Masaryk University
Faculty of Informatics

Statistical testing of lighweight cryptography based pseudo-random number generators

Master's Thesis

Michal Hajas

Brno, Spring 2018

This is where a copy of the official signed thesis assignment and a copy of the Statement of an Author is located in the printed version of the document.

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Michal Hajas

Advisor: RNDr. Petr Švenda, Ph.D.

Acknowledgements

Thank all.

Computational resources were supplied by the Ministry of Education, Youth and Sports of the Czech Republic under the Projects CESNET (Project No. LM2015042) and CERIT-Scientific Cloud (Project No. LM2015085) provided within the program Projects of Large Research, Development and Innovations Infrastructures.

We also acknowledge the support of Czech Science Foundation, the project GA16-08565S.

Abstract

Abstract to be done

Keywords

randomness testing, cryptanalysis, block functions, lightweight cryptography, pseudo-radnom number generators

Contents

1 Introduction

2 Theory

3 CryptoStreams testing methodology

CryptoStreams is a tool which is developed and maintained by team of people[1] at the Centre for Research on Cryptography and Security, Masaryk University [1]. It is tool for automatic generation of data with certain properties. Each output is generated by so called streams which are producers of data. Retrieving data from stream in loop and storing them results in data file with desired binary data. Each call for new data from stream will return data of adjustable size, so called *test-vector size*. By setting size of test-vector and number of test vector it is possible to set size of resulting data file. CryptoStreams contains following types of streams.

Streams based on round-reduced cryptoprimitives. It is also possible to configure them with various types of plaintext and key inputs. For example, it is possible to obtain data from AES with 3 rounds which are generated by encrypting all zeroes plaintext with key consisting of all ones. This means all test-vectors will be equal and resulting data file will contain desired number of test-vectors.

Streams outputing data of exact structure which are mostly used as an input streams such as plaintext or key stream. Those might be for example all zeroes stream, all ones stream or low hamming weight stream (small amount of ones) etc. Random (pseudo-random) streams also belong to this category.

Manipulating streams are those which are configured with some source stream or more streams and manipulate them in some desired way. For example, repeating stream is repeating one output specified number of times. Another example is tuple stream that is getting more streams as an input and for each call it returns data consisting of all streams next to each other. By tuple stream it is possible to receive data which consists of plaintexts followed by corresponding ciphertexts.

3.1 History

Initial implementation of CryptoStreams project was part of tool EACirc [2] which is a tool for automatic randomness testing based on genetic programming. At that moment it served only as an input to EACirc and was not possible to use it separately outside of this project. After some time we decided that CryptoStreams might be potentially interesting also as a separate tool. That is why EACirc-streams project was introduced in 2017 and then in 2018 EACirc sub-name was completely abandoned and project was renamed to nowadays name, CryptoStreams.

3.2 Idea

Main idea behind CryptoStreams is easy producing data from crypto-primitives which are somehow reduced, either by rounds or by providing them input with bad randomness properties. By analyzing such outputs, for example, with statistical batteries or other

1. The team of randomness testing involves following people: Radka Cieslarová, Michal Hajas, Dušan Klinec, Matúš Nemec, Jiří Novotný, Ľubomír Obrátil, Marek Sýs, Petr Švenda, Martin Ukrop and others.

randomness testing tools, it is possible to discover potential weaknesses in investigated functions. In this thesis we will show such analysis on cryptography functions designed for internet of things, that is why we call them **lightweight functions**.

3.3 Conducted experiments

There are more ways how we reduce functions so that we find out where are theirs limitations.

Special type of input, mostly with some bad randomness properties, is provided to functions and then results are compared with random or other special inputs. For puposes of this thesis we used following types of streams.

1. Counter stream is a stream in which each test-vector is addition of one to previous test-vector in number representation.
2. Low hamming weight stream returns outputs with least count of ones it is possible. Starting with all zeroes. Then only one true bit on each position etc.
3. Strict avalanche criterion is a type of stream in which first test-vector is randomly generated and every next call is just previous test-vector with one flipped bit.

Round reduced function, is such function, which is composed of very same sequence of operations performed multiple times, mostly in a loop. For example, AES [3] has recommended number of rounds 10, 12 or 14 based on key length as you can see in Figure 3.1. Each round in AES consists of ***SubBytes***, ***ShiftRows***, ***MixColumns***, ***AddRoundKey*** operations.

	Key Length *(Nk words)*	**Block Size** *(Nb words)*	**Number of Rounds** *(Nr)*
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Figure 3.1: Table of rounds based on key size taken from [3], word size is 32 bits.

By analyzing differences between randomness of individual rounds it is possible to find out so called **security margin**. Denoting security margin involves finding out how many rounds is breakable by any kind of attack, the less rounds we are able to break, the higher security margin is. In this thesis we will denote security margin by randomness properties of individual round output.

Plaintext ciphertext stream produce pairs of input and output to function. With statistical testing performed on such output, it is possible to investigate dependency be-

Function	Round	Block size	Key size	Reduction comments
Chaskey	16	16	16	All
Fantomas	12	16	16	All
HIGHT	32	8	16	Only full
LBlock	32	8	10	Even round numbers
LEA	24	16	16	>= 21 rounds
LED	48	8	10	All
Piccolo	25	8	10	All
PRIDE	20	8	16	Round 2 and full
PRINCE	12	8	16	All
RC5-20	20	8	10	All
RECTANGLE-K80	25	8	16	Only full
RECTANGLE-K128	25	8	16	Only full
RoadRunneR-K80	10	8	10	All
RoadRunneR-K128	12	8	16	All
Robin	16	16	16	All
RobinStar	16	16	16	All
SPARX-B64	8	8	16	All
SPARX-B128	8	16	16	All
TWINE	35	8	10	All

Table 3.1: List of all investigated functions, where sizes are given in Bytes and Reduction comments means for what round encryption followed by decryption works.

tween plaintext and ciphertext. However we need to be careful with deciding what input we will choose since it is part of output it must have perfect randomness properties otherwise it may cause some interference in testing result.

3.4 Investigated functions

Implementation of investigated functions were taken over from project FELICS [4] developed by Daniel Dinu and his group at University of Luxembourg. This project is conducting performance analysis of those functions intended for internet of things devices. We have taken over only C++ implementation of function, as we were not interested in implementations optimized for other architectures. Also we needed to implement round reduction of functions ourselves as the project contained only full round implementation. However, provision of round reduction was not hard, mostly functions were prepared in round reduction in mind and we needed to only replace constant in loop with our variable. Besides the main loop function mostly contains also key scheduling, initial and final part. Key scheduling is taking care of creating round key based on encryption key. Initial and final part serves for initialization and finalization of process. We did not round reduced any of those parts as we wanted to avoid some memory problems like uninitialized or wrongly cleaned memory.

Table 3.1 contains all investigated functions.

Idea behind cryptostream: Reduce functions (rounds, weak plaintext, key stream) Test with statistical batteries/ RTT, find out security margin

3.5 Configuration

JSON, test vectors (size/count), seeding

3.6 Content of CryptoStreams

All functions, exact streams (block, hash, stream, prng), manipulating streams mentioned above (maybe move to this subsection).

Testing of streams (Google tests), testing with test vectors, functional tests

4 Results of evaluation of statistical randomness properties

Bibliography

[1] Petr Švenda et al. *CryptoStreams. Tool for generation of data from round-reduced crypto-primitives*. Centre for Research on Cryptography and Security, Masaryk University. 2017. URL: `https://github.com/crocs-muni/CryptoStreams` (visited on 08/20/2018).

[2] Petr Švenda et al. *EACirc. Framework for automatic search for problem solving circuit via Evolutionary algorithms*. Centre for Research on Cryptography and Security, Masaryk University. 2012. URL: `https://github.com/crocs-muni/eacirc` (visited on 08/25/2018).

[3] *Federal Information Processing Standards Publication (FIPS 197). Advanced Encryption Standard (AES)*. 2001.

[4] Daniel Dinu et al. "Felics–fair evaluation of lightweight cryptographic systems". In: *NIST Workshop on Lightweight Cryptography*. Vol. 128. 2015.