Curupira, a block cipher for constrained platforms

Paulo S. L. M. Barreto[1]**, Marcos A. Simplício Jr.**[1]

[1] Departamento de Engenharia de Computação e Sistemas Digitais (PCS),
Escola Politécnica, Universidade de São Paulo, Brazil.

`pbarreto@larc.usp.br, marcos.simplicio@poli.usp.br`

Abstract. *We present* Curupira, *a special-purpose block cipher tailored for platforms where power consumption and processing time are very constrained resources, like sensor and mobile networks or systems heavily dependent on tokens or smart cards.* Curupira *is an instance of the Wide Trail family of algorithms which includes the AES cipher, and displays both* involutional structure, *in the sense that the encryption and decryption modes differ only in the key schedule, and* cyclic key schedule, *whereby the round subkeys can be computed in-place in any order.*

1. Introduction

Battery-powered sensor networks impose several constraints on the cryptographic algorithms that can be effectively deployed for such systems. Processing speed is typically low compared to general processors to save batteries; complex all-purpose algorithms will not only take longer to run but also consume more energy. On the other hand, messages exchanged between sensors and central servers or among the sensors themselves are usually small, a typical packet being 30 bytes in length. *Ad-hoc* networks involving mobile equipment (and often tokens and smart cards as well) also impose restrictions on code size and bandwidth occupation.

The TinySec [Karlof et al. 2004] link layer architecture was designed to provide security for sensor networks and is gaining wide acceptance in this role. TinySec relies on a block cipher to achieve its goals regarding access control, message integrity, and confidentiality adopting the Skipjack [NSA 1998] cipher by default. Several alternatives have been analyzed by [Law et al. 2006], confirming that Skipjack is the most energy-efficient of all surveyed ciphers and recommending that MISTY1 [Matsui 1997] or the Advanced Encryption Standard (AES) [NIST 2001] might used in scenarios with higher security requirements. However, Skipjack uses relatively small (80-bit) keys and has a very low margin of security [Biham et al. 1999]; MISTY1 is encumbered by patents, and the AES has larger code, memory, and energy consumption requirements.

In this paper we address these issues by presenting the special-purpose block cipher Curupira. It operates on a 96-bit data block and accepts keys of size 96, 144, or 192 bits, with a variable number of rounds. The cipher design follows the Wide Trail strategy [Daemen 1995]. The most well-known member of the Wide Trail family of ciphers is the AES itself. Curupira offers a variety of implementation trade-offs according to the resources available on the target platform. In face of the constrained environment to which it is targeted, Curupira was designed to exhibit *involutional structure* and *cyclic key schedule*.

Involutional structure is found in many cipher designs. All classical Feistel networks [Feistel 1973] have this property, as do some more general block ciphers like Skipjack and MISTY1 (but not the AES). Involutional ciphers similar to CURUPIRA were described and analyzed in [Barreto and Rijmen 2000a, Barreto and Rijmen 2000b, Daemen et al. 2000, Youssef et al. 1996, Youssef et al. 1997]. The importance of involutional structure resides in the equivalent security of both encryption and decryption and in the advantages for implementation (the latter being critical in the case of TinySec, due to its adoption of CBC mode). The usual way to make a cipher of the Wide Trail family involutional is by including matrix transposition (restricting the block size to perfect squares) or multiplication by a large unitary matrix (incurring large overheads) as components in the round function; the solution adopted in CURUPIRA is novel (see section 3.2.), and avoids those drawbacks. CURUPIRA is closely related to BKSQ [Daemen and Rijmen 1998] (another member of the Wide Trail family), designed with smart cards in mind but lacking involutional structure.

The key schedule adopted by a block cipher is called cyclic or periodic if (1) it iterates some function on the cipher key to derive the round subkeys, and (2) that function becomes the identity after a certain number of iterations. It is important to notice that, with this setting, the subkeys need not be computed incrementally: they can be computed backwards as often needed by the decryption process, or in a random order, or even all in parallel, a useful feature if the cipher is implemented in dedicated hardware, for instance to be deployed on servers. They can also be computed in-place, contrary to schemes where all subkeys must be precomputed and stored in a large table.

As is *de rigueur* for any encryption algorithm proposal, we focus our attention on a detailed security analysis, without losing sight of actual implementation issues.

This document is organized as follows. We introduce basic mathematical tools in section 2. The CURUPIRA cipher is described in section 3. Security issues of the resulting design are discussed in section 4. Implementation techniques together with efficiency considerations and comparisons are presented in section 5. We conclude in section 6.

2. Mathematical preliminaries and notation

2.1. Finite fields

The finite field $GF(2^8)$ will be represented as $GF(2)[x]/p(x)$, where $p(x) = x^8 + x^6 + x^3 + x^2 + 1$ is the only primitive pentanomial of degree 8 over $GF(2)$ for which a primitive cube root of unity is represented as a quartic trinomial (namely, $c(x) = x^{85} \bmod p(x) = x^4 + x^3 + x^2$), which is the simplest form achievable. Since multiplications by the generator $g(x) = x$ of $GF^*(2^8)$ and by $c(x)$ both occur in the algorithm, it is important that $c(x)$ be as simple as possible for efficiency reasons.

An element $u = u_7 x^7 + u_6 x^6 + u_5 x^5 + u_4 x^4 + u_3 x^3 + u_2 x^2 + u_1 x + u_0$ of $GF(2^8)$ where $u_i \in GF(2)$ for all $i = 0, \ldots, 7$ will be denoted by the numerical value $u_7 \cdot 2^7 + u_6 \cdot 2^6 + u_5 \cdot 2^5 + u_4 \cdot 2^4 + u_3 \cdot 2^3 + u_2 \cdot 2^2 + u_1 \cdot 2 + u_0$, written in hexadecimal notation. For instance, the polynomial $c(x) = x^4 + x^3 + x^2$ is written 1C; by extension, the reduction polynomial $p(x)$ is written 14D.

2.2. MDS codes

We provide a few relevant definitions regarding the theory of linear codes. For a more extensive exposition on the subject we refer to [MacWilliams and Sloane 1977].

The Hamming distance between two vectors u and v from the n-dimensional vector space $GF(2^p)^n$ is the number of coordinates where u and v differ. The Hamming weight $w_h(a)$ of an element $a \in GF(2^p)^n$ is the Hamming distance between a and the null vector of $GF(2^p)^n$, i.e. the number of nonzero components of a.

A *linear* $[n, k, d]$ *code* over $GF(2^p)$ is a k-dimensional subspace of the vector space $GF(2^p)^n$, where the Hamming distance between any two distinct subspace vectors is at least d (and d is the largest number with this property).

A *generator matrix G* for a linear $[n, k, d]$ code C is a $k \times n$ matrix whose rows form a basis for C. A generator matrix is in *echelon* or *standard* form if it has the form $G = [I_{k \times k} A_{k \times (n-k)}]$, where $I_{k \times k}$ is the identity matrix of order k. We write simply $G = [I\,A]$ omitting the indices wherever the matrix dimensions are irrelevant for the discussion, or clear from the context.

Linear $[n, k, d]$ codes obey the *Singleton bound*: $d \leqslant n - k + 1$. A code that meets the bound, i.e. $d = n - k + 1$, is called a *maximal distance separable* (MDS) code. A linear $[n, k, d]$ code C with generator matrix $G = [I_{k \times k} A_{k \times (n-k)}]$ is MDS if, and only if, every square submatrix formed from rows and columns of A is nonsingular (cf. [MacWilliams and Sloane 1977, chapter 11, § 4, theorem 8]).

2.3. Matrices over $GF(2^m)$

The set of all $3 \times n$ matrices over $GF(2^m)$ is denoted by $\mathbb{M}_n$, with O and I standing for the zero and identity matrices, respectively.

Consider the following matrices:

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}, \ B = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \ C = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

A straightforward inspection reveals that A and B are nilpotent and C is idempotent over $GF(2^m)$ ($A^2 = B^2 = O, C^2 = C$), and also that $AB = BA = O$.

As a consequence, all matrices of form $D = I + aA + bB$ are involutions, i.e. $D^2 = I, \forall a, b \in GF(2^m)$. One can show by direct enumeration that the determinants of all square submatrices formed from rows and columns of D take one of the forms $\{a, a + 1, b, b + 1, a + b, a + b + 1\}$, so that D is MDS iff $a \neq 0, 1, b \neq 0, 1$, and $b \neq a, a + 1$. The simplest such matrix to display the MDS property is thus $D = I + 2A + 4B$.

Furthermore, a simple calculation shows that a matrix of form $E = I + cC$ is invertible iff $c \neq 1$, in which case $E^{-1} = I + \left(\frac{c}{c+1}\right)C$. The determinants of all square submatrices formed from rows and columns of E take one of the forms $\{1, c, c + 1\}$, so that E is MDS iff $c \neq 0, 1$. If c is a primitive cube root of unity, i.e. if $c^2 + c + 1 = 0$ then, on the one hand $c^3 = 1 \Rightarrow c^2 = 1/c$, and on the other hand $c^2 = c + 1 \Rightarrow 1/(c + 1) = 1/c^2 = c \Rightarrow c/(c + 1) = c^2 = c + 1$. Thus E^{-1} assumes a particularly simple form, namely, $E^{-1} = I + (c + 1)C$.

Matrices D and E play important roles in Curupira (see sections 3.3. and 3.7.).

2.4. Cryptographic properties

A product of m distinct Boolean variables is called an m-th order product of the variables. Every Boolean function $f : \mathrm{GF}(2)^n \to \mathrm{GF}(2)$ can be written as a sum over GF(2) of distinct m-order products of its arguments, $0 \leqslant m \leqslant n$; this is called the algebraic normal form of f. The *nonlinear order* of f, denoted $\nu(f)$, is the maximum order of the terms appearing in its algebraic normal form.

A *linear* Boolean function is a Boolean function of nonlinear order 1, i.e. its algebraic normal form only involves isolated arguments. Given $\alpha \in \mathrm{GF}(2)^n$, we denote by $\ell_\alpha : \mathrm{GF}(2)^n \to GF(2)$ the linear Boolean function consisting of the sum of the argument bits selected by the bits of α:

$$\ell_\alpha(x) = \bigoplus_{i=0}^{n-1} \alpha_i \cdot x_i.$$

A mapping $S : \mathrm{GF}(2^n) \to \mathrm{GF}(2^n)$, $x \mapsto S[x]$, is called a *substitution box*, or S-box for short. An S-box can also be viewed as a mapping $S : \mathrm{GF}(2)^n \to \mathrm{GF}(2)^n$ and therefore described in terms of its component Boolean functions $s_i : \mathrm{GF}(2)^n \to \mathrm{GF}(2), 0 \leqslant i \leqslant n-1$, i.e. $S[x] = (s_0(x), \ldots, s_{n-1}(x))$.

The *nonlinear order* of an S-box S, denoted ν_S, is the minimum nonlinear order over all linear combinations of the components of S:

$$\nu_S = \min_{\alpha \in \mathrm{GF}(2)^n} \{\nu(\ell_\alpha \circ S)\}.$$

The *difference table* of an S-box S is defined as

$$e_S(a, b) = \#\{c \in \mathrm{GF}(2^n) \mid S[c \oplus a] \oplus S[c] = b\}.$$

The *δ-parameter* of an S-box S is defined as

$$\delta_S = 2^{-n} \cdot \max_{a \neq 0, b} e_S(a, b).$$

The product $\delta_S \cdot 2^n$ is called the *differential uniformity* of S.

The *correlation* $c(f, g)$ between two Boolean functions f and g can be calculated as follows:

$$c(f, g) = 2^{1-n} \cdot \#\{x \mid f(x) = g(x)\} - 1.$$

The *λ-parameter* of an S-box S is defined as the maximal value for the correlation between linear functions of input bits and linear functions of output bits of S:

$$\lambda_S = \max_{(i,j) \neq (0,0)} c(\ell_i, \ell_j \circ S).$$

The *branch number $\mathcal{B}$* of a linear mapping $\theta : \mathrm{GF}(2^n) \to \mathrm{GF}(2^n)$ is defined as

$$\mathcal{B}(\theta) = \min_{a \neq 0}\{w_h(a) + w_h(\theta(a))\}.$$

Table 1. The P and Q mini-boxes

u	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$P[u]$	3	F	E	0	5	4	B	C	D	A	9	6	7	8	2	1
$Q[u]$	9	E	5	6	A	2	3	C	F	0	4	D	7	B	1	8

2.5. Miscellaneous notation

Given a sequence of functions $f_m, f_{m+1}, \ldots, f_{n-1}, f_n$, $m \leqslant n$, we adopt the notation

$$\overset{n}{\underset{r=m}{\bigcirc}} f_r \equiv f_n \circ f_{n-1} \circ \cdots \circ f_{m+1} \circ f_m$$

and

$$\overset{r=n}{\underset{m}{\bigcirc}} f_r \equiv f_m \circ f_{m+1} \circ \cdots \circ f_{n-1} \circ f_n.$$

If $m > n$, both expressions stand for the identity mapping.

3. Description of the CURUPIRA primitive

The CURUPIRA cipher is an iterated block cipher that operates on a 96-bit *cipher state* organized as a matrix in $\mathbb{M}_4$. It uses a 48t-bit ($2 \leqslant t \leqslant 4$) *cipher key* organized as a matrix in $\mathbb{M}_{2t}$. In the following we will individually define the component mappings and constants that build up CURUPIRA, then specify the complete cipher in terms of these components.

3.1. The nonlinear layer γ

Function $\gamma : \mathbb{M}_n \rightarrow \mathbb{M}_n$ consists of the parallel application of a nonlinear S-box $S : GF(2^8) \rightarrow GF(2^8)$ to all bytes of the argument individually:

$$\gamma(a) = b \iff b_{i,j} = S[a_{i,j}],\ 0 \leqslant i < 3,\ 0 \leqslant j < n.$$

The CURUPIRA S-box is the same one defined for the ANUBIS and KHAZAD ciphers [Barreto and Rijmen 2000a, Barreto and Rijmen 2000b], designed to display $\delta_S = 2^{-5}$, $\lambda_S = 2^{-2}$, and $\nu_S = 7$ and hence to thwart differential, linear, and interpolation attacks. It was also constructed so that $S[S[x]] = x, \forall x \in GF(2^8)$. Therefore, γ is an involution.

The actual S-box can be computed on demand with algorithm 1 from two mini-boxes (see table 1) that fit in just 16 bytes. Alternatively and more commonly, if space is available it can be precomputed (either at compilation time or at system startup) and stored in 256 bytes.

Algorithm 1 Computing $S[u]$ from the mini-boxes P and Q

INPUT: $u(x) \in GF(2^8)$, represented as a byte u.
OUTPUT: $S[u]$.

1: $u_h \leftarrow P[(u \gg 4)\,\&\,\mathsf{F}]$, $u_l \leftarrow Q[u\,\&\,\mathsf{F}]$
2: $u'_h \leftarrow Q[((u_h\,\&\,\mathsf{C}) \oplus ((u_l \gg 2)\,\&\,3)]$, $u'_l \leftarrow P[((u_h \ll 2)\,\&\,\mathsf{C}) \oplus (u_l\,\&\,3)]$
3: $u_h \leftarrow P[(u'_h\,\&\,\mathsf{C}) \oplus ((u'_l \gg 2)\,\&\,3)]$, $u_l \leftarrow Q[((u'_h \ll 2)\,\&\,\mathsf{C}) \oplus (u'_l\,\&\,3)]$
4: **return** $(u_h \ll 4) \oplus u_l$

3.2. The permutation layer π

Permutation $\pi : \mathbb{M}_n \to \mathbb{M}_n$ swaps each columns of its argument according to the rule:

$$\pi(a) = b \Leftrightarrow b_{i,j} = a_{i,i \oplus j}, \ 0 \leqslant i < 3, \ 0 \leqslant j < n.$$

Its easy to see that π is an involution.

3.3. The linear diffusion layer θ

The diffusion layer $\theta : \mathbb{M}_n \to \mathbb{M}_n$ is a linear mapping based on the $[6, 3, 4]$ MDS code with generator matrix $G_D = [I \ D]$ where $D = I + 2A + 4B$, i.e.

$$D = \begin{bmatrix} 3 & 2 & 2 \\ 4 & 5 & 4 \\ 6 & 6 & 7 \end{bmatrix},$$

so that

$$\theta(a) = b \Leftrightarrow b = D \cdot a.$$

Since D is self-inverse, θ is an involution.

Circulant matrices [Daemen et al. 1997, NIST 2001] are not suitable for the diffusion layer because no circulant matrix can be involutional. Cauchy matrices [Youssef et al. 1996, Youssef et al. 1997] might have been an option, but the resulting coefficients are in general very complex, impairing efficient implementation. The actual choice above involves the simplest possible coefficients, in the sense of minimum polynomial degree and minimum Hamming weight.

3.4. The key addition $\sigma[k]$

The affine key addition $\sigma[k] : \mathbb{M}_n \to \mathbb{M}_n$ consists of the bitwise addition of a key matrix $k \in \mathbb{M}_n$, i.e.

$$\sigma[k](a) = b \Leftrightarrow b_{i,j} = a_{i,j} \oplus k_{i,j}, \ 0 \leqslant i < 3, \ 0 \leqslant j < n.$$

This mapping is also used to introduce schedule constants in the key schedule. The key addition so defined is obviously an involution.

3.5. Key representation

A $48t$-bit user key $\mathcal{K}$, $2 \leqslant t \leqslant 4$, externally stored as a byte array of length $6t$, is internally represented as a matrix $K \in \mathbb{M}_{2t}$ such that

$$K_{i,j} = \mathcal{K}[i + 3j], \ 0 \leqslant i < 3, \ 0 \leqslant j < 2t.$$

In other words, the user key is mapped to the cipher key by columns (not by rows).

3.6. Schedule constants

The *schedule constants* $q^{(s)}$ are constant matrices defined as $q^{(0)} = 0$ and, for $s > 0$ and $0 \leqslant j < 2t$,

$$q_{i,j}^{(s)} = \begin{cases} S[2t(s-1) + j], & \text{if } i = 0, \\ 0, & \text{otherwise.} \end{cases}$$

Good schedule constants should not be equal for all bytes in a state, and also not equal for all bit positions in a byte. They should also be different in each round. The actual choice meets these constraints. Taking values from the S-box itself avoids the need for any extra storage. In fact, exactly t S-box entries are used per round.

3.7. The key evolution ψ_s

The cipher key is updated during the cipher operation by a reversible transform defined by two linear operations as follows.

Let $\xi : \mathbb{M}_{2t} \to \mathbb{M}_{2t}$ be the linear transform such that

$$\xi(a) = b \Leftrightarrow \begin{cases} b_{0,j} &=& a_{0,j}, \\ b_{1,j} &=& a_{1,(j+1) \bmod 2t}, \\ b_{2,j} &=& a_{2,(j-1) \bmod 2t}, \end{cases}$$

i.e. keeps the first row of its argument unchanged, rotates the second row one position to the left, and rotates the third row one position to the right.

Let $\mu : \mathbb{M}_n \to \mathbb{M}_n$ be the linear transform such that

$$\mu(a) = E \cdot a,$$

where $E = I + cC$, c being the primitive cube root of the unity $c(x) = x^4 + x^3 + x^2 \in \mathrm{GF}(2^8)$. Its inverse is $\mu^{-1}(a) = E^{-1} \cdot a$ where $E^{-1} = I + (c+1)C$.

Define $\omega \equiv \mu \circ \xi$, and let ω^m denote the composition of ω with itself m times. Let $a \in \mathbb{M}_{2t}$. By direct inspection one can verify that the period of ω over $\mathbb{M}_{2t}$ is $m = 6t$ for $2 \leqslant t \leqslant 4$. In other words, $m = 6t$ is the smallest positive integer such that $\omega^m(a) = a$, $\forall a \in \mathbb{M}_{2t}$. The idea is to compute ω^m on a basis of $\mathbb{M}_{2t}$, e.g. $\{e^{(kl)} \mid e_{ij}^{(kl)} = \delta_{ki}\delta_{lj}\}$, and verifying that $\omega^m(e^{(kl)}) \neq e^{(kl)}$ for $1 \leqslant m < 6t$ but $\omega^m(e^{(kl)}) = e^{(kl)}$ for $m = 6t$.

Define the accumulated schedule constant $Q^{(s)} = \sum_{i=0}^{s} \omega^{s-i+1}(q^{(i)})$. Let $K \in \mathbb{M}_n$ be the cipher key, and let the initial key stage to be $K^{(0)} \equiv K$. The *key evolution function* $\psi_r : \mathbb{M}_n \to \mathbb{M}_n$ computes key stage $K^{(r)}$ from key stage $K^{(r-1)}$. It is defined as $\psi_r \equiv \omega \circ \sigma[q^{(r)}]$, i.e.

$$K^{(r)} = \psi_r(K^{(r-1)}) = \left(\bigcirc_{i=1}^{r} \omega \circ \sigma[q^{(i)}] \right)(K) = \omega^r(K) + Q^{(r)}.$$

The key schedule derives subkeys from key stages $K^{(0)}$ through $K^{(m-1)}$. It is clear that an extra application of ω and subsequent addition of $Q^{(m-1)}$ recovers the original cipher key, i.e. $K = \omega(K^{(m-1)}) + Q^{(m-1)}$. Only $Q^{(m-1)}$ needs to be stored for sequential processing; in all scheduling steps but this finalization the simpler constants $q^{(r)}$ can be used. This *cyclic schedule* behavior resets the cipher; therefore, the key evolution process can be conducted in-place; no extra storage is needed for the key stages, which can always be computed on-the-fly at low computational cost.

3.8. The key selection ϕ_r

The effective round subkeys $\kappa^{(r)}$ needed by the cipher are computed (either directly from the cipher key K or indirectly from key stage $K^{(r)}$) via the *key selection function*, $\phi_r : \mathbb{M}_n \to \mathbb{M}_n$, defined so that:

$$\kappa^{(r)} = \phi_r(K) \Leftrightarrow \kappa_{0,j}^{(r)} = S[K_{0,j}^{(r)}] \text{ and } \kappa_{i,j}^{(r)} = K_{i,j}^{(r)} \text{ for } i > 0, \ 0 \leqslant j < 4.$$

The key selection function introduces nonlinearity in the key schedule, ensuring 8 applications of the S-box between any two rounds subkeys and $4(R+1)$ applications of the S-box for the whole key schedule, where R is the number of rounds.

3.9. The complete cipher

Curupira is defined for the cipher key $K \in \mathbb{M}_n$ and R rounds as the mapping Curupira$[K]$: $\mathbb{M}_4 \to \mathbb{M}_4$ given by

$$\text{Curupira}[K] \equiv \sigma[\kappa^{(R)}] \circ \pi \circ \gamma \circ \left(\overset{R-1}{\underset{r=1}{\bigcirc}} \sigma[\kappa^{(r)}] \circ \theta \circ \pi \circ \gamma \right) \circ \sigma[\kappa^{(0)}].$$

The initial key addition $\sigma[\kappa^{(0)}]$ is called *whitening*. The composite mapping $\rho[\kappa^{(r)}] \equiv \sigma[\kappa^{(r)}] \circ \theta \circ \pi \circ \gamma$ is called the *round function* for the r-th round; by convenience, the related mapping $\rho'[\kappa^{(R)}] \equiv \sigma[\kappa^{(R)}] \circ \pi \circ \gamma$ is called the *last round function*, although it is not the same as the round function.

An R-round iterated cipher needs $R + 1$ subkeys. The key evolution has period $m = 6t$, the number of rounds is at most $6t - 1$. The actual number of rounds is variable: for $48t$-bit keys, it is any value in range $4t - 2 \leqslant R \leqslant 6t - 1$ (see table 2). The cyclic property of the key schedule is only fully available (in the sense of automatically resetting the cipher after each encryption) with the maximum number of rounds for each key size, but the round subkeys can always be computed independently in any desired order.

Table 2. Allowed number of rounds for each key size

key size (bits)	min # rounds	max # rounds
96	10	11
144	14	17
192	18	23

3.10. The inverse cipher

We now show that Curupira is an involutional cipher, in the sense that the only difference between the cipher and its inverse is in the key schedule.

Theorem 1. *Let* $\alpha[\kappa^{(0)}, \ldots, \kappa^{(R)}]$ *stand for* Curupira *encryption under the sequence of round keys* $\kappa^{(0)}, \ldots, \kappa^{(R)}$, *and let the decryption keys be defined as* $\bar{\kappa}^{(0)} \equiv \kappa^{(R)}$, $\bar{\kappa}^{(R)} \equiv \kappa^{(0)}$, *and* $\bar{\kappa}^{(r)} \equiv \theta(\kappa^{(R-r)})$ *for* $0 < r < R$. *Then* $\alpha^{-1}[\kappa^{(0)}, \ldots, \kappa^{(R)}] = \alpha[\bar{\kappa}^{(0)}, \ldots, \bar{\kappa}^{(R)}]$.

Proof. We start from the definition of $\alpha[\kappa^{(0)}, \ldots, \kappa^{(R)}]$:

$$\alpha[\kappa^{(0)}, \ldots, \kappa^{(R)}] = \sigma[\kappa^{(R)}] \circ \pi \circ \gamma \circ \left(\overset{R-1}{\underset{r=1}{\bigcirc}} \sigma[\kappa^{(r)}] \circ \theta \circ \pi \circ \gamma \right) \circ \sigma[\kappa^{(0)}].$$

Since the component functions are involutions, the inverse transform is obtained by applying them in reverse order. However, we notice that $\gamma \circ \pi = \pi \circ \gamma$, since π moves but does not mix elements of its argument while γ only affects individual elements, independently of their coordinates. Besides, $\theta \circ \sigma[\kappa^{(r)}] = \sigma[\theta(\kappa^{(r)})] \circ \theta$, since $(\theta \circ \sigma[\kappa^{(r)}])(a) = \theta(\kappa^{(r)} + a) = \theta(\kappa^{(r)}) + \theta(a) = (\sigma[\theta(\kappa^{(r)})] \circ \theta)(a)$, for any input a. Therefore we can write:

$$\alpha^{-1}[\kappa^{(0)}, \ldots, \kappa^{(R)}] = \sigma[\kappa^{(0)}] \circ \left(\overset{r=R-1}{\underset{1}{\bigcirc}} \pi \circ \gamma \circ \sigma[\theta(\kappa^{(r)})] \circ \theta \right) \circ \pi \circ \gamma \circ \sigma[\kappa^{(R)}].$$

Substituting $\bar{\kappa}^{(r)}$ in the above equation and slightly changing the grouping of operations we arrive at the desired result:

$$\alpha^{-1}[\kappa^{(0)}, \ldots, \kappa^{(R)}] = \sigma[\bar{\kappa}^{(R)}] \circ \pi \circ \gamma \circ \left(\bigcirc_{r=1}^{R-1} \sigma[\bar{\kappa}^{(r)}] \circ \theta \circ \pi \circ \gamma \right) \circ \sigma[\bar{\kappa}^{(0)}] = \alpha[\bar{\kappa}^{(0)}, \ldots, \bar{\kappa}^{(R)}].$$

$\square$

4. Analysis

For all allowed key lengths, the security goals are that CURUPIRA is *K-secure* and *hermetic* in the sense of [Daemen 1995]. We now present the results of our security analysis.

4.1. Differential and linear cryptanalysis

One can show [Daemen and Rijmen 1998, section 3.1] that every four-round differential characteristic or linear approximation has at least 16 active S-boxes. As a consequence, no four-round differential characteristic has probability larger than $\delta_S^{16} = (2^{-5})^{16} = 2^{-80}$, and no four-round linear approximation has input-output correlation larger than $\lambda_S^{16} = (13 \times 2^{-6})^{16} \approx 2^{-36.8}$. This makes classical differential or linear attacks, which need characteristics with probability larger than 2^{-95} or input-output correlations larger than 2^{-48} over *all* rounds, as well as some advanced variants like differential-linear attacks, very unlikely to succeed for the full cipher.

Attacks based on linear cryptanalysis can sometimes be improved by using nonlinear approximations [Knudsen and Robshaw 1996]. However, with the current state of the art the application of nonlinear approximations seems limited to the first and/or the last round of a linear approximation. This seems to be even more so for ciphers using strongly nonlinear S-boxes, like CURUPIRA.

4.2. Truncated differentials

A truncated differential attack similar to that described in [Daemen and Rijmen 1998, section 3.2 and appendix A] against 7 rounds of BKSQ applies to CURUPIRA reduced to the same number of rounds. Since the probability of the truncated differential for right pairs of plaintexts over six rounds is very low (2^{-48}, the same as that of a random pairs of plaintexts), no truncated differential faster than exhaustive key search has been found for 8 or more rounds.

4.3. Interpolation attacks

Interpolation attacks [Jakobsen and Knudsen 1997] generally depend on the cipher components (particularly the S-box) having simple algebraic structures that can be combined to give expressions with manageable complexity. The complex algebraic expression of the pseudo-randomly generated S-box in $GF(2^8)$, in combination with the effect of the diffusion layer, makes these types of attack infeasible for more than a few rounds.

4.4. Weak keys and related-key cryptanalysis

Weak keys are keys that result in a block cipher mapping with detectable weaknesses. Typically, this occurs for ciphers in which the nonlinear operations depend on the actual

key value. This is not the case for Curupira, where keys are applied using XOR and all nonlinearity is in the fixed S-box.

Related-key attacks generally rely upon slow diffusion and/or symmetry in the key schedule. The Curupira key schedule was itself designed according to the Wide Trail strategy and features fast, nonlinear diffusion of cipher key differences to the round keys. This makes related-key attacks, including advanced variants like that described in [Ferguson et al. 2001, section 4], exceedingly unlikely.

For key lengths that are larger than the length of one round key, it is inevitable that there exist sets of keys that produce identical values for at least one round key. We leave it as an open problem to determine whether classes of keys can be determined that produce identical round key values for two or more consecutive rounds. Even so, it is unclear how such keys could possibly be used successfully in a related-key attack.

4.5. Saturation attacks

Saturation or integral attacks are among the most effective attacks against ciphers of the Wide Trail strategy with a reduced number of rounds. Adapting the basic saturation attacks against reduced-round Rijndael [Daemen and Rijmen 2002, section 10.2] to work against Curupira is straightforward.

The partial sum improvement [Ferguson et al. 2000] is a dynamic programming technique that trades computational effort for storage by reorganizing the intermediate computations. While it still requires the same amount of chosen plaintexts as a basic 6-round attack, the computational effort drops by a factor around 2^{14} at the cost of about 2^{16} bits of extra storage. There is a very complex extension of the partial sum technique called the herds attack. It is not really clear whether it works against Curupira; if so, it would break 7 rounds at the cost of 99.6% of all possible plaintexts, an offline effort of 2^{88} encryptions, and a huge amount of storage.

Table 3 summarizes the complexities of saturation attacks against Curupira.

Table 3. Complexity of saturation attacks against Curupira

rounds (n)	plaintexts	n-round encryptions
4	2^9	2^9
5	2^{11}	2^{35}
6	2^{27}	2^{37}
7	$2^{96} - 2^{87}$	2^{88}

4.6. The Gilbert-Minier attack

The Gilbert-Minier attack [Gilbert and Minier 2000] breaks 7 rounds of Curupira with 2^{24} guesses for one column of the first round key $\times 2^{12}$ c-sets $\times 48$ S-box lookups per entry $\times 2^{64}$ entries/table $\times 2$ tables, or about 2^{110} S-box lookups (2^{104} 7-round encryptions), plus 2^{32} chosen plaintexts. It is unclear whether the speedup against 7-round Rijndael with 128-bit keys can be modified to work against 7-round Curupira with 96-bit keys; if so, this attack would become marginally faster than exhaustive search.

4.7. A general extension attack

Any n-round attack can be extended against $n + 1$ or more rounds for long keys by simply guessing the whole $\kappa^{(n+1)}$ round key and proceeding with the n-round attack [Lucks 2000]. Each extra round increases the complexity by a factor 2^{96} S-box lookups. Since the complexity of a 6-round saturation attack with the partial sum improvement is 5×2^{40} S-box lookups, a 7-round extension costs 5×2^{136} S-box lookups, or about 2^{132} 7-round encryptions, faster than exhaustive search for 144-bit and 192-bit keys.

4.8. Algebraic and other attacks

There is no consensus regarding the effectiveness of algebraic attacks like the XSL method [Courtois and Pieprzyk 2002]. Because of the heuristic nature of such attacks, theoretical complexity analyses remain controversial, while experimental evidence tends to deny their viability. At the time of this writing no such attack has ever been demonstrated, not even against simplified versions of AES, whose S-box has a far simpler algebraic structure than the S-box used in CURUPIRA. It is pointed out, however, that this very S-box in a different context has already been subjected to third-party scrutiny, with the explicit goal of assessing its resistance against algebraic attacks [Biryukov and DeCannière 2003], and no evidence of weaknesses was found.

Boomerang attacks [Wagner 1999] benefit from ciphers whose strength is different for encryption and decryption; this is hardly the case for CURUPIRA, due to its involutional structure. Slide attacks [Biryukov and Wagner 1999] are thwarted by the asymmetry introduced in the key schedule by the schedule constants.

Summarizing, no attack faster than exhaustive key search could be found against CURUPIRA.

5. Implementation issues

On an 8-bit processor with a limited amount of RAM, the nonlinear substitution layer is performed bytewise, combined with the $\sigma[k]$ transformation. For θ and ψ_s, it is necessary to implement matrix multiplication.

Multiplication by the polynomial $g(x) = x$ in $GF(2^8)$ is of central importance. It can be implemented with one shift and one conditional XOR, or more efficiently using a 256-byte table X such that $X[u] \equiv x \cdot u$, if space is available [Daemen and Rijmen 2002, section 4.1.1]. Whatever the choice, we denote by `xtimes(u)` the result of $x \cdot u(x)$. Equally important is multiplication by the polynomial $c(x) = x^4 + x^3 + x^2$ in $GF(2^8)$. We denote by `ctimes(u)` the result of $c(x) \cdot u(x)$. It can be implemented as `ctimes(u)` $\equiv$ `xtimes(xtimes(xtimes(xtimes(u)` $\oplus$ `u)` $\oplus$ `u))`, using 2 XORs and 4 `xtimes` operations.

Algorithm 2 calculates $D \cdot a$ for a vector $a \in \mathbb{M}_1$ at the cost of 6 XORs and 2 `xtimes` operations.

Algorithm 3 calculates $E \cdot a$ for a vector $a \in \mathbb{M}_1$ (or a column of a matrix in $\mathbb{M}_n$) at the cost of 5 XORs and 1 `ctimes` operation, or 7 XORs and 4 `xtimes` operations. It also calculates $E^{-1} \cdot a$ at the cost of one extra XOR.

5.1. Efficiency considerations

We now compare the relative efficiency of CURUPIRA and Skipjack. This is quite representative in context of ciphers designed for sensor networks in face of the results

---

Algorithm 2 Computing $D \cdot a$

INPUT: $a \in \mathbb{M}_1$.
OUTPUT: $D \cdot a$.
1: $v \leftarrow \texttt{xtimes}(a_0 \oplus a_1 \oplus a_2), \ w \leftarrow \texttt{xtimes}(v)$
2: $b_0 \leftarrow a_0 \oplus v, \ b_1 \leftarrow a_1 \oplus w, \ b_2 \leftarrow a_2 \oplus v \oplus w$
3: **return** b

Algorithm 3 Computing $E \cdot a$ or $E^{-1} \cdot a$

INPUT: $a \in \mathbb{M}_1$.
INPUT: e, a Boolean flag signaling whether $E \cdot a$ (**true**) or $E^{-1} \cdot a$ (**false**) is to be computed.
1: $v \leftarrow a_0 \oplus a_1 \oplus a_2$
2: **if** e **then**
3: $\quad v \leftarrow \texttt{ctimes}(v)$
4: **else**
5: $\quad v \leftarrow \texttt{ctimes}(v) \oplus v$
6: **end if**
7: $b_0 \leftarrow a_0 \oplus v, \ b_1 \leftarrow a_1 \oplus v, \ b_2 \leftarrow a_2 \oplus v$
8: **return** b

---

of [Law et al. 2006]. Thus, a similar advantage exists with respect to MISTY1 and even more so regarding the AES or the other ciphers considered in that survey.

The encryption cost of R-round CURUPIRA is $3R - 1$ XORs, $3(R - 1)/2$ xtimes operations and R S-box lookups per byte, not counting the cost of the key schedule. For instance, 10-round CURUPIRA with a 96-bit key takes 29 XORs, 6 xtimes operations and 10 S-box lookups per encrypted byte. By comparison, Skipjack takes 48 XORs and 16 F-table lookups per encrypted byte, not counting counter increments and key index updates for the key schedule. Assuming that the cost of any of the basic operations for each cipher is roughly the same, we estimate CURUPIRA to cost $45/64 \approx 70\%$ as much as Skipjack on average. We are currently working on CURUPIRA implementation and a comparison between the ciphers, specially for constrained platforms, will be released soon.

The CURUPIRA key schedule is admittedly more complex than its counterparts. Still, it amounts to less than 1/2 S-box lookup, 1/6 of a ctimes operation and 2 XORs per key byte and per round, and it thwarts several kinds of related-key attacks.

We point out that adoption of CURUPIRA also allows for a noticeable reduction in energy consumption and processing time if a standard MAC like CBCMAC, which demands a full encryption per authenticated data block, is replaced by a MAC of the ALRED family like Pelican [Daemen and Rijmen 2005], which requires only $4/R$ encryptions per block plus two full encryptions for the whole message.

6. Conclusion

We have described CURUPIRA, a new, special-purpose block cipher targeted at applications with heavily constrained resources, particularly sensor networks and *ad-hoc* networks involving mobile equipment. CURUPIRA was designed according to the Wide Trail strategy and displays both involutional structure and cyclic key schedule, two important features to overcome the limitations of the target platforms. The resulting cipher is compact and efficient, yet the security analysis indicates that it is about as strong a cipher as can be designed under the assumed operational constraints.

References

Barreto, P. S. L. M. and Rijmen, V. (2000a). The ANUBIS block cipher. In *First open NESSIE Workshop*, Leuven, Belgium. NESSIE Consortium.

Barreto, P. S. L. M. and Rijmen, V. (2000b). The KHAZAD legacy-level block cipher. In *First open NESSIE Workshop*, Leuven, Belgium. NESSIE Consortium.

Biham, E., Biryukov, A., and Shamir, A. (1999). Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials. In *Advances in Cryptology – Eurocrypt'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 55–64. Springer.

Biryukov, A. and DeCannière, C. (2003). Block ciphers and systems of quadratic equations. In *Fast Software Encryption – FSE'2003*, volume 2887 of *Lecture Notes in Computer Science*, pages 274–289. Springer.

Biryukov, A. and Wagner, D. (1999). Slide attacks. In *Fast Software Encryption – FSE'99*, volume 1636 of *Lecture Notes in Computer Science*, pages 245–259, Rome, Italy. Springer.

Courtois, N. and Pieprzyk, J. (2002). Cryptanalysis of block ciphers with overdefined systems of equations. In *Advances in Cryptology – Asiacrypt'2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 267–287. Springer.

Daemen, J. (1995). *Cipher and hash function design strategies based on linear and differential cryptanalysis*. Doctoral dissertation, Katholiek Universiteit Leuven.

Daemen, J., Knudsen, L. R., and Rijmen, V. (1997). The block cipher SQUARE. In *Fast Software Encryption – FSE'97*, volume 1267 of *Lecture Notes in Computer Science*, pages 149–165, Haifa, Israel. Springer.

Daemen, J., Peeters, M., Assche, G. V., and Rijmen, V. (2000). The NOEKEON block cipher. In *First open NESSIE Workshop*, Leuven, Belgium. NESSIE Consortium.

Daemen, J. and Rijmen, V. (1998). The block cipher bksq. In *Smart Card Research and Applications – CARDIS'98*, volume 1820 of *Lecture Notes in Computer Science*, pages 236–245. Springer.

Daemen, J. and Rijmen, V. (2002). *The Design of Rijndael: AES – The Advanced Encryption Standard*. Springer, Heidelberg, Germany.

Daemen, J. and Rijmen, V. (2005). The pelican mac function. Cryptology ePrint Archive, Report 2005/088. Available from `http://eprint.iacr.org/2005/088`.

Feistel, H. (1973). Cryptography and computer privacy. *Scientific American*, 228(5):15–23.

Ferguson, N., Kelsey, J., Lucks, S., Schneier, B., Stay, M., Wagner, D., and Whiting, D. (2000). Improved cryptanalysis of RIJNDAEL. In *Fast Software Encryption – FSE'2000*, volume 1978 of *Lecture Notes in Computer Science*, pages 213–230, New York, USA. Springer.

Ferguson, N., Kelsey, J., Lucks, S., Schneier, B., Stay, M., Wagner, D., and Whiting, D. (2001). Improved cryptanalysis of rijndael. In *Fast Software Encryption – FSE'2000*, volume 1978 of *Lecture Notes in Computer Science*, pages 213–230. Springer.

Gilbert, H. and Minier, M. (2000). A collision attack on seven rounds of Rɪᴊɴᴅᴀᴇʟ. In *Third Advanced Encryption Standard Candidate Conference – AES3*, pages 230–241, New York, USA. NIST.

Jakobsen, T. and Knudsen, L. R. (1997). The interpolation attack on block ciphers. In *Fast Software Encryption – FSE'95*, volume 1267 of *Lecture Notes in Computer Science*, pages 28–40, Haifa, Israel. Springer.

Karlof, C., Sastry, N., and Wagner, D. (2004). Tinysec: a link layer security architecture for wireless sensor networks. In *2nd International Conference on Embedded Networked Sensor Systems – SenSys'2004*, pages 162–175, Baltimore, USA. ACM.

Knudsen, L. R. and Robshaw, M. J. B. (1996). Non-linear approximations in linear cryptanalysis. In *Advances in Cryptology – Eurocrypt'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 224–236. Springer.

Law, Y. W., Doumen, J., and Hartel, P. (2006). Survey and benchmark of block ciphers for wireless sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 2(1):65–93.

Lucks, S. (2000). Attacking seven rounds of Rɪᴊɴᴅᴀᴇʟ under 192-bit and 256-bit keys. In *Third Advanced Encryption Standard Candidate Conference – AES3*, pages 215–229, New York, USA. NIST.

MacWilliams, F. J. and Sloane, N. J. A. (1977). *The theory of error-correcting codes*, volume 16. North-Holland Mathematical Library.

Matsui, M. (1997). New block encryption algorithm MISTY. In *Fast Software Encryption – FSE'97*, volume 1267 of *Lecture Notes in Computer Science*, pages 54–68. Springer.

NIST (2001). *Federal Information Processing Standard (FIPS 197) – Advanced Encryption Standard (AES)*. National Institute of Standards and Technology. `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`.

NSA (1998). *Skipjack and KEA Algorithm Specifications, version 2.0*. National Security Agency. `http://csrc.nist.gov/CryptoToolkit/skipjack/skipjack.pdf`.

Wagner, D. (1999). The boomerang attack. In *Fast Software Encryption – FSE'99*, volume 1636 of *Lecture Notes in Computer Science*, pages 156–170, Rome, Italy. Springer.

Youssef, A. M., Mister, S., and Tavares, S. E. (1997). On the design of linear transformations for substitution permutation encryption networks. In *Selected Areas in Cryptography – SAC'97*, Proceedings, pages 40–48.

Youssef, A. M., Tavares, S. E., and Heys, H. M. (1996). A new class of substitution-permutation networks. In *Selected Areas in Cryptography – SAC'96*, Proceedings, pages 132–147.