**Simple Fast Quality 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 function and PRNG have been proposed in last decades. However, there is still no consensus non-cryptographic hash function and PRNG that possese both quality, speed, simplicity and portability. We propose wyhash and wyrand as candidates of consensus 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 hash function of the V and Zig langugage. wyhash and wyrand are completely free with The Unlicense at <https://github.com/wangyi-fudan/wyhash>.

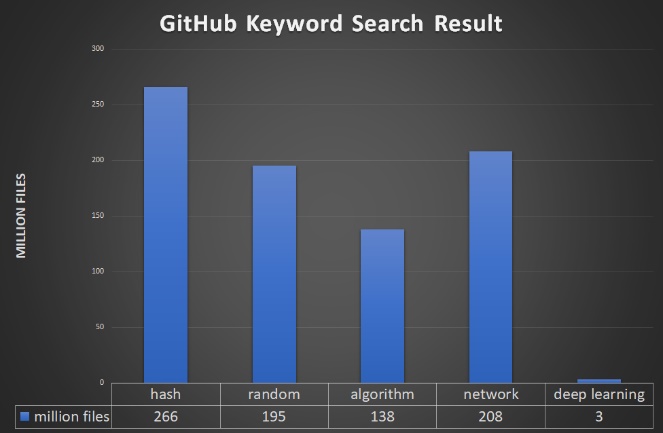
**Introduction**

A hash function is a function to convert arbitrary data to a fixed-length integer. 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).

A pseudo-random number generator (PRNG) is an algorithm that can generate a stream of numbers which appears random. 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).

To roughly access the popularity of hash function and PRNG, we searched GitHub. Figure 1 shows the number of GitHub files that contains the several keywords respectively. To our superise, hash and random is at least as important as algorithm and network which is well known to be key importance. Due to their popularity and importance, numerous hash functions (2) and PRNGs (3) have been designed in last decades. As one of the efforts, Google is still developing hash functions in last decade such as highwayhash, cityhash and farmhash.

***Figure 1: 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 possese 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 aestheticly amusing. In practice we also need portability which means the hash function and PRNG should support different machine archtectures such as 32-bit/64-bit, little/big endian, aligned/unaligned memory etc.

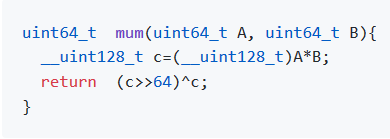
To approach the idealy hash function and PRNG, we continously developed wyhash and wyrand for one year. They are of high quality that pass SMHasher, PractRand amd BigCrush. They are the fastest hash function and PRNG that are both high quality and portable (eg. no AVX instruction). Their code sizes are small especially for wyrand. They are conventional and portable for 32-bit/64-bit, little/big endian, aligned/unaligned machine architectures. Considering the balance of these features, we bravely suggest wyhash and wyrand to become candidates of consensus non-cryptographic hash function and PRNG respectively. wyhash and wyrand is an open source project and was distributed under The Unlicense (<https://unlicense.org/>) which means completely free for the public.

With 18 monthes of exposure to public, wyhash and wyrand already have impacts on downstream applications. They have become the default hash function for the V and Zig language. For the V languge wyhash become a game changer which make hashmap faster than B-tree implementation (https://github.com/vlang/v/pull/3591). Remote destop software xorgxrdp (https://github.com/neutrinolabs/xorgxrdp/pull/167) got 3X speedup of 4K screen by simply replacing CRC hash function with wyhash. Microsoft Hololen project (https://github.com/microsoft/MixedReality-Sharing/issues/115) get “much faster” from wyhash on X86 CPU. Mergerfs (<https://github.com/trapexit/mergerfs/pull/805>) avoids segment fault by replacing fasthash64 with wyhash.

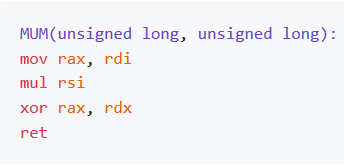
**Method**

***Mix Function:***

wyhash 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 predefined 64-bit integer with 32 1bits and seed is current status with random number of 1bits. The masked-MUM can protect the hasher, randomize biased 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: A finalization part and minibatch part and a batch part. The finalization part processes the tail bytes (<=16), 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 code is shown below where the \_wyr\* function reads \* byte from the key and \_wymix is an alias of mum.



***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.



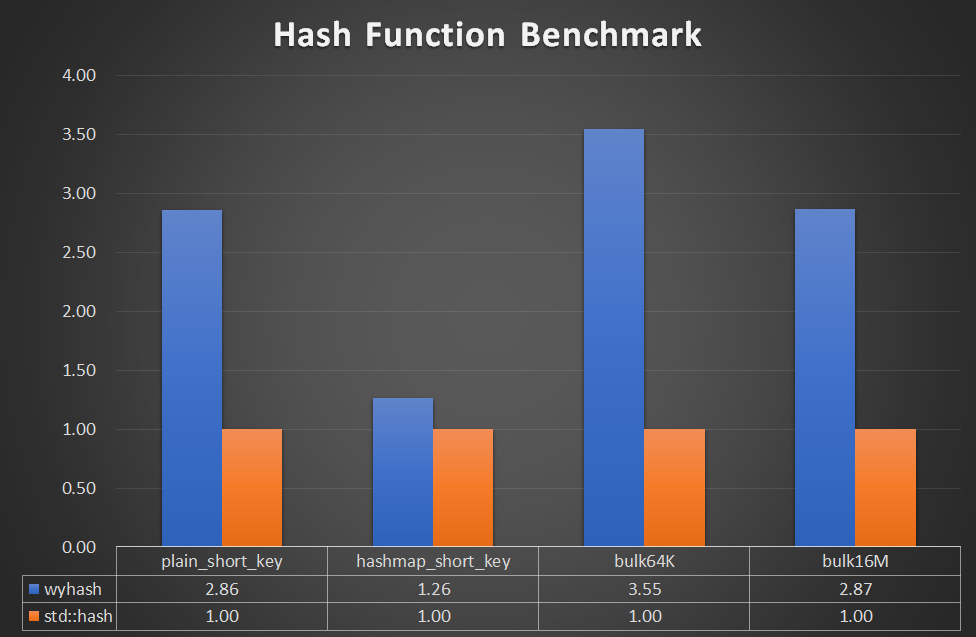
**Result**

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).

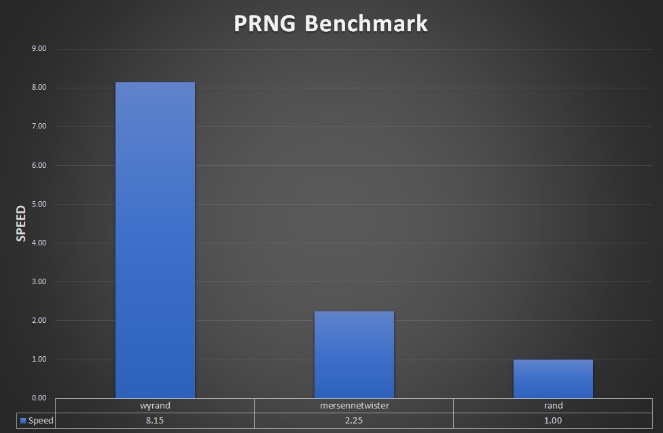
We benchmark the speed of wyhash against the C++ library hash function std::hash. Four benchmarks were carried: the plain short key hashing, the hashmap short key hashing, the 64KB bulk key hashing and the 16MB bulk key hashing. We employ /usr/shared/dict/words as testing corpus. Relative speed is shown 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***



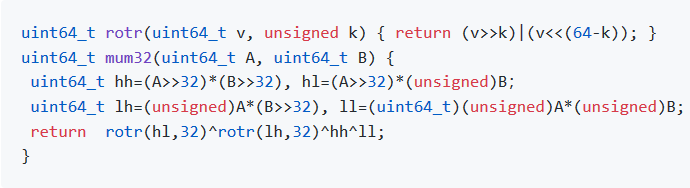
We benchmarked the speed of wyrand with rand and Mersenne Twister in testingRNG suite. We employed inlined benchmark code instead of testingRNG. Relative speed is shown in figure 4. Notably, wyrand is 8X faster than C library function rand and 3.5X faster than Mersenne Twister.

***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. As a solution to this problem, we evolved to the masked-MUM. By keeping the mask as secrets or randomized value, MUM can not be cracked trivially in non-cryptographic applications. Further protection on extreame rare chance (2-64) that the data^mask=0 is also possible by defining a higher security level and invoke secure-MUM(A,B)=MUM(A,B)^A^B. It is obvious that For A=0, secure-MUM(A,B)=B.

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 pariticular 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>