

Software vs. Hardware RNGs

If I had a dollar for every time I've been asked "Can you recommend an RNG?"... Well, you know then ending of that sentence, so I'll help you with the rest. There are two basic types of RNGs: hardware and software. What is surprising is that many gaming operators do not clearly understand the strengths and weaknesses associated with each, let alone the meaningful differences between the two. Improper design of your RNG, be it software or hardware, could leave you wide open to a number of vulnerabilities, including predictability – the proverbial Achilles' heel of electronic gaming. These attacks can be very costly and embarrassing to operators, software suppliers, and regulatory bodies alike.

So how do you choose which type of RNG is right for you? In order to make this decision, one must first obtain a clear understanding of the inner workings of each. One must also consider methods used by attackers to predict game outcomes. To be cognisant of how to defend against these attacks, one must think like an attacker. In doing so, RNGs should always be designed and implemented defensively.

Software RNGs

At the heart of every software RNG is an algorithm. Software RNGs are also known as pseudo-RNGs because these algorithms generate outcomes that only *appear* to be random. Some would argue that the pseudo-random behaviour of software RNGs makes them inferior to hardware RNGs. This is a classic misconception. Just because software RNGs are only pseudo-random, doesn't mean they can't do the job. When implemented correctly, software RNGs can be sufficiently random to thwart even the most expert and informed attacks. Conversely, when designed poorly, software RNGs can put you out of business.

The first matter to consider is the choice of algorithm. An algorithm is a complex mathematical equation, which given any particular input will in turn produce a specific output. Not all algorithms are created equal. Some algorithms are entirely inadequate for high-risk applications like gaming. Other algorithms may be able to support simple games, but will fail miserably when called upon to support the more complex ones. It will likely take a skilled mathematician to advise you on your choice of algorithm.

Under the normal operation of a software RNG, the algorithm will generate an ongoing string of outcomes. As each outcome is generated and sent to a game, that same value is also fed back into the algorithm, which will in turn be used as the seed to produce the next outcome. Eventually, after a gargantuan number of outcomes have been generated, the algorithm will begin to generate exactly the same string of outcomes all over again. The number of outcomes generated by the algorithm before it starts to repeat itself is called the 'Period'. The Period of stronger algorithms is typically far greater than that of weaker algorithms. Since more complex games characteristically demand a greater Period from the RNG, the types of games you have should directly impact your choice of algorithm.

Since each outcome is systematically fed back into the algorithm to be used as the seed to create the next outcome, how do you create the first outcome to set the process in motion? The process of initializing a software RNG upon start-up is called 'Seeding'. In order to get things going, the software RNG must look to a specified location to find its initial seed values. These values must be sufficiently random and secure to be effectively hidden from attackers. If an attacker can learn the initial seed values for your RNG, they may be able to predict the resultant string of outcomes. That in mind, a good source of Seeding is your first layer of defence against prediction attacks.

The second, and most critical layer of defence against predictability, is called 'Background Cycling'. Unfortunately, this is also the most common area for costly mistakes in software RNG design. With Background Cycling implemented properly, a software RNG will generate outcomes, at a highly accelerated and variable rate, whether or not outcomes are actually required by a game at any given point in time. Effectively, what this means is that the algorithm is in a constant state of motion, thus preventing an attacker from locking down the Period and thereby determining what the next outcome will be.

The final major issue to be considered in your design is Scaling & Mapping. Most software RNGs output extremely large numbers; 32-bit values are not uncommon ($2^{32} = 4,294,967,296$). Naturally, these titanic numbers must be scaled down to more useable values, such as 52 for a deck of cards. Subsequent to Scaling, each number must be mapped to a symbol used in a game, for example the number 52 could be mapped to the Ace of Spades. This is another common area for errors in RNG design. Scaling in particular often wreaks havoc with the quality of output, causing biases to particular outcomes. Scaling & Mapping must be carefully designed, and properly implemented in order to ensure that the software RNG operates correctly.

Considering all of these facts about software RNGs, is a software RNG right for your gaming system? Let's look at the pros and cons...

Pros:

1. Since software RNGs are created from source code, they do not grow old, wear out, or break down,
2. You don't have to buy a software RNG because algorithms are free, and
3. Since software RNGs are far less dependent on the hardware upon which they operate, they have a higher potential for portability from one gaming system to the next.

Cons:

1. Due to the pseudo-random nature of software RNGs, special care must be taken to prevent the outcomes from being predictable, and
2. Software RNGs are only pseudo-random. Although they are random for all intents and purposes, the layperson may perceive otherwise.

Hardware RNGs

Hardware RNGs are altogether very different. Hardware RNGs are comprised of a physical hardware device (usually an electronic card that plugs into a computer) complete with special interface software. Hardware RNGs are capable of truly random output. Since hardware RNGs do not depend on an algorithm, such factors as Period, Seeding and Background Cycling simply do not apply. However, there are other issues that must be addressed when using hardware RNGs.

The first matter to consider is the choice of physical hardware device. There is a wide selection of devices on the market today, most of which are based on taking samples of real-life random events such as radioactive decay or atmospheric noise. Much like algorithms, not all physical hardware devices are created equal. Some may not actually exhibit the advertised randomness, or may have strict requirements as to their compatibility with computer hardware and operating systems. Your best bet is to choose a reputable manufacturer with a tried and tested product.

The second issue that must be addressed is hardware to software interfacing. Care must be taken to ensure that the physical hardware device outcomes are not being adversely manipulated by the interface software. Each operating system will have different port drivers, which will in turn interact differently with physical hardware devices.

The final matter to consider is again Scaling & Mapping. Just like software RNGs, hardware RNGs also output extremely large numbers. These numbers must be scaled down to more useable values, and then mapped to symbols used in the game. As before, this remains a common area for potential errors in RNG design.

Considering all of these facts about hardware RNGs, is a hardware RNG right for your gaming system? Let's look at the pros and cons.

Pros:

1. Since hardware RNGs do not depend on an algorithm, factors such as Period, Seeding and Background Cycling need not be addressed, and
2. Hardware RNGs are truly random. This can go a long way with investor and public perception.

Cons:

1. Since hardware RNGs are comprised of physical hardware devices, they can potentially wear out and break down over time,
2. You must purchase a hardware RNG from a suitable manufacturer, and
3. Hardware RNGs are more dependent on the computer upon which they operate, so they have less potential for portability.

Conclusion

You must weigh out the pros and cons of each type of RNG before you can decide which one is right for you. With either type of RNG, as long as you design it correctly, and seek independent verification of your implementation, you can avoid being taken for a costly ride.



Noah Turner is the Laboratory Manager of Technical Systems Testing (TST), an internationally recognized Accredited Testing Facility (ATF) offering evaluation and consultation services for land-based and interactive-based gaming, lottery and IT industries.