## User

For generating Hardware Based Random Numbers (HRNG), I have been advised as under for rust code:

If you need to generate random numbers using the hardware-based DRNG, you may consider using a crate like getrandom which provides a cross-platform interface for secure random number generation, including utilizing hardware-based random number generators when available.

Does this crate is more authentic for generating HRNG or the good old intel intrinsic rdrand64\_step?

## Bing

The getrandom crate and the rdrand64\_step intrinsic both provide ways to generate hardware-based random numbers, but they have different considerations:

* **getrandom crate: This crate provides a cross-platform interface for secure random number generation [4]. It's designed to be easy to use and to work across a wide range of platforms. It can utilize hardware-based random number generators when available [4].**
* **rdrand64\_step intrinsic: This is a specific intrinsic provided by Intel for their processors [1] [3]. It directly uses the RDRAND instruction provided by the CPU to generate a random number [1] [3]. However, it's worth noting that this intrinsic is specific to Intel (and compatible) processors [1], and may not be available or may work differently on processors from other manufacturers.**

In terms of authenticity, both methods should provide high-quality random numbers if used correctly. The choice between them may depend more on your specific needs, such as whether you need cross-platform compatibility (in which case getrandom might be more suitable), or whether you're specifically targeting Intel processors and want to use the features they provide (in which case rdrand64\_step might be more suitable). It's also worth noting that the getrandom crate may use rdrand64\_step under the hood on platforms where it's available [4].

Please consult the documentation and consider your specific use case and requirements when choosing a method for random number generation.

Source: Conversation with Bing, 2/2/2024

1. [rand 0.8.5 - Docs.rs](https://docs.rs/crate/rand/latest)
2. [Availability of \_rdrand\_step intrinsics under Microsoft compilers?](https://stackoverflow.com/questions/33072605/availability-of-rdrand-step-intrinsics-under-microsoft-compilers)
3. [\_rdrand64\_step in core::arch::x86\_64 - Rust - Learn Rust](https://doc.rust-lang.org/beta/core/arch/x86_64/fn._rdrand64_step.html)
4. [How to add RDRAND instruction into 64-bit code compiled with VS 2008?](https://stackoverflow.com/questions/38044484/how-to-add-rdrand-instruction-into-64-bit-code-compiled-with-vs-2008)

## User

Okay, but what is the methodology of getrandom crate for generating HRNG because I do not want to use Pseudo Random Numbers (software based) (PRNG)? I mean whether getrandom crate captures some kind of random signals from hardware and convert them into bits / bytes and finally to digits or is there some other method?

## Bing

The getrandom crate in Rust provides an interface to the operating system's random number generator [2]. The exact methodology it uses to generate hardware-based random numbers (HRNG) depends on the target operating system [2]. Here are a few examples:

* **On Linux and Android, it uses the** getrandom **system call if available, otherwise it reads from** /dev/urandom **after successfully polling** /dev/random **[2].**
* **On Windows, it uses the** BCryptGenRandom **function [2].**
* **On macOS, it uses the** getentropy **function [2].**

In general, these methods work by interfacing with the hardware of the system to gather entropy, which is then used to generate random numbers [3]. The exact source of this entropy can vary, but it often includes things like timings of hardware interrupts, disk operations, and other hard-to-predict events.

It's worth noting that while the getrandom crate can utilize hardware-based random number generators when available, it also includes fallbacks for systems where a hardware-based generator is not available [2]. However, these fallbacks are designed to still provide a high level of randomness and security [2].

So, in essence, the getrandom crate does capture some kind of random signals from the hardware and convert them into bits/bytes and finally to digits, but the exact process can vary depending on the operating system and the specific hardware of the system. It's always a good idea to consult the documentation for the getrandom crate and your specific operating system for more detailed information.

Source: Conversation with Bing, 2/2/2024

1. [getrandom - Rust - Docs.rs](https://docs.rs/getrandom/latest/getrandom/)
2. [How are AES-256 keys generated using hardware random number generator?](https://security.stackexchange.com/questions/254175/how-are-aes-256-keys-generated-using-hardware-random-number-generator)
3. [rand 0.8.5 - Docs.rs](https://docs.rs/crate/rand/latest)

## User

You said:

On Windows, it uses the BCryptGenRandom function.

Is that not software based pseudo RNG?

## Bing

The BCryptGenRandom function in Windows is part of the Cryptography API: Next Generation (CNG) and is used to generate random numbers [2]. While it is a software function, it's important to note that it doesn't generate pseudo-random numbers in the traditional sense. Instead, it uses a pseudo-random number generator algorithm that is fed by an entropy source provided by the Windows kernel [3]. This entropy source can include various hardware events, which makes the generated numbers hard to predict and suitable for cryptographic purposes [3].

The BCryptGenRandom function implements an algorithm for generating random numbers that complies with the NIST SP800-90 standard, specifically the CTR\_DRBG portion of that standard [2] [5]. This means that while the function itself is software-based, the random numbers it generates are of a high quality and suitable for cryptographic use [2] [5].

So, while BCryptGenRandom is a software function, it's designed to provide secure, high-quality random numbers by utilizing entropy from the underlying system, which can include hardware sources [3]. It's not a pseudo-random number generator in the sense of being entirely deterministic or predictable like some simpler software-based random number generators can be. It's always a good idea to consult the documentation for the BCryptGenRandom function and your specific operating system for more detailed information.

Source: Conversation with Bing, 2/2/2024

1. [BCryptGenRandom function (bcrypt.h) - Win32 apps | Microsoft Learn](https://learn.microsoft.com/en-us/windows/win32/api/bcrypt/nf-bcrypt-bcryptgenrandom)
2. [Difference between CryptGenRandom and CNG BCryptGenRandom APIs](https://stackoverflow.com/questions/37290697/difference-between-cryptgenrandom-and-cng-bcryptgenrandom-apis)
3. [Conformance of Windows Random Number Generator (SP800-90A) with Key ...](https://crypto.stackexchange.com/questions/108104/conformance-of-windows-random-number-generator-sp800-90a-with-key-lifecycle-se)
4. [Random number generator security: BCryptGenRandom vs ...](https://stackoverflow.com/questions/19201461/random-number-generator-security-bcryptgenrandom-vs-rngcryptoserviceprovider)
5. [Secure Random Generators (CSPRNG) - Practical Cryptography for ... - Nakov](https://cryptobook.nakov.com/secure-random-generators/secure-random-generators-csprng)

## User

Okay - That is about random numbers. My next question is does rust provide symbolic expressions library like python (sympy) which is actually missing in C / C++. So for example, if I want to find integration of logx without limits (indefinite integration) I cannot do that in C / C++ but python has ready library - sympy. Does rust has any such library?

Another question is whether rust has any crates for calculus related problems in general - like limits, derivatives, integration, vectors, metrices, etc.

## Bing

Yes, Rust does have libraries for symbolic mathematics and calculus:

1. **Symbolic Mathematics: The** rusymbols **crate is a Rust library for symbolic mathematics [1] [3]. It aims to become a full-featured computer algebra system (CAS), similar to SymPy in Python [1]. Another option is the** symbolic\_math **library [4], which facilitates representation and manipulation of mathematical expressions.**
2. **Calculus: The** calculus **crate in Rust provides some functionality for calculus [5]. However, it's worth noting that the Rust ecosystem for mathematical and scientific computing is not as mature as Python's, so some functionality might be missing or not as fully developed.**

For more advanced mathematical operations like working with vectors and matrices, you might want to look into the nalgebra crate [6], which provides comprehensive support for linear algebra.

Remember to always consult the documentation for these crates to understand their capabilities and limitations. Also, keep in mind that the Rust ecosystem is continuously evolving, so new libraries and features are being added regularly..

Source: Conversation with Bing, 2/2/2024

1. [rusymbols - Rust - Docs.rs](https://docs.rs/rusymbols/latest/rusymbols/)
2. [rusymbols — Rust math library // Lib.rs](https://lib.rs/crates/rusymbols)
3. [GitHub - nathan-barry/symbolic\_math: A Rust library for performing ...](https://github.com/nathan-barry/symbolic_math)
4. [calculus - Rust - Docs.rs](https://docs.rs/calculus/latest/calculus/)
5. [Curated list of must know rust crates : r/rust - Reddit](https://www.reddit.com/r/rust/comments/ul66bh/curated_list_of_must_know_rust_crates/)
6. [Computer Algebra System in Rust](https://users.rust-lang.org/t/computer-algebra-system-in-rust/49016)
7. [crates.io: Rust Package Registry](https://crates.io/)
8. <https://github.com/s3bk/bullet>
9. <https://github.com/s3bk/bullet/blob/master/tests/diff.rs>
10. [en.wikipedia.org](https://en.wikipedia.org/wiki/Rust_(programming_language))

## User

Does C / C++ have any equivalent of getrandom crate in Rust?

## Bing

In C and C++, random numbers are typically generated using functions provided by the standard library, such as rand() in C and std::rand() in C++. However, these functions are not equivalent to the getrandom crate in Rust because they do not provide a direct interface to the operating system's random number generator.

For generating random numbers that are more suitable for cryptographic use, you might use platform-specific APIs. For example, on Linux, you could read from the /dev/urandom or /dev/random devices [1]. On Windows, you could use the CryptGenRandom function or the newer BCryptGenRandom function [1].

However, these methods require more work and platform-specific knowledge than using the getrandom crate in Rust, which abstracts away these details and provides a consistent interface across platforms [1].

There are also libraries available for C and C++ that provide higher-level interfaces for random number generation, such as OpenSSL's RAND\_bytes function. But again, these are not direct equivalents to the getrandom crate.

In summary, while C and C++ do provide ways to generate random numbers, they do not have a direct equivalent to the getrandom crate in Rust. The method you choose to use will depend on your specific needs and the platforms you are targeting. It's always a good idea to consult the documentation for the functions and libraries you are using for more detailed information.

Source: Conversation with Bing, 2/2/2024

1. [getrandom - Rust - Docs.rs](https://docs.rs/getrandom/latest/getrandom/)
2. [rand\_core - Rust - Docs.rs](https://docs.rs/rand_core/latest/rand_core/)
3. [getrandom 0.2.11 - Docs.rs](https://docs.rs/crate/getrandom/latest)
4. [fastrand - Rust - Docs.rs](https://docs.rs/fastrand/latest/fastrand/)

## User

Do CryptGenRandom and BCryptGet Random generate HRNG like getrandom crate does in rust?

## Bing

Yes, both CryptGenRandom and BCryptGenRandom are designed to generate cryptographically secure random numbers in Windows, similar to the getrandom crate in Rust [1].

* CryptGenRandom is a function provided by the Windows Cryptography API (CryptoAPI). It generates a random sequence of bytes [2].
* BCryptGenRandom is a function provided by the Windows Cryptography API: Next Generation (CNG). It also generates a random sequence of bytes [1].

Both functions use a pseudo-random number generator algorithm that is fed by an entropy source provided by the Windows kernel [1]. This entropy source can include various hardware events, which makes the generated numbers hard to predict and suitable for cryptographic purposes [1].

However, it's unclear if they use a hardware random source as seed (or part of the seed) [1]. So while they are designed to generate cryptographically secure random numbers, the exact process can vary depending on the operating system and the specific hardware of the system [1].

It's always a good idea to consult the documentation for these functions and your specific operating system for more detailed information. Please note that the quality of the random numbers generated by these functions should be sufficient for most cryptographic purposes [1].

Source: Conversation with Bing, 2/2/2024

1. [Random number generator security: BCryptGenRandom vs ...](https://stackoverflow.com/questions/19201461/random-number-generator-security-bcryptgenrandom-vs-rngcryptoserviceprovider)
2. [CryptGenRandom function (wincrypt.h) - Win32 apps](https://learn.microsoft.com/en-us/windows/win32/api/wincrypt/nf-wincrypt-cryptgenrandom)
3. [Difference between CryptGenRandom and CNG BCryptGenRandom APIs](https://stackoverflow.com/questions/37290697/difference-between-cryptgenrandom-and-cng-bcryptgenrandom-apis)
4. [CryptGenRandom output not getting same as rand() call](https://stackoverflow.com/questions/16901731/cryptgenrandom-output-not-getting-same-as-rand-call)