

So You Need a Random Number Generator

Charles Wright

If you are looking for a quality random number generator (RNG) for your cryptographic needs, then let this guide help you along the way. Before you start, you must know what a random number generator is. Then you must be able to differentiate between classifications of RNGs. Lastly, you must be able to determine the qualities you need in a RNG in your cryptographic system.

Random Number Generators

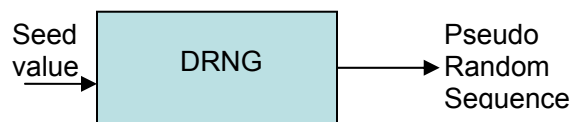
First of all, a random number generator is, like its name suggests, a method of generating a random number. The RNG (random number generator) will provide a sequence of numbers that are random. Generally the RNG will provide a sequence of random numbers not just one number.

Different Random Number Generators

RNGs fall into three different categories: deterministic RNGs (DRNGs), also known as pseudorandom, true (physical) random RNGs (TRNGs), and hybrid of DRNG and TRNG. These three classifications of RNGs are generally referenced in the field of cryptography because they have different qualities that serve different purposes.

Deterministic Number Generators

Pseudorandom deterministic number generators are attributed with the quality that the sequence of numbers a DNRG generate are



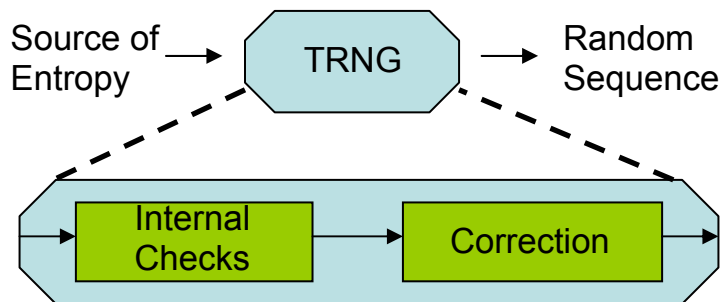
predetermined from an equation. DNRGs need some input value (seed value) to start generating the random sequence. This seed value will allow different sequences with different seeds. The DNRG, however complex and complicated, will always produce the same output from the exact same seed and internal states.

Generally, DNRGs are software implementations. DNRGs lend themselves to be coded since there are no requirements of specialized hardware. All that is needed is the seed value. The DNRG then outputs a pseudo random sequence.

True (Physical) Random Number Generators

True Random Number Generators require a source of entropy (randomness). TRNGs generally measure some source of noise that is nearly indeterminable, so that the measurements are likely to be any value within a certain range. For example, the amount of radiation that a Geiger counter detects in a normal setting will be random. At any given time, the amount background radiation detected will be random.

Just to make sure that the entropy is usable, TRNGs have internal checks to make sure that entropy is still random and that it is still measurable.



Then the bits of the raw data from the entropy source should be corrected for any abnormalities or bias that might exist. The correction should at least include an XOR corrector or a Von Neumann corrector.

The XOR corrector will take 2 consecutive bits and XOR them together. This simple process will reduce most of the bias toward ones or zeros. The Von Neumann corrector will examine the same 2 consecutive bits as before but this time it will see if the two bits are the same. If they are both the same, then both bits are ignored; if they are different, then the first bit is used. The Von Neumann corrector will be able to detect

periods of 0s or 1s where the DRNG might be malfunctioning. An example of the two correctors in action is presented below from Robert Davies.

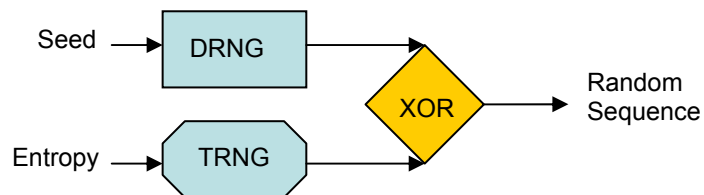
Original	10	11	00	10	10	01	00	01	10	10	01	01	11	00	10	00	10	10	01	01
XOR	1	0	0	1	1	1	0	1	1	1	1	1	0	0	1	0	1	1	1	1
Von Neumann	1			1	1	0		0	1	1	0	0			1		1	1	0	0

The first sequence is an example of input the TRNG might be given. The XOR corrector deletes half of the original sequence’s number of bits. The Von Neumann corrector, on the other hand deletes approximately ¾ of the original sequences of bits.

Hybrid Random Number Generators

Hybrid Random Number Generators are a combination of a TRNG and a DRNG. Hybrid RNGs can either be a TRNG and a DRNG XORed together, or a DRNG with the seed created by a TRNG. The first of the two combinations is more common. A hybrid RNG made from a combination of a TRNG and DRNG usually mixes the two outputs with an XOR or similar mixing function. As stated above, the XOR decreases the output

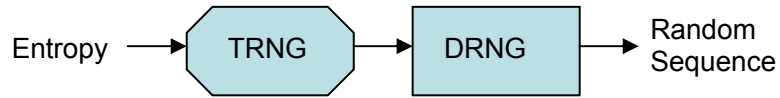
by a half, but since there are two sources of information, the output length would equal the sequences of either the DRNG or the TRNG.



This configuration also provides a way to allow either the DRNG or the TRNG to temporarily provide nonrandom numbers. If the TRNG were to be temporarily affected by an electromagnetic field or the like, the DRNG would still provide the randomness for the sequence. It also makes an attack on a cryptographic system implementing a hybrid RNG much more difficult. The attack must not only be physical but at the same time

software based. The attacker must be able to determine what the DRNG will provide and must be able to prevent the TRNG from properly functioning.

The next type of hybrid RNG is much less common; DRNGs are usually implemented with software because it is easier and cheaper than implementing a DRNG with hardware. This type of hybrid RNG does exist so it will be mentioned. A DRNG is provided a seed value



determined by the

TRNG. This allows more confidence that the DRNG will provide a random sequence. Since the seed from the DRNG is random, the output will be random as long as the DRNG is cryptographically secure. This configuration usually assumes that the TRNG will be used far less than the DRNG. This type of hybrid RNG is actually implemented by Intel® and RSA Data Security® in the Intel Random Number Generator in the 8XX series chipsets.

Your Random Number Generator Needs

For use in cryptography, the needs of a RNG may vary significantly. You must be able to know exactly which qualities you need most. After prioritizing your needs in a RNG, you can then choose which RNG suits your needs best. This guide will be able to help you distinguish the general needs of most RNG implementations in cryptosystems.

Most RNGs in cryptosystems have the same general qualities that allow them to be implemented. These RNGs have to be secure enough to be cryptographically strong. They must all produce their sequences of random numbers without allowing any attackers to observe. If the random numbers are observed and recorded, the entire cryptosystem is

compromised. RNGs must somehow be able to be determined good RNGs for use in cryptosystems.

How to Determine a Good RNG

In the field of cryptography, the requirements of a certain random number generator vary from application to application. The general requirements of a RNG include providing all possible values needed with an equal probability. Another good quality of a RNG that is used in the cryptography field is that it should produce numbers that are independent from previous values

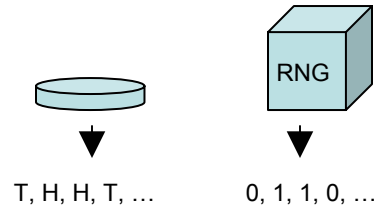
Requirement 1: Statistically Correct
Requirement 2: Hard to predict

and that future values should be not able to be guessed.

Equal Probability

A random number generator should be able to have the quality of giving an equally probably random number.

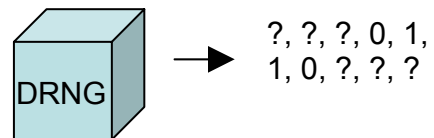
Just as flipping a coin should produce heads or tails at the same probability or a dice should produce one of its sides at



an equal chance as its others, a random number generator should produce a random number within its range that is equally probable. Each RNG, therefore, must be tested so that their output shall be statistically acceptable.

Indeterminist Values

A good quality of a RNG in the field of cryptography is that it generates numbers that seem



to not be related to previous or future values. If given a previous value generated by an RNG in a crypto-application, the current or later values should not be able to be guessed.

If they were, it would compromise any future uses of the crypto-application. If the application were used for encryption, all previous and future encrypted information could be determined.

Testing

Each RNG must be tested to provide a high level of confidence that the RNG in question will provide a random sequence most of the time. There are problems with testing RNGs; each RNG is different. DRNGs and TRNGs must be tested differently because they have different qualities. A reasonable test for DRNGs will not necessarily be a reasonable test for TRNGs and vice versa. Also, each RNG is different from other ones of its own type. One kind of TRNG is different than another TRNG and will perform differently to the same tests. A RNG could be designed to pass all of the designated tests but still not pass the second requirement of being hard to guess. A sequence generator could be created so that the sequence is not random at all. For example, a sequence of primes could be tested using all known test suites. In all tests, the prime numbers will pass well enough; it is a purely chaotic sequence. The problem is that the pattern is known, all previous and future values can be attained.

Despite these pitfalls of testing, the test still must be conducted to provide the high level of confidence in the RNGs that is necessitated in the field of cryptography. Common tests for all three types of RNGs are the George Marsaglia's DIEHARD Tests and National Institute of Standards and Technology's statistical tests. Both test suites have their advantages and flaws, but remember these tests don't prove that the RNG tested is a suitable RNG; it just gives you a high confidence that it is.

Example Random Number Generators

Listed below are some of the examples of RNGs that are available. To help you decide which RNG is appropriate these example RNGs have a description in the level of confidence that should be placed on them. The RNGs are evaluated according all of the suggestive information presented earlier. The information that the RNGs are evaluated upon is supplied by the manufacturer or designer of the RNG. A link to the product is also supplied so that you can compare my evaluation of the RNG to your expectations of an RNG in your cryptosystem. Not all of the following RNGs will be appropriate for your needs. This list is not a comprehensive list of RNGs available. The list is merely for comparative purposes.

SG100 TRNG by Protego

http://www.protego.se/sg100_en.htm

First of all, this RNG is outdated in that it relies on a serial port interface. If older models of computers are running your cryptosystem, then this might be a feasible option for you. Unfortunately, the speed of the SG100 is quite slow: “High output speed: 9.2 kBytes/s.” On the plus side, this product has proof of passing the DIEHARD test suite. Unfortunately, this product has been tested with the NIST test suite. On the plus side, this product has its complement of documentation. An interesting note is provided. “The current version of the SG100 do not work on some multi-function chipsets for serial/parallel ports. These chipsets are common on Pentium II computers.”

R200-USB™ TRNG by Protego

http://www.protego.se/sg200_d.htm

This product is a newer in that it uses a USB interface. Some red flags are raised on this product though. “The Output from the R200:s are processed by the HCIA

Technology to guarantee perfect output statistics for any application area.” If perfect output statistics are guaranteed, how are true randomness guaranteed. What raise the most alarm is in the HCIA manual. The HCIA manual does not show how it functions. Everything in cryptography depends not on the secrecy of the cryptographic process but the secret keys that are involved. By not providing the specification of the HCIA process, the level of confidence in this product is greatly reduced. No matter how much testing is involved, there is no guarantee that this product does not provide random sequences. Another disturbing feature of this product is that it is too general purpose.

The R210-USB, R220-USB, R230-USB True Random Number Generators are designed to provide easy access to random numbers for Lottery applications, key generation in Computer Security applications, and random input data for Monte Carlo and Simulated Annealing Optimizations.

In general, cryptographic systems require specialized hardware and software because the cryptosystems need to meet very stringent requirements.

RBG 1210 by Tundra

<http://www.tundra.com/Products/8000Series/index.cfm>

This TRNG is unfortunately discontinued. This product, on the other hand, describes in full detail of its processes. It has diagrams and descriptions of the random noise source this product uses. It only has a bit generation of “up to 20,000 bits/second.” With respect to testing, the description for the RBG 1210 simply states, “The RBG 1210 has been successfully proven for true randomness against an extensive set of recognized tests including Chi squared, KS test, serial test, poker test, collision test, and picturing randomness.” Test results are supposed be supplied by Tundra Semiconductor Corporation.

ZRANDOM USB by Westphal

<http://home.t-online.de/home/p.westphal/zranusbe.htm>

This TRNG/Hybrid RNG uses a USB 2.0 interface allowing “40.000 random bits / s (physical) 220.000 random bits / s (XOR)” in its two different modes. The first mode appears to be solely a TRNG mode, while the XOR mode is the first of the two hybrid RNGs mentioned above. The testing documentation of ZRANOM USB does not provide enough tests it sample data. First, only four different tests are done on this product. Second, a very small sample is tested in each case. Only 20 packets of 20000 bits are shown to be tested.

LavaRnd™ by LavaRnd

<http://lavarand.com/what/index.html>

The LavaRnd software is just that, software. Its distinguishing feature is that the software is open source, meaning that it is essentially free. The only cost associated with the LavaRnd system is the actual parts needed to complete the system. The LavaRnd software depends on an outside source, a closed lens digital camera or webcam. The problem with this system is that it claims to be a “cryptographically sound random number generator” by presenting the data that the LavaRnd has great statistical qualities. LavaRnd does not provide evidence that supports meeting the second requirement of RNGs. The information passed to the computer from the webcam could be intercepted and that data could comprise any cryptosystem implementing LavaRnd.

Random.org

<http://www.random.org/>

This website provides random numbers and sequences using radio signals as a source of entropy for this TRNG. This site explicitly says, “Numbers from random.org shouldn't be used for this purpose (generating cryptographic keys) because they might be observed by a third party while in transit.” Despite this, random.org's numbers could be

used as seeds of DRNGs as long as enough data is sampled. Although is not a great method of providing seeds for DRNGs, this method does ensure that the different sequences provided by those DRNGs won't be correlated because of the seeds.

Blum Blum Shub by Lenore Blum, Manuel Blum and Michael Shub

The Blum Blum Shub algorithm is a DRNG that is proven to be secure with a correct implementation. The only problem with the BBS is that it is extremely slow. The reason for its strong security and its lack of speed is that it generates pair of large sized prime numbers. As long as the prime numbers are nearly prime, then the amount of work to decrypt a BBS random number sequence is at least as hard as factoring large integers. Since the generation of large primes is much faster than factoring the product of two of the said primes, this algorithm is cryptographically secure. Unfortunately it is extremely slow compared to other RNGs.

HENKOS by Marius Oliver Gheorghita

This DRNG won grand prize “ ‘E-IDEA’ for software ideas/products at binary 2002, Bucharest-Romania” HENKOS also boasts a bit generation of “276.62 Mbps, for a seed of 640 bits,” tested on a PII 300MHz with 32 MB RAM. This DRNG also proudly displays its test results from numerous test suites including NIST's and DIEHARD. The problem with the HENKOS is that it does not go into any detail into the type of functions this DRNG utilizes. The product only mentions that “The HENKOS generator is a new type of generator that implements an own algorithm.” Even with the impressive test results of HENKOS, this DRNG cannot be assured to be a sufficient random number generator. “The security (of the HENKOS generator) is not sacrificed for the speed, the statistic tests confirming that the generator can be used in cryptographic applications.” It

passes requirement 1 but not 2; the above statement does not sufficiently prove nor satisfy requirement 2.

Sources Cited

- Davies, Robert. "Exclusive OR (XOR) and hardware random number generators". 28 Feb 2002. <<http://www.robertnz.net/pdf/xor2.pdf>>.
- "Hardware random number generators". Statistics Research Associates Limited. 14 Oct 2000. <<http://www.robertnz.net/hwrng.htm>>.
- "True random number generators". 1997. <http://www.robertnz.net/true_rng.html>.
- Gheorghita, Marius Oliver. "GRAND PRIZE AT NATIONAL CONTEST "E-IDEA" FOR SOFTWARE IDEAS/PRODUCTS AT BINARY 2002, BUCHAREST-ROMANIA -- HENKOS PSEUDO-RANDOM NUMBER GENERATOR". <<http://crypto.8k.com/>>.
- Guisado, Ernesto. "Cryptographic Random Numbers". <<http://triumvir.org/rng/>>.
- Narasimhan, B. "DIEHARD". <<http://stat.fsu.edu/~geo/diehard.html>>.
- Nazario, Jose. "What Crypto is Good For (And Why it Matters)". May 2001. <<http://monkey.org/~jose/docs/crypto.txt>>.
- Network Working Group. "Randomness Recommendations for Security". Ohio State. Dec 1994. <<http://www.cse.ohio-state.edu/cgi-bin/rfc/rfc1750.html>>.
- Noll, Landon Curt, Simon Cooper, and Mel Pleasant. "LavaRnd Frequently Asked Questions". <<http://lavarand.com/faq/index.html>>.
- "LavaRnd process in detail". <<http://lavarand.com/what/process.html>>.
- "LavaRnd: A Cryptographically Sound Random Number Generator". <<http://lavarand.com/what/how-it-works.html>>.
- "Terms & Definitions: cryptographically sound random number generator". <http://lavarand.com/faq/crypto_sound.html>.
- "What is LavaRnd? LavaRnd is a Random Number Generator". 2003. LavaRnd. <<http://lavarand.com/what/index.html>>
- Protego Information AB. "R200-USB™ TRNG". 25 April 2003. <http://www.protego.se/sg200_d.htm>.
- "SG100 TRNG". 4 March 2004. <http://www.protego.se/sg100_en.htm>.

- “The HCIA Cipher Technology”. 16 February 2003.
<<http://www.protego.se/pdf/hcia.pdf>>.
- Random.org. “True Random Number Service”. <<http://www.random.org/>>.
- Foley, Louise. “Analysis of an On-line Random Number Generator”. Trinity College, Dublin. THE DISTRIBUTED SYSTEMS GROUP, Computer Science Department, TCD. April 2001. <<http://www.random.org/report/Report.pdf>>.
- Haahr, Mads. “Introduction to Randomness and Random Numbers”. June 1999.
<<http://www.random.org/essay.html>>.
- “Statistical Analysis Report”. June 2001.
<<http://www.random.org/report/>>.
- “Who is using Random.org?”. Feb 2000.
<<http://www.random.org/users.html>>.
- RSA Data Security. “Hardware-based Random Number Generation -- An RSA Data Security White Paper”. Security Dynamics. 1999.
<http://cnscenter.future.co.kr/resource/crypto/algorithm/random/intel_rng_v2.pdf>.
- Szabo, Nick. “The Abuse of Statistics in Cryptography”. 1999.
<<http://szabo.best.vwh.net/statistics.html>>.
- Tundra Semiconductor Corp. “8000 Series & DES Components -- Random Bit Generator (RBG 1210)”. 2004.
<<http://www.tundra.com/Products/8000Series/index.cfm>>.
- “RBG 1210 Random Bit Generator”. Encryption Products Catalogue. 9 August, 2004. pgs 3-71 – 3-74.
- Uner, Eric. “Generating random numbers”. Embedded Systems Programming. 24 May 2004. <<http://www.embedded.com/showArticle.jhtml?articleID=20900500>>.
- Wagner, David. “Randomness for Crypto”. Berkeley.
<<http://www.cs.berkeley.edu/~daw/rnd/>>.
- Ware, Will. “Hardware Random Bit Generator”. <<http://willware.net:8080/hw-rng.html>>.
- Westphal Electronic. “ZRANDOM USB - True Random Number Generator”. 16 August 2004. <<http://home.t-online.de/home/p.westphal/zranusbe.htm>>.
- Wikipedia. “Blum Blum Shub”. 4 Nov 2004.
<http://en.wikipedia.org/wiki/Blum_Blum_Shub>.

“Pseudorandom Number Generator”. 7 Nov 2004.
<http://en.wikipedia.org/wiki/Pseudorandom_number_generator>.

Wright, Charles. “Deterministic Random Number Generators:”. Oct. 2004.