On the Optimization of an Advanced Encryption Standard Algorithm on Processing-in-Memory

Hyunseob Shin*, Seunghyun Lee*
School of Electrical Engineering
Korea University, South Korea

Jahyun Koo
Department of EECS
DGIST, South Korea

Jaeha Kung
School of Electrical Engineering
Korea University, South Korea

***Abstract*—Demand for on-chip encryption is high to secure data integrity for numerous tasks. Although the advanced encryption standard is frequently used for memory encryption, data-intensive characteristic causes high energy consumption and low throughput. Thus implementing the AES algorithm with a processing-in-memory technique is emerging to mitigate those challenges. Still, optimizing for minimal area and power consumption while improving overall performance remains tasks to be addressed. In this paper, we propose to improve the AES process on the PIM architecture by introducing on-the-fly round key generation, enhanced MixColumns, and flexible cryptographic algorithm support with a RISC-V controller.**

***Index Terms*—security, cryptography, processing in memory, advanced encryption standard**

I. Introduction

There are countless attack methods to disturb the designed operation of a hardware system. Some critical memory attacks, e.g., cold boot attacks and bus snooping, can contaminate data and break signal integrity, increasing the demand for better on-chip encryption methods. Usually, Advanced Encryption Standard (AES) is used to encrypt data for secure data communication, but the AES algorithm requires a significant amount of computation which may increase the communication latency. To alleviate this issue, prior works have proposed in-memory encryption/decryption solutions [1]–[3] for higher energy efficiency. While those methods have optimized the encryption or decryption process to reduce latency and improve performance, we propose several methods to get even more performance improvement by optimizing the sub-steps of the AES algorithm. Another benefit of running encryption/decryption algorithms would be protecting the secure data, i.e., private data or encryption keys, from the bus probing attack that directly reads data coming out of the on-chip memory.

II. Background

A. Advanced Encryption Standard

Advanced Encryption Standard (AES) [4] is an encryption method for electronic data by the U.S. National Institute of Standards and Technology. AES is a symmetric-key algorithm, that uses identical keys on both the encryption and decryption processes. A key is randomized bits with a length of 128, 192, or 256 bits. It is highly recommended to use a true random number generator (TRNG), not a pseudo-random number generator (PRNG), due to security reasons. Each AES process can handle 128 bits of raw data, called *state*.

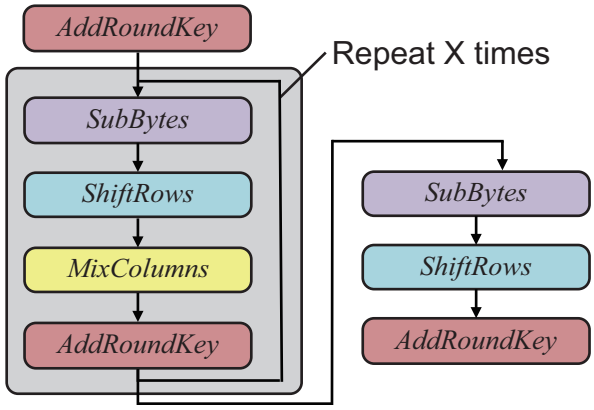

Fig. 1. Advanced Encryption Standard procedure. X equals 9, 11, and 13 times on key lengths of 128, 192, and 256 bits. `MixColumn` process does not occur on final iteration.

AES contains the repeated `AddRoundKey`, `SubBytes`, `ShiftRows` and `MixColumns` processes as shown in Fig. 1. `AddRoundKey` process, XOR result of the current state and corresponding round key is stored as a new state. For the `AddRoundKey` process, a new round key is used. Round keys are generated independently from the AES process (`KeyExpansion`). `SubBytes` is a non-linear substitution from a lookup table. The size of the lookup table is 8×256 since each value from the state can have 256 values, and the result of the substitution is also 8 bits. `ShiftRows` is a cyclic transposition of bytes within a single state. `MixColumns` is a linear mixing operation targeting columns, combining all values to the new state.

B. Processing-in-Memory (PIM)

Processing-in-memory (PIM) is a paradigm aimed at overcoming the memory bottleneck inherent in conventional von Neumann architectures by performing computations within or near the memory devices. In traditional computing systems, the fact that the computation unit needs to transfer data from the memory unit gives rise to the *memory wall* problem. This issue grows exponentially due to improvements in processor speed, which outpaces the access speed of memory. This inef-

* H. Shin and S. Lee are equally contributed authors.

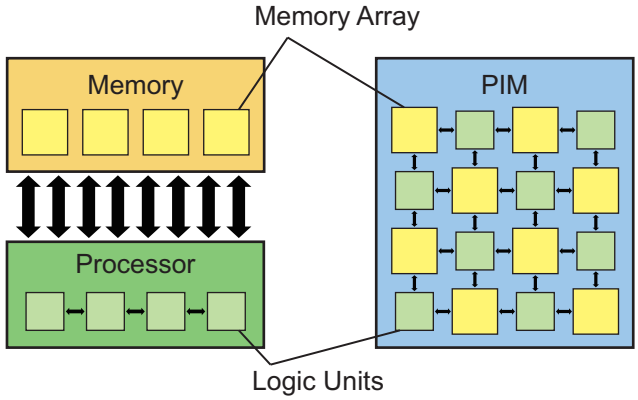

Fig. 2. Conventional von Neumann architecture (left), and an example of Processing-in-Memory architecture (right).

ficiency worsens overall system performance and contributes to significant energy inefficiency [5].

PIM addresses these challenges by integrating processing elements into memory units, allowing data to be processed at its location, thus minimizing the need for data movement between the memory system and processors. As a result, the latency associated with data transfers is reduced, and the memory wall problem is alleviated [6]. PIM is typically implemented in two distinct ways: one approach involves embedding bit-line computing units near memory, and the other is to include computation units inside memory cells. These architectures can offer significant benefits with respect to performance, energy efficiency, and security for various applications, especially those that are data-intensive or memory-bound [7].

III. Processing-in-Memory for AES

A. Previous Work

Numerous research efforts have been devoted to the integration of the AES algorithm with the PIM architectures. The AES encryption entails intricate operations, including substitution, XOR calculation, and mixing, executed on large data blocks. We can enhance the performance and energy efficiency of the AES by minimizing data movement and leveraging parallelism through PIM. However, achieving high throughput in AES PIM implementations while mitigating significant overhead in area and power consumption remains a challenge.

One approach to reduce power consumption and improve performance involves the utilization of look-up tables (LUT) [2], [3]. P. R. Sutradhar et al. have introduced an architecture that optimizes the time-consuming `SubBytes` and `MixColumns` processes by precomputing and storing the required bytes in LUTs, including data like S-box for `SubBytes` and multiplied values for `MixColumn` [2]. Although this strategy yields benefits in terms of high performance, it cannot support decryption and other cryptographic algorithms. There is another paper addressing such limitations. D. Reis et al. proposed IMCRYPTO, a PIM architecture that incorporates a RISC-V controller for orchestration of the AES operations [3]. IMCRYPTO supports AES encryption and decryption and other encryption algorithms like SHA-256 through a RISC-V coprocessor which enables in-memory operations to be executed in arbitrary order. Nevertheless, both of these architectures contend with considerable area overhead due to the extensive utilization of LUTs to exploit parallelism.

Zhang et al. have presented an alternative structure that eliminates the need for LUTs [8]. Their approach involves organizing the S-box, plaintext, keys, and intermediate data required by the AES algorithm within the same memory subarray. Nonetheless, this approach also introduces challenges, including large peripherals needed to exploit parallelism, and concerns related to security and the efficient utilization of data bits due to all the round keys within the memory subarray.

Xie et al. have implemented computation inside memory for the Non-volatile Main Memory (NVMM) [1]. While their work is focused on low energy and the fast process of the AES, they separated each column of the data matrix into a separate memory array. Also, they optimized substeps of `MixColumns` to boost the speed of encryption. Still, there are some margins to improve performance.

IV. Potential Enhancements for AES PIM

Building upon prior research and addressing limitations identified in the prior work, we propose enhancements to the AES PIM architectures, with a specific emphasis on improving efficiency and security, especially based on the Sealer architecture [8].

A. Proposal 1. On-the-fly Round Key Calculation

Instead of the data organization in the Sealer architecture, which stores all the required data in the same memory subarray, we propose to optimize memory utilization and enhance security by calculating the round key on-the-fly using an external 128-bit register and ALUs, and storing only a single round key per round within the subarray. Consequently, the security risks associated with storing all round keys in the memory are mitigated, while preserving operational efficiency and improving utilization of data bits in the subarray.

B. Proposal 2. Enhanced MixColumns

The `MixColumn` process in the AES encryption introduces a processing bottleneck, primarily due to its intensive XOR computations. To accelerate this process, we propose to employ two additional sense amplifiers (SAs) for performing 4-bit XOR calculation, as illustrated in Fig. 3 (a). While the current two SAs are capable of detecting three voltage levels of BL and BLB (activated when the voltage of BL or BLB surpasses V_{ref} and both voltages exceed or fall below V_{ref}), we intend to incorporate two supplementary SAs to identify five bit cell data patterns. Fig. 3 (b) demonstrates how these extra SAs distinguish five patterns. Since the output of the SA is determined by comparing two input voltages, the first two bits of SA output (representing the output of SA_1 and SA_2) indicate whether both voltages are the same, and the last two

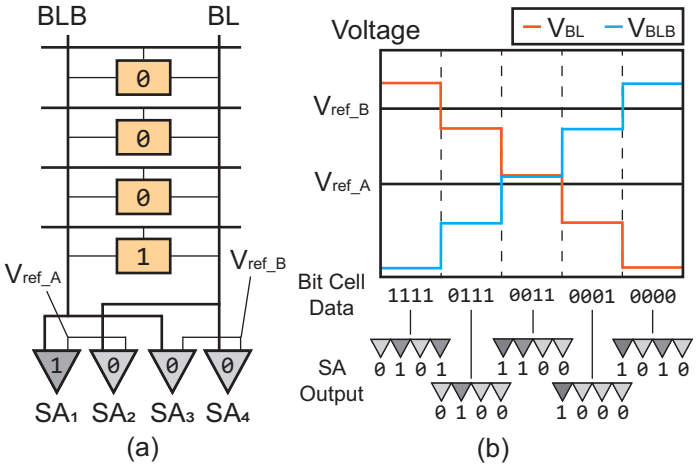

Fig. 3. (a) Enhanced MixColumns using two additional SAs, and (b) voltage levels for 5 data patterns.

bits show whether the voltage of BL or BLB is set to 1 or not.

C. Proposal 3. Flexible Cryptographic Algorithm Support with RISC-V Controller

Inspired by the IMCRYPTO architecture, we propose an extension incorporating a RISC-V coprocessor to orchestrate in-memory operations. Through coprocessor integration and customized RISC-V instructions, the AES PIM architecture gains the ability to support diverse algorithms, including the AES decryption and other cryptographic algorithms like SHA-2 or SHA-3. This enhancement empowers the PIM architecture to execute in-memory operations in any order, facilitating cryptographic algorithms that heavily utilize XOR and bit-shifting operations.

V. Conclusion

In this paper, we reviewed prior work on the PIM architecture for running the AES algorithm and proposed some possible improvements on the AES PIM hardware that we can delve into. Our future work would be implementing the proposed AES PIM architecture to maximize the hardware throughput and support various cryptography algorithms to enhance the security level with minimal silicon overhead.

References

[1] M. Xie, S. Li, A. O. Glova, J. Hu, and Y. Xie, "Securing emerging nonvolatile main memory with fast and energy-efficient aes in-memory implementation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 11, pp. 2443–2455, 2018.

[2] P. R. Sutradhar, K. Basu, S. M. P. Dinakarrao, and A. Ganguly, "An ultra-efficient look-up table based programmable processing in memory architecture for data encryption," in *2021 IEEE 39th International Conference on Computer Design (ICCD)*, 2021, pp. 252–259.

[3] D. Reis, H. Geng, M. Niemier, and X. S. Hu, "Imcrypto: An in-memory computing fabric for aes encryption and decryption," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 5, pp. 553–565, 2022.

[4] P. Chown, "Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)," RFC 3268, Jul. 2002. [Online]. Available: https://www.rfc-editor.org/info/rfc3268

[5] X. Zou, S. Xu, X. Chen, L. Yan, and Y. Han, "Breaking the von neumann bottleneck: architecture-level processing-in-memory technology," *Science China Information Sciences*, vol. 64, no. 6, p. 160404, Apr 2021. [Online]. Available: https://doi.org/10.1007/s11432-020-3227-1

[6] O. Mutlu, S. Ghose, J. Gómez-Luna, and R. Ausavarungnirun, "A modern primer on processing in memory," 2022.

[7] D. Kim, C. Yu, S. Xie, Y. Chen, J.-Y. Kim, B. Kim, J. P. Kulkarni, and T. T.-H. Kim, "An overview of processing-in-memory circuits for artificial intelligence and machine learning," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 12, no. 2, pp. 338–353, 2022.

[8] J. Zhang, H. Naghibijouybari, and E. Sadredini, "Sealer: In-sram aes for high-performance and low-overhead memory encryption," in *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*, 2022, pp. 1–6.