New Features of Latin Dances:
Analysis of Salsa, ChaCha, and Rumba

Jean-Philippe Aumasson[1] Simon Fischer[1]
Shahram Khazaei[2]
Willi Meier[1] Christian Rechberger[3]

[1]FHNW, Windisch, Switzerland
[2]EPFL, Lausanne, Switzerland
[3]IAIK, Graz, Austria

Fast Software Encryption
Lausanne, Switzerland, Feb. 2008

Outline

- Targets of our attcks:
  - Two stream ciphers: Salsa and ChaCha
  - A compression function: Rumba
- Our Contribution:
  - Introducing the concept of Probabilistic Neutral Bits (PNB)
  - Attack on reduced rounds of Salsa, ChaCha and Rumba
  - The first break of Salsa20/8

Part I
Analysis of Salsa and ChaCha

Description of Salsa

$$X = \begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{pmatrix} = \begin{pmatrix} c_0 & k_0 & k_1 & k_2 \\ k_3 & c_1 & v_0 & v_1 \\ t_0 & t_1 & c_2 & k_4 \\ k_5 & k_6 & k_7 & c_3 \end{pmatrix}$$

A keystream block Z is defined as $Z = X + \mathrm{Round}^{20}(X)$ with Round being the round function of Salsa20 defined as:

- Rotates the j^{th} column of its input X of j positions up,
- Transforms each column $(x_0, x_1, x_2, x_3)^\dagger$ to $(z_0, z_1, z_2, z_3)^\dagger$ by

$$\begin{aligned} z_1 &= x_1 \oplus \left[(x_3 + x_0) \lll 7\right] \\ z_2 &= x_2 \oplus \left[(x_0 + z_1) \lll 9\right] \\ z_3 &= x_3 \oplus \left[(z_1 + z_2) \lll 13\right] \\ z_0 &= x_0 \oplus \left[(z_2 + z_3) \lll 18\right], \end{aligned}$$

- Rotates back the j^{th} column of j positions down,
- Transpose matrix

Description of ChaCha

The same as Salsa except for the non-linear transformation:

$$\begin{aligned} b_0 &= x_0 + x_3 & z_0 &= b_0 + b_3 \\ b_1 &= (x_1 \oplus b_0) \lll 7 & z_1 &= (b_1 \oplus z_0) \lll 13 \\ b_2 &= x_2 + b_1 & z_2 &= b_2 + z_1 \\ b_3 &= (x_3 \oplus b_2) \lll 9 & z_3 &= (b_3 \oplus z_2) \lll 18 \end{aligned}$$

Bernstein mentions: It brings better confusion with the same number of operations compared with Salsa.

This is the early version; new version to be proposed at SASC'08 has different rotation values.

Attack Overview

Analysis of Salsa and ChaCha reduced to R rounds:

- Identify an optimal choice for truncated differentials (over the first r rounds)
- Guess partially the key and detect the bias backwardly from last round to r-th round ($R - r$ rounds).

Differential Attack: More Details

Two steps:

- Finding an r-round truncated bias differential with $\mathcal{ID}\Delta^0$:

  $\Pr_{v,t}([\text{Round}^r(X) \oplus \text{Round}^r(X')]_{p,q} = 1 \mid \Delta^0) = \frac{1}{2}(1 + \varepsilon_d)\}$

- Backward computation:

  $f(k, v, t, Z, Z') := [\text{Round}^{r-R}(Z-X) \oplus \text{Round}^{r-R}(Z'-X')]_{p,q}$

Hypotheses Testing

$$H_0 : \hat{k} = k$$
$$H_1 : \hat{k} \neq k$$

$$\Pr\{f(\hat{k}, v, t, Z, Z') = 1 \mid H_0\} = \tfrac{1}{2}(1 + \varepsilon_d)$$
$$\Pr\{f(\hat{k}, v, t, Z, Z') = 1 \mid H_1\} = \tfrac{1}{2}$$

Classical way: try all 2^{256} guesses for $\hat{k}$

New Idea:

Find an approximation $g(k, v, t, Z, Z')$ of f which effectively depends only on m (< 256) key bits.

New Idea:

Find an approximation $g(k, v, t, Z, Z')$ of f which effectively depends only on m (< 256) key bits.

Motivation: Reducing the search space from 2^{256} to 2^m

How to find g?

Probabilistic Neutral Bits

Our approach: 1) Divide the key bits into two sets: significant key bits and non-significant ones. 2) replace non-significant key bits with some fixed values.

Probabilistic Neutral Bits

Our approach: 1) Divide the key bits into two sets: significant key bits and non-significant ones. 2) replace non-significant key bits with some fixed values.

Definition: For a function f, the neutrality measure of the key bit k_i is defined as $\gamma_i = 2p_i - 1$, where p_i is the probability that complementing the key bit k_i does not change the output of f.

Probabilistic Neutral Bits

Our approach: 1) Divide the key bits into two sets: significant key bits and non-significant ones. 2) replace non-significant key bits with some fixed values.

Definition: For a function f, the neutrality measure of the key bit k_i is defined as $\gamma_i = 2p_i - 1$, where p_i is the probability that complementing the key bit k_i does not change the output of f.

Non-significant key bits: all key bits with $|\gamma_i| > \gamma$ for some γ.

Detection of the Bias

Function Approximation:

$$\mathrm{Pr}_{v,t}\{f(k,v,t,Z,Z') = g(k,v,t,Z,Z')\} = \tfrac{1}{2}(1+\varepsilon_a)$$

Differential Bias:

$$\mathrm{Pr}_{v,t}\{f(k,v,t,Z,Z') = 1\} = \tfrac{1}{2}(1+\varepsilon_d)$$

Detection of the Bias

Function Approximation:

$$\Pr_{v,t}\{f(k,v,t,Z,Z') = g(k,v,t,Z,Z')\} = \frac{1}{2}(1+\varepsilon_a)$$

Differential Bias:

$$\Pr_{v,t}\{f(k,v,t,Z,Z') = 1\} = \frac{1}{2}(1+\varepsilon_d)$$

$$\Rightarrow \Pr_{v,t}\{g(k,v,t,Z,Z') = 1\} = \frac{1}{2}(1+\varepsilon), \ \varepsilon = \varepsilon_a \cdot \varepsilon_d$$

Detection of the Bias

Function Approximation:

$$\Pr_{v,t}\{f(k,v,t,Z,Z') = g(k,v,t,Z,Z')\} = \frac{1}{2}(1+\varepsilon_a)$$

Differential Bias:

$$\Pr_{v,t}\{f(k,v,t,Z,Z') = 1\} = \frac{1}{2}(1+\varepsilon_d)$$

$$\Rightarrow \Pr_{v,t}\{g(k,v,t,Z,Z') = 1\} = \frac{1}{2}(1+\varepsilon), \quad \varepsilon = \varepsilon_a \cdot \varepsilon_d$$

To detect the bias with $p_{\text{nd}} = 1.3 \times 10^{-3}$ and p_{fa}:

Samples: $N \approx \left(\frac{\sqrt{-2\log p_{\text{fa}}}+3\sqrt{1-\varepsilon^2}}{\varepsilon}\right)^2$

Time: $N2^m$

Attack

- **Precomputation**
- **Effective (or on-line) attack**

Precomputation

1. Find a high-probability r-round truncated differential (i.e. Δ^0 and bit position index (p, q)).
2. Choose a threshold γ.
3. Construct the function f.
4. Estimate the neutrality measure γ_i of each key bit.
5. Put all those key bits with $|\gamma_i| < \gamma$ in the significant key bits set of size m.
6. Construct the approximation function g.
7. Estimate the bias ε.
8. Estimate the required number of samples N.

Effective attack

1. For an unknown key, collect N pairs of keystream blocks commited to the input difference Δ^0.
2. For each choice of the subkey (i.e. the m significant key bits) do:
   2.1 Compute the bias of g using the N keystream block pairs.
   2.2 If the optimal distinguisher legitimates the subkeys candidate as a (possibly) correct one, perform an additional exhaustive search over the $256 - m$ non-significant key bits to check the correctness of this filtered subkey and to find the non-significant key bits.
   2.3 If the right key is found stop and output the recovered key.

Time complexity: $2^m(N + 2^{256-m}p_{\text{fa}}) = 2^m N + 2^{256} p_{\text{fa}}$

Simulation Results

	Salsa20/7	Salsa20/8	ChaCha6	ChaCha7
γ	0.6	0.2	0.55	0.4
m	131	228	117	208
ε	0.006	0.004	0.004	0.002
N	2^{23}	2^{21}	2^{24}	2^{23}
Before	2^{190}	2^{255}	2^{255}	2^{255}
Now	2^{153}	2^{249}	2^{140}	2^{231}

Simulation Results

	Salsa20/7	Salsa20/8	ChaCha6	ChaCha7
γ	0.6	0.2	0.55	0.4
m	131	228	117	208
ε	0.006	0.004	0.004	0.002
N	2^{23}	2^{21}	2^{24}	2^{23}
Before	2^{190}	2^{255}	2^{255}	2^{255}
Now	2^{153}	2^{249}	2^{140}	2^{231}

Part II
Analysis of Rumba Compression Function

Description of Rumba

- Maps 1536-bit (48-word) message to a 512-bit (16-word) value
- $M = (M_0, M_1, M_2, M_3)$
- Consists of four instances of Salsa with different diagonal constants: $F_i(M_i) = (X_i + \mathrm{Round}^{20}(X_i))$

$$\mathrm{Rumba}(M) = F_0(M_0) \oplus F_1(M_1) \oplus F_2(M_2) \oplus F_3(M_3)$$

Collision Attack on Rumba20

Differential based attack involving two message blocks M and M' satisfying:

$$M_0 \oplus M'_0 = M_2 \oplus M'_2,\ M_1 = M'_1 \text{ and } M_3 = M'_3.$$

Collision Attack on Rumba20

Differential based attack involving two message blocks M and M' satisfying:

$$M_0 \oplus M'_0 = M_2 \oplus M'_2,\ M_1 = M'_1 \text{ and } M_3 = M'_3.$$

$$F_0(M_0) \oplus F_0(M'_0) = F_2(M_2) \oplus F_2(M'_2)$$

This suggests us to look for high probability differentials for F_i

Notations

$\Delta_i^0 = X_i \oplus X_i'$: Initial input difference for F_i

$\Delta_i^r = \text{Round}^r(X_i) \oplus \text{Round}^r(X_i')$: Difference after r round without FF

Attack procedure

- Find a High-Probability Differential $(\Delta_i^r \mid \Delta_i^0)$
  - Use a linearized version of Rumba by replacing '$+$' with '$\oplus$'
  - Find low weight input differentials

Attack procedure

- Find a High-Probability Differential $(\Delta_i^r \mid \Delta_i^0)$
  - Use a linearized version of Rumba by replacing '$+$' with '$\oplus$'
  - Find low weight input differentials
- Enlarge the probabilities
  - Linearization method in the first round
  - Neutral bits technique in the second round

Our Low Weight Differential

$$\Delta_i^0 = \begin{pmatrix} 0 & 0 & \texttt{00000002} & 0 \\ \texttt{00080040} & 0 & \texttt{00000020} & 0 \\ \texttt{80000000} & 0 & 0 & 0 \\ \texttt{80001000} & 0 & \texttt{01001000} & 0 \end{pmatrix}$$

Our Low Weight Differential

$$\Delta_i^0 = \begin{pmatrix} 0 & 0 & \texttt{00000002} & 0 \\ \texttt{00080040} & 0 & \texttt{00000020} & 0 \\ \texttt{80000000} & 0 & 0 & 0 \\ \texttt{80001000} & 0 & \texttt{01001000} & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 2 & 0 & 2 & 0 \end{pmatrix} \xrightarrow{\text{Round}} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix} \xrightarrow{\text{Round}}$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 2 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \xrightarrow{\text{Round}} \begin{pmatrix} 2 & 2 & 3 & 1 \\ 0 & 3 & 4 & 2 \\ 1 & 1 & 7 & 3 \\ 1 & 1 & 1 & 6 \end{pmatrix} \xrightarrow{\text{Round}} \begin{pmatrix} 8 & 3 & 2 & 4 \\ 5 & 10 & 3 & 4 \\ 9 & 11 & 13 & 7 \\ 6 & 9 & 10 & 9 \end{pmatrix}$$

Our Low Weight Differential

$$\Delta_i^0 = \begin{pmatrix} 0 & 0 & \texttt{00000002} & 0 \\ \texttt{00080040} & 0 & \texttt{00000020} & 0 \\ \texttt{80000000} & 0 & 0 & 0 \\ \texttt{80001000} & 0 & \texttt{01001000} & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 2 & 0 & 2 & 0 \end{pmatrix} \xrightarrow[2^{-4}]{\text{Round}} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix} \xrightarrow[2^{-7}]{\text{Round}}$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 2 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \xrightarrow[2^{-41}]{\text{Round}} \begin{pmatrix} 2 & 2 & 3 & 1 \\ 0 & 3 & 4 & 2 \\ 1 & 1 & 7 & 3 \\ 1 & 1 & 1 & 6 \end{pmatrix} \xrightarrow[2^{-194}]{\text{Round}} \begin{pmatrix} 8 & 3 & 2 & 4 \\ 5 & 10 & 3 & 4 \\ 9 & 11 & 13 & 7 \\ 6 & 9 & 10 & 9 \end{pmatrix}$$

Attack Complexity

Without using linearization and neutral bits technique:

	Rumba20/3	Rumba20/4
Without FF	2^{41}	2^{194}
With FF	2^{85}	2^{313}

Improving with Linearization and Neutral Bits

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 2 & 0 & 2 & 0 \end{pmatrix} \xrightarrow[2^{-4}]{\text{Round}} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix} \xrightarrow[2^{-7}]{\text{Round}} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 2 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

Improving with Linearization and Neutral Bits

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 2 & 0 & 2 & 0 \end{pmatrix} \xrightarrow[1]{\text{Round}} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix} \xrightarrow[2^{-3}]{\text{Round}} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 2 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

Try until a good message pair with lots of 2-neutrals bits has been found ...

Our Message pairs

- For F_0:

$$X_0 = \begin{pmatrix} \texttt{73726966} & \texttt{00000400} & \texttt{00000080} & \texttt{00200001} \\ \texttt{00002000} & \texttt{6d755274} & \texttt{000001fe} & \texttt{02000008} \\ \texttt{00000040} & \texttt{00000042} & \texttt{30326162} & \texttt{10002800} \\ \texttt{00000080} & \texttt{00000000} & \texttt{01200000} & \texttt{636f6c62} \end{pmatrix}$$

with 251 neutral bits and a 2-neutral set of size 147. Conforming probability is $\mathrm{Pr} = 0.52$.

- For F_2:

$$X_2 = \begin{pmatrix} \texttt{72696874} & \texttt{00000000} & \texttt{00040040} & \texttt{00000400} \\ \texttt{00008004} & \texttt{6d755264} & \texttt{000001fe} & \texttt{06021184} \\ \texttt{00000000} & \texttt{00800040} & \texttt{30326162} & \texttt{00000000} \\ \texttt{00000300} & \texttt{00000400} & \texttt{04000000} & \texttt{636f6c62} \end{pmatrix}$$

with 252 neutral bits and a 2-neutral set of size 146, and conforming probability $\mathrm{Pr} = 0.41$.

Improving with Linearization and Neutral Bits

$$\Delta_i^2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 2 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \xrightarrow[{\color{red}2^{-34}}]{\text{Round}} \Delta_i^3 = \begin{pmatrix} 2 & 2 & 3 & 1 \\ 0 & 3 & 4 & 2 \\ 1 & 1 & 7 & 3 \\ 1 & 1 & 1 & 6 \end{pmatrix}$$

Improving with Linearization and Neutral Bits

$$\Delta_i^2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 2 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \xrightarrow[2^{-34}]{\text{Round}} \Delta_i^3 = \begin{pmatrix} 2 & 2 & 3 & 1 \\ 0 & 3 & 4 & 2 \\ 1 & 1 & 7 & 3 \\ 1 & 1 & 1 & 6 \end{pmatrix}$$

Attack Complexity using linearization and neutral bits technique:

	Rumba20/3
Without FF	2^{35}
With FF	2^{79}

Summary

Summary

- Introducing the concept of Probabilistic Neutral Bits
- Breaking Salsa20/8 and ChaCha7
- Collision attack on Rumba20/3 in time 2^{79}
- Samples of near collision attack on Rumba20/3 and Rumba20/4

Thank You for your Attention!

Shoot me your Questions :)