

# RISC-V based Cryptographic ECC-AES Hybrid Core for Secure Communications

S Bharath<sup>1</sup> and Veena M B<sup>2</sup>

<sup>1,2</sup>BMSCE Dept of Electronics and Communication, Bangalore, India  
Email: bharaths.lal23@bmsce.ac.in, veenamb.ece@bmsce.ac.in

**Abstract—** In the era of pervasive connectivity, secure communication has become a fundamental requirement in embedded system design. The growing number of resource-constrained devices demands lightweight yet efficient cryptographic engines that balance performance, security, and scalability. This paper presents the design and simulation of a RISC-V-based processing platform integrated with a hybrid Elliptic Curve Cryptography (ECC) and Advanced Encryption Standard (AES) engine. Implemented in Verilog HDL, the architecture comprises a RISC-V CPU, an ECC module for 256-bit key generation, and an AES encryption/decryption core. The ECC-generated public key (Qx) is directly employed as the AES key, while a memory-mapped interface enables seamless communication between the processor and cryptographic modules. The CPU executes preloaded instructions to emulate arithmetic and memory operations, thereby producing realistic switching activity for performance and power evaluation. Simulation results validate the functional correctness of the platform. Encryption of a single plaintext required 36 cycles in both standalone and integrated configurations. For multiple plaintexts, the standalone system required 144 cycles for five blocks, whereas the integrated platform achieved the same in 124 cycles, yielding a 13.89% reduction in average cycles per block. Power analysis showed 164 mW for the standalone design and 198 mW for the integrated version. The proposed RISC-V-integrated ECC-AES platform thus achieves cycle efficiency with modest power overhead, making it a scalable and modular hardware solution for secure embedded applications.

**Index Terms—** RISC-V, Elliptic Curve Cryptography (ECC), Advanced Encryption Standard (AES), Hardware Acceleration, Cryptographic Processor, Secure Communications.

## I. INTRODUCTION

Cryptographic systems serve as the fundamental backbone of secure communication in the digital era, playing an indispensable role in safeguarding data integrity, confidentiality, and authenticity. These systems are designed to protect sensitive information against a myriad of threats, including eavesdropping, tampering, and unauthorized access, during both storage and transmission. At their core, cryptographic systems employ sophisticated mathematical algorithms to transform data into an unreadable format (encryption) and subsequently revert it to its original state (decryption). Cryptographic techniques are broadly categorized into three main types: symmetric-key algorithms, asymmetric-key algorithms, and hash functions. Symmetric-key algorithms, such as the Advanced Encryption Standard (AES), utilize a single, shared key for both encryption and decryption, offering high performance and simplicity but necessitating secure key distribution mechanisms.

Asymmetric-key algorithms, including RSA and Elliptic Curve Cryptography (ECC), employ a pair of keys—a public key and a private key—to facilitate secure communication without requiring prior key exchange, making them essential for public-key infrastructure (PKI) systems. The advent of hybrid encryption schemes, which strategically combine multiple cryptographic techniques, represents a significant advancement. These schemes effectively address challenges such as computational overhead and key management inefficiencies inherent in single-algorithm approaches, leading to an enhancement in overall system performance and security.

The pervasive expansion of digital communication and the proliferation of connected devices, particularly within the Internet of Things (IoT) and embedded systems, have led to an escalating demand for high-speed processing and low-power consumption in modern cryptographic systems. Traditional software-based cryptographic implementations often fall short in meeting these stringent requirements, leading to performance bottlenecks and increased power consumption. In modern secure systems, hardware-based cryptographic accelerators have become essential as they offload complex computations from the main processor, achieving higher throughput and lower latency. Traditional algorithms like RSA increasingly struggle with large key sizes and high computational demands, resulting in performance, energy, and storage inefficiencies—making them unsuitable for resource-constrained platforms such as IoT and mobile devices. Hence, there is a growing need for a lightweight and power-efficient cryptographic approach like Elliptic Curve Cryptography (ECC) to ensure secure communication under strict performance and energy limits. Studies by Mojtaba Bisheh et al. [1] and Bao et al. [2] highlight ECC's efficiency in encryption, key exchange, and secure communications, noting its faster key generation and lower memory usage compared to traditional schemes. Similarly, [3] emphasize ECC's effectiveness in applications like cloud computing, e-health, and electronic voting, showcasing its superior security and scalability over RSA and Diffie-Hellman. Together, these works establish ECC as a robust and efficient solution for modern cryptographic needs.

Yang et al. [4] and Zheng et al. [5] presented RISC-V-based designs emphasizing hardware–software co-design for cryptographic applications. Yang et al. proposed a low-power IoT processor with a hybrid encryption accelerator, achieving significant power savings and efficient cryptographic operations. Zheng et al. developed a cryptographic SoC using RISC-V virtual prototyping, which accelerated design iterations and optimized hardware–software integration. Both works demonstrated the flexibility of RISC-V and the importance of co-design in achieving efficient and secure implementations. Azad and Yang [6] designed a RISC-V-based SoC aimed at accelerating encryption and decryption on edge devices, particularly emphasizing homomorphic encryption. Their work addressed the challenges of implementing computationally intensive algorithms in resource-limited environments by introducing performance and energy optimizations, demonstrating the practicality of integrating homomorphic encryption accelerators into IoT and edge systems. Ignatius et al. [7] investigated the susceptibility of RISC-V-based cryptographic cores to power side-channel attacks, a major security concern in modern cryptographic hardware. They identified potential vulnerabilities in RISC-V implementations and proposed advanced countermeasures, such as randomization techniques and secure key management, to mitigate these threats, ensuring the reliability of cryptographic operations in high-security applications like defence and financial systems.

Zhao et al. [10] developed energy-efficient AES hardware architectures implemented in 65nm CMOS technology, achieving the lowest reported energy per encryption operation using voltage scaling and clock gating. Their study provided valuable insights into the trade-offs between power, area, and performance, offering design strategies for energy-constrained applications. Similarly, Dao et al. [11] introduced an optimized AES encryption core for IoT systems, employing low-power design techniques and efficient dataflow architectures to achieve a balance between energy efficiency and computational performance, making it suitable for battery-powered devices. Jain et al. [12] presented an inner-round pipelined AES hardware architecture designed to achieve high throughput through parallel processing. This pipelining approach enables multiple AES rounds to be executed concurrently, reducing latency and enhancing speed, making the architecture well-suited for real-time applications such as secure communications, video streaming, and financial transactions.

### III. METHODOLOGY

The proposed architecture is designed to bring together the strengths of two well-established cryptographic techniques: Elliptic Curve Cryptography (ECC) and the Advanced Encryption Standard (AES). ECC is used for generating secure 256-bit keys with minimal memory usage and reduced computational effort, which makes it highly suitable for lightweight embedded platforms. AES, on the other hand, is adopted for its ability to perform high-speed encryption and decryption of large data blocks. By combining these two methods, the design ensures that the system benefits from ECC's secure and efficient key generation along with AES's proven capability to

handle fast and reliable data protection. This hybrid integration not only improves security but also optimizes performance, offering a practical and scalable solution for modern embedded applications that require confidentiality and integrity without compromising efficiency.

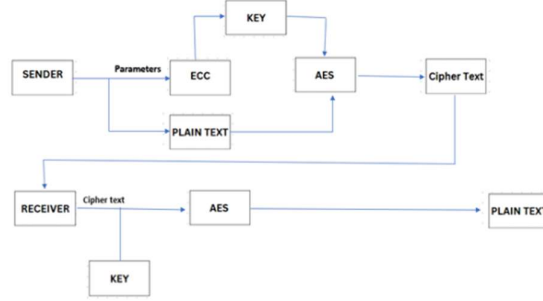


Fig 1: Flow chart of ECC-AES Operation

A RISC- V-based cryptographic processor with elliptic curve acceleration provides a scalable and secure solution by leveraging hardware accelerators for ECC operations. This methodology outlines the design and implementation of such a processor to achieve high performance.

#### A. Elliptical Curve Cryptography

ECC is a public-key encryption method introduced by Koblitz and Miller and its popularity stems from the strong level of security it offers while requiring significantly smaller key sizes compared to other public-key cryptosystems such as RSA. This efficiency makes ECC highly suitable for modern cryptographic applications where performance and security are critical.

The mathematical operations in ECC are carried out over finite fields. The use of finite fields ensures accurate and efficient results, which are fundamental for cryptographic computations.

The elliptic curve  $E(a, b)$  defined over a finite field  $F_p$  is represented mathematically as:

$$y^2 = x^3 + ax + b \pmod{p} \quad (1)$$

Here, 'a' and 'b' are integers less than a sufficiently large prime number 'p'. The chosen parameters must satisfy the following conditions to ensure the validity of the curve:

$$4a^3 + 27b^2 \neq 0 \pmod{p} \quad (2)$$

The x and y coordinates of the elliptic curve  $E(a, b)$  are determined using Equation (1). The group order 'n' of  $E(a, b)$  over the finite field  $F_p$  is equal to the total number of points on the curve, including the point at infinite 'O'. The base point G on the elliptic curve is chosen based on the group order, and it plays a central role in encryption and decryption operations. The operations performed with the elliptic curve points on the finite field are summarized

- **Point Addition:** When adding two distinct points  $P(x_1, y_1)$  and  $Q(x_2, y_2)$  on the elliptic curve to obtain a third point  $Q_3(x_3, y_3)$ , specific formulas are applied. Geometrically, this operation can be visualized by drawing a straight line through points P and Q. This line will intersect the elliptic curve at a third point, which is then reflected across the x-axis to yield the result R. The formulas for the slope  $\lambda$  and the coordinates  $(x_3, y_3)$  are:

$$Q_3(x_3, y_3) = Q_3(x_1, y_1) + Q_3(x_2, y_2) \quad (3)$$

If  $x_1 \neq x_2$ , the calculations of  $x_3$  and  $y_3$  are performed below,

$$x_3 = \lambda^2 - x_1 - x_2 \quad (4)$$

$$y_3 = \lambda(x_1 - x_2) - y_1 \quad (5)$$

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} \quad (6)$$

If  $x_1 = x_2$ , then the infinity point  $O$  is formed.

- Point Subtracting: Consider two points  $Q_1(x_1, y_1)$  and  $Q_2(x_2, y_2)$  on the elliptic curve  $E(a,b)$  over the finite field. To subtract these points in ECC, the first step is to determine the negative of  $Q_2(x_2, y_2)$ . Once this is obtained, the subtraction operation is carried out by performing point addition as expressed in Eq. (8).

$$Q_2(x_2, y_2) = Q_2(x_2, -y_2) \quad (7)$$

$$Q_3(x_3, y_3) = Q_1(x_1, y_1) + Q_2(x_2, -y_2) \quad (8)$$

- Point Doubling: Consider two points  $Q_1(x_1, y_1)$  and  $Q_2(x_2, y_2)$  on the elliptic curve  $E(a,b)$  defined over the finite field  $F_p$ . When both points are the same, i.e.,  $Q_1(x_1, y_1) = Q_2(x_2, y_2)$ , the operation performed is called point doubling. This process is a fundamental step in ECC and is mathematically represented using Eq. (9). The process involves finding the tangent line to the elliptic curve at point  $Q$ .

$$Q_3(x_3, y_3) = Q_1(x_1, y_1) + Q_1(x_1, y_1) = 2 Q_1(x_1, y_1) \quad (9)$$

The calculations of  $x_3$  and  $y_3$  are performed as follows:

$$x_3 = \lambda^2 - 2x_1 \quad (10)$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \quad (11)$$

$$\lambda = \frac{3x_1^2 + a}{2y_1} \quad (12)$$

- Scalar Multiplication: Let  $Q_1(x_1, y_1)$  be a point on the elliptic curve  $E(a,b)$  over the finite field  $F_p$ . Scalar multiplication in ECC is carried out by repeatedly applying the point addition and point doubling operations. For example, multiplying the point by 10, i.e.,  $10 Q_1$ , is computed step by step using these operations, as shown in Eq. (13).

$$10 Q_1 = 2(2(Q_1 + Q_1) + Q_1) \quad (13)$$

- Key Generation (Private Key): In ECC, the private key ( $d$ ) is a randomly chosen integer. This integer must fall within the range  $[1, n-1]$ , where  $n$  represents the order of the base point  $G$ . The order  $n$  signifies the number of times the base point  $G$  can be added to itself before the operation cycles back to the point at infinity, which functions as the zero element in elliptic curve arithmetic.
- Key Computation (Public Key): The public key ( $Q$ ) is derived from the private key ( $d$ ) through a process called scalar multiplication. This involves multiplying the private key  $d$  by the base point  $G$  on the elliptic curve, resulting in  $Q = dG$ . This scalar multiplication operation is computationally intensive and relies on repeated applications of point addition and point doubling operations on the elliptic curve.
- Shared Secret ( $S$ ): The shared secret in ECC is a common secret value computed independently by two parties during key exchange (for example, in the Elliptic Curve Diffie–Hellman (ECDH) protocol). It ensures that both parties can derive the same secret key without ever transmitting it over the communication channel.

$$S = d_A Q_B = d_A (d_B G) = d_B (d_A G) = d_B Q_A \quad (14)$$

### B. AES Algorithm

The AES is a symmetric block cipher that has been widely adopted globally since its development by the National Institute of Standards and Technology (NIST) in 2001. It is a cornerstone of modern cryptography, designed to secure data by transforming it into an unreadable ciphertext without the correct cryptographic key. AES offers robust protection against unauthorized access and supports various key lengths: 128, 192, or 256 bits, providing flexibility in security strength.

The AES encryption process consists of a structured sequence of operations applied iteratively over multiple rounds:

- Add Round Key: The initial round key is XORed with the plaintext to establish the first level of key dependence. This step guarantees that the encryption process cannot proceed independently of the secret key, introducing security from the very beginning.

- **SubBytes:** Each byte of the state undergoes a non-linear transformation through an S-box. This introduces confusion in the cipher by ensuring that the mapping between input and output bytes is highly non-linear, thereby protecting against linear and differential cryptanalysis.
- **ShiftRows:** The state matrix rows are shifted cyclically by fixed offsets. Although simple in operation, this stage enhances diffusion by dispersing byte dependencies across multiple columns, preventing localized influence of individual bytes.
- **MixColumns:** Each column of the state is multiplied by a fixed polynomial in GF(28). This operation mixes the four bytes of every column, ensuring that changes in one byte propagate across the entire column, significantly improving diffusion.
- **Add Round Key:** The final step of a round applies a unique round key, XORed with the state. Since each round key is derived from the main key through the key expansion algorithm, this stage reinforces key dependence and ensures round-to-round variability

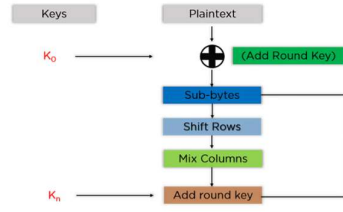


Fig 2: AES Encryption/Decryption Flow

### C. RISC V ECC-AES Integration

The ECC (Elliptic Curve Cryptography) module is integrated into the RISC-V processor by modifying the datapath and control logic to support secure cryptographic operations. The ECC is implemented as an external hardware IP block, interfaced through multiplexers and control signals.

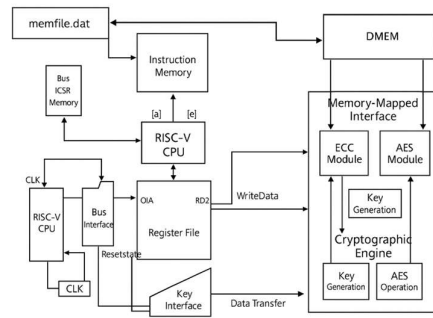


Fig 3: RISC V ECC-AES System Architecture

|    |          |
|----|----------|
| 1  | E59F0000 |
| 2  | E3A010A1 |
| 3  | E5801004 |
| 4  | E3A010B2 |
| 5  | E5801004 |
| 6  | E3A010C3 |
| 7  | E5801004 |
| 8  | E3A010D4 |
| 9  | E5801000 |
| 10 | E3A010E5 |
| 11 | E5801004 |
| 12 | E3A010F6 |
| 13 | E5801004 |
| 14 | E5801004 |
| 15 | E3A01118 |
| 16 | E5801000 |
| 17 | E59F0000 |
| 18 | E3A01001 |
| 19 | E5801000 |
| 20 | EAF00000 |
| 21 | 10000000 |
| 22 | 10000010 |
| 23 | 10000020 |

Fig 4: Instructions in memfile.dat

- **Datapath Modification:** Multiplexers are added to route data between the registers, memory, and the ECC module. The module performs modular arithmetic operations (e.g., point addition, scalar multiplication) using internal ALUs, activated only when ECC-specific instructions are decoded.
- **Instruction Decoding & Control:** The control unit is extended with custom opcodes or functional fields for ECC. When detected, the decoder enables the ECC IP, configures multiplexers, and ensures proper write-back of results, while standard instructions follow the normal RISC-V datapath.
- **Memory-Mapped Interface:** The cryptographic modules are accessed through memory-mapped registers, providing a standardized interface for software interaction. The AES module accepts 128-bit plaintext input through four 32-bit registers at specific memory addresses, while control signals for triggering operations are written to dedicated control registers.
- **dat File for ECC Triggering:** Machine instructions, stored in a .dat file, include both standard RISC-V and ECC opcodes. When an ECC instruction is fetched, the decoder asserts ECC control signals, routes operands, executes the cryptographic operation, and writes back the result. This provides a programmable interface to test ECC without altering hardware.

➤ Flow of Operation:

- At system startup, a clock and reset initialize the CPU, instruction memory, and cryptographic modules. The instruction memory is preloaded with a program designed to handle multiple plaintexts sequentially, while data memory is cleared to buffer intermediate results.
- Once reset is deasserted, the CPU begins instruction fetch from memory. Each instruction (LW, SW, arithmetic, control) experiences fetch and decode delays before its effect appears in the pipeline. This staged execution becomes more visible during multi-plaintext processing, as the CPU alternates between fetching, decoding, operand preparation, and memory access.
- The ECC module operates in parallel, performing scalar multiplication on elliptic curve points to generate a 256-bit key. Importantly, the CPU does not stall during ECC execution—it continues fetching and issuing instructions, preparing plaintext blocks for AES. This overlap demonstrates efficient parallelism between computation and instruction flow.
- AES input is handled through memory-mapped writes: the CPU issues successive SW instructions to load 32-bit chunks of each plaintext into AES registers, followed by a control flag write to trigger encryption. For multiple plaintexts, this sequence repeats, with observable fetch delays separating successive store operations. (~10 cycles),

#### IV. RESULTS AND DISCUSSIONS

When evaluating the standalone ECC–AES block against the RISC-V integrated design, it is important to account for all stages in the processing flow, namely plaintext fetch, ECC key generation, AES encryption, and AES decryption.

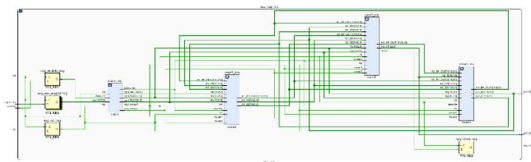


Fig 5. Schematic of ECC Core

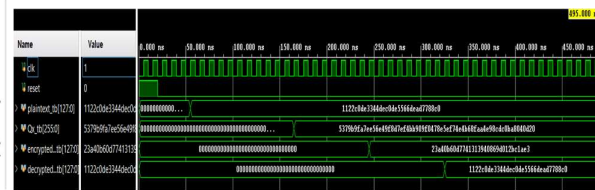


Fig 6. Simulation of Standalone ECC-AES block

From the waveform analysis of the standalone ECC-AES cryptographic block, the cycle requirements for different operations were observed and are discussed below:

- Fetch Operation – 4 Cycles The fetch stage, which involves retrieving the plaintext or instruction from memory, consistently required 4 clock cycles. This is expected due to the pipeline depth of the RISC-V core and memory access latency. Since the fetch process overlaps with the ongoing cryptographic computation in later cycles, it contributes minimally to overall throughput degradation.
- ECC Key Generation (Qx Computation) – 9 Cycles The generation of the ECC public key component Qx required 9 cycles. This stage primarily involves modular arithmetic (point multiplication on the elliptic curve), which is more computationally intensive compared to linear memory fetch
- AES Encryption – 11 Cycles the AES encryption of a 128-bit plaintext block took 11 clock cycles. This aligns well with the round-based structure of AES-128 (10 rounds + initial key addition).
- AES Decryption – 12 Cycles the AES decryption stage required 12 cycles, which is slightly higher than encryption. This is consistent with theoretical expectations, as AES decryption involves additional operations such as inverse mix columns, which are more complex than their encryption counterparts

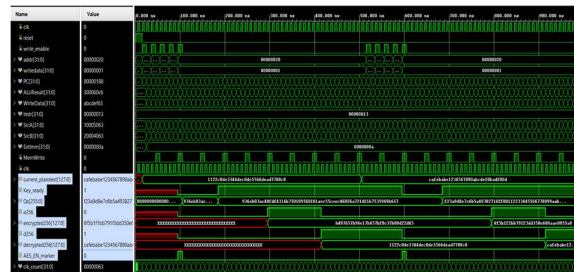


Fig 7. Simulation RISC V integrated ECC-AES Block

In the standalone implementation, these stages occur sequentially, meaning every fetch (4 cycles each) and the one-time key generation cost (9 cycles) are visible along with the encryption (11 cycles) and decryption (12 cycles) for each plaintext. For a set of five plaintexts, this results in a total of 144 cycles. In contrast, the RISC-V integrated design hides subsequent fetch operations (after the first) by overlapping them with AES processing, thereby reducing the visible fetch cost while keeping the key generation, encryption, and decryption cycles unchanged.

TABLE I. COMPARISON TABLE FOR PERFORMANCE AND POWER METRICS

| Workload             | Standalone ECC-AES | RISC V Integrated ECC-AES | % Reduction |
|----------------------|--------------------|---------------------------|-------------|
| Single Plaintext     | 36 Cycles          | 36 Cycles                 | 0%          |
| 5 Plaintexts         | 144 Cycles         | 124 Cycles                | 13.89%      |
| Avg Cycles per block | 28 cycles          | 24 Cycles                 | 13.89%      |
| Power Consumption    | 168mV              | 198mV                     | -           |

This reduces the total execution time to 124 cycles for five plaintexts. This integration thus provides a reduction of 20 cycles, corresponding to an overall efficiency gain of about 13.89%. This demonstrates how pipeline-style parallelism in RISC-V improves throughput, especially when processing multiple plaintexts, while the one-time ECC key generation remains constant in both approaches.

## V. CONCLUSION AND FUTURE SCOPE

The integration of the ECC–AES cryptographic module with a RISC-V processor demonstrates a significant improvement in execution efficiency compared to the standalone implementation. While the standalone design requires additional cycles for repeated key generation and plaintext fetch operations, the RISC-V integrated approach reduces this overhead, resulting in a 13.89% cycle reduction when processing multiple plaintexts. This highlights the advantage of tightly coupling cryptographic modules with a processor core, where data and keys can be efficiently managed within the pipeline. The implementation has been verified through simulation in ModelSim and Vivado, where the encryption and decryption processes were validated against expected outputs. In the future, this platform can be expanded to support post-quantum cryptography and include hardware acceleration for SHA-3 and other algorithms for stronger security. By adding side-channel attack protections such as masking and power balancing, the design can become more robust.

## REFERENCES

- [1] Mojtaba Bisheh Niasar, Reza Azarderakhsh, and Mehran Mozaffari-Kermani. 2021. Cryptographic accelerators for digital signature based on Ed25519. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 29, 7 (2021), 1297–1305.
- [2] Bao, Jiaxu. “Research on the security of elliptic curve cryptography.” 2022 7th International Conference on Social Sciences and Economic Development (ICSED 2022). Atlantis Press, 2022.
- [3] Yang, S.; Shao, L.; Huang, J.; Zou, W. Design and Implementation of Low-Power IoT RISC-V Processor with Hybrid Encryption Accelerator. *Electronics* 2023, 12, 4222. <https://doi.org/10.3390/electronics12204222>.
- [4] Xin Zheng "Hardware/Software Co-Design of Cryptographic SoC Based on RISC-V Virtual Prototype," in *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS*, VOL. 70, NO. 9, SEPTEMBER 2023.
- [5] Zahra Azad, Guowei Yang "RISE: RISC-V SoC for En/Decryption Acceleration on the Edge for Homomorphic Encryption," *IEEE TRANSACTIONS ON VERY LARGE-SCALE INTEGRATION (VLSI) SYSTEMS*, VOL. 31, NO. 10, OCTOBER 2023.
- [6] Titu Mary Ignatius, "Power Side-Channel Attacks on Crypto-Core Based on RISC-V ISA for High-Security Applications," *Digital Object Identifier 10.1109/ACCESS.2024.3477961* in open access journal on 22 October 2024.
- [7] F. Rahman, M. Farmani, M. Tehranipoor, and Y. Jin, “Hardware assisted cybersecurity for IoT devices,” in *Proc. 18th Int. Workshop Microprocessor SOC Test Verification (MTV)*, Dec. 2017, pp. 51–56.
- [8] D. A. Osvik, J. W. Bos, D. Stefan, and D. Canright, *Fast Software AES Encryption*. Berlin, Germany: Springer, 2010, pp. 75–93.
- [9] M. Kashif, S. Mehruz, I. Shakeel and S. Ahmad, "Employing an ECC-Based Hybrid Data Encryption Method to Improve Multitenancy Security in Cloud Computing," 2023 REEDCON, New Delhi, India, 2023, pp. 79-83, doi: 10.1109/REEDCON57544.2023.10150684.
- [10] D. S. Dayana, R. Pandian, A. R. Babu, S. Nirmalraj, S. D. S. Jebaseelan and M. V, "Elevating Security in Wireless Sensor Networks using ECC and AES Cryptographic Techniques," 2023 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICES), Chennai, India, 2023, pp. 1-6, doi: 10.1109/ICES60034.2023.10465296.

- [11] B. Sarala, H. Shine, S. D. Devi, M. Shanmugapriya and S. Ravi, "Effective data security for Vehicular Adhoc Network using a Novel ECC-based Authentication and Key Exchange," 2023 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems(ICSES), 10.1109/ICSES60034.2023.10465460.
- [12] A. Jain, D. Jain, A. Katiyar, and G. Kaur, "An inner round pipeline architecture hardware core for AES," in Proc. 2nd Asian Conf. Innov. Technol. (ASIANCON), Aug. 2022, pp. 1–5.
- [13] Hamad Alrimeih and Daler Rakhmatov. 2014. Fast and flexible hardware support for ECC over multiple standard prime fields. IEEE Transactions on Very Large-Scale Integration (VLSI) Systems 22, 12 (2014), 2661–2674.
- [14] "ARMV8. architecture reference manual, for Armv8-A, 2020," Arm Limited, Tech. Rep. ARM DDI 0487F.c (ID072120), <https://developer.arm.com/documentation/ddi0487/fc/> 2020. [Online]. Available:
- [15] Joppe W. Bos, Craig Costello, Patrick Longa, and Michael Naehrig. 2016. Selecting elliptic curves for cryptography: An efficiency and security analysis. Journal of Cryptographic Engineering 6 (2016), 259–286.