Integrity and Data Confidentiality in Trustworthy Supercomputing

ASCR Cybersecurity for Scientific Computing Integrity Workshop

Jian Yin, John R. Johnson, Nick Multari
(jian.yin@pnnl.gov, john.johnson@pnnl.gov, nick.multari@pnnl.gov)
Pacific Northwest National Laboratory

Many scientific fields can benefit from fine-grained simulation and big data analytics enabled by extreme-scale supercomputers. Moreover, computing integrity and data confidentiality are critical in many scientific endeavors. Integrity is important because computed results must be trusted, accurate, and untainted. Data confidentiality could become prerequisite for data to be shared and to enable large-scale simulation and big data analytics applications.

However, existing techniques cannot adequately protect supercomputing infrastructures from faults or malicious tampering. Prevalent approaches include isolation, hardening of software and hardware, and active monitoring. However, supercomputing infrastructures generally are built to serve many diverse applications run by scientists around the world. Thus, isolation is not feasible. In addition, for any reasonably complex real-world system, it is not possible to eliminate every software and hardware bug and security vulnerability, even with the best possible engineering practices. Supercomputer architectures are performance-driven and often use cutting-edge hardware and software components. Thus, software and hardware bugs and security vulnerabilities that an adversary can exploit are expected. Once an attacker exposes a system, at least some aspects of confidentiality and integrity are immediately compromised.

This system weakness validates the need for inherently secure schemes to protect integrity and data confidentiality in a provably secure manner: one that affords an attempt to guess the content of confidential data is no better than random guessing, subject to standard computation hardness assumptions, yet allows required computations to be effectively performed on transformed data.

Regarding integrity, traditional approaches replicate computation across several machines, assuming not all machines are compromised. Then, computed results are checked against each other. However, once an adversary controls a leadership-class machine, that entity can tamper with computation results in the same way, making them agree. To address this, a dual-encryption approach, similar to those proposed in [1, 2, 3], can be employed. In this scheme, a computation is encrypted differently from the data and is performed directly on encrypted data, known as *homomorphic computing*. Because the computation is encrypted, attackers face a second layer of protection. Of note, this approach addresses integrity in the case of tampering or fault. However, it does not prevent complete data destruction. To resist data destruction, additional data replication and backup techniques need to be explored.

Extreme-scale scientific applications are compute intensive and provide unique scalability challenges to existing homomorphic computing approaches [4, 5, 6] that can introduce too much overhead. Two possible approaches can be exploited to address these issues and make inherently secure computation feasible for extreme-scale scientific applications.

First, multiple features of the high-performance computing (HPC) platform can be exploited. For example, various levels of parallelism, including inter- and intra-node and data parallel instructions, can be used for each portion of homomorphic computation. Because the amount of computation required for each gate (in homomorphic computation, computation generally is modeled as arithmetic circuits) generally is independent of the input data, good load balancing can be achieved if the cost of per-gate computation can be accurately estimated when load is distributed. Hence, good scaling can be realized as long as the impact of communication overhead is mitigated. When designing arithmetic circuits for an application, the size and amount of coarse-grained parallelism that can be extracted from

Jian Yin, John R. Johnson, Nick Multari
(jian.yin@pnnl.gov, john.johnson@pnnl.gov, nick.multari@pnnl.gov)
Pacific Northwest National Laboratory

them must be considered. For implementing a given function, there is a trade-off between the size and depth of arithmetic circuits. Performance can be optimized by customizing the trade-off based on the amount of parallelism provided. The DoubleCRT format significantly reduces time complexity for multiplication. Converting ciphertext to DoubleCRT format with the fast Fourier transform (FFT) and modulus reduction can provide efficient multiplication and modulus switching in homomorphic computation. Wide vector units and the many cores in modern processors offer good opportunities to significantly speedup FFT and modulus reduction on number-theoretic fields. Concurrent computation and data movement must be coordinated to maximize cache-hit rates. In addition, a field that allows efficient computation needs to be chosen and combined with Montgomery reduction to reduce or eliminate modulus operation overhead. Significant performance gains also can be derived by improving efficiency of memory access of homomorphic computation for the cache architecture of modern processors. In modern architectures, memory access is most efficient when aligned with cache lines. However, homomorphic computation often relies on ciphertext packing to improve performance. With ciphertext packing, permutations are required to rotate the positions of plaintexts to interact with each other. Permutation of plaintext positions is mapped to permutation of coefficients of polynomials that use ciphertext packing to encode the plaintexts. Moving data not only introduces overhead, but is also may cause unaligned memory access and pollute caches. A two-step approach can be used to address this issue. First, based on arithmetic circuits, the positions can be arranged to minimize permutations. In case some data are read-only, replicating them into multiple ciphertexts can be considered to increase the chance that at least one of them does not require rotation. This also may be achieved via virtual permutation, which can be implemented as indirections, or the algorithm can be altered to seek data in their original places rather than rotating the data. Zero-copy noncontiguous data transfers supported by various HPC network interface cards can be leveraged to rearrange data when they are transferred from one node to another node for next-stage processing.

Second, application-specific data processing characteristics can be exploited to reduce overhead. Depending on the processing required for the data in specific applications, less general but more efficient computation schemes can be used. Using a specialized scheme can be as efficient as computing over data that have not been transformed. Moreover, the computation can be decomposed into several parts, with some parts using a general scheme and others employing more efficient, less general schemes.

References

[1] H. Wang, J. Yin, C. S. Perng, and P. S. Yu, Dual Encryption for Query Integrity Assurance. Proceedings of the 17th ACM conference on Information and knowledge management, pp. 863-872 (2008). DOI: 10.1145/1458082.1458196.
[2] M. Xie, H. Wang, J. Yin, and X. Meng, Providing Freshness Guarantees for Outsourced Databases. Proceedings of the 11th International Conference on Extending Database Technology: Advances in Database Technology, pp. 323-332 (2008). DOI: 10.1145/1353343.1353384.
[3] M. Xie, H. Wang, J. Yin, and X. Meng, Integrity Auditing of Outsourced Data. 33rd International Conference on Very Large Data Bases, pp. 782-793 (2007).
[4] C. Gentry, Fully Homomorphic Encryption Using Ideal Lattices. Proceedings of the 41st ACM Symposium on Theory of Computing, pp. 169-178 (2009). DOI: 10.1145/1536414.1536440.
[5] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, (Leveled) Fully Homomorphic Encryption without Bootstrapping. Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, pp. 309-325 (2012). DOI: 10.1145/2090236.2090262.
[6] S. Halevi and V. Shoup, Algorithms in HElib, Chapter 31 in *Advances in Cryptology-CRYPTO 2014*, pp. 554-571, Springer Berlin Heidelberg (2014). DOI: 10.1007/978-3-662-44371-2_31.