SCIENCE CHINA
Information Sciences

• LETTER •

September 2021, Vol. 64 199103:1–199103:3
https://doi.org/10.1007/s11432-018-9648-x

Hashing multiple messages with SM3 on GPU platforms

Shuzhou SUN[1,2], Rui ZHANG[1,2*] & Hui MA[1]

[1]*State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing* 100093*, China;*
[2]*School of Cyber Security, University of Chinese Academy of Sciences, Beijing* 100049*, China*

Received 30 June 2018/Revised 9 September 2018/Accepted 12 November 2018/Published online 16 March 2021

Citation Sun S Z, Zhang R, Ma H. Hashing multiple messages with SM3 on GPU platforms. Sci China Inf Sci, 2021, 64(9): 199103, https://doi.org/10.1007/s11432-018-9648-x

Dear editor,

The SM3 cryptographic hash algorithm [1] was approved as the only standard hash function in China and standardized as a Chinese National Standard (GB standard) in 2016. Soon after, SM3 was standardized by the International Organization for Standardization (ISO). The structure of SM3 is similar to SHA-2 family [2], which based on the Merkle-Damgård construction, with the addition of several strengthening features including a more complex step function and stronger message dependency than SHA-2.

As a standard cryptographic algorithm, the performance of SM3 on a variety of platforms should be comprehensively evaluated. Currently, there exist implementations on x86 processor [1], FPGA architecture [3, 4] and other platforms [5]. We introduce related work in Appendix A. However, to the best of our knowledge, there is no implementation on the graphics processing units (GPUs) platform, furthermore, it is desirable to improve previously known speed of the algorithm. This work fills the gap by presenting the first implementation of SM3 hash algorithm on NVIDIA's GPU devices.

Simultaneous hashing a large amount of independent messages has many potential scenarios. In a large data storage server, e.g., Amazon's S3, data deduplication techniques are always used to reduce volumes. During the deduplication phase, a great number of files are scanned and divided to chunks, which are further hashed to identify data redundancy. Another example is the TLS/SSL server workload where hash functions are used to authenticate the session data of each user. For a large e-commerce website (e.g., Amazon), or a social networking site (e.g., Facebook), a huge amount of web traffic occurs thus requiring efficient implementation of concurrent hashing.

The contributions of this study are three-fold. First, this study presents the first implementation of SM3 hash algorithm on the GPU platform. The SM3 algorithm and the CUDA framework are briefly reviewed in Appendix B. To be specific, two dedicated computation schemes are proposed to utilize the parallel power of both CPUs and GPUs. Second, to obtain a high throughput, the SM3 kernel and the memory transaction are extremely optimized. Finally, we conduct comprehensive experiments of two types of data on two types of graphics cards. For the fixed size data of 32 KB (resp. 64 KB), the throughput is 86.86 Gbps (resp. 82.58 Gbps) on GTX 1080 (resp. TITAN Xp). This performance result shows that our implementation obtains 13.28× (resp. 12.63×) speedup compared with the best SM3 implementation previously known. Meanwhile, the performance is 4.91× (resp. 4.67×) faster than the best known SHA-3 implementation. For the arbitrary length data, our implementation achieves 6.59 and 6.90 Gbps on TITAN Xp and GTX 1080, which is also comparable with the best SM3 implementations.

Implementation architecture. We propose two dedicated computation schemes here and present our optimization techniques in Appendix C.

As the hash algorithm can take an arbitrary length message as input, we build our kernels for two types of data. The first type is that all the messages have the same length which may correspond to the block-level data deduplication scenario. The second type is that each message has its own length. Hereafter, we refer these two types of data as fixed length data and arbitrary length data.

The SM3 hash algorithm contains mainly two stages: the padding stage, and the iterative expansion and compression stage. The padding stage appends some bits in the end of a message where the inserting position is determined by the message's length. When hashing multiple independent messages, if the lengths of messages are the same, no divergence occurs. However, when the lengths are different, such a length-dependent padding procedure results in unavoidable divergence on the GPU platform thus decreases the performance. Therefore, we propose two different computation modes for this two types of data.

In a simultaneous hashing scenario, multiple messages, which can be network traffic or local files, are first copied to the CPU memory. Then according to the type of data,

* Corresponding author (email: r-zhang@iie.ac.cn)

Table 1 Performance comparison with related work

Data type	Implementation	Platform	Latency (ms)	Throughput (Gbps)
Fixed length	SM3 ([6])	ASIC	–	6.54
	Keccak-256 ([7])	GTX 295	–	17.70
	Plain-SM3	GTX 1080	3.38	39.71
		TITAN Xp	2.52	53.26
	Optimized-SM3	GTX 1080	395.59	86.86
		TITAN Xp	832.17	82.58
Arbitrary length	Plain-SM3	GTX 1080	632.83	3.26
		TITAN Xp	525.26	3.92
	Optimized-SM3	GTX 1080	298.45	6.90
		TITAN Xp	312.87	6.59

different computation procedures are applied:

• Fixed size data. Since all messages have the same length, no divergence will occur in the SM3 algorithm including the padding stage. Thus the messages are transferred to the GPU device directly. The whole SM3 algorithm is executed on the device.

• Arbitrary size data. As explained before, the padding procedure results in divergent executions. It turns out that the padding is more efficient on the CPU side. Thus we let the CPU first add padding to each independent message. Since the number of messages can be very large, we also parallel this padding stage by using multiple threads on the CPU side. After padding, all messages are transferred to the GPU device and further compressed by the iterative expansion and compression stage on the GPU side.

Finally, all computed digests are copied back to CPU side. These two working modes obtain divergence-free implementations on GPUs which is crucial for the performance.

Implementation analysis. Two performance metrics are considered: throughput and latency. Throughput is the data rate, i.e., how many bits we can process in one second. Hereafter, let Mbps (one million bits per second) denote this metric. The latency is the running time of following stages: copying data from CPUs to GPUs, the SM3 kernel and copying data from GPUs to CPUs. In the case of arbitrary length data, the CPU padding time is also included in the latency. Our overall goal is to achieve a high throughput with a reasonable latency (i.e., less than 1 s).

The benchmark of our implementation is performed on two platforms: an Intel Xeon CPU E5-2609 v4 @1.7 GHz with a NVIDIA GeForce GTX 1080 GPU, and an Intel Xeon CPU E5-2667 v4 @3.2 GHz with a NVIDIA TITAN Xp GPU. On both platforms, the CPU has 128 GB memory and 8 cores. We refer these two platforms as the GTX 1080 and the TITAN Xp. For compiling we use `nvcc` (version 7.5.17) with flags `-O3`.

The benchmark procedure on two types of data is presented in Appendix D. We present the performance comparison in Table 1. The state-of-the-art implementation of SM3 algorithm is owned to [6] which achieved 6.54 Gbps on ASIC. Our implementation achieves at most 86.86 Gbps for the fixed size data of 32 KB on GeForce GTX 1080, which is 13.28× faster than [6]. On TITAN Xp, the peak throughput is 82.58 Gbps for the fixed size of 64 KB, which is 12.63× improvement. For the arbitrary length data, our achieved throughput is also comparable with [6]. We also compare the performance with state-of-the-art implementation of SHA-3 [7]. On GTX 1080 and TITAN Xp, our implementation achieves 4.91× and 4.67× improvements compared with [7].

Besides compared with exist softwares, we also conduct a plain implementation to verify that our purposed schemes and optimizations are in fact practical. Concretely, the basic implementation involves the whole SM3 algorithm on the GPU side. Meanwhile, it does not adopt the proposed optimizations. We benchmark the fixed size data and arbitrary length data. It turns out that the plain implementation achieves peak throughput 39.71 Gbps (resp. 53.26 Gbps) on GTX 1080 (resp. TITAN Xp) for the fixed size data of 8 KB (resp. 4 KB). In the arbitrary length case, 3.26 Gbps (resp. 3.92 Gbps) throughput is obtained on GTX 1080 (resp. TITAN Xp). Compared with the performance of optimized implementation, these performance result proves that our optimizations effectively improve the performance.

Conclusion. As the only approved hash algorithm in China and an ISO standard, SM3 currently has no implementation on the GPU platforms. This study fills the gap by give the first SM3 implementations on NVIDIA's GPUs. We proposed two dedicated computation modes to fully utilize the parallel computing power of both CPUs and GPUs and allow pipeline executions. We then developed a few general and hardware-specific optimization techniques to build an efficient SM3 kernel, and decrease the memory transaction latency. Finally, we conducted experiments that achieve a throughput of 82.58 and 86.86 Gbps on GTX 1080 and TITAN Xp cards, respectively, which are 12.63× and 13.28× speedup compared with the best known SM3 implementations.

Acknowledgements This work was supported in part by National Natural Science Foundation of China (Grant Nos. 61632020, 61472416, 61772520, 61602468, 61802392) and Key Research Project of Zhejiang Province (Grant No. 2017C01062).

Supporting information Appendixes A–D. The supporting information is available online at info.scichina.com and link.springer.com. The supporting materials are published as submitted, without typesetting or editing. The responsibility for scientific accuracy and content remains entirely with the authors.

References

1 Wang X Y, Yu H B. SM3 cryptographic hash algorithm. J Inform Secur Res, 2016, 2: 983–994
2 US Department of Commerce, National Institute of Standards and Technology (NIST). Secure hash standard (SHS). FIPS 180-2. 2012
3 Ao T Y, He Z Q, Dai K, et al. A compact hardware implementation of SM3. In: Proceedings of the 13th International Conference on Trust, Security and Privacy in Computing and Communications, 2014
4 Ma Y, Xia L N, Lin J Q, et al. Hardware performance optimization and evaluation of SM3 hash algorithm on FPGA.

In: Proceedings of the 14th International Conference, 2012. 105–118

5 Hu Y, Wu L J, Wang A, et al. Hardware design and implementation of SM3 hash algorithm for financial IC card. In: Proceedings of the 10th International Conference on Computational Intelligence and Security, 2014. 514–518

6 Du X J, Li S G. The ASIC implementation of SM3 hash algorithm for high throughput. In: Proceedings of IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, 2016. 1481–1487

7 Bos J W, Stefan D. Performance analysis of the SHA-3 candidates on exotic multi-core architectures. In: Proceedings of the 12th International Workshop, 2010. 279–293