**Modern Hash Function and Pseudorandom Number Generator**

Yi Wang1,2, Diego Barrios Romero3, Daniel Lemire4, Li Jin1,2\*

1. Ministry of Education Key Laboratory of Contemporary Anthropology, Collaborative Innovation Center for Genetics and Development, School of Life Sciences, Shanghai, China.
2. Human Phenome Institute, Fudan University, Shanghai, China.
3. Fraunhofer Institute for Digital Medicine MEVIS, Bremen, Germany.
4. Université du Québec (TÉLUQ) Montreal, Canada

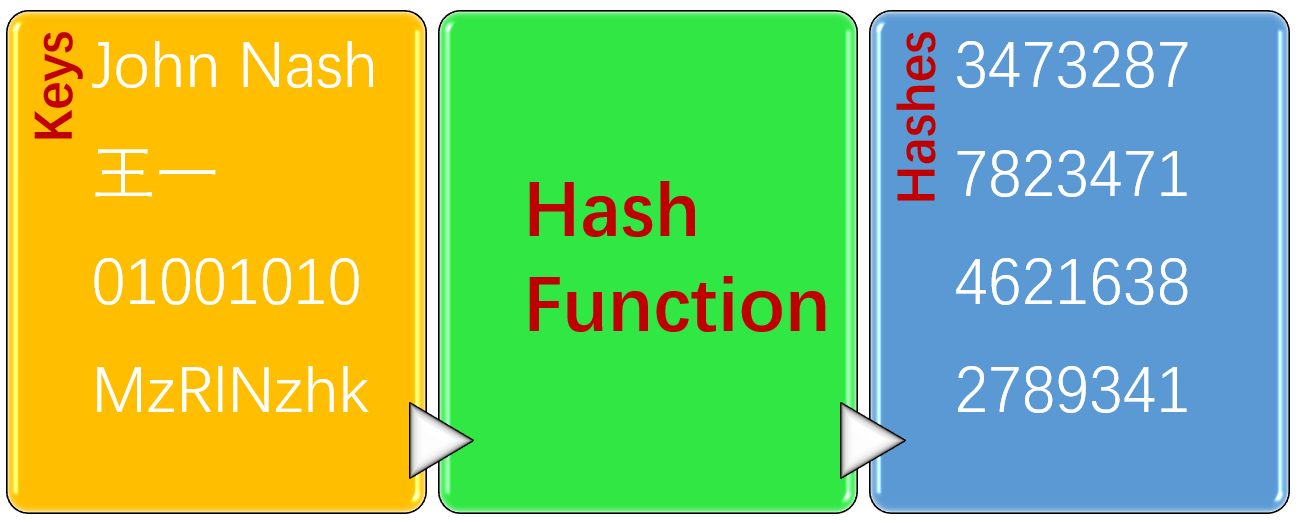
**Abstract**

Hash function and pseudorandom number generator (PRNG) are two fundamental functions in computer science with numerous applications. Due to their importance, hundreds of hash functions and PRNGs have been proposed in last decades. However, there is still no consensus non-cryptographic hash function and PRNG that possess both quality, speed, simplicity and portability. We propose wyhash and wyrand as modern non-cryptographic hash function and PRNG respectively. They are of high quality that pass SMHasher PractRand and BigCrush. Benchmark and user feedback suggest a significant speedup by replacing existing hash function and PRNG with them. They have been packed into Debian software source and become the default of the V and Zig language. wyhash and wyrand are completely free under The Unlicense at <https://github.com/wangyi-fudan/wyhash>.

**Introduction**

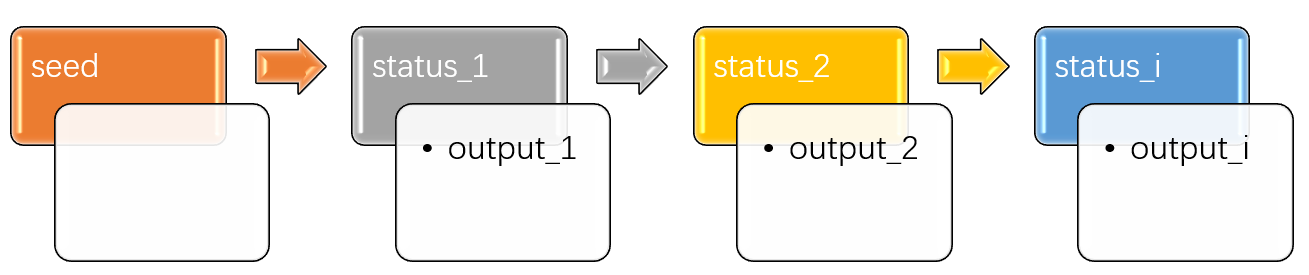
A hash function is a function that convert arbitrary data to a fixed-length integer (Figure1). The input data was called the “keys” and the output integer was called the “hashes”. Hash is a cornerstone of computer science and has numerous applications: fast algorithms, hash table, file checksum, duplication/collision detection, password storage, unique ID generation, proof-of-work, etc. (1).

***Figure 1: Illustration of hash function***



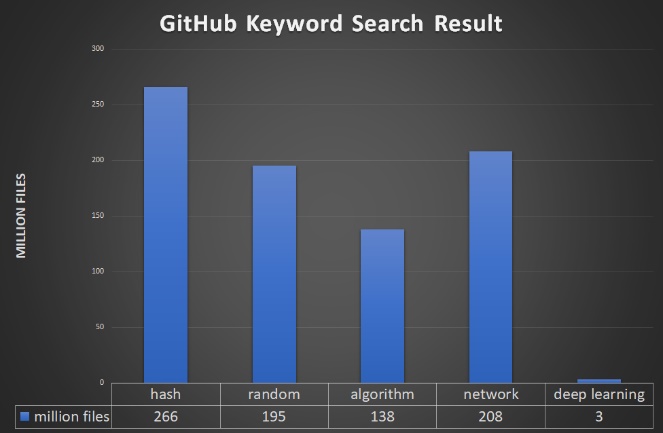
A pseudo-random number generator (PRNG) is an algorithm that can generate a stream of numbers which appears random (Figure 2). The PRNG-generated sequence is not truly random, because it is completely determined by an initial value, called the “seed”. PRNG brings “randomness” to a deterministic computer, thus has wide applications: randomized algorithm, statistical sampling, simulation, gaming, cryptography, lottery, music and art, etc. (1).

***Figure 2: Illustration of pseudo-random number generator***



To roughly access the popularity of hash function and PRNG, we searched GitHub. Figure 3 shows the number of GitHub files that contains the several keywords respectively. To our surprise, hash and random is at least as popular as algorithm and network which is well known to be key importance in computer science. Due to their popularity and importance, numerous hash functions (2) and PRNGs (3) have been designed in last decades.

***Figure 3: Number of GitHub files that contain keywords***



Despite the richness of hash functions and PRNGs, there is still no consensus non-cryptographic hash function and PRNG that possess both quality, speed, simplicity and portability. The quality of hash function and PRNG is characterized by their uniformity and independence of distribution. It is the premise of hash function and PRNG and can be evaluated by SMHasher, PractRand and BigCrush. The speed is characterized by the number of function calls per second or GB per seconds. In practice short key hashing speed attracts more attention as real key length distribution is biased to short ones. Simplicity is measured by number of instructions after compilation. Simple hash function and PRNG are not only cache efficient but also aesthetically amusing. In practice we also require portability which means the hash function and PRNG should support different machine architectures such as 32-bit/64-bit, little/big endian, aligned/unaligned memory etc.

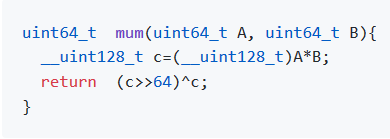
To approach the ideal hash function and PRNG, we propose wyhash and wyrand. They are of high quality that pass SMHasher, PractRand and BigCrush. They are the fastest conventional (e.g. no AVX instruction) hash function and PRNG at the premise of high quality. Their code sizes are small. They are portable to both 32-bit/64-bit, little/big endian, aligned/unaligned machine architectures. Considering the balance of these features, we bravely name wyhash and wyrand the “modern” non-cryptographic hash function and PRNG respectively. wyhash and wyrand are open sourced and were distributed under The Unlicense (<https://unlicense.org/>) which means completely free for the public.

After 18 months of exposure to public, wyhash and wyrand already have impacts on downstream applications. They have become the default for the V and Zig language. For the V language wyhash become a game changer which make its hash map faster than B-tree implementation (https://github.com/vlang/v/pull/3591). Remote desktop software xorgxrdp (https://github.com/neutrinolabs/xorgxrdp/pull/167) got 3X speedup on 4K screen by simply replacing CRC hash function with wyhash. Microsoft HoloLens project (https://github.com/microsoft/MixedReality-Sharing/issues/115) becomes “much faster” on X86 CPU by applying wyhash. Mergerfs (<https://github.com/trapexit/mergerfs/pull/805>) avoids segment fault by replacing fasthash64 with wyhash. Based on these report, we call for wider applications of wyhash and wyrand to speed up software and hence reduce the emission of carbon dioxide.

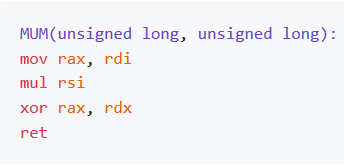
**Method**

***Mix Function:***

Wyhash and wyrand is based on a mix function call MUM that mix two 64-bit integer A and B to produce a 64-bit integer C: MUM (A, B) => C. @vnmakarov released the original version of MUM on Mother’s Day.



Despite the nominal 128-bit multiplication, the actual instructions on 64-bit machines are as simple as follow:



Our further improvements on MUM is the masked-MUM: MUM (A^secret, B^seed), where secret is a predefined 64-bit integer with 32 1bits and seed is current status with a uniform distributed number of 1bits. The masked-MUM can protect the MUM from being zero (Discussion), randomize the distribution of real data and produce an avalanche effect. We observed experimentally that just two rounds of masked-MUM suffice to pass all statistical tests.

***wyhash Hash Function***

wyhash hash function is based on masked-MUM and contains three parts: The batch part the minibatch part and finalization part. The batch part processes most of the data as 64-byte blocks while the minibatch part process the reminder of 64 bytes blocks as 16 bytes mini blocks before finalization. The finalization part processes the tail bytes (<=16). The code is shown below where the \_wyr# functions reads # byte from the key.



***wyrand PRNG***

Our PRNG is named wyrand is even simpler. It keeps a 64-bit internal status and updates it by adding a 64-bit prime. The internal status is mixed with masked itself by MUM function to produce a pseudorandom number. It is obvious that its cycle length is 2^64 as p0 is a large prime.



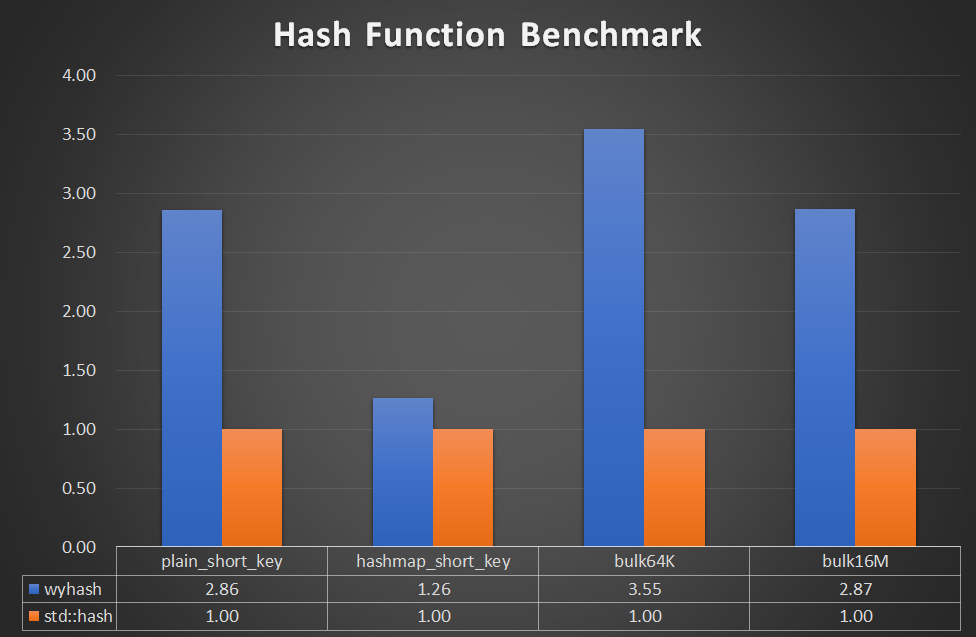
**Result**

We perform statistical quality test on wyhash by SMHasher. wyhash passed all quality tests. (supplemental material).

We performed statistical quality test of wyrand by PractRand and BigCrush via testingRNG suite. wyrand passed all tests despite of few suspicious false positives due to multiple testing (supplemental material).

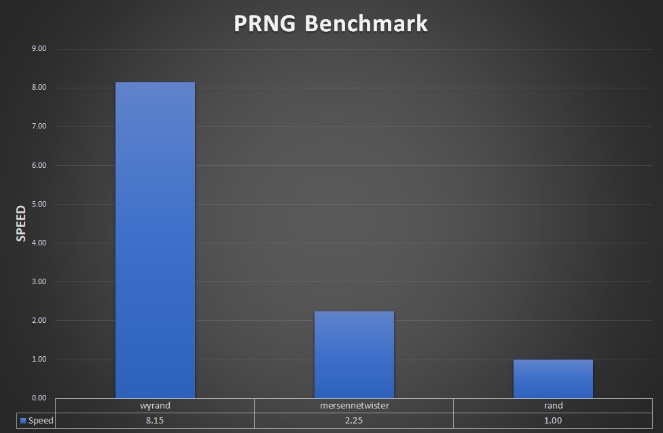
We benchmark the speed of wyhash against the C++ library hash function std::hash. Four benchmarks were carried: the plain short key hashing, the hashmap short key hashing, the 64KB bulk key hashing and the 16MB bulk key hashing. We employ /usr/shared/dict/words as testing corpus. Relative speed is shown in figure 3. SMHasher also lists benchmark of 150 hash functions, in which wyhash is the fastest one without quality problem.

***Figure 3: Hash function benchmark***



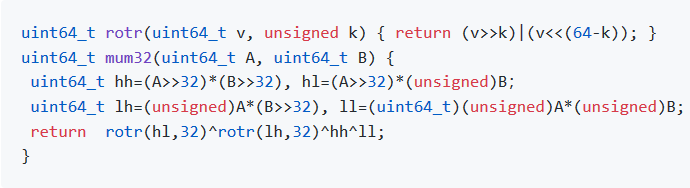
We benchmarked the speed of wyrand with rand and Mersenne Twister in testingRNG suite. We employed inlined benchmark code instead of testingRNG. Relative speed is shown in figure 4. Notably, wyrand is 8X faster than C library function rand and 3.5X faster than Mersenne Twister.

***Figure 4: PRNG Speed Benchmark***



**Discussion**

The MUM function is very fast on 64-bit machine but is slow on 32-bit machine. We proposed an alternative MUM32 function to be efficient of 32-bit machine.



As @leo-yuriev pointed out, MUM function without xoring mask is vulnerable, as MUM(0, X)=0 for any X which losses entropy. As a solution to this problem, we evolved to the masked-MUM. By keeping the mask as secrets or randomized value, MUM cannot be cracked trivially in non-cryptographic applications. However, in rare case (2-64) data^mask=0 is possible. Further protection against such cases is also available at a bit cost of speed by defining a higher security level and invoke the secure-MUM(A,B)=MUM(A,B)^A^B. It is obvious that For A=0, secure-MUM(A,B)=B will not loss entropy.

wyhash use memcpy to access memory safely. It does not do unaligned memory access which is unsafe on some machines. Wyhash does not depend on the “read through” method that read across memory bound. However, in particular cases where the short key hashing speed is very important, wyhash can use such method by defining a lower security level.

The benchmark process, in fact, is complex. There are several benchmark suits available. There are common pitfalls in these suites. First, they employ function pointers, which disables powerful function inlining. Without function inlining, the speed is compromised. Second, they use artificial data, which is either predictable or unrealistic. To avoid these pitfalls, we developed our own simple benchmark code (supplement material) and benchmark it using /user/shared/dict/word.

**Acknowledgements**

We sincerely thank the following names due to their contributions to wyhash development: Reini Urban, Dietrich Epp, Joshua Haberman, Tommy Ettinger, Otmar Ertl, cocowalla, leo-yuriev, Diego Barrios Romero, paulie-g, dumblob, Yann Collet, ivte-ms, hyb, James Z.M. Gao.

**References**

1. https://github.com/
2. <https://github.com/rurban/smhasher>
3. <https://github.com/lemire/testingRNG>