Analysis of Salsa and Chacha

Santanu Sarkar

IIT Madras, Chennai

Outline of the talk

- Salsa20 Description
- Differential Analysis
- Salsa20/8 Cryptanalysis
- New results

The eStream Project

- An effort to get some secure stream ciphers satisfying the current requirements.
- http://www.ecrypt.eu.org/stream/
- This is the home page for eStream, the ECRYPT Stream Cipher Project. This multi-year effort running from 2004 to 2008 has identified a portfolio of promising new stream ciphers.
- "For the future, we expect that research on the eSTREAM submissions in general, and the portfolio ciphers in particular, will continue. We therefore welcome any ongoing contributions to any of the eSTREAM submissions. It is also possible that changes to the eSTREAM portfolio might be needed in the future. If so, any future revisions will be made available via these pages."

The eSTREAM Portfolio

The eSTREAM Portfolio (revision 1) (September 2008): The eSTREAM portfolio has been revised and the portfolio now contains the following ciphers:

Profile 1 (SW)	Profile 2 (HW)
HC-128	Grain v1
Rabbit	MICKEY v2
Salsa20/12	Trivium
SOSEMANUK	

Summary of 2nd review (January 2012)

Salsa

Introduction

- Salsa20 was designed by Bernstein in 2005 as a candidate for eStream
- `http://cr.yp.to/snuffle.html`
- Salsa20/12 has been accepted in the eStream software portfolio
- Attacks till 8 rounds are known

Data Structure

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

Each word is of 32 bits. Total 16 words.

256-bit key

- $c_0 = 0x61707865, c_1 = 0x3320646e,$
  $c_2 = 0x79622d32, c_3 = 0x6b206574,$
- 256-bit key $k_0, \ldots, k_7$
- 64-bit nonce v_0, v_1
- 64-bit counter t_0, t_1
- Since this is with 256-bit keys, we can refer it as 256-bit Salsa20.

128-bit key

- One can use the same cipher with 128-bit key, where $k_i = k_{i+4}$, for $0 \leq i \leq 3$
- $c_0 = 0x61707865, c_1 = 0x3120646e$, $c_2 = 0x79622d36, c_3 = 0x6b206574$.
- One may note the little differences in c_1, c_2 for the 128-bit and 256-bit version.

Quarter Round

- The basic nonlinear operation of Salsa20 is the quarterround function.
- Each quarterround(a, b, c, d) consists of four ARX rounds, each of which comprises of one addition (A), one cyclic left rotation (R) and one XOR (X) operation as given below.

$$\left.\begin{aligned} b &= b \oplus ((a + d) \lll 7), \\ c &= c \oplus ((b + a) \lll 9), \\ d &= d \oplus ((c + b) \lll 13), \\ a &= a \oplus ((d + c) \lll 18). \end{aligned}\right\} \tag{1}$$

Row & Column Round

- Each columnround works as four quarterrounds on each of the four columns of the state matrix
- Each rowround works as four quarterrounds on each of the four rows of the state matrix
- columnround, rowroundworks one after another
- Salsa20/20: 10 columnround, 10 rowround interleaved

Column Round & Transpose

- In each round, first apply quarterround on all the four columns in the following order:

$$\text{quarterround}(x_0, x_4, x_8, x_{12}),\ \text{quarterround}(x_5, x_9, x_{13}, x_1),$$
$$\text{quarterround}(x_{10}, x_{14}, x_2, x_6),\ \text{quarterround}(x_{15}, x_3, x_7, x_{11}),$$

- And then a transpose(X) as follows:

$$X = \begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{pmatrix} \rightarrow X^T = \begin{pmatrix} x_0 & x_4 & x_8 & x_{12} \\ x_1 & x_5 & x_9 & x_{13} \\ x_2 & x_6 & x_{10} & x_{14} \\ x_3 & x_7 & x_{11} & x_{15} \end{pmatrix}.$$

Number of Rounds

- By $X^{(r)}$, we mean that r such rounds have been applied on the initial state X.
- $X^{(0)}$ is the same as the initial state X.
- Finally, after R rounds we have $X^{(R)}$.
- Then a keystream block of 16 words or 512 bits is obtained as

$$Z = X + X^{(R)}.$$

- For Salsa20, $R = 20$.
- The one accepted in eStream software portfolio is Salsa20/12, where $R = 12$.
- Naturally, more rounds will provide better security and less rounds will provide higher speed.

Reversible State Transition

- Each Salsa20 round is reversible as the state-transition operations are reversible.
- If $X^{(r+1)} = \text{round}(X^{(r)})$, then $X^{(r)} = \text{reverseround}(X^{(r+1)})$, where reverseround is the inverse of round and consists of first transposing the state and then applying the inverse of quarterround for each column as follows.

$$\left.\begin{array}{rcl} a & = & a \oplus ((d + c) \lll 18), \\ d & = & d \oplus ((c + b) \lll 13), \\ c & = & c \oplus ((b + a) \lll 9), \\ b & = & b \oplus ((a + d) \lll 7). \end{array}\right\} \quad (2)$$

Differential Analysis

Differential

- Start with two Salsa20 states
  - Same at most of the positions
  - Different in a very few positions (may be just one bit)
- Apply a few rounds
- Check whether any correlation can be obtained between two states
- Same key, different IV

Notations

- x_i: the i-th word of the matrix X
- $x_{i,j}$, we mean the j-th bit (0-th bit is the least significant bit) of x_i
- Consider two states $X^{(r)}, X'^{(r)}$
- $\Delta_i^{(r)} = x_i^{(r)} \oplus x_i'^{(r)}$
- $\Delta_{i,j}^{(r)} = x_{i,j}^{(r)} \oplus x_{i,j}'^{(r)}$, we mean the difference between two states at the j-th bit of the i-th word after r many rounds
- $\Delta_{7,31}^{(0)}$ means that we have two initial states $X^{(0)}, X'^{(0)}$ that differ at the 31-st (most significant) bit of the 7-th word (a nonce word) and they are same at all the other bits of the complete state.

Biases

- One may obtain significant biases after a few rounds.
- Input a differential at the initial state (call it Input Differential or $\mathcal{I}D$) and then try to obtain some bias corresponding to combinations of some output bits (call it Output Differential or $\mathcal{O}D$).
- One can compute

$$\Pr(\Delta_{p,q}^{(r)} = 0 | \Delta_{i,j}^{(0)}) = \frac{1}{2}(1 + \epsilon_d)$$

Interesting Observations

- $\Pr(\Delta^{(4)}_{1,14} = 0 | \Delta^{(0)}_{7,31}) = \frac{1}{2}(1 + 0.1314)$ (FSE 2008, Aumasson et al)
- Some other: $\Pr(\Delta^{(4)}_{11,17} = 0 | \Delta^{0}_{8,31}) = \frac{1}{2}(1 - 0.1642)$,
  $\Pr(\Delta^{(4)}_{6,26} = 0 | \Delta^{(0)}_{7,31}) = \frac{1}{2}(1 + 0.1954)$,
  $\Pr(\Delta^{(4)}_{11,26} = 0 | \Delta^{0}_{8,31}) = \frac{1}{2}(1 + 0.1914)$.
- Combining $\mathcal{O}D$'s:
  $\Pr(\Delta^{(4)}_{1,0} \oplus \Delta^{(4)}_{2,9} = 0 | \Delta^{0}_{7,26}) = \frac{1}{2}(1 - 0.60)$ (FSE 2008, could not be used in attacks yet)
- Acceptable conditions: $\mathcal{I}D$s are in counters or nonces only

Differential State Matrix

Given two states $X^{(r)}, X'^{(r)}$, we represent the differential state matrix as

$$\Delta^{(r)} = \begin{pmatrix} \Delta_0^{(r)} & \Delta_1^{(r)} & \Delta_2^{(r)} & \Delta_3^{(r)} \\ \Delta_4^{(r)} & \Delta_5^{(r)} & \Delta_6^{(r)} & \Delta_7^{(r)} \\ \Delta_8^{(r)} & \Delta_9^{(r)} & \Delta_{10}^{(r)} & \Delta_{11}^{(r)} \\ \Delta_{12}^{(r)} & \Delta_{13}^{(r)} & \Delta_{14}^{(r)} & \Delta_{15}^{(r)} \end{pmatrix},$$

where $\Delta_i^{(r)} = x_i^{(r)} \oplus x_i'^{(r)}$.

Cryptanalytic Strategies

- Put some input differential at the initial state and then try to obtain certain bias in some output differential.
- Once you can move forward a few rounds with the above strategy, try to come back a few rounds from a final state after a certain rounds obtaining further non-randomness.
- Combining the above ideas, one may find some non-randomness in Salsa20 for forward plus backward many rounds.

State of the art results

- Cryptanalysis by Crowley (SASC 2006)
  `http://www.ciphergoth.org/crypto/salsa20/`
  Move forward 3 rounds from the initial state and to come back 2 rounds from the final state for an attack on 5-round Salsa20.
- Aumasson, Fischer, Khazaei, Meier, Rechberger: FSE 2008
  Biases till 4 forward rounds have been exploited and then the attack considered moving backwards from the final state for 4 rounds to obtain an 8-round attack on Salsa20.

Initial Ideas (FSE 2008, Aumasson et al)

- Known plain text only attack
- The 512-bit key stream of Salsa20/R, $X + X^{(R)}$, is available to the attacker
- The 256 key bits are not available, thus only rest 256 bits of X are known: 4 constant words, 2 counter words, 2 nonces
- The motivation is to guess the 256 key bits of Salsa20/R with a key search complexity less than 2^{256}, the complexity for exhaustive search.

Setting up

- X, X' are two valid initial states
- Given $\mathcal{I}D$ $\Delta^{(0)}_{i,j}$, one observes a high bias ϵ_d in an $\mathcal{O}D$ $\Delta^{(r)}_{p,q}$ after $r < R$ Salsa rounds
- $\Pr(\Delta^{(r)}_{p,q} = 0 | \Delta^{(0)}_{i,j}) = \frac{1}{2}(1 + \epsilon_d)$, where $\Delta^{(r)} = X^{(r)} \oplus X'^{(r)}$.
- The two keystream blocks after R rounds are given by $Z = X + X^{(R)}$ and $Z' = X' + X'^{(R)}$.
- $r = 4, R = 8$
  $\Pr(\Delta^{(4)}_{1,14} = 0 | \Delta^{(0)}_{7,31}) = \frac{1}{2}(1 + 0.1314)$

Probabilistic Neutral Bit (PNB)

- In X and X', a particular key bit position k is complemented to yield the states $\overline{X}$ and $\overline{X'}$
- Next, one can reverse the states $Z - \overline{X}$ and $Z' - \overline{X'}$ by $R - r$ rounds to yield the states Y and Y' respectively.
- Let $\Gamma_{p,q} = Y_{p,q} \oplus Y'_{p,q}$.
- If the bias in the event $(\Gamma_{p,q} = 0 | \Delta_{i,j}^{(0)})$ is high, then we call the key bit k a *probabilistic neutral bit* (PNB)
- If $\Pr(\Gamma_{p,q} = 0 | \Delta_{i,j}^{(0)}) = \frac{1}{2}(1 + \gamma_k)$, then γ_k is called the *neutrality measure* of the key bit.

How to obtain PNBs

- One should run the experiment for each key bit several times over randomly chosen nonces and counters (e.g., 2^{24} samples corresponding to each key bit, which is sufficient to identify the biases)
- By repeating this for each of the 256 key bits, a subset of the key bits is identified, which are called the PNBs.
- Typically, a threshold probability $\frac{1}{2}(1+\gamma)$ is chosen to filter the PNBs.
- If $\gamma_k \geq \gamma$, then the key bit k is included in the set of the PNBs.

The idea of the attack

- Let the number of PNBs be n and therefore the number of non-PNB bits are $m = 256 - n$
- The main idea behind the key recovery is to search these two sets separately
- The actual attack considers search over the key bits which are not PNBs and then by considering a distinguisher it is possible when the correct keys can be identified

In case of the correct key

- Let the correct key values have been assigned to the m non-PNB key bits and random binary values are assigned to the n PNB key bits in both X and X' to yield the states $\hat{X}$ and $\hat{X}'$ respectively.
- Then one can reverse the states $Z - \hat{X}$ and $Z' - \hat{X}'$ by $R - r$ rounds to yield the states $\hat{Y}$ and $\hat{Y}'$ respectively.
- Let $\hat{\Gamma}_{p,q} = \hat{Y}_{p,q} \oplus \hat{Y}'_{p,q}$
- $\Pr(\hat{\Gamma}_{p,q} = 0 | \Delta_{i,j}^{(0)}) = \frac{1}{2}(1 + \hat{\epsilon})$.
- A higher value of $\hat{\epsilon}$ means that the PNBs have been chosen properly and even without knowing the n PNBs it is possible to identify the rest of the key bits which are not PNBs.

Estimating the complexity of the attack

If the number of samples used is N and if the probability of false alarm is $P_{fa} = 2^{-\alpha}$, the complexity of the attack is then given by

$$2^m(N + 2^n P_{fa}) = 2^m N + 2^{256-\alpha}, \tag{3}$$

where the required number of samples is

$$N \approx \left(\frac{\sqrt{\alpha \log 4} + 3\sqrt{1 - \epsilon^2}}{\epsilon} \right)^2 \tag{4}$$

for probability of non-detection $P_{nd} = 1.3 \times 10^{-3}$.

Complexity of the attack (FSE 2008)

- Taking $\gamma = 0.12$, $n = 36$ PNBs have been reported
- estimated $\epsilon = 0.00015$ median
- $m = 256 - 36 = 220$
- Taking $\alpha = 8$, one obtains $N \approx 2^{31}$
- the total complexity becomes $2^{220}(2^{31} + 2^{36} \cdot 2^{-8})$, which is approximately 2^{251}
- Slight improvement to 2^{250} in ICISC 2012

Cipher	Round	Attack complexity	Reference
Salsa	8	$2^{251.0}$	Aumasson et al. FSE 2008
		$2^{250.0}$	Shi et al. ICISC 2012
		$2^{247.2}$	Maitra et al. WCC 2015
		$2^{245.5}$	Maitra DAM 2016
		$2^{244.9}$	Choudhuri & Maitra FSE 2017
		$2^{243.7}$	Dey & Sarkar DAM

Chacha

ChaCha20 has been adopted by Google, to be used in OpenSSL, replacing RC4

ChaCha

$$X = \begin{pmatrix} X_0 & X_1 & X_2 & X_3 \\ X_4 & X_5 & X_6 & X_7 \\ X_8 & X_9 & X_{10} & X_{11} \\ X_{12} & X_{13} & X_{14} & X_{15} \end{pmatrix} = \begin{pmatrix} c_0 & c_1 & c_2 & c_3 \\ k_0 & k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 & k_7 \\ t_0 & t_1 & v_0 & v_1 \end{pmatrix}.$$

Here $c_0 = 0x61707865, c_1 = 0x3320646e, c_2 = 0x79622d32, c_3 = 0x6b206574$.

Round Function:

$$a = a + b,\ d = ((d \oplus a) \lll 16),$$
$$c = c + d,\ b = ((b \oplus c) \lll 12),$$
$$a = a + b,\ d = ((d \oplus a) \lll 8),$$
$$c = c + d,\ b = ((b \oplus c) \lll 7).$$

- ChaCha applies the function along column and diagonals.
- Along column the order is $(X_0, X_4, X_8, X_{12}), (X_1, X_5, X_9, X_{13}), (X_2, X_6, X_{10}, X_{14})$ and $(X_3, X_7, X_{11}, X_{15})$.
- Also along diagonal the order is $(X_0, X_5, X_{10}, X_{15})$, $(X_1, X_6, X_{11}, X_{12})$, (X_2, X_7, X_8, X_{13}) and (X_3, X_4, X_9, X_{14}).

Cipher	Round	Attack complexity	Reference
Chacha	7	$2^{248.0}$	Aumasson et al. FSE 2008
		$2^{246.5}$	Shi et al. ICISC 2012
		$2^{238.9}$	Maitra DAM 2016
		$2^{237.7}$	Choudhuri & Maitra FSE 2017
		$2^{235.2}$	Dey & Sarkar DAM

IIT Madras

Looking bright & motivated student

Thank You