B & B & B & B \\ B & B & B & B \\ B & B & B & B \end{pmatrix} \qquad (4)$$

The distinguisher (4) shows that the encryption framework of Curupira takes at least three rounds to achieve complete diffusion (compared to two rounds for the AES [AES 1997]). This slower diffusion impacts the extent to which key-recovery attacks can be performed on the cipher.

3.1. Higher-Order Multiset Distinguisher

Using larger words, instead of bytes, we arrive at higher-order multiset distinguishers, such as (5), where 'A^*' stand for a 24-bit active word. Similarly, 'E^*' denote 24-bit even words, namely, all 24-bit values appear an even number of times.

$$\begin{pmatrix} A^* & P & P & P \\ P & A^* & P & P \\ P & P & A^* & P \end{pmatrix} \overset{\theta\circ\pi\circ\gamma\circ\sigma}{\longrightarrow} \begin{pmatrix} A^* & P & P & P \\ A^* & P & P & P \\ A^* & P & P & P \end{pmatrix} \overset{\pi\circ\gamma\circ\sigma}{\longrightarrow} \begin{pmatrix} E^* & P & P & P \\ P & E^* & P & P \\ P & P & E^* & P \end{pmatrix}$$
$$\overset{\theta}{\longrightarrow} \begin{pmatrix} E_1^* & E_2^* & E_3^* & P \\ E_1^* & E_2^* & E_3^* & P \\ E_1^* & E_2^* & E_3^* & P \end{pmatrix} \overset{\pi\circ\gamma\circ\sigma}{\longrightarrow} \begin{pmatrix} E_1^* & E_2^* & E_3^* & P \\ E_2^* & E_1^* & P & E_3^* \\ E_3^* & P & E_1^* & E_2^* \end{pmatrix} \overset{\theta}{\longrightarrow} \begin{pmatrix} A^* & E^* & E^* & E^* \\ A^* & E^* & E^* & E^* \\ A^* & E^* & E^* & E^* \end{pmatrix}$$
$$\overset{\pi\circ\gamma\circ\sigma}{\longrightarrow} \begin{pmatrix} A^* & E^* & E^* & E^* \\ E^* & A^* & E^* & E^* \\ E^* & E^* & A^* & E^* \end{pmatrix} \overset{\sigma\circ\theta}{\longrightarrow} \begin{pmatrix} B^* & B^* & B^* & B^* \\ B^* & B^* & B^* & B^* \\ B^* & B^* & B^* & B^* \end{pmatrix} \qquad (5)$$

The distinguisher (5) reaches 4.25 rounds, and it has been confirmed experimentally, for randomly selected user keys.

An attack on 5-round Curupira, using (5) proceeds as follows:

- create a pool of 2^{24} plaintexts $\{P_i\}$ in which byte positions 0, 4 and 8 of the state contain all 24-bit values exactly once, while the remaining bytes contain fixed, random values. This construction represents the 'A^*' word in (5);
- encrypt the pool $\{P_i\}$ across 5-round Curupira, obtaining the corresponding pool $\{C_i\}$, $0 \leq i < 2^{24}$;

- the last round after the distinguisher consists of the transformation $\sigma_{k_5} \circ \pi \circ \gamma \circ \sigma_{k_4}$. It does not contain θ, but even if θ was present, it could be undone, since $\theta \circ \sigma = \sigma' \circ \theta$ (Sect. 2.). Likewise, the last π layer can be undone, since it is key independent. Thus, the last round can be simplified, resulting in $\sigma'_{k_5} \circ \gamma \circ \sigma_{k_4}$, where σ'_{k_5} consists of a permuted version of σ_{k_5};
- guess σ'_{k_5} bytewise, apply $\gamma^{-1} = \gamma$, and check the exclusive-or over all 2^{24} ciphertexts in a pool. If the result is a balanced byte (the xor is zero), then the guessed σ'_{k_5} byte is a candidate for the correct value. The expected number of false positives is $2^8/2^{24} < 1$.

The attack complexity is $2^{24} \cdot 2^8 = 2^{32}$ 1-round partial decryptions per subkey byte guessed. For all twelve subkey bytes of σ'_{k_5} it means $12 \cdot 2^{32}/5 \approx 2^{33}$ 5-round computations. The data complexity is 2^{24} CP. The memory complexity is 2^{24} text blocks.

An attack on 6-round Curupira, using (5), recover subkeys from two rounds beyond the distinguisher:

- consider the last two rounds after the distinguisher, $\sigma_{k_6} \circ \pi \circ \gamma \circ \sigma_{k_5} \circ \theta \circ \pi \circ \gamma$. The last π layer can be undone. Moreover, σ_{k_5} can move across θ and π, resulting in $\sigma'_{k_6} \circ \gamma \circ \theta \circ \pi \circ \sigma'_{k_5} \circ \gamma$;
- guess three bytes of σ'_{k_6} and one byte of σ'_{k_5} in order to partially decrypt $\sigma'_{k_6} \circ \gamma \circ \theta \circ \pi \circ \sigma'_{k_5} \circ \gamma$ until arriving at one byte at the bottom end of (5);
- check if this byte is balanced (an 8-bit condition), then the guessed 32-bit subkey is a candidate to the correct value.

For each 32-bit subkey value, this procedure costs $2^{24} \cdot 2^{32} = 2^{56}$ 2-round partial decryptions, or about $2^{56} \cdot 4$ S-box computations. Since there are 12 S-boxes per round, it results in $2^{56} \cdot 4/(12 \cdot 6) \approx 2^{52}$ 6-round computations. There is a 2^{-8} chance of false positives for 2^{32} subkey values. After five pools, the number of false positives becomes $(2^{-8})^5 \cdot 2^{32} < 1$. The data complexity is $5 \cdot 2^{24}$ CP. The time complexity is $5 \cdot 2^{52} \approx 2^{54}$ 6-round computations. The memory complexity is 2^{24} text blocks.

4. Impossible-Differential Attack

Unlike the differential [Biham and Shamir 1991] and linear [Matsui 1994] techniques, which look for events such as text patterns or statistical correlations of high probability (or high bias), the Impossible Differential (ID) method looks for events that never happen. The ID technique is a chosen-plaintext (CP) attack formerly proposed by Knudsen in [Knudsen 1998b] against the DEAL block cipher, and further applied to Skipjack [Biham et al. 1998], IDEA [Biham et al. 1999], Khufu [Biham et al. 1999], AES [Biham and Keller 2000] and many other ciphers. ID distinguishers currently reported in the literature use the miss-in-the-middle technique [Biham et al. 1998] that requires two differentials (denoted ∇ and Δ) both holding with certainty (probability one). In ∇, the difference patterns propagate in the encryption direction. In Δ, the difference patterns propagate in the decryption direction. Both differentials are constructed such that the output difference pattern of ∇ is incompatible with the input difference pattern of Δ, in the sense that ∇ cannot cause Δ (and vice-versa). This contradiction explains the term miss-in-the-middle, and this incompatibility between the two differentials is denoted $\nabla \not\to \Delta$. Symmetrically, $\Delta \not\to \nabla$ (the latter works in a chosen-ciphertext (CC) setting).

In byte-oriented ciphers such as Curupira, it is typical to use truncated differentials [Knudsen and Berson 1996] to construct Δ and ∇, because truncated difference patterns hold with certainty. Moreover, they are also independent of the particular S-boxes used in the cipher. In truncated differentials, one only distinguishes between zero and nonzero differences, namely, the exact value of the nonzero difference is irrelevant. For bytewise difference patterns, as in the AES, a nonzero byte difference will be denoted 'δ'. In contrast, a zero byte difference will be denoted simply '0'. Notice that although 'δ' is used throughout the distinguisher, it does not mean that all these bytes contain the same difference value. It only means that the difference value in nonzero. The difference operator used for Curupira is exclusive-or.

Differential [Biham and Shamir 1991] and linear [Matsui 1994] distinguishers (among others) recognize the correct key by comparing difference patterns or linear relations that most often satisfy the distinguishers. ID distinguishers operate the other way around. The keys that actually satisfy the ID distinguisher are wrong values, and the (single) key value not suggested by the distinguisher is the correct one.

For Curupira we have found a 5-round ID distinguisher, depicted in (6). The symbol '$?$' denotes either a zero or nonzero byte difference (its exact status is unknown). Arrows indicate the direction of propagation of differences. Recall that all four round transformations are involutions (thus, for instance, $\theta^{-1} = \theta$). To avoid clutter, we simplify σ_k as σ.

Notice that before the third θ layer in (6), the rightmost column of the state contains differences $(\delta, \delta, ?)$. But after this θ layer, the state contains the differences $(0, 0, 0)$. This situation contradicts the fact that θ has branch number four, because the sum of nonzero byte differences before and after this θ layer is at most three. Whether the value of '?' is zero or nonzero, the branch number is smaller than four. Thus, (6) holds with probability zero.

Notice that the output difference of (6) is satisfied with probability $(2^{-8})^9 = 2^{-72}$ by a random permutation.

$$\begin{pmatrix} \delta & \delta & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \xrightarrow{\theta\circ\pi\circ\gamma\circ\sigma} \begin{pmatrix} \delta & \delta & 0 & 0 \\ \delta & \delta & 0 & 0 \\ \delta & \delta & 0 & 0 \end{pmatrix} \xrightarrow{\pi\circ\gamma\circ\sigma} \begin{pmatrix} \delta & \delta & 0 & 0 \\ \delta & \delta & 0 & 0 \\ 0 & 0 & \delta & \delta \end{pmatrix} \xrightarrow{\theta}$$
$$\begin{pmatrix} ? & ? & \delta & \delta \\ ? & ? & \delta & \delta \\ ? & ? & \delta & \delta \end{pmatrix} \xrightarrow{\pi\circ\gamma\circ\sigma} \begin{pmatrix} ? & ? & \delta & \delta \\ ? & ? & \delta & \delta \\ \delta & \delta & ? & ? \end{pmatrix} \overset{\theta}{\not\rightarrow} \begin{pmatrix} ? & 0 & 0 & 0 \\ 0 & ? & 0 & 0 \\ 0 & 0 & ? & 0 \end{pmatrix}$$
$$\xleftarrow{\pi\circ\gamma\circ\sigma} \begin{pmatrix} ? & 0 & 0 & 0 \\ ? & 0 & 0 & 0 \\ ? & 0 & 0 & 0 \end{pmatrix} \xleftarrow{\theta} \begin{pmatrix} \delta & 0 & 0 & 0 \\ \delta & 0 & 0 & 0 \\ \delta & 0 & 0 & 0 \end{pmatrix} \xleftarrow{\sigma\circ\pi\circ\gamma\circ\sigma} \begin{pmatrix} \delta & 0 & 0 & 0 \\ 0 & \delta & 0 & 0 \\ 0 & 0 & \delta & 0 \end{pmatrix} \qquad (6)$$

A key-recovery ID attack on 6-round Curupira using (6), discovers the first round subkey (the distinguisher is placed in the last 5 rounds), and works as follows:

(i) create a poll of 2^{48} plaintexts $\{P_i\}$ with all possible values in positions 0, 1 and 2 of the state, and fixed values in the other positions;

(ii) encrypt this pool across 6-round Curupira, and obtain a corresponding ciphertext pool $\{C_i\}$, $0 \leq i < 2^{48}$;

(iii) form about $2^{48}(2^{48} - 1)/2 \approx 2^{95}$ pairs $C_i \oplus C_j$, with $i \neq j$, and check whether byte positions 1, 2, 3, 5, 6, 7, 9, 10, 11 of the ciphertext state contain zero byte difference; if so, guess 48 bits of σ_{k_0} corresponding to positions 0, 1, 3, 4, 8, 11, and partially decrypt the first round of (P_i, P_j), up to the leftmost two columns of the state. If there is only a single nonzero byte difference after θ in each of these columns, then the guessed 48-bit subkey is wrong, because it satisfies the input difference to the ID distinguisher (6);

(iv) output the single 48-bit subkey, not eliminated by the filtering in (iii).

Step (i) creates a pool of plaintexts that lead to nonzero byte differences only in the two leftmost columns of the state after one round. Step (ii) creates the corresponding ciphertext pool, from which pairs will be selected in step (iii) that satisfy the output difference of (6). Moreover, step (iii) also filters wrong subkey candidates that lead to the input difference of (6). Such wrong subkeys must be discarded. The forbidden output difference for $C_i \oplus C_j$ contain nonzero byte differences in byte positions $(1, 2, 3, 5, 6, 7, 9, 10, 11)$, or $(0, 2, 4, 5, 6, 7, 8, 9, 10)$, or $(0, 1, 3, 4, 5, 7, 8, 9, 11)$ or $(0, 1, 2, 3, 4, 6, 8, 10, 11)$ (recall the byte numbering in 1). The expected number of ciphertext pairs that satisfy the output difference pattern in (6) is $2^{95}/(4 \cdot (2^8)^9) = 2^{21}$ since there are four difference patterns with nine zero byte differences in each. So, we expect to test about 2^{21} pairs in step (iii). Each 48-bit subkey candidate that satisfies (iii) lead to the two leftmost columns of the intermediate state with a single δ in each. Thus, one expects that $2^{48}/2^{32} = 2^{16}$ wrong subkeys are suggested per pair. After four plaintext pools are processed, the expected number of wrong subkeys remaining is $2^{48}(1 - 2^{-16})^{2^{23}} \approx 2^{48}(e^{-1})^{2^7} = 2^{48}/e^{128} <<< 1$, so, it is expected that only the correct subkey value remains.

In step (ii) each right pair is partially decrypted for 2^{48} subkeys. This is equivalent to $2^{48} \cdot 2^{23} = 2^{71}$ 1-round decryptions, or $2^{71}/6$ 6-round computations. This procedure must be repeated twice to recover the full σ_{k_0}. Thus, the effort is $2 \cdot 2^{71}/6 \approx 2^{69.5}$ 6-round computations. The data complexity is $2 \cdot 4 \cdot 2^{48} = 2^{51}$ chosen plaintexts (CP). The memory required is about 2^{23} blocks to store the right pairs, and 2^{48} bits (or $2^{41.5}$ blocks) to store the subkeys candidates.

5. Boomerang Analysis

The boomerang technique is a chosen-plaintext adaptively chosen-ciphertext (CPACC) attack, developed by Wagner [Wagner 1999]. The boomerang technique exploits encryption schemes parameterized by a secret key K, denoted E_K, and which can be decomposed into $E_K = E_1 \circ E_0$. Moreover, it is assumed that there is no known high probability differential covering E_K, but E_0 is a weak transformation in the encryption direction, and E_1 is weak in the decryption direction (that is the reason for the CPACC setting). In this particular context, the term weak means that a truncated differential propagates across both E_0 and E_1^{-1} with a relatively high probability. Notice that the probabilities of propagation of truncated differentials depend on the direction of propagation of the differences (encryption or decryption direction). The difference operator is bitwise exclusive-or.

A boomerang attack on 5-round Curupira is depicted in Fig. 1, and works as follows:

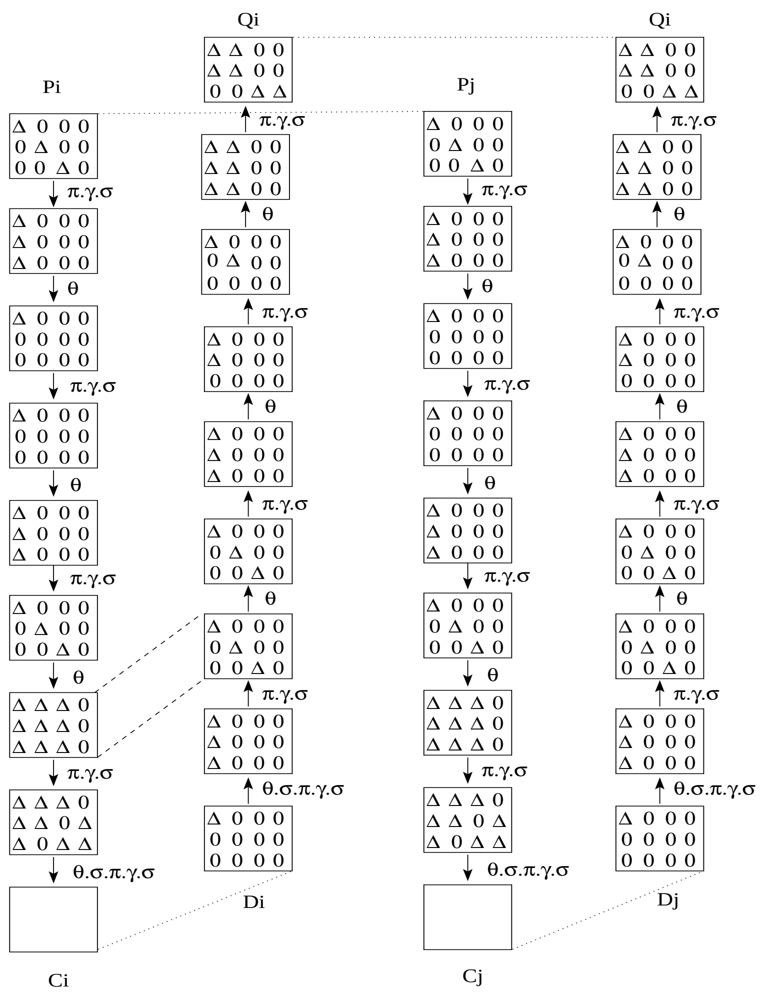

Figure 1. 5-round boomerang distinguisher for Curupira.

(1) consider a pool $\{P_i\}$ of 2^{24} blocks with all possible 24-bit values in byte positions 0, 4, 8 of the plaintext state, while the remaining byte positions contain fixed, arbitrary values;
(2) encrypt the pool $\{P_i\}$ across 5-round Curupira, and obtain the corresponding ciphertext pool $\{C_i\}$, $0 \leq i < 2^{24}$;
(3) construct a pool of ciphertexts $\{D_i\} = \{C_i \oplus \nabla\}$, where ∇ is a fixed nonzero difference with only one nonzero byte difference (Fig. 1); decrypt the pool $\{D_i\}$ to obtain a plaintext pool $\{Q_i\}$. Sort $\{Q_i\}$ by the bytes corresponding to nine zero byte differences. Pick only those pairs $Q_i \oplus Q_j$, $i \neq j$, which have zero difference in these nine bytes (Fig. 1); if none found, then go back to step (1);
(4) for each quartet (P_i, P_j, Q_i, Q_j) that satisfy step (3) guess the 24-bit subkey value that are xored to the three Δ byte differences at σ_{k_0};
(5) using the guessed 24-bit subkey, partially encrypt one round and check that the resulting difference has a single Δ byte. This is a 16-bit condition for each pair (P_i, P_j) and (Q_i, Q_j). This gives a 32-bit condition for both sides of the boomerang in the case of common 3-tuples of active S-boxes. But, with probability half, there will be no common 3-tuples. We will then pick subkey candidates that are suggested most often.

The pool $\{P_i\}$ provides about $2^{24}(2^{24} - 1)/2 \approx 2^{47}$ text pairs with nonzero difference in byte positions 0, 4, 8. After one round, about $2^{47}/2^{16} = 2^{31}$ pairs lead to a state with a single nonzero byte difference in the leftmost column of the state. After the second round, there will be three nonzero byte differences in this leftmost column. After the third round, nine nonzero byte differences will be nonzero (Fig. 1)

From the bottom-up direction, we will have one round crossed with probability one, with a truncated differential that starts with a single nonzero byte difference, and ends with three nonzero byte differences. At this point, we need that after the next γ layer, the difference in these three bytes be the same. This happens with probability 2^{-24}. Then, we switch to the last face of the boomerang where the effect of the mixing of the third round can be undone with certainty, and we will get three nonzero byte differences after γ of the third round. There is a probability of $3 \cdot 2^{-8}$ for three nonzero byte differences to turn into two nonzero byte differences after γ and θ. From that point, the truncated differential propagates freely with probability one. As a result we obtain a new pair of plaintexts with nonzero difference in six bytes and zero difference in the remaining six bytes. This is a $(48 - \log 3)$-bit filtering condition. The $-\log 3$ appears since the position of the two nonzero byte differences are not fixed. From 2 pools, we will have $2^{31} \cdot 2 \cdot 3 \cdot 2^{-8} \cdot 2^{-24} = 3$ good boomerangs returning. The average amount of false quartets which satisfy our initial filtering condition is $2^{47} \cdot 2 \cdot 2^{-48} \cdot 3 = 3$. In the simplest case, when the boomerang returns in the same three bytes as it was sent, we guess the 24 bits of the first subkey and check it against the two sides (P_i, P_j) and (Q_i, Q_j), whether in both cases, it leads to a single nonzero byte difference after one round. This gives a 32-bit filtering condition which leaves only the correct key guess with probability $1 - 2^{-8}$. However, with probability $1/2$, it may happen that for the two boomerang pairs, active 3-tuples of the output pair (Q_i, Q_j) will be different from those of the input pair (P_i, P_j). In this case, we independently guess 24 bits of the subkey corresponding to the input 3 bytes for each pair, and leave only those subkey values that lead to a single nonzero byte difference after the first round. To recover the other half of the first subkey, just repeat the attack by chaning the nonzero byte difference in Fig. 1 appropriately. The attack complexity is $2 \cdot 2 \cdot 2^{24} = 2^{26}$ chosen-plaintexts adaptively-chosen ciphertexts (CPACC), and 2^{26} 1-round computations, or $2^{26}/5 \approx 2^{23}$ 5-round computations. The memory required is 2^{24} blocks.

An attack on 6-round Curupira works similarly to the one on 5 rounds, but we guess subkeys at both ends of the distinguisher Fig. 1. We guess additionally, 24 bits of the 6th-round subkey. We double the number of pools from 2 to 4, to get 4–6 good quartets for better filtration. We expect at least 2 to 3 good quartets will have overlapping 3-tuples between the P_i's and Q_i's, which provide 2^{-8} filtration power. Thus, we will get about $2^{24} \cdot 2^{-8} = 2^{16}$ candidates for 48-bit subkeys: 24 bits at the top and 24 bits at the bottom. The correct subkey will be suggested at least twice, and the wrong subkeys would likely be suggested only once. Thus, we expect that all the wrong pairs will be filtered at the key recovery step. The attack complexity is 2^{26} CP, $4 \cdot 2^{24} \cdot 2^{24} = 2^{50}$ ACC and 2^{50} 1-round computations, or $2^{50}/6 \approx 2^{47.5}$ 6-round computations, and 2^{24} blocks of memory.

6. Plaintext Leakage

Due to the birthday paradox [Menezes et al. 1997], after about $2^{n/2}$ encryptions, either in ECB or CBC modes, an n-bit block cipher starts to leak information about the plaintext [Knudsen 1998a], in a ciphertext-only (CO) setting. For Curupira, this leakage happens after $2^{96/2} = 2^{48}$ block encryptions (or decryptions), which sets an upperbound on the number of plaintext blocks encrypted before the key has to be changed. Modern ciphers, such as the AES [AES 1997], already use 128-bit blocks to counter this drawback, which depends only on the block size, namely, is independent of the cipher structure, the number of rounds and internal cipher components. For Curupira, this vulnerability depends on the application allowing or not 2^{48} block encryptions (under the same key).

7. Related-Key Attack

In [Biham 2002], Biham developed an attack method on arbitrary block ciphers, that depends only on the key size (namely, it is independent of the number of rounds). His attack is supported by the birthday paradox, and has complexity $2^{k/2}$ encryptions for a k-bit user key. For Curupira, the attack complexities for the different key sizes are $2^{96/2} = 2^{48}$; $2^{144/2} = 2^{72}$ and $2^{192/2} = 2^{96}$ encryptions, which set a lower upperbound on the number of text blocks than the complexity of an exhaustive key search. For Curupira, this attack may be effective or not depending on the application allowing $2^{k/2}$ encryptions (of the same block) under different keys.

Concerning related-key attacks, notice that the operations in the key schedule of Curupira are all linear. Even the non-linear S-box, that is applied to only a few bytes of each subkey, can be undone since the S-box is invertible. These facts motivate an analysis of the difference propagation across the key schedule, meaning how many subkeys it takes until all subkey bytes depend on all user key bytes. This analysis involves tracing the (xor) difference propagation across the key schedule (just like in the encryption framework) and is left as an open problem.

8. Related-Cipher Attack

In [Wu 2002], Wu described an attack on block ciphers with a variable number of rounds, or in which the number of rounds could be somehow manipulated by an

Table 1. Attack complexities on (reduced-round) Curupira.

#Rounds	Time	Data	Memory	Attack	Source
4	2^9	2^9 CP	2^8	multiset	[Barreto and Simplício Jr 2007]
5	2^{23}	2^{26} CPACC	2^{24}	boomerang	Sect. 5.
5	2^{33}	2^{24} CP	2^{24}	multiset	Sect. 3.1.
5	2^{35}	2^{11} CP	2^8	multiset	[Barreto and Simplício Jr 2007]
6	2^{38}	2^{27} CP	2^8	multiset	[Barreto and Simplício Jr 2007]
6	$2^{47.5}$	2^{50} CPACC	2^{24}	boomerang	Sect. 5.
6	2^{54}	2^{26} CP	2^{24}	multiset	Sect. 3.1.
6	$2^{69.5}$	2^{51} CP	$2^{41.5}$	imp.diff.	Sect. 4.
7	2^{88}	$2^{96} - 2^{87}$ CP		multiset	[Barreto and Simplício Jr 2007]
7	2^{104}	2^{32} CP	2^{32}	Gilbert-Minier	[Barreto and Simplício Jr 2007]

CP: Chosen Plaintext; CPACC: Chosen-Plaintext Adaptively Chosen-Ciphertext

adversary. The rationale is similar to a slide attack [Biryukov and Wagner 2000, Biryukov and Wagner 1999]. A suggested countermeasure is to embed the number of rounds in the cipher, for instance, in the key schedule, so that cipher instances with different number of rounds are not useful for this attack. Curupira could become vulnerable to this attack, since its key schedule does not depend on the number of rounds. Nonetheless, this vulnerability can only be exploited depending on the application environment.

9. Conclusion and Open Problems

This paper discussed impossible-differential, boomerang and higher-order multiset, plaintext leakage and related-key attacks on the Curupira cipher. These attacks have not being discussed before, not even by the cipher designers. Table 1 summarizes the attack complexities and compares them with other known attacks. The results in this paper do not threaten the full Curupira cipher, for any of the defined key sizes, but complement the analyses provided by its designers.

The fact that all round components in Curupira are involutions, just like in Khazad [Barreto and Rijmen 2000], raises the question of attacks based on symmetries of the computational framework, such as [Biryukov 2003]. For instance, consider 4-round Curupira, and group the round transformations as follows

$$\theta \circ \pi \circ \gamma \circ \sigma_{k_3} \circ \theta \circ \pi \circ \gamma \circ \sigma_{k_2} \circ \theta \circ \pi \circ \gamma \circ \sigma_{k_1} \circ \theta \circ \pi \circ \gamma \circ \sigma_{k_0} =$$
$$(\theta \circ \pi \circ \gamma \circ \theta \circ \pi) \circ (\sigma_{k'_3} \circ \sigma_{k_2}) \circ (\theta \circ \pi \circ \gamma \circ \theta \circ \pi) \circ (\sigma_{k'_1} \circ \sigma_{k_0}),$$

where $\sigma_{k'_3} = \theta \circ \pi(k_3)$, that is, we transformed k_3 through θ and π in order to move it through these two transformations. Notice, the key-independent transformation $\tau = \theta \circ \pi \circ \gamma \circ \theta \circ \pi$, interleaved with $\sigma_{k'} \circ \sigma_k$. This is the same phenomenon observed by Biryukov in [Barreto and Rijmen 2000]. This symmetry shall be analysed together with the key schedule of Curupira, and is left as an open problem.

References

AES (1997). The advanced encryption standard development process. http://csrc.nist.gov/encryption/aes/.

Barreto, P. and Rijmen, V. (2000). The KHAZAD Legacy-Level Block Cipher. 1st NESSIE Workshop, Heverlee, Belgium. http://cryptonessie.org.

Barreto, P. and Simplício Jr, M. (2007). Curupira, a block cipher for constrained platforms. 25th Brazilian Symposium on Computer Networks and Distributed Systems (SBRC).

Biham, E. (2002). How to decrypt or even substitute DES-encrypted messages in 2^{28} steps. *Information Processing Letters*, 84(3):117–124.

Biham, E., Biryukov, A., and Shamir, A. (1998). Cryptanalysis of Skipjack Reduced to 31 Rounds using Impossible Differentials. Tech Report CS0947 revised, Technion, CS Dept.

Biham, E., Biryukov, A., and Shamir, A. (1999). Miss-in-the-Middle Attacks on IDEA, Khufu and Khafre. In Knudsen, L., editor, *6th Fast Software Encryption Workshop*, LNCS 1636, pages 124–138. Springer-Verlag.

Biham, E. and Keller, N. (2000). Cryptanalysis of Reduced Variants of Rijndael. 3rd AES Conference, New York, USA. http://csrc.nist.gov/encryption/aes/round2/conf3/aes3papers.html.

Biham, E. and Shamir, A. (1991). Differential Cryptanalysis of DES-like Cryptosystems. *Journal of Cryptology*, 4(1):3–72.

Biryukov, A. (2003). Analysis of involutional ciphers: Khazad and anubis. In Johansson, T., editor, *10th Fast Software Encryption Workshop*, LNCS 2887. Springer-Verlag.

Biryukov, A. and Shamir, A. (2001). Structural Cryptanalysis of SASAS. In Pfitzmann, B., editor, *Advances in Cryptology, Eurocrypt'01*, LNCS 2045, pages 394–405. Springer-Verlag.

Biryukov, A. and Wagner, D. (1999). Slide Attacks. In Knudsen, L., editor, *6th Fast Software Encryption Workshop*, LNCS 1636, pages 245–259. Springer-Verlag.

Biryukov, A. and Wagner, D. (2000). Advanced Slide Attacks. In Preneel, B., editor, *Advances in Cryptology, Eurocrypt'00*, LNCS 1807, pages 589–606. Springer-Verlag.

Daemen, J. (1995). *Cipher and Hash Function Design – Strategies based on Linear and Differential Cryptanalysis*. PhD thesis, Dept. Elektrotechniek, Katholieke Universiteit Leuven, Belgium.

Daemen, J., Knudsen, L., and Rijmen, V. (1997). The Block Cipher SQUARE. In Biham, E., editor, *4th Fast Software Encryption Workshop*, LNCS 1267, pages 149–165. Springer-Verlag.

Hu, Y., Zhang, Y., and Xiao, G. (1999). Integral cryptanalysis of safer+. *Electronic Letters*, 35(17):1458–1459.

Knudsen, L. (1998a). Block Ciphers – a Survey. In Preneel, B. and Rijmen, V., editors, *State of the Art in Applied Cryptography*, LNCS 1528, pages 18–48. Springer-Verlag.

Knudsen, L. (1998b). Deal – a 128-bit block cipher. Tech Report #151, University of Bergen, Dept. of Informatics, Norway.

Knudsen, L. and Berson, T. (1996). Truncated Differentials of SAFER. In Gollmann, D., editor, *3rd Fast Software Encryption Workshop*, LNCS 1039, pages 15–26. Springer-Verlag.

Knudsen, L. and Wagner, D. (2002). Integral cryptanalysis. In Daemen, J. and Rijmen, V., editors, *9th Fast Software Encryption Workshop*, LNCS 2365, pages 112–127. Springer-Verlag.

Lucks, S. (2001). The saturation attack – a bait for twofish. In Matsui, M., editor, *8th Fast Software Encryption Workshop*, LNCS 2355, pages 1–15. Springer-Verlag.

Matsui, M. (1994). Linear Cryptanalysis Method for DES Cipher. In Helleseth, T., editor, *Advances in Cryptology, Eurocrypt'93*, LNCS 765, pages 386–397. Springer-Verlag.

Menezes, A., van Oorschot, P., and Vanstone, S. (1997). *Handbook of Applied Cryptography*. CRC Press.

NIST (2001). Advanced Encryption Standard (AES). FIPS PUB 197 Federal Information Processing Standard Publication 197, U.S. Department of Commerce.

Wagner, D. (1999). The Boomerang Attack. In Knudsen, L., editor, *6th Fast Software Encryption Workshop*, LNCS 1636, pages 156–170. Springer-Verlag.

Wu, H. (2002). Related-cipher attacks. In Deng, R., editor, *ICICS 2002*, LNCS 2513, pages 447–455. Springer-Verlag.

A Appendix A

We have discovered 2-round iterative truncated differentials of Curupira. One of these differentials is described in (7), and it holds with probability[2] 2^{-64}. Nonetheless, a random permutation satisfies the output difference of (7) with this same probability. In (7), the symbol 'δ' denotes an arbitrary nonzero xor difference byte, and '0' a zero xor difference byte. We use simply σ without distinguishing any particular round subkey, since its exact value is not relevant for this analysis.

$$\begin{pmatrix} \delta & \delta & \delta & \delta \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \overset{\theta\circ\pi\circ\gamma\circ\sigma}{\longrightarrow} \begin{pmatrix} \delta & \delta & \delta & \delta \\ \delta & \delta & \delta & \delta \\ \delta & \delta & \delta & \delta \end{pmatrix} \overset{\theta\circ\pi\circ\gamma\circ\sigma}{\longrightarrow} \begin{pmatrix} \delta & \delta & \delta & \delta \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{7}$$

There are analogous iterative truncated differentials with nonzero byte differences in the second and third rows of the state.

A dual differential to (7) is (8), with the same probability of 2^{-64}.

$$\begin{pmatrix} \delta & \delta & \delta & \delta \\ \delta & \delta & \delta & \delta \\ \delta & \delta & \delta & \delta \end{pmatrix} \overset{\theta\circ\pi\circ\gamma\circ\sigma}{\longrightarrow} \begin{pmatrix} \delta & \delta & \delta & \delta \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \overset{\theta\circ\pi\circ\gamma\circ\sigma}{\longrightarrow} \begin{pmatrix} \delta & \delta & \delta & \delta \\ \delta & \delta & \delta & \delta \\ \delta & \delta & \delta & \delta \end{pmatrix} \tag{8}$$

[2]The value 2^{-64} is due to $(2^8/2^{24})^4 = (2^{-16})^4$ since it is the probability that three nonzero bytes in a column of the state before θ turn into a single nonzero byte (in a specific position).

We corroborate that there are at least sixteen active S-boxes in differential characteristics across four rounds of Curupira. An example is depicted in (9).

$$\begin{pmatrix} \delta & 0 & 0 & 0 \\ 0 & \delta & 0 & 0 \\ 0 & 0 & \delta & 0 \end{pmatrix} \overset{\pi\circ\gamma\circ\sigma}{\longrightarrow} \begin{pmatrix} \delta & 0 & 0 & 0 \\ \delta & 0 & 0 & 0 \\ \delta & 0 & 0 & 0 \end{pmatrix} \overset{\pi\circ\gamma\circ\sigma\circ\theta}{\longrightarrow} \begin{pmatrix} \delta & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \overset{\gamma\circ\sigma\circ\theta}{\longrightarrow}$$
$$\begin{pmatrix} \delta & 0 & 0 & 0 \\ \delta & 0 & 0 & 0 \\ \delta & 0 & 0 & 0 \end{pmatrix} \overset{\pi}{\longrightarrow} \begin{pmatrix} \delta & 0 & 0 & 0 \\ 0 & \delta & 0 & 0 \\ 0 & 0 & \delta & 0 \end{pmatrix} \overset{\gamma\circ\sigma\circ\theta}{\longrightarrow} \begin{pmatrix} \delta & \delta & \delta & 0 \\ \delta & \delta & \delta & 0 \\ \delta & \delta & \delta & 0 \end{pmatrix} \overset{\sigma\circ\pi}{\longrightarrow} \begin{pmatrix} \delta & \delta & \delta & 0 \\ \delta & \delta & 0 & \delta \\ \delta & 0 & \delta & \delta \end{pmatrix} \quad (9)$$