




# mmh3: A Python extension for MurmurHash3

Hajime Senuma <sup>1</sup>

<sup>1</sup> National Institute of Informatics, Japan

DOI: [10.21105/joss.06124](https://doi.org/10.21105/joss.06124)

## Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

---

Editor: [Vincent Knight](#)  

## Reviewers:

- [@mrshu](#)
- [@JPenuchot](#)
- [@cassiersg](#)

Submitted: 22 May 2023

Published: 18 January 2025

## License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

## Summary

In recent years, artificial intelligence (AI) has rapidly evolved, particularly in natural language processing (NLP) with services like OpenAI's ChatGPT. Likewise, the Internet of Things (IoT) continues to grow as a key area of ubiquitous computing, exemplified by Shodan, the first IoT search engine.

Underlying these advancements are high-performance algorithms and data structures relying on non-cryptographic hash functions, which are characteristically fast, produce statistically well-distributed bits, exhibit an avalanche effect (where a one-bit change in the input alters at least half of the output), and are collision resistant. Because cryptographic strength is unnecessary in these cases, they benefit from the efficiency of non-cryptographic hashes.

MurmurHash3 and its test suite, SMHasher, was developed by Appleby (2011) and is one of the earliest and most continuously popular hash functions specifically designed to implement the characteristics mentioned above.

mmh3 was launched in 2011 as a Python extension for MurmurHash3 and has been maintained ever since. Its API is simple to use for Python programmers, as it offers both one-shot hash functions and hasher classes that allow incremental updating, whose methods are compliant to `hashlib`, a part of the Python Standard Library. The library provides Python wheels (i.e., pre-built binary packages) for immediate use on various platforms, including Linux (x86\_64, aarch64, i686, ppc64le, and s390x), Windows (win32, win\_amd64, and win\_arm64), and macOS (Intel Mac and Apple Silicon). From version 4.0.0, mmh3 has been published under the MIT License, an OSI-approved permissive open-source license.

As of September 1, 2024, mmh3 was being downloaded more than 4 million times per month, and it ranks as the 973th most downloaded PyPI package (of around 566,000 projects), showing that only 0.17% of the remaining packages in the PyPI ecosystem are more popular (Van Kemenade et al., 2024). According to PePy, as of September 1, 2024, the total downloads of this library exceeded 130 millions.

Libraries and organizations that use mmh3 include Shodan, Microsoft Azure SDK for Python, Apache Iceberg (open table format for analytic datasets), Feast (feature store for machine learning), PyMilvus (Python SDK for Milvus, an open-source vector database), and pocsuite3 (open-source remote vulnerability testing framework).

## Statement of need

### AI and High-Performance Computing

AI is one of the most resource-demanding fields in computer science and engineering. To mitigate this problem, various techniques are employed under main systems, in which non-cryptographic hash functions play key roles in a number of algorithms and data structures.

A notable technique is *feature hashing* (Shi et al., 2009; Weinberger et al., 2009). In its simplest usage, when given a string-indexed data vector, it converts the vector into an integer-indexed data vector in which each index is the hash result of the original string index; collision values are summed. Despite its simple and intuitive usage, a machine-learning process with feature hashing is statistically guaranteed to be nearly as accurate as its original process. Feature hashing has been shown to be useful for various situations, including K-means clustering (Senuma, 2011) and succinct model learning (Senuma & Aizawa, 2016).

Other popular techniques that leverage non-cryptographic hash functions include *Bloom Filter* (Bloom, 1970), a compact data structure that tests whether an element is a member of a certain set (with false positive matches), and *MinHash* (Broder, 1997), an algorithm that quickly estimates the similarity of two sets.

mmh3 appears in scholarly papers on various topics, including Indian language NLP suites (Kakwani et al., 2020), a secure system based on probabilistic structures (Adja et al., 2021), as well as secure ciphertext deduplication in cloud storage (Tang & Jin, 2024). It has also appeared in technical books and computer science texts (Gorelick & Ozsvald, 2020; Kumar & Miglani, 2021; Medjedovic et al., 2022).

## Internet of Things

mmh3 is applicable to the IoT field. According to Shodan (2021), Shodan (Matherly, 2017) uses mmh3 as its fingerprint for a favicon (i.e., an icon associated with a web page or website). Matherly (2024) explained the adoption of mmh3 due to its speed and compact hash size, noting that cryptographic guarantees provided by md5 and other hashes were not necessary for their use case. ZoomEye, another popular IoT search engine, follows Shodan's convention.

For cybersecurity, Kopriva (2021) reported a method of discovering possible phishing websites by searching websites with Shodan, whose favicon's mmh3 hash value was the same as that of a genuine one. Another use case of mmh3 in this area includes open-source intelligence (OSINT) activities, such as measuring the popularity of web frameworks and servers, as some users do not change their default favicon settings specified by applications (Faraday Security, 2022).

## Related software

PYMMH (Kihlander & Gusani, 2013) is a pure Python implementation of the MurmurHash3 algorithms. Among various other Python bindings for non-cryptographic hashes, python-xxhash by Yue Du (Du, 2014) is another popular hash library, featuring xxHash developed by Yan Collet (Collet, 2014).

## Benchmarks

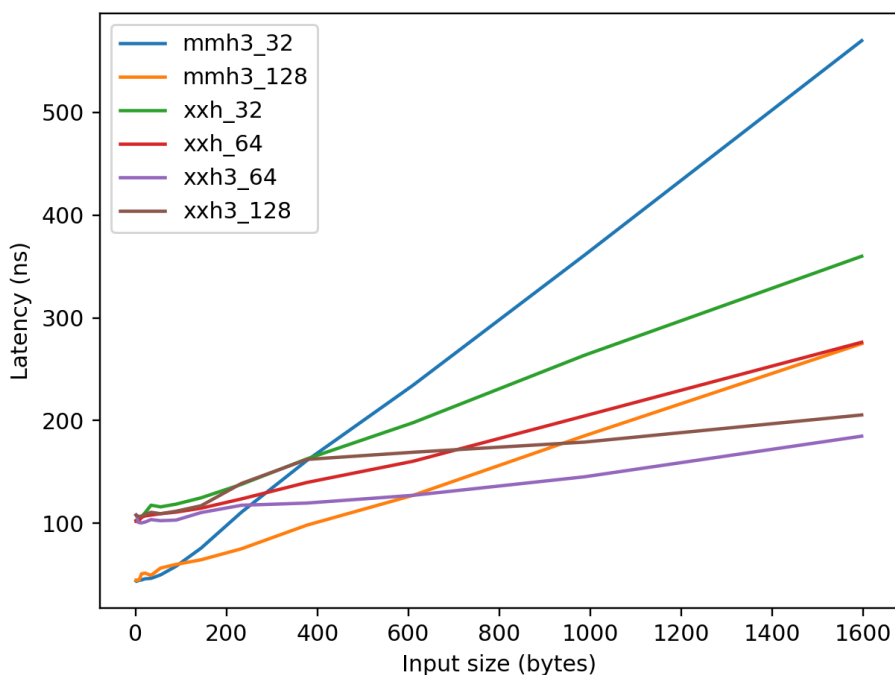
We conducted microbenchmarking experiments to compare the efficiency of Python-C hash libraries, balancing accuracy, reproducibility, and reliability. Our methodology follows practices from microbenchmarking literature, including works by Peters (2002), Stinner (2016), Collet (2020), Gorelick & Ozsvald (2020), Rodríguez-Guerra (2021), and Bernhardt (2023).

Table 1 and Figure 1 summarize the benchmarking results. While the xxh3 family in python-xxhash 3.5.0 shows superior performance for large inputs, the mmh3 5.0.0 implementation excels with smaller inputs (common scenarios for non-cryptographic hashes), due to its use of METH\_FASTCALL, an overhead-reducing interface introduced in Python 3.7.

For details, see the documentation of the project: <https://mmh3.readthedocs.io/en/latest/benchmark.html>. Additionally, the benchmarking results are publicly available as JSON files in the repository: <https://github.com/hajimes/mmh3-benchmarks>.

**Table 1:** Benchmarking results for Python extensions. Small data velocity is defined as the inverse of the mean latency (in microseconds) for inputs in the range of 1–256 bytes. Collet (2020) refers to the results of original C implementations experimented by the author of xxHash, using a CPU clocked at 3.6–4.9 GHz (ours: 2.4–3.3 GHz).

Hash	Width	Bandwidth	Small Data Velocity	cf. Collet (2020)
xxh3_128	128 bits	<b>22.42 GiB/s</b>	8.96	29.6 GiB/s
xxh3_64	64 bits	22.41 GiB/s	9.5	31.5 GiB/s
xxh_64	64 bits	8.90 GiB/s	9.3	9.1 GiB/s
<b>mmh3_128</b>	128 bits	6.91 GiB/s	<b>19.04</b>	N/A
xxh_32	32 bits	6.15 GiB/s	8.91	9.7 GiB/s
<b>mmh3_32</b>	32 bits	2.86 GiB/s	18.41	3.9 GiB/s
sha1	16 bits	1.63 GiB/s	2.4	0.8 GiB/s
md5	128 bits	0.65 GiB/s	1.95	0.6 GiB/s



**Figure 1:** Latency for small to medium-sized inputs. Lower is better.

## Acknowledgements

The author extends sincere gratitude to Akiko Aizawa for her helpful comments on this paper. Appreciation is also given to all those involved in the development and maintenance of mmh3. Special thanks go to Micha Gorelick, who made the first pull request to the project and later introduced the library in her technical book (Gorelick & Ozsvald, 2020).

## References

- Adja, Y. C. E., Hammi, B., Serhrouchni, A., & Zeadally, S. (2021). A blockchain-based certificate revocation management and status verification system. *Computers & Security*, 104, 102209. <https://doi.org/10.1016/j.cose.2021.102209>
- Appleby, A. (2011). *MurmurHash3 and SMHasher*. <https://github.com/aappleby/smhasher>
- Bernhardt, M. (2023). *On pinning and isolating CPU cores*. <https://manuel.bernhardt.io/posts/2023-11-16-core-pinning/>
- Bloom, B. H. (1970). Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7), 422–426. <https://doi.org/10.1145/362686.362692>
- Broder, A. Z. (1997). On the resemblance and containment of documents. *Proceedings. Compression and Complexity of SEQUENCES 1997*, 21–29. <https://doi.org/10.1109/SEQUEN.1997.666900>
- Collet, Y. (2014). *xxHash*. <https://github.com/Cyan4973/xxHash>
- Collet, Y. (2020). *xxHash: Performance comparison (2020)*. <https://github.com/Cyan4973/xxHash/wiki/Performance-comparison>
- Du, Y. (2014). *xxhash*. <https://github.com/ifduyue/python-xxhash>
- Faraday Security. (2022). *Understanding Spring4Shell*. <https://faradaysec.com/understanding-spring4shell/>
- Gorelick, M., & Ozsvald, I. (2020). *High performance Python: Practical performant programming for humans* (2nd edition). O'Reilly Media. ISBN: 978-1-4920-5502-0
- Kakwani, D., Kunchukuttan, A., Golla, S., N. C., G., Bhattacharyya, A., Khapra, M. M., & Kumar, P. (2020). IndicNLP Suite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for Indian languages. *Findings of the Association for Computational Linguistics: EMNLP 2020*, 4948–4961. <https://doi.org/10.18653/v1/2020.findings-emnlp.445>
- Kihlander, F., & Gusani, S. (2013). *PYMMH3*. <https://github.com/wc-duck/pymmh3>
- Kopriva, J. (2021). *Hunting phishing websites with favicon hashes*. SANS Internet Storm Center. <https://isc.sans.edu/diary/27326>
- Kumar, N., & Miglani, A. (2021). *Probabilistic data structures for blockchain-based Internet of Things applications*. CRC Press. <https://doi.org/10.1201/9781003080046>
- Matherly, J. (2017). *Complete guide to shodan: Collect. Analyze. Visualize. Make internet intelligence work for you*. (Version 2017-08-23). Shodan.
- Matherly, J. (2024). *Deep dive: http.favicon*. Shodan. <https://blog.shodan.io/deep-dive-http-favicon/>
- Medjedovic, D., Tahirovic, E., & Dedovic, I. (2022). *Algorithms and data structures for massive datasets*. Manning. ISBN: 978-1-61729-803-5
- Peters, T. (2002). Algorithms: introduction. In A. Martelli & D. Ascher (Eds.), *Python cookbook* (1st edition). O'Reilly Media.
- Rodríguez-Guerra, J. (2021). *Is GitHub actions suitable for running benchmarks?* Quansight Labs. <https://labs.quansight.org/blog/2021/08/github-actions-benchmarks>
- Senuma, H. (2011). K-means clustering with feature hashing. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Student Session*, 122–126.
- Senuma, H., & Aizawa, A. (2016). Learning succinct models: Pipelined compression with

- L1-regularization, hashing, Elias–Fano indices, and quantization. *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 2774–2784.
- Shi, Q., Petterson, J., Dror, G., Langford, J., Smola, A., & Vishwanathan, S. V. N. (2009). Hash kernels for structured data. *Journal of Machine Learning Research*, 10, 2615–2637.
- Shodan. (2021). It's the MMH3 hash of the http.html property. See: PyPI mmh3. In *Twitter*. <https://twitter.com/shodanhq/status/1395501365456261122>
- Stinner, V. (2016). *My journey to stable benchmark*. <https://vstinner.github.io/journey-to-stable-benchmark-system.html>
- Tang, X., & Jin, L. (2024). Data splitting based double layer encryption for secure ciphertext deduplication in cloud storage. *2024 IEEE 17th International Conference on Cloud Computing (CLOUD)*, 153–163. <https://doi.org/10.1109/CLOUD62652.2024.00027>
- Van Kemenade, H., Si, R., & Dollenstein, Z. (2024). *Hugovk/top-pypi-packages: Release 2024.09* (Version 2024.09). Zenodo. <https://doi.org/10.5281/zenodo.13624792>
- Weinberger, K., Dasgupta, A., Langford, J., Smola, A., & Attenberg, J. (2009). Feature hashing for large scale multitask learning. *Proceedings of the 26th International Conference on Machine Learning*. <https://doi.org/10.1145/1553374.1553516>