# POSTER: Accelerating High-Precision Integer Multiplication used in Cryptosystems with GPUs

Zhuoran Ji
Shandong University

Zhaorui Zhang
HK Polytechnic University

Jiming Xu
Ant Group

Lei Ju
Shandong University

## Abstract

High-precision integer multiplication is crucial in privacy-preserving computational techniques but poses acceleration challenges on GPUs due to its complexity and the diverse bit lengths in cryptosystems. This paper introduces *GIM*, an efficient high-precision integer multiplication algorithm accelerated with GPUs. It employs a novel segmented integer multiplication algorithm that separates implementation details from bit length, facilitating code optimizations. We also present a computation diagram to analyze parallelization strategies, leading to a series of enhancements. Experiments demonstrate that this approach achieves a 4.47× speedup over the commonly used baseline.

***CCS Concepts:*** • **Computing methodologies** → **Massively parallel algorithms**; • **Security and privacy**;

***Keywords:*** GPU computing, big integer multiplication

**ACM Reference Format:**
Zhuoran Ji, Zhaorui Zhang, Jiming Xu, and Lei Ju. 2024. POSTER: Accelerating High-Precision Integer Multiplication used in Cryptosystems with GPUs. In *The 29th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming (PPoPP '24), March 2–6, 2024, Edinburgh, United Kingdom.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3627535.3638495

## 1 Introduction

In light of the escalating volume of digital information and rising data security risks [4, 5, 8], many privacy-preserving computation techniques have emerged. The fundamental building blocks of these techniques are cryptosystems, with many relying on hard number theory problems [2]. The primary computations of these cryptosystems involve high-precision integer multiplication, a compute-intensive operation. Currently, most cryptosystems require at least 2048-bit integers to ensure security [1], necessitating 4096 32-bit wide-multiply operations per multiplication. A single cryptographic operation requires thousands of multiplications, and most applications require processing many data items. There are parallelization opportunities both within a single multiplication and among multiple multiplications. It provides a strong incentive to accelerate them with GPUs [3, 7].

However, parallelizing integer multiplication in cryptosystems is challenging. The bit lengths involved are too extensive for single-thread processing but not large enough for effectively utilize the Number Theoretic Transform (NTT) algorithm. Thus, the schoolbook multiplication method is still preferred, but it requires distribution across multiple threads to achieve parallel processing. This distribution involves a complex balance among factors like on-chip resource usage, communication costs, parallelism, workload balance, and memory access patterns. Moreover, the bit length, which varies with the cryptosystem and desired security level, heavily influences these factors. A strategy effective for computations with $N$-bit integers may not work well for $2N$-bit integers, leading to potential performance issues.

## 2 Decouple Working Set from Bit Length

We propose a segmented integer multiplication algorithm to decouple the working set from bit length, much like tiled matrix multiplication. As shown in Figure 1, the basic unit is a fixed-size vanilla integer multiplication function, denoted as UINT_MUL. The integers are divided into fixed-size segments, and the multiplications of various bit lengths are performed via a sequence of UINT_MUL calls, followed by accumulation. The UINT_MUL operation is executed in parallel with $N_T$ threads, while the remaining operations are processed serially by the same $N_T$ threads. The parallel component is fully encapsulated within UINT_MUL. Thus, the on-chip resource usage is determined by the segment size, which is independent of the bit length. It allows the segment size to be strategically selected to facilitate performance optimization.

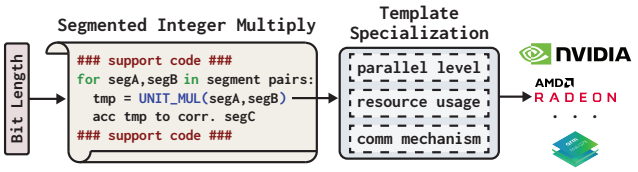


**Figure 1.** Segmented high-precision integer multiplication

## 3 Modeling with Computation Diagram

A computation diagram is proposed to illustrate the process of parallel integer multiplication, thus elucidating the challenges. As shown in Figure 3, each element represents a


*PPoPP '24, March 2–6, 2024, Edinburgh, United Kingdom*

ACM ISBN 979-8-4007-0435-2/24/03.
https://doi.org/10.1145/3627535.3638495

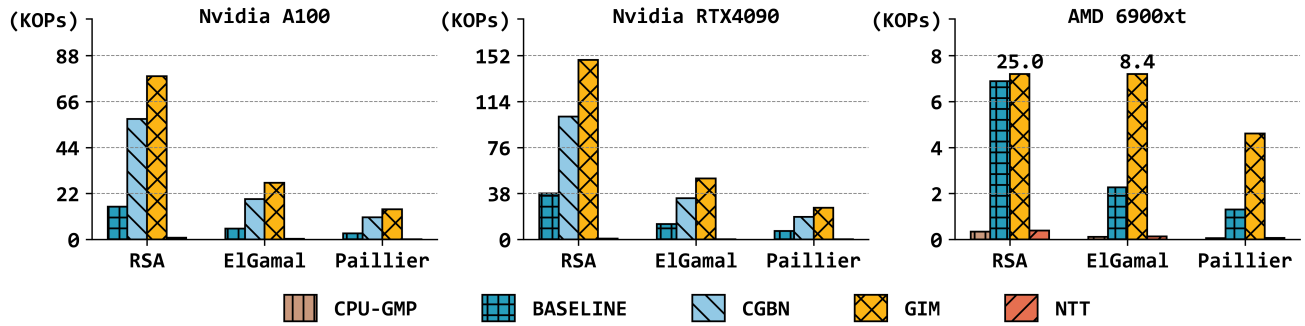


**Figure 2.** Overall throughput (encryption+decryption) of the evaluated methods for different cryptosystems

wide-multiply instruction, while the red lines symbolize addition operations. The addition operations executed along the diagonal lines facilitate the propagation of the higher 32 bits and the associated carry. The enclosed blue shape delineates the workload apportioned to each thread. The intersection between blue and red lines indicates inter-thread data dependency, and thus communication. Variations among the shapes associated with different threads, including the geometry and orientation, indicate divergence. The area of each shape functions as a measure of the allocated workload.

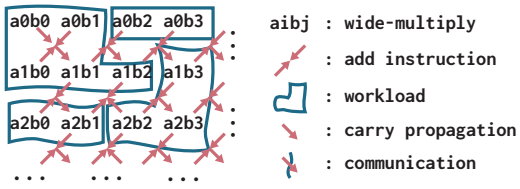


**Figure 3.** Example of a computation diagram

The computation diagram effectively illuminates the interplay among the factors that have significant impacts on performance. It promotes a two-dimensional parallelization strategy, where the calculation of each word is distributed among several threads, and each thread is also responsible for computing the partial products of several words. It opens up the potential for further optimizing on-chip resource usage, communication cost, parallelism degree, workload balance, and memory access pattern.

## 4 Evaluation

We compared *GIM* with *BASELINE*, *CGBN*, and *NTT*, along with a CPU benchmark *CPU-GMP*. *BASELINE* assigns each thread an individual integer multiplication. *CGBN* is a state-of-the-art high-precision integer arithmetic library released by Nvidia [6]. *NTT* applies an NTT-based algorithm. The tests were on Nvidia A100, RTX4090 GPUs, an AMD Radeon 6900XT, and an Intel 13900KF CPU.

Our results show *GIM*'s superior performance in RSA, ElGamal, and Paillier cryptosystems (2048-bit keys). It outperforms *CGBN* by 1.41 to 1.49 ×. Performance varied across GPUs, with greater speedups on GPUs having more integer ALUs. Remarkably, *BASELINE* surpassed *CPU-GMP* by 294.3 ×, affirming GPUs' efficiency in high-precision integer multiplication for cryptography. Both *GIM* and *CGBN* significantly outperform *BASELINE* and *NTT*, reflecting the inadequacy of single-thread and NTT approaches for current cryptographic bit lengths. *GIM*'s compatibility extends to non-CUDA GPUs but showed reduced performance on the AMD 6900XT. This drop is due to the need to replace warp data shuffle operations with shared memory data exchanges, which are less efficient.

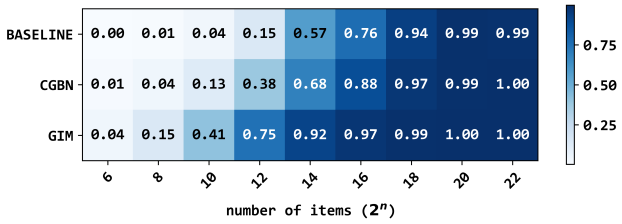


**Figure 4.** Throughput across different parallelism degrees

The previous experiments set the number of input elements to $2^{22}$, which is sufficient for all methods to fully utilize the GPU resources. We then investigate the minimum number of input elements required for each method to achieve its maximum potential throughput. Figure 4 illustrates the throughput of each method with varying numbers of input elements, normalized by the throughput achieved when the number of input elements is $2^{23}$. In order to achieve at least 97% of the maximum attainable throughput, *BASELINE*, *CGBN*, and *GIM* require a minimum of $2^{20}$, $2^{18}$, and $2^{16}$ input elements, respectively. This highlights the limited efficacy of inter-operation parallelism and underscores the need to delve into intra-operation parallelism.

## 5 Conclusion

This paper *GIM*, an efficient high-precision integer multiplication algorithm with segmented integer multiplication and two-dimensional parallelization. We hope that our study can spur further research in accelerating high-precision integer multiplication with GPUs.

## References


[1] Elaine Barker, William Barker, William Burr, William Polk, and Miles Smid. 2007. NIST special publication 800-57. *NIST Special publication* 800, 57 (2007), 1–142.

[2] Steven D Galbraith. 2012. *Mathematics of public key cryptography*. Cambridge University Press.

[3] Owen Harrison and John Waldron. 2009. Efficient acceleration of asymmetric cryptography on graphics hardware. In *Progress in Cryptology–AFRICACRYPT 2009: Second International Conference on Cryptology in Africa, Gammarth, Tunisia, June 21-25, 2009. Proceedings 2*. Springer, 350–367.

[4] Priyank Jain, Manasi Gyanchandani, and Nilay Khare. 2016. Big data privacy: a technological perspective and review. *Journal of Big Data* 3 (2016), 1–25.

[5] Fenghua Li, Hui Li, Ben Niu, and Jinjun Chen. 2019. Privacy computing: concept, computing framework, and future development trends. *Engineering* 5, 6 (2019), 1179–1192.

[6] NVIDIA Research Projects. 2021. CGBN: CUDA Accelerated Multiple Precision Arithmetic using Cooperative Groups. https://github.com/NVlabs/CGBN.

[7] Robert Szerwinski and Tim Güneysu. 2008. Exploiting the power of GPUs for asymmetric cryptography. In *Cryptographic Hardware and Embedded Systems–CHES 2008: 10th International Workshop, Washington, DC, USA, August 10-13, 2008. Proceedings 10*. Springer, 79–99.

[8] Duygu Sinanc Terzi, Ramazan Terzi, and Seref Sagiroglu. 2015. A survey on security and privacy issues in big data. In *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*. IEEE, 202–207.