ARTICLE IN PRESS

Discrete Applied Mathematics ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Probability distributions for the Linux entropy estimator

Yongjin Yeom, Ju-Sung Kang *

Department of Mathematics and Financial Information Security, Kookmin University, 77 Jeongneung-Ro, Seongbuk-Gu, Seoul, 02707, Republic of Korea

ARTICLE INFO

Article history:
Received 29 January 2016
Received in revised form 19 June 2016
Accepted 12 July 2016
Available online xxxx

Keywords:
Random number generator
Entropy
Entropy estimator
Linux pseudo-random number generator

ABSTRACT

We propose a mathematical model of the entropy estimator in the Linux random number generator. First, we construct a probability model for random event times in entropy sources, and then precisely derive probability distributions for the first, second, and third time differences. Second, we obtain the probability distribution for the minimum of absolute values of these differences, which is used for the estimated entropy in the Linux system. Moreover, we provide several simulations that display the accuracy of our results for various parameters.

1. Introduction

Random numbers are indispensable to modern cryptography. The security of encryption algorithms and protocols in cryptosystem is guaranteed by assuming the existence of *ideal* random number generators (RNGs). For instance, encryption keys should be selected from the uniformly distributed key space, and nonces in challenge–response protocols are required to be sufficiently close to full entropy.

RNG should generate an *unbiased*, *independent*, and *unpredictable* sequence of numbers. However, these goals are difficult to achieve in a real system. A typical model of RNG [7] collects entropy from noise sources and produces pseudo-random numbers using a deterministic algorithm called a PRNG. PRNG uses small input called *seed* to produce output sequences, which are hardly statistically distinguished from random numbers.

Linux PRNG (LRNG) is one of the most popular RNGs. LRNG is widely adopted in Linux and embedded systems as well as in several OSes for mobile devices. LRNG has been a part of the Linux kernel since 1994 and its core structure has hardly changed to date. Nevertheless, the design rationale of LRNG remains unclear. The structure can be divided into three parts: collecting entropy, managing entropy pools, and generating pseudo-random numbers. Several security analyses of LRNG have been conducted [5,9,14]. In 2006, Gutterman et al. [5] provided a full description of LRNG and reported the weakness in entropy pools and the random number generating algorithm. They checked the entropy collecting part and experimentally show that entropy estimation in LRNG is highly conservative. In 2012, Lacharme [9] proposed new observations on the entropy collecting part in the revised LRNG of version 3.X [10]. Empirical entropy analysis shows that the estimated entropy computed by LRNG is lower than the entropy based on empirical frequencies. A formal security model for PRNG with input was recently considered by Dodis et al. [4]. They defined new PRNG construction and related security notions. As a result, LRNG is not *robust* in their model because of the weakness in the entropy estimator and internal mixing function. However, mathematical results based on the probability distributions for the entropy estimator in LRNG are still unavailable.

* Corresponding author. Fax: +82 29104739.
E-mail addresses: salt@kookmin.ac.kr (Y. Yeom), jskang@kookmin.ac.kr (J.-S. Kang).

http://dx.doi.org/10.1016/j.dam.2016.07.019
0166-218X/

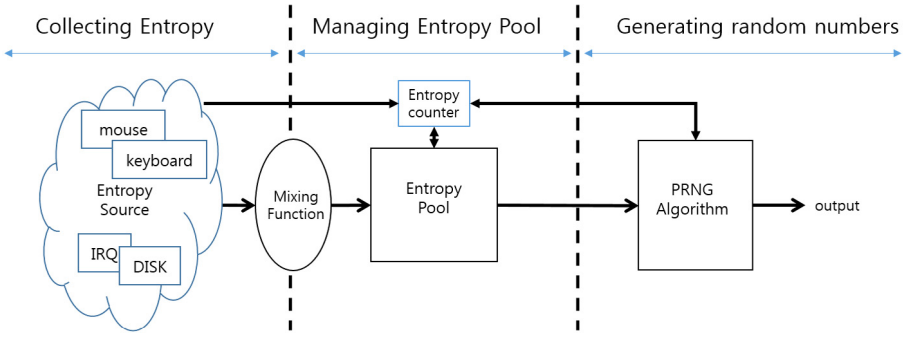

Fig. 1. Brief structure of the Linux PRNG.

In the assessment of cryptographic modules, entropy estimation is a major concern for both developers and testing labs. However, in security evaluation systems such as cryptographic module validation program (CMVP) [3,11] and Common Criteria (CC) [2], which are used to validate the security of RNG in a module or product, statistical tests [13] are more common than entropy estimation. As the output of LRNG is completely determined by the entropy sources, it is very important to estimate the amount of entropy harvested from the noise sources. Consequently, the entropy estimator and the entropy counter play key roles in the security of LRNG. Recently, NIST announced the second draft of SP 800-90B [1], which provides systematic approaches for testing entropy sources. An ISO standard document for entropy estimations [6] is currently being developed and will be published as international standard methods for validating RNGs in the cryptographic modules.

Our contributions:

Under the assumption that a random event occurs with probability p at a discrete time, we derive the probability distribution for the entropy estimator in LRNG. From the sequence of random variables for event times, we precisely calculate the distributions of their differences up to order three and derive the probability distribution for the minimum of absolute values of these differences which is used for estimating entropy in LRNG.

Our result contributes to mitigate the security concern caused by incorrect entropy estimations, particularly when entropy is harvested using the noise sources generated from generic random time differences.

2. Random number generation in Linux

2.1. Structure of the LRNG

The RNG in the Linux system is implemented in the kernel and managed by the operating system. In the Linux system, OS collects entropy from the environment and generates random numbers. Whenever a user needs random numbers, two devices, called `/dev/random` and `/dev/urandom`, produce the required bytes of random data. The difference between the two devices lies in managing their output pools.

As depicted in Fig. 1, the process of random number generation consists of three parts as follows:

- Collecting entropy
- Managing entropy pools
- Generating random numbers.

In the entropy collecting part, LRNG collects data from various entropy sources including interrupt information, disk timing and input devices. Entropy inputs are plunged into the entropy pool through the mixing function. Meanwhile, the entropy estimator evaluates input data and determines the amount of entropy. For each input, the entropy counter is adjusted to indicate how much entropy the pool holds.

Three pools are managed in LRNG. The input pool holds data accumulated from the entropy sources. Two output pools are dedicated to `/dev/random` and `/dev/urandom` devices, respectively. They are independent of each other and have their own entropy counters.

When a user extracts data from the device `/dev/random` using the output function, the corresponding entropy counters are adjusted. The output function is designed to generate output by hashing data in the output pool and simultaneously updates the pool by the feedback function.

If the counter needs to decrease to zero, the output request is blocked. To produce high quality random bits, the output is not allowed unless the pool holds sufficient entropy. By contrast, `/dev/urandom` returns as many bytes as required without blocking regardless of the status of its entropy counter. The whole structure of the LRNG is described in [9].

2.2. Entropy estimator in LRNG

For each event, the LRNG estimates the amount of entropy harvested from the event. The entropy estimator only uses information on the event time using *jiffy count*. The jiffy is a time interval determined by the Linux kernel. The kernel uses

Please cite this article in press as: Y. Yeom, J.-S. Kang, Probability distributions for the Linux entropy estimator, Discrete Applied Mathematics (2016), http://dx.doi.org/10.1016/j.dam.2016.07.019