Introduction

There is no random number by itself, the concept of random number Is talking about sequence of random distribution.

Ex) if you pick a number from 0 to 10 ,let's say you pick 7, I can't predict the next number you may pick and that is it one outcome doesn't tell anything about the next one !.

Random Number Types

1) True RN like physical process in nature or when you flip a coin or a dice serval times ...etc.

Note: SWs can't generate TRN without using extra HW (e.g. circuit to measure thermal noise), SW are deterministic they only perform math with numbers stored in the memory and take decisions.

So True RN generating isn't fast or predicated or reproducible and of course not periodic.

2)Pseudo RN which depends on math functions and needs a seed

like C rand () function

C rand () implementation is S(i+1) =111035245 S(I)+12345 mod 2^31

Where s(I) is the I/p

S(I+1) is the o/p

12345 is the fixed seed

The fixed seed issue makes the next outcome pretty predictable and reproducible so pseudo RNG with fixed seed can't be used in cryptography.

What we can do to make pseudo RN predictability less? Using different seed each time + complex algorithm+ generating a really large number if we have 256bit number we have a period of 2^256 for each seed !

examples : Mersenne Twister ,TinyMt , Xorshift ,PCG,…..

3) Cryptographically secure pseudo RN they are using

Cryptographic functions which are very very secure(one way functions) like hash ,ciphers ,... so they can't be predicted.

Examples : openSSl library.

How to choose a better PRNG algorithm??

It should be:

-*easy to use.*

-*very fast.*

-*occupy little space.*

-*small code size.*

-*very good performance in* statistical *tests.*

- less predictable and thus more secure.

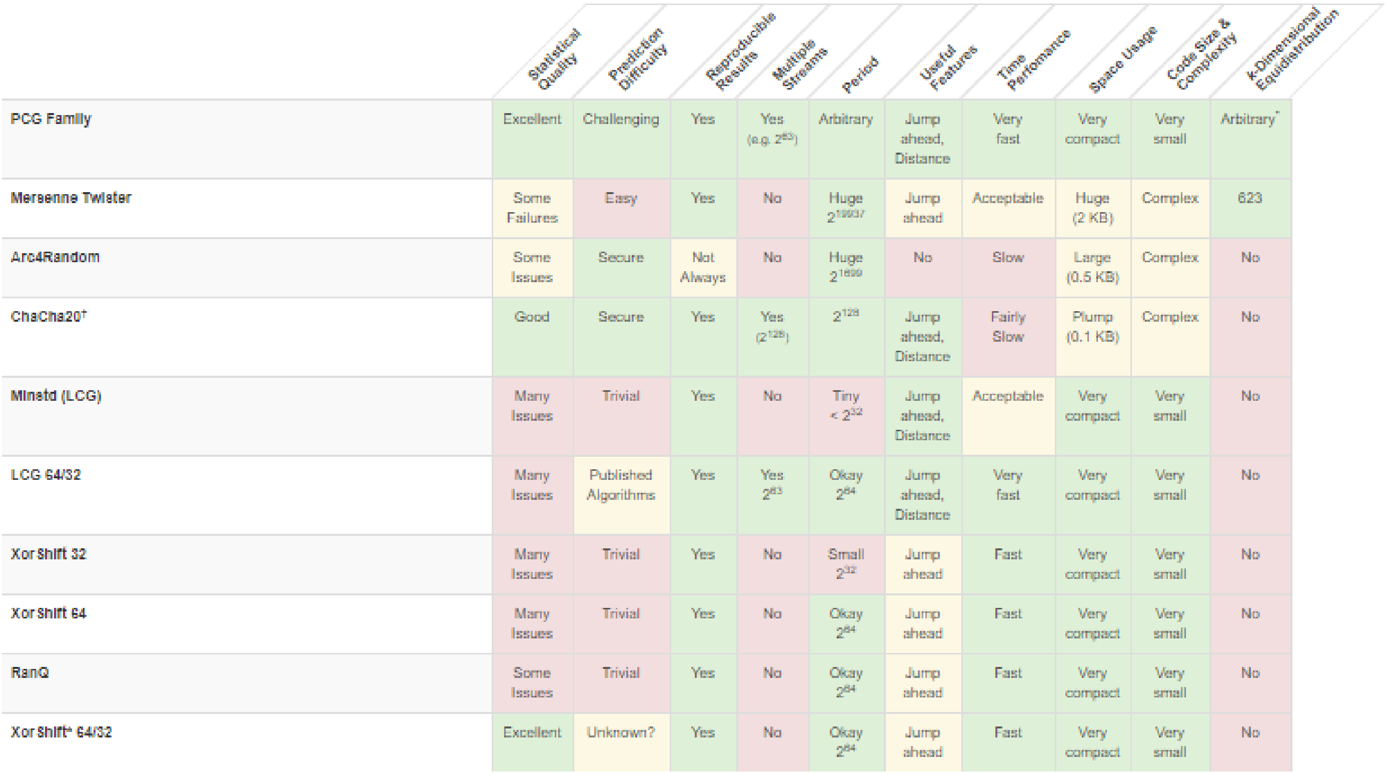
Note :

if a given PRNG does *not* look good statistically, then it is utterly proven to be pure junk. On the other hand, good statistical randomness does not tell you much with regards to cryptographic security. Cryptographic security is about whether the PRNG output could be predicted by a sentient attacker who knows the in and outs of your algorithm (but not its internal state). **Statistical randomness** is about whether the PRNG output could be predicted by a trained monkey.

For more info about RNG testing , please visit

https://crypto.stackexchange.com/questions/394/what-tests-can-i-do-to-ensure-my-random-number-generator-is-working-correctly

Why PCG algorithm??



	Statistical Quality	Prediction Difficulty	Reproducible Results	Multiple Streams	Period	Useful Features	Time Perfomance	Space Usage	Code Size & Complexity	k-Dimensional Equidistribution
PCG Family	Excellent	Challenging	Yes	Yes (e.g. 2^{63})	Arbitrary	Jump ahead, Distance	Very fast	Very compact	Very small	Arbitrary*
Mersenne Twister	Some Failures	Easy	Yes	No	Huge 2^{19937}	Jump ahead	Acceptable	Huge (2 KB)	Complex	623
Arc4Random	Some Issues	Secure	Not Always	No	Huge 2^{1699}	No	Slow	Large (0.5 KB)	Complex	No
ChaCha20†	Good	Secure	Yes	Yes (2^{128})	2^{128}	Jump ahead, Distance	Fairly Slow	Plump (0.1 KB)	Complex	No
Minstd (LCG)	Many Issues	Trivial	Yes	No	Tiny $< 2^{32}$	Jump ahead, Distance	Acceptable	Very compact	Very small	No
LCG 64/32	Many Issues	Published Algorithms	Yes	Yes 2^{63}	Okay 2^{64}	Jump ahead, Distance	Very fast	Very compact	Very small	No
XorShift 32	Many Issues	Trivial	Yes	No	Small 2^{32}	Jump ahead	Fast	Very compact	Very small	No
XorShift 64	Many Issues	Trivial	Yes	No	Okay 2^{64}	Jump ahead	Fast	Very compact	Very small	No
RanQ	Some Issues	Trivial	Yes	No	Okay 2^{64}	Jump ahead	Fast	Very compact	Very small	No
XorShift* 64/32	Excellent	Unknown?	Yes	No	Okay 2^{64}	Jump ahead	Fast	Very compact	Very small	No

For more detailed comparisons between the algorithms in the above pictures please visit https://www.pcg-random.org/