

# Dissonant Numbers

Altan Orhon and Jonathan E. Magen

December 5, 2006

This handout to accompany a talk of the same title delivered on 2006-12-5 to the Data and Information program at The Evergreen State College by Altan Orhon and Jonathan E. Magen. For a full research description, the authors direct you to the complete paper.

## 1 Pseudo-Random Number Generation Algorithms

The PRNGs we have selected can be thought of as black-box functions mapping sources of entropy to bytes:

$$f : S \rightarrow \{0 \dots 2^8 - 1\} \quad (1)$$

### 1.1 Linear Congruential Generators

Linear congruential generators are one of the fastest and most popular methods of generating pseudo-random sequences. A linear congruential sequence is defined [5] as

$$X_{n+1} = (a \times X_n + c) \bmod m \quad (2)$$

where

$$\begin{aligned} m &> 0. \\ 0 &\leq a < m. \\ 0 &\leq c < m. \\ 0 &\leq X_0 < m. \end{aligned}$$

#### 1.1.1 UNIX rand()

The UNIX `rand()` function first proposed by Brian Kernigan and Dennis Ritchie in the first edition of *The C Programming Language* in 1972 [2]. Since then both the BSD and GNU C libraries have included LCG implementations along with the ANSI C library standard.

### 1.2 Quadratic Congruential Generators

The linear congruential generator can be generalized to the quadratic form [5]:

$$X_{n+1} = (dX_n^2 + aX_n + c) \bmod m \quad (3)$$

#### 1.2.1 Middle Square

The first known variant is the *middle-square method*, suggested by John von Neumann in 1946. The algorithm is based on extracting  $n$  digits from the middle of a written representation of the square of the previous number in the sequence [7].

### 1.3 Mersenne Twister

The Mersenne Twister is a 623-dimensionally equidistributed variant of the twisted generalized feedback shift register operating in 623 dimensions [6]. It is quite fast and produces a sequence of pseudo-random numbers with an enormous period and uniform distribution, which makes it suitable for use in simulations. However, the Mersenne twister is not suitable for cryptographic use: by analyzing a sufficiently large size of the output, the state of the algorithm can be predicted by a simple linear transformation [6]. Even so, the Mersenne Twister algorithm is perfectly suited for large-scale statistical simulation.

## 2 OS Level PRNG Mechanisms

### 2.1 GNU/Linux /dev/random and /dev/urandom

The GNU/Linux kernel implements a cryptographically safe PRNG mechanism through the `/dev/random` and `/dev/urandom` device drivers. The kernel monitors various system I/O drivers and continually adds data to the *system entropy pool*, which is used as input for both PRNGs [3].

Consider the cryptographic and economic reasons for having two PRNGs: `random` expends the contents of the entropy pool as numbers are generated and it will wait until enough entropy has accumulated before progressing. `urandom` (standing for “unlimited random”) will use or reuse data available in the pool and generate numbers as fast as possible. Accordingly, the sequence produced by `urandom` has significantly less entropy than that produced by `random` and should not be used when a high degree of randomness is required [8].

## 2.2 XNU /dev/random

Yarrow is a family of cryptographic PRNGs with the primary of goals of resistance to attack, efficiency and ease of deployment. The Yarrow algorithm relies on a one-way hash function, a block cipher, and an entropy pool. The design of the algorithm is such that once the PRNG is initialized with a secure key, an attacker will not be able to predict the sequence from entropy sources or previous output values. Yarrow-160 is the most popular variant and relies on the SHA1 hash and the three-key triple-DES cipher. [4]

The XNU (Mac OS X) kernel implements the Yarrow-160 compound algorithm for its system PRNG. The entropy pool in XNU is fed by system I/O events. Unlike Linux, write access to the pool is not protected - output is not predictable from pool contents or previous output; though quality will suffer if the entropy-gathering process fails for any reason [1].

## 3 Links

<http://svn.yonkeltron.com/randomtest>  
Project code and document repository.

[http://yonkeltron.com/processing/random\\_unity](http://yonkeltron.com/processing/random_unity)  
Randomly generated Pop-art

<http://pdos.csail.mit.edu/scigen> SCiGen - An Automatic CS Paper Generator.

<http://random.org> Random.org offers true random numbers to anyone on the internet.

## 4 Acknowledgments

The authors would like to thank Richard Weiss and Yoshiya Moritani for their guidance, help, input and patience as well as the Free software community at

large. Additionally, none of this research would have been possible without the gracious accommodation of Sherri Shulman and the Advanced Operating Systems Laboratory.

## 5 License

This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/2.5/> or send a letter to Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA

## References

- [1] Mac `random(4)` manpage. Retrieved 2006-12-5 from <http://www.hmug.org/man/4/RANDOM.php>.
- [2] Brian W. Kernigan and Dennis M. Ritchie. *The C Programming Language*. Prentice Hall, 2 edition, 1988.
- [3] Zvi Gutterman, Benny Pinkas, and Tzachy Reinman. Analysis of the linux random number generator. Cryptology ePrint Archive, Report 2006/086, 2006. Retrieved 2006-11-23 from <http://eprint.iacr.org/2006/086>.
- [4] J. Kelsey, B. Schneier, and N. Ferguson. Yarrow-160: Notes on the Design and Analysis of the Yarrow Cryptographic Pseudorandom Number Generator. *Sixth Annual Workshop on Selected Areas in Cryptography, August, 1999*.
- [5] D. E. Knuth. *Seminumerical algorithms*, volume 2 of *The art of computer programming*. Reading, Mass., 2 edition, 1981.
- [6] M. Matsumoto and T. Nishimura. Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. *ACM Trans. on Modeling and Computer Simulation*, 8(1):3-30, January 1998.
- [7] Stephen Wolfram. *A New Kind of Science*. Wolfram Media, Inc., 2002. Retrieved 2006-12-1 from <http://www.wolframscience.com/nksonline>.
- [8] The Linux Kernel Documentation. `random(4)` man page. <http://www.kernel.org>.