Dynamic XTEA Optimization and Secure Key Management for Embedded Microcontroller-Based SDN Systems in Smart Cities

Sunil Kumar Shah[1]*, Raghvendra Sharma[2], and Neeraj Shukla[3]

Abstract—**The rapid expansion of smart city infrastructures necessitates robust and efficient security mechanisms for embedded processor-based Software-Defined Networking (SDN) nodes. Hence, this research introduces the Adaptive Secure XTEA for Embedded Microcontrollers (ASX-EM), a novel encryption method designed to address these environments' unique security and performance needs. Existing encryption implementations often neglect proper padding validation, leading to vulnerabilities such as the Padding Oracle Attack (POA). The proposed Context-Aware Key Expansion and Secure Padding Validation (CAKE-SPV) technique customizes key scheduling based on node-specific parameters and employs a robust padding verification mechanism, significantly enhancing encryption security. Moreover, the computational demands of XTEA, with its multiple rounds of operations, are inefficient on resource-constrained 8-bit microcontrollers, leading to increased latency and reduced system responsiveness. To optimize performance, the Adaptive Round and Parallel Processing (ARPP) method is developed that dynamically adjusts encryption rounds based on system metrics and employs bit-slice processing with precomputed lookup tables for efficient arithmetic operations. The results show that the proposed model has a low encryption time of 2.7s a decryption time of 2.6s, and a high encryption throughput of 22,000 KB/sec and, a decryption throughput of 17600 KB/sec, compared to other existing models.**

Index Terms—**Smart cities, Padding Oracle Attack, Software-Defined Networking, Extended Tiny Encryption Algorithm, precomputed lookup tables.**

I. INTRODUCTION

The infrastructure of smart cities is being revolutionized by Software-Defined Networking (SDN), which provides a programmable and centralized method of controlling intricate and diverse metropolitan networks. SDN offers the required agility and control in the context of a smart city, where the integration of several IoT devices, traffic management systems, energy grids, and public safety networks produces a highly dynamic environment. SDN design allows for centralized control via a software-based controller by severing the network control plane from the data plane. Because it is in charge of the whole network, this controller enables dynamic resource allocation, automatic traffic management, and real-time monitoring. Because SDN is programmable, network policies may be quickly altered in response to shifting circumstances or new threats, guaranteeing ongoing security and optimization. To improve network flow and lower latency for vital applications like emergency response systems, SDN, for example, dynamically redirects data from bottleneck locations in traffic management. By modifying the distribution in response to real-time supply and demand data, SDN in energy management can help make it easier to integrate renewable energy sources. Furthermore, by offering a centralized platform for executing security procedures and identifying irregularities throughout the network, SDN's centralized architecture improves cybersecurity [1-4].

Enormous networks of interlinked sensors, devices, and communication systems underpin smart cities, and secure data transfer is necessary to preserve confidentiality, integrity, and trust. The XTEA (Extended Tiny Encryption Algorithm) is a durable and lightweight symmetric key block cipher that is well-suited for use in smart city applications. Because of its small size and ability to function on 64-bit blocks with a 128-bit key, XTEA is perfect for resource-constrained contexts, such as embedded systems and Internet of Things devices, which are commonly found in smart city infrastructure. Strong cryptographic security is provided by its straightforward structure, which consists of a sequence of bitwise shifts, XOR operations, and modular additions, this offers little computing cost. Because of its iterative method, which typically consists of 64 rounds, XTEA is more resistant to cryptanalysis attempts, which makes it a dependable option for protecting sensitive data such as traffic data, utility use statistics, and personal information. Although XTEA has many benefits, there are security risks and vulnerabilities associated with its usage in smart city infrastructure that need to be properly addressed [5-8].

One major issue is that XTEA is subject to differential cryptanalysis, especially if it is not implemented correctly and with enough rounds (64 rounds is the ideal amount, but implementations with fewer rounds are still vulnerable). Additionally, if weak keys are utilized, XTEA's key scheduling technique is extremely basic and vulnerable to cryptanalytic assaults. Physical assaults, like side-channel attacks, present a significant concern in the context of smart cities since gadgets frequently function in unsupervised and sometimes unsafe settings. To determine the encryption key, these attacks make use of data that is disclosed during the encryption process, such as power usage or electromagnetic emissions. Another risk

[1]* Amity University Madhya Pradesh, Maharajpura (Opposite Airport) Gwalior, Madhya Pradesh, India (e-mail: sunil.ggits@gmail.com)

[2] Amity University Madhya Pradesh, Maharajpura (Opposite Airport) Gwalior, Madhya Pradesh, India (e-mail: rsharma3@gwa.amity.edu)

[3] Gyan Ganga College of Technology, Jabalpur, Madhya Pradesh, India (e-mail: neerajshukla28@gmail.com)

DOI: 10.36244/ICJ.2025.4.8

is replay attacks, in which a hacker intercepts encrypted data and sends it again to trick the target system into carrying out commands or actions that are not authorized. Robust key management procedures, such as regular key rotation and the use of powerful, randomly generated keys, must be put in place to reduce these risks [9–12]. Smart cities may use XTEA's advantages while reducing possible weaknesses by tackling these risks with all-encompassing security solutions.

The XTEA implementation in embedded microcontrollers inside an SDN framework for smart city infrastructure has several technical difficulties that need to be carefully considered. One major problem is that embedded microcontrollers have restricted resources by nature; they include memory, computational power, and energy availability. Despite the lightweight nature of XTEA, cryptographic operations can still put a load on these few resources, which might affect the responsiveness and performance of vital smart city applications. In an SDN environment, where real-time data flow management and fast reconfiguration are critical, this is especially pertinent. The computational expense of XTEA causes latency problems, impacting the speed at which data is sent and decisions are made in the SDN, particularly when the necessary 64 rounds for security are implemented. The integration of XTEA inside the SDN control plane and data plane separation paradigm is another major hurdle. Embedding XTEA encryption techniques that dynamically adapt to the network's changing topology and traffic patterns is necessary to provide smooth and secure communication across various planes [13–15]. It is imperative to optimize the implementation of XTEA for low-power operations to overcome these technological obstacles. This can be achieved, for example, by using software libraries that are specifically designed for microcontroller architectures or by using hardware acceleration. Ultimately, an integrated approach that achieves a balance between security, performance, and resource limitations while guaranteeing the flexibility and scalability of the whole infrastructure is needed for the effective implementation of XTEA in embedded microcontrollers inside an SDN framework for smart cities.

A. Main objective of this paper

The following methodological and experimental contributions have been achieved by this paper:

- To mitigate POA vulnerabilities, CAKE-SPV is implemented within the XTEA encryption, in which Context-Aware Key Expansion Scheduling algorithm is utilized for customizing the key scheduling process, and OAEP with HIPV mechanism is utilized for secure padding with integrated validation, thereby ensuring secure communication and preventing information leakage through POA vulnerabilities.
- To optimize the XTEA for 8-bit Microcontrollers-based smart city infrastructure, an ARPP is presented, which utilizes adaptive round adjustment for an optimal number of encryption rounds and parallel processing for dividing the data into smaller bit-level blocks thereby enhancing the efficiency of XTEA encryption methods to operate effectively on 8-bit microcontrollers, reducing chip area, power consumption, and processing time.

B. Organization of study

The arrangement of the paper is as follows. In Section 2, relevant literature is reviewed; in Section 3, the methodology of the proposed system is explained; in Section 4, experiments, datasets, comparison, and evaluation methodologies are covered; in Section 5, suggestions for future developments and limits of the approach are made.

II. LITERATURE SURVEY

For the protection of sensitive data in a variety of applications, including RFID systems and smart cities, the security and effectiveness of encryption methods are essential. Enhancing the XTEA algorithm and its variations to handle certain security flaws and performance limitations in various scenarios has been the subject of recent research.

Ahmed et al [16] employed an enhanced S-box to boost security and thwart a variety of assaults, resulting in a new and reliable version of the original XXTEA. To achieve the one-time pad idea and provide an extra degree of protection, the M-XXTEA was also combined with a chaotic key-generating system. In contrast to the original XXTEA and AES, the cipher keys were dynamically updated for every block of plaintext throughout the encryption process, offering a more reliable security method. The M-XXTEA works with multiple text block sizes and key sizes in addition to improving data security. To compare the M-XXTEA's performance with that of the original XXTEA and AES, many experiments were carried out. The results showed that M-XXTEA surpassed AES by 60% in terms of encryption and decryption time efficiencies. The addition of new elements, including the chaotic key generation, results in unanticipated weaknesses, even if the M-XXTEA already counters several assaults.

Manikandan et al [17] addressed the XTEA's security issues by using domain-specific customization, random number generation, and hidden key renewal processes. RXMAP-1 and RXMAP-2, two different encoder architectures for the Renovated XTEA Mutual Authentication Protocol (RXMAP), were proposed. Their foundation was the replacement of accurate computational blocks with approximations. The proposed RXMAP protocol's computational and storage overhead was evaluated, and it was tested against a variety of security threats using BAN logic in both formal and informal verification. The proposed encoder designs are simulated for functional verification, and ASIC implementation is carried out on a 132 nm manufacturing node. However, because of the customization, use of random numbers, and key renewal procedures, the suggested protocol resulted in computational and storage overhead.

Zeesha Mishra and Bibhudendra Acharya [18] constructed optimal lightweight ciphers to implement the cipher in hardware by modeling the design characteristics. To accomplish the intended result, the TEA, XTEA, and XXTEA ciphers were developed, put into practice, and optimized utilizing specialist hardware platforms including Application

Specific Integrated Circuit (ASIC) and Field Programmable Gate Array (FPGA). Through the execution of designs for four hardware architectures TEA (T1), XTEA (T2), XXTEA (T3), and hybrid model (T4) many elements, including block sizes, implementation rounds, and crucial scheduling components, have been explored. The percentage gains in frequency for T1, T2, and T3 using a pipelined method are 75.9%, 162%, and 89%, respectively. Nevertheless, when optimizations are carried out, their scope and effects on other aspects like as security or resource use are not thoroughly investigated.

Neha Khute et al [19] proposed a round-based XXTEA-192-bit architecture to reduce the implemented hardware's space. This design had cheap cost and small space required, and it was meant for RFID applications. Simple shifting, addition, and XOR operations are among the fundamental and logical operations used by XXTEA. These simple activities allowed the architecture to be low-area and extremely efficient by design. Performance analysis was carried out on several FPGA device families, including Spartan-3, Virtex-7, Virtex-5, and Virtex-4, assessing variables including throughput and efficacy. Nevertheless, further optimization is needed in terms of speed, power usage, or reduced area.

Dzaky Zakiyal et al [20] developed a distributed MQTT (message queuing telemetry transmission) brokers-optimized architecture. For edge resources, a distributed MQTT broker might reduce latency and network traffic by managing only topics that were consumed on the network. An integer non-linear code was created to optimize container placement and minimize the wastage of edge computing resources. This architecture with the existing distributed MQTT middleware design with random and greedy container placement was simulated through rigorous modeling. When it came to lowering synchronization overhead, power use, network utilization, and deployment failure rates, this design fared better than the others. Nevertheless, the limited memory, processing power, and storage of edge devices affect the solution's viability and efficiency.

Keshari et al [21] suggested using the Grey Wolf Optimization Affinity Propagation (GWOAP) algorithm to arrange many controllers in smart city networks. The network's linked smart devices' traffic was controlled by the controllers. The OS3E network architecture is used to mimic the suggested method. To minimize processing delays and regulate the controller's traffic load, the controller deploys in the OS3E network topology by executing AP and GWO optimization algorithms that split the network into subdomains. IoT-enabled smart switches are better distributed throughout clusters using GWOAP, and node equalization was distributed evenly among all controllers in the deployed architecture. The traffic load of IoT-enabled devices in smart city networks is intelligently balanced across controllers by employing the suggested technique.

Anusha, and Shastrimath [22] developed and put into use a low-cost FPGA RFID-Mutual Authentication (MA) system with XTEA security. By offering Reader's and Tag's challenge and Response utilizing XTEA security, the RFID-MA incorporated Reader and Tag authentication. The RFID-MA procedure was completed faster overall because of XTEA's pipelined design, which combined parallel execution of key scheduling with encryption and decryption processes. RFID incorporated the XTEA with Cypher block chaining (CBC) for protected MA applications. Based on the challenge and response between the Reader and Tag utilizing XTEA-CBC, the authentication procedure was successful. The security of XTEA is constrained by its vulnerability to complex cryptographic techniques such as differential cryptanalysis. For long-term security, more powerful encryption algorithms and frequent upgrades are required.

Chen et al [23] focused on DDoS attack traceback techniques in SDN-based SC. Relevant reports from the past few years were analysed, and it was discovered that the current approaches were less adaptable overall and require more time and resources. As a result, this research provided a simple traceback system based on anomaly trees. By examining network traffic fluctuations, this approach created an anomaly tree. It then calls on several detection algorithms that satisfy the necessary conditions to reduce the tree and ultimately identify the attack path. The main weakness in the method is that it is vulnerable to erroneous data from hacked base station nodes, which might result in imprecise anomaly identification and traceability of attack paths. It is additionally susceptible to noise and inconsistencies since it depends on consistent network traffic patterns.

Abdulkadhim et al [24] presented a more advanced, lightweight Modified XTEA Algorithm that protected against node abuse attacks and side-channel vulnerabilities. Provide a design in this work that used chaotic systems to create encryption keys, making them more unpredictable and random. This research's main goal is to strengthen security protocols against a variety of modern attack methods, ensuring complete defense, unpredictable behavior, and resilience. The purpose of implementing strategic defenses and strategies is to protect important resources from potential harm. Even with these improved security measures, the complexity of chaotic key generation still causes the updated XTEA method to operate poorly on very limited hardware.

Ragab et al [25] demonstrated that the XXTEA lightweight block cypher used fewer memory and computing cycles, so it is a better fit for usage in IoT smart devices for message encryption. Additionally, the elliptic curve cryptography (ECC) asymmetric cipher was chosen over RSA because it provides a higher level of bit security at smaller key sizes. To ensure authenticity, integrity, and non-repudiation, the ECC cipher was employed. For secrecy, the XXTEA block cipher was employed. Additionally, each time data is encrypted, the script hashing algorithm is utilized to confirm data integrity and produce numerous keys. By combining ECC, XXTEA, and script, the suggested hybrid cryptosystem satisfies the four primary requirements of cryptography: secrecy, authenticity, integrity, and non-repudiation. However, the physical setup of the suggested hybrid cryptosystem needs to be addressed.

From this review, it is noted that [16] introduces unexpected weaknesses that can impact reliability, in [17] results in increased computational and storage overhead, and in [18]

observed that their optimized hardware implementations while improving performance, did not thoroughly address the trade-offs between security and resource usage. [19] it still requires further optimization for speed, power consumption, and area reduction, in [20] faces challenges due to the limited memory and processing power of edge devices, in [21] highlighted that their GWOAP-based controller arrangement, while balancing IoT traffic, could be limited by the complexity of the optimization process, in [22] vulnerable to advanced cryptographic attacks and requires more robust encryption solutions. [23] reported that their anomaly tree-based DDoS traceback method, while simplifying detection, is susceptible to inaccuracies from erroneous data and noise, in [24] suffers from poor performance on highly constrained hardware due to the complexity of chaotic key generation.

III. Adaptive Secure XTEA for Embedded Microcontrollers in Smart Cities

As smart cities rely more on networked devices for crucial urban infrastructure, the demand for strong security measures grows. XTEA is a lightweight encryption solution designed for resource-constrained contexts, making it a good fit for embedded processor-based SDN nodes. However, using XTEA in these systems involves specific obstacles that need to be overcome to maintain effective security. Hence, an **Adaptive Secure XTEA for Embedded Microcontrollers (ASX-EM)** is proposed to address the goals of mitigating vulnerabilities caused by POA, optimizing performance for 8-bit microcontrollers, and developing customized XTEA encryption methods for embedded processor-based SDN nodes in smart cities. Many existing embedded processor-based SDN controllers ignore adequate padding validation to improve decryption performance. This carelessness makes them vulnerable to Padding Oracle Attacks, which allow attackers to change ciphertext and exploit incorrect answers from the SDN controller. By iterating through potential modifications, attackers can decrypt sensitive data block by block, compromising the integrity and confidentiality of the communication. To mitigate POA vulnerabilities, the Context-Aware Key Expansion and Secure Padding Validation (CAKE-SPV) are introduced. Here, A Context-Aware Key Expansion Scheduling technique is developed, which customizes the key scheduling process to node-specific factors such as MAC address and node ID. This ensures that even if one key is compromised; the security of other nodes remains intact. Additionally, the OAEP with HIPV mechanism provides secure padding and integrated validation. OAEP adds random padding and a cryptographic hash to the plaintext before encryption. During decryption, the HIPV checks hash consistency, aborting if the message is tampered with, thus ensuring ciphertext integrity and preventing information leakage through padding oracle attack vulnerabilities. This technique adds an extra layer of security to XTEA encryption, safeguarding against potential POA risks.

Furthermore, smart city infrastructure frequently relies on a large number of interconnected devices, many of which are powered by 8-bit microcontrollers since they are inexpensive and consume little power. Such devices manage important tasks including public safety, environmental monitoring, and traffic control. Many existing XTEA designs, which include multiple rounds of complicated arithmetic and bitwise operations, are not appropriate for 8-bit microcontrollers. These devices struggle with computational overhead, causing considerable delays during encryption and decryption. The restricted processing power and memory increase latency difficulties, preventing real-time data transfer. As a result, essential applications in smart city infrastructure may encounter delays, jeopardising the security and efficiency of sensitive data exchange. This inefficiency presents a substantial difficulty for implementing strong encryption in resource-limited contexts. Hence, an Adaptive Round and Parallel Processing (ARPP) method is introduced to optimize XTEA for 8-bit microcontrollers in smart city infrastructure. The Dynamic Round Adjustment technique uses Threshold-based Adaptive Control Logic to monitor system parameters such as CPU load, memory, and network traffic, and then adjusts the number of encryption rounds in real-time. This lowers the need for huge buffers or storage spaces, reducing the necessary chip area. During periods of low system load or restricted resources, fewer rounds accelerate processing without risking security. Furthermore, Bit-Slice Processing with Precomputed Lookup Tables accelerates encryption/decryption by processing bit-level blocks concurrently and obtaining precomputed values. This reduces arithmetic operations while dramatically increasing XTEA efficiency on 8-bit microcontrollers, making it suitable for smart city applications.

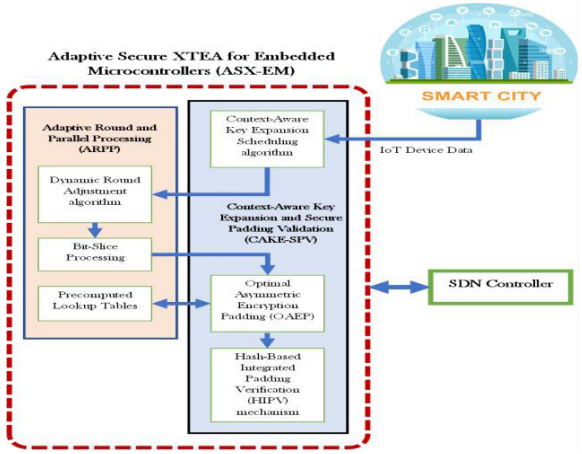

Fig.1: Overall flow diagram of the proposed model

The overall flow diagram of the suggested model is illustrated in the figure 1. IoT devices in the smart city transmit data to the ASX-EM framework. Initially, the Context-Aware Key Expansion method produces unique keys for each node depending on predefined factors. The Dynamic Round Adjustment program then analyses system parameters and modifies the number of encryption rounds accordingly. The XTEA algorithm is used to encrypt data, which is optimized with Bit-Slice Processing and Precomputed Lookup Tables. Furthermore, OAEP provides padding and a hash for safe transmission, while the HIPV technique maintains integrity

during decryption. Finally, the encrypted data is decrypted, and the HIPV detects manipulation. If the data is genuine, it is processed; otherwise, the operation is aborted. The SDN Controller manages the whole process, guaranteeing optimal resource allocation and network management.

A. Context-Aware Key Expansion Scheduling

The Context-Aware Key Expansion Scheduling method is proposed to improve the security of the XTEA encryption system by creating unique encryption keys for specific nodes in a dispersed network. This approach incorporates node-specific factors into the key scheduling process, resulting in unique, pseudo-random encryption keys for each node. This customization enables each node to produce unique encryption keys, which improves security, particularly in dispersed networks. This solution dramatically enhances the security of XTEA encryption by preventing a key breach from influencing the security of other nodes.

The algorithm begins by generating a base key K_b from node-specific parameters. let MAC be the node's MAC address and NodeID be its unique identification. These parameters serve as the basis for generating unique keys for each node. The unique key for each node is generated by combining the base key with the node-specific parameters, which is mathematically represented as in equation (1)

$$K_b = f(MAC, NodeID) \quad (1)$$

Where K_b is the base key. f is a cryptographic function that combines the base key with the node-specific parameters, which is designed using a hash function to ensure that the output is pseudo-random and unique for each node. Once the base key is generated, the key scheduling process begins. The key scheduling function K_{sch} takes the base key K_b and a context parameter C_i. The context parameter is a combination of node-specific parameters or an additional random value for added security.

$$K_{sch}(K_b, C_i) \rightarrow \{K_b^{(1)}, K_b^{(2)}, K_b^{(3)}, \dots, K_b^{(n)}\} \quad (2)$$

The basic key K_b is extended into round keys $\{K_b^{(1)}, K_b^{(2)}, K_b^{(3)}, \dots, K_b^{(n)}\}$ for XTEA encryption rounds. The Context-Aware Key Expansion Scheduling algorithm generates these round keys in a manner that incorporates node-specific context. The key scheduling is adaptive; this is change based on the operational context of the node. To further strengthen the uniqueness of the keys, the algorithm modifies the round keys based on contextual information:

$$C_i = h(MAC, NodeID, V_i) \quad (3)$$

Where V_i is a nonce to ensure the uniqueness of the context parameter, and h adjusts the round key based on the current context, ensuring that even under similar conditions, the keys remain distinct. The key scheduling function K_{sch} uses a pseudo-random number generator (PRNG) seeded with the base key K_b and context parameter C_i to produce the round keys. The PRNG ensures that the round keys are unique and pseudo-random.

$$K_f = PRNG(C_i, V_i) \quad (4)$$

Here, the V_i is a unique number used to prevent replay attacks, ensuring that each key generated is distinct even if the same base key is used. The Context-Aware Key Expansion Scheduling approach generates round keys that are subsequently employed in the XTEA encryption process.

Algorithm 1: Context-Aware Key Expansion Scheduling algorithm

Input: MAC Address, Node ID, Context Parameters C_i, Nonce V_i, and
Step 1: Generate a unique key for node i
Step 2: Generate the context parameter $C_i$using the function h
Step 3: Initialize RoundKeys array
Step 4: Key scheduling using PRNG seeded with base key K_b and context parameter C_i
    for i from 0 to n do
        $K[i] \leftarrow K_{sch}((K_b, C_i))$
    end for
Step 5: Key scheduling function K_{sch} uses a PRNG: $K[i] = PRNG(C_i, V_i)$
Step 6: Return unique context-aware key K_f
Output: A series of unique, context-aware keys for encryption.

This promises that even if one node's key breaches, the security of other nodes is preserved through unique sub-keys formed from node-specific factors. This pseudo-random and unique key expansion considerably improves the security of the XTEA encryption technique. This customization makes it difficult for an attacker to derive keys for other nodes even if one key is compromised.

B. Dynamic Round Adjustment Algorithm

The Dynamic Round Adjustment technique optimizes the XTEA encryption process for 8-bit microcontrollers used in smart city infrastructures. With this approach, XTEA encryption security and performance are optimally balanced since the number of encryption rounds is constantly adjusted based on real-time system parameters, even under varying computational loads and resource availability. Threshold-based adaptive control logic monitors various system metrics and makes decisions about the number of encryption rounds required at any given time.

In the first step, the system predefined the number of encryption rounds minimum (R_{min}) and maximum (R_{max}). The intended security level and the microcontroller's capabilities are used to define these values. Additionally, threshold values are set for network traffic, CPU load, and memory availability. The number of encryption rounds will be increased or decreased based on these thresholds. These thresholds help the algorithm decide when to adjust the number of encryption rounds.

The algorithm continuously monitors real-time data on the identified system metrics (CPU load, available memory, and network traffic). The monitored values are compared against predefined thresholds to evaluate the system's current state.

CPU Load Evaluation

- If the CPU load is below a low threshold, it indicates low processing demand, allowing the system to afford more encryption rounds for enhanced security.
- If the CPU load is above a high threshold, it suggests high processing demand, prompting a reduction in the number of rounds to free up processing power.

Memory Evaluation

- High available memory allows for more encryption rounds without risking memory overflow or significant slowdowns.
- Low available memory necessitates reducing the number of rounds to conserve resources.

Network Traffic Evaluation

- Low network traffic permits more encryption rounds as the system can handle additional processing without impacting transmission speed.
- High network traffic requires fewer encryption rounds to maintain timely and efficient data transmission.

The algorithm calculates the optimal number of encryption rounds (R_{op}) based on a weighted function of the monitored metrics:

$$R_{op} = R_{min}\left(\frac{\text{CPU load+memory+Traffic factor}}{3}\right) \times (R_{max} - R_{min}) \quad (5)$$

The calculated R_{op} is then used to update the number of encryption rounds in real time. The number of encryption rounds is dynamically adjusted in real time based on the ongoing assessment of system metrics. This adjustment helps keep the encryption process both efficient and secure, even as system conditions change. Finally, the algorithm applies the updated number of rounds to the XTEA encryption process. This continuous modification ensures that the system keeps running without any problems, dynamically adjusting to the present situation. The encryption and decryption operations now proceed with the adjusted rounds, ensuring that the system operates efficiently without compromising security. The Dynamic Round Adjustment Algorithm is shown in the following algorithm 2.

Algorithm 2: Dynamic Round Adjustment Algorithm

1. Start

2. Initialize Parameters

Set R_{min}, R_{max}, and Define thresholds for CPU load, memory availability, and network traffic

3. Monitor System Metrics

Continuously collect data on CPU load, available memory, and network traffic

4. CPU Load Evaluation

If CPU load < low threshold, increase encryption rounds

If CPU load > high threshold, decrease encryption rounds

5. Memory Evaluation

If available memory is high, increase encryption rounds

If available memory is low, decrease encryption rounds

6. Network Traffic Evaluation

If network traffic is low, increase encryption rounds

If network traffic is high, decrease encryption rounds

7. Calculate Optimal Number of Rounds

$$R_{op} = R_{min}\left(\frac{\text{CPU load} + \text{memory} + \text{Traffic factor}}{3}\right) \times (R_{max} - R_{min})$$

Adjust the number of encryption rounds to R_{op}

Implement the adjusted number of rounds in the XTEA encryption process

8. Repeat from step 3

9.End

By altering the number of encryptions rounds dynamically, the Dynamic Round Adjustment algorithm method reduces the need for huge buffers or storage locations. This is especially important for 8-bit microcontrollers, which have limited memory and computing capability.

Once the number of encryption rounds is updated, the Bit-Slice Processing is used to divide data into smaller bit-level segments, allowing several encryption processes, which is explained in the following section 3.3.

C. Bit-Slice Processing

Conventional byte-oriented processing processes data in 8-bit (or larger) chunks, resulting in inefficiencies while performing concurrent activities. Hence the bit-slice processing is used in this research, which divides data into discrete bits such that many bits carry out operations concurrently. This method greatly increases efficiency and speed by enabling the simultaneous execution of many encryptions or decryption operations.

The specific architecture of the 8-bit microcontroller determines how bits are efficiently processed in parallel. This includes the availability of parallel execution units and the capability to handle bit-level operations. By dividing the data into slices, multiple bit-wise operations are executed simultaneously, improving throughput and reducing processing time on 8-bit microcontrollers. The Bit-Split function is defined as follows in equation (6)

$$B_s(Q(y)) = B_k \oplus B_{k-1} \oplus, \ldots, \oplus B_i \quad (6)$$

Here, $\oplus$ denotes the bitwise XOR operation, and B_k, B_{k-1},…, B_i represent the different segments obtained by splitting the bit sequence. These bit-level blocks are then processed simultaneously.

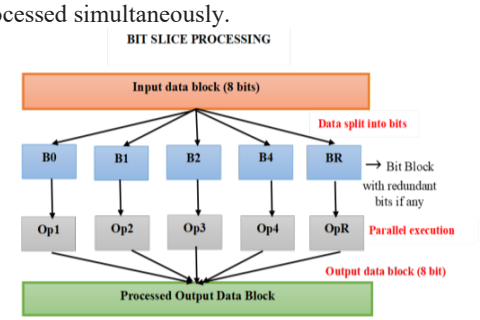

Fig.2: Bit-Slice Processing

Figure 2 depicts the bit slice processing of the proposed model. This implies that multiple encryption/decryption operations are performed at the same time, making use of parallel processing units within the microcontroller. This bit-slice processing offers a flexible and scalable approach to processor architecture, allowing for effective execution of arithmetic and logical operations in parallel. This method is especially useful in resource-constrained situations, such as 8-bit microcontrollers, where optimal performance and resource utilization are crucial. Bit-slice processing improves flexibility, speed, and efficiency in microprocessor designs by breaking down the data route into smaller slices. Once the bit-slicing operation is done, OAEP is utilized for proper encryption, which is explained in the following section 3.4.

D. Optimal Asymmetric Encryption Padding

To provide a secure padding, the OAEP is utilized for encrypting the plaintext message from the IoT device. OAEP is a padding strategy that is frequently used in conjunction with asymmetric encryption algorithms to increase security, particularly against POAs and other cryptographic vulnerabilities. The process of OAEP requires numerous phases, from appending padding to the plaintext to validating the message's integrity after decryption.

Begin with the plaintext message needs to be encrypted. Let M be the original plaintext message. The first step is to determine the required length of the padding. The total length of the padded message needs to match the block size of the encryption algorithm. Apply padding to M to ensure the total length is a multiple of the block size n required by the encryption algorithm. Extra bytes are added to the plaintext to ensure that it fits the encryption algorithm's required block size. This stage also includes adding random bytes to the message to ensure that the same message encrypted several times produces distinct ciphertexts, hence increasing security. Let $P(M)$ denote the operation of adding padding P to M:

$$P(M) = P \oplus M \tag{7}$$

Where $\oplus$ denotes concatenation, and P is the random padding added to make the message length compliant with block size requirements. The padding is random, enhancing security by making it difficult for attackers to predict the padding structure. Then compute a cryptographic hash $H(M)$ of the original plaintext M, which is expressed in the following equation ()

$$h = H(M) \tag{8}$$

This hash h is important for verifying the integrity of the message during the decryption process. The generated hash value is attached to the message along with the random padding. Concatenate the padded message $P(M)$ with the hash h, which is expressed in the following equation ()

$$D_M = P(M) \oplus h \tag{9}$$

This combined message D_M includes both the padded plaintext and the hash, ensuring both data integrity and security. Encrypt the concatenated message D_M using the XTEA encryption algorithm with a key K_f.

$$C = E_{K_f}(D_M) \tag{10}$$

Here, K_{K_f} is the encryption function, and C is the resulting ciphertext. The encryption key K_f is uniquely generated for each node using the Context-Aware Key Expansion Scheduling algorithm, ensuring that the keys are pseudo-random and node-specific. The encryption process transforms the combined message into ciphertext, ensuring its confidentiality during transmission. The encrypted message (ciphertext C) is transmitted to the intended recipient. By including random padding in the message, OAEP promises that even if the same plaintext is encrypted numerous times, the resultant ciphertext is unique each time. This randomization makes it far more difficult for an attacker to anticipate or manipulate the ciphertext. OAEP is designed to work effectively within the constraints of 8-bit microcontrollers. The use of a lightweight hash function and efficient padding mechanisms ensures that the encryption process remains fast and resource-efficient. The algorithm for OAEP is explained in Algorithm 3.

Algorithm 3: Optimal Asymmetric Encryption Padding

Inputs: Plaintext message M, Block size n of the encryption algorithm, encryption key K_f generated using the Context-Aware Key Expansion Scheduling algorithm, and Hash function H

Output: Ciphertext C

1. Compute the length of the padding P needed to make M fit the block size n.
2. Generate a random padding P of appropriate length.
3. Concatenate the random padding P with the plaintext message M:
4. Compute the cryptographic hash h of the original plaintext M
5. Combine the padded message P(M) with the hash h
6. Encrypt the combined message D_M using the XTEA encryption algorithm:

The resulting ciphertext C is returned.

Send the ciphertext C to the intended recipient.

To optimize the XTEA encryption algorithm by using precomputed lookup tables, reducing real-time computational overhead, and improving processing efficiency on 8-bit microcontrollers. Frequently used arithmetic operations in the XTEA algorithm are precomputed and stored in lookup tables. During encryption or decryption, instead of performing the computation in real time, the algorithm retrieves the result from the table. The main purpose is to speed up encryption and decryption by avoiding repetitive computations, especially for computationally expensive operations. By accessing precomputed values, the need for real-time arithmetic operations is minimized, leading to faster encryption and decryption. Minimizes processor cycles and memory usage, making it suitable for 8-bit microcontrollers with limited resources.

The OAEP method is enhanced by HIPV, which checks the integrity of the decrypted message. By recalculating the hash after decryption and comparing it to the initial hash, the system

assures that the message has not been altered, which is explained in the following section 3.5.

E. Hash-Based Integrated Padding Verification

HIPV is designed as an additional layer for improving the security of encrypted data by including padding validation directly into the decryption process. It aims to prevent vulnerabilities corresponding to the POA by ensuring that only valid ciphertexts are decrypted.

Upon receiving the ciphertext, the SDN controller decrypts it using the XTEA decryption algorithm. This restores the padded and hashed plaintext message. The recipient receives the ciphertext that needs to be decrypted. Decrypt the incoming ciphertext to get the padded plaintext and appended hash. The ciphertext C is decrypted using the XTEA decryption function $D_{K_f}(\cdot)$ with key K_f, restoring the combined message M'.

$$M' = D_{K_f}(C) \tag{11}$$

The decrypted message is divided into two parts: the original message with padding and the hash value. This extracted hash was appended during the encryption phase and serves as a reference for integrity verification. The decrypted message M' is split into the padded plaintext P' and the extracted hash H_P'.

Then the random padding bytes are removed from the decrypted message to extract the original plaintext. This step restores the plaintext to its original form before padding and hashing. The padding bytes q are removed from P', yielding the extracted plaintext P. A new hash is calculated from the extracted plaintext using the same cryptographic hash function as in the encryption phase. This recalculated hash (H_P'') is compared with the extracted hash to verify the integrity of the message. Calculate the hash of the extracted plaintext P', which is expressed in the following equation ()

$$H_P'' = H(P') \tag{12}$$

Compare the recalculated hash H_P'' with the extracted hash H_P'. If the recalculated hash matches the retrieved hash, it means that the message was not tampered with. The plaintext is regarded as valid. If the hashes do not match, it means that the message was tampered with. In this situation, the decryption operation is terminated, and the plaintext is not utilized. This hash consistency check detects ciphertext manipulation and padding issues during decryption. If the hashes do not match, the decryption is halted, ensuring the message's integrity

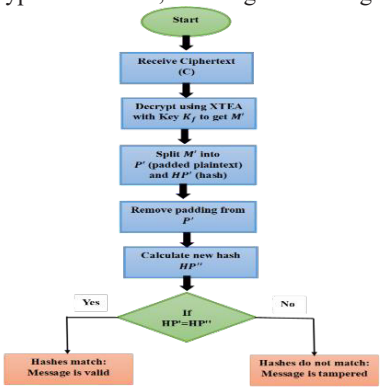

Fig.3: Flow chart for Hash-Based Integrated Padding Verification

Figure 3 illustrates the flow chart for HIPV. HIPV offers an adequate framework for maintaining the integrity and security of encrypted messages. By combining hashing and padding verification, it overcomes significant vulnerabilities including the POA while remaining efficient, making it ideal for embedded processor-based smart city systems. This structured security technique ensures that only genuine and untampered ciphertexts are handled, which considerably improves the encryption scheme's overall resilience. The OAEP with HIPV approach improves the security of encrypted data by ensuring that the plaintext is securely padded and verifiable.

Overall, the proposed approaches solve a variety of difficulties while considerably improving the security, performance, and efficiency of XTEA encryption for integrated processor-based SDN nodes in smart cities.

IV. RESULT AND DISCUSSION

The performance of the proposed system and the implementation findings are explained in depth in this section, which also includes a comparison section to verify that the suggested technique is appropriate for data security in embedded processor-based SDN nodes in smart cities.

A. System Configuration

The proposed data security methodology has been simulated in MATLAB. The evaluation is conducted by varying the data size correspondingly.

Software	: MATLAB
OS	: Windows 10 (64-bit)
Processor	: Intel i5
RAM	: 8GB RAM

The simulation in MATLAB was conducted using an embedded microcontroller model, which includes hardware-supported operations such as bitwise logic, arithmetic computations, and memory access. The microcontroller supports fixed-point arithmetic and instruction-level optimizations, enabling efficient execution of encryption algorithms.

B. Performance of the proposed model

This section discusses the experimental results from the initial setup of the suggested model for ASX-EM to mitigate POA vulnerabilities, and optimize performance for 8-bit microcontrollers in embedded processor-based SDN nodes in smart cities.

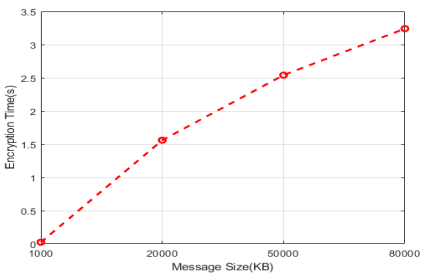

Fig.4: Encryption time of the suggested model

The suggested model's encryption time is displayed in Figure 4. When the message size is 80000KB, the suggested model achieves an encryption time of 3.25s, while when the message size is 1000KB, it achieves an encryption time of 0.07s. If the message size increases, the encryption time of the proposed model also increases. OAEP uses a lightweight hash function and efficient padding procedures to keep the encryption process speedy and resource-efficient, thereby reducing the encryption time.

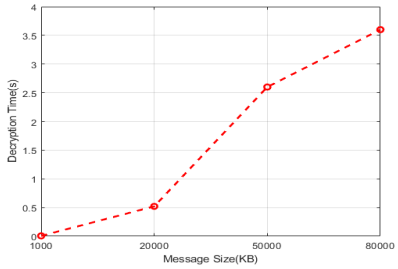

Fig.5: Decryption time of the suggested model

Figure 5 shows the suggested approach's decryption time. The suggested model achieves the fastest decryption time of 3.6 seconds when the message size is 80000KB and the fastest encryption time of 0.07 seconds when the message size is 1000KB. By processing many bits at once and utilizing parallel execution units, the bit slice process technique shortens the time required for encryption and decryption without increasing clock speed.

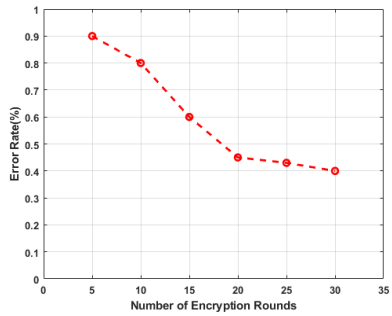

Fig.6: Error rate of the suggested model

Figure 6 presents the error rate of the suggested model across different encryption rounds. At 25 encryption rounds, the model exhibits an error rate of 0.9%, whereas at 5 encryption rounds, the error rate decreases to 0.4%. By verifying that only authentic and unaltered ciphertexts are decrypted, CAKE-SPV in conjunction with HIPV lowers the error rate associated with improper decryption.

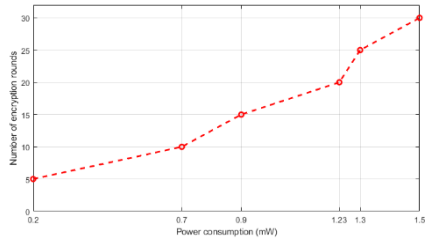

Fig.7: Power consumption of the suggested model

Figure 7 displays the power consumption of the recommended model for varying encryption rounds. The graph indicates that as encryption rounds increase, so does the power consumption. When the encryption rounds are at 5, the power consumption is 0.2mW, whereas when the encryption rounds are increased to 30, the power consumption of the suggested model is 1.5mW. By reducing the computational complexity and number of required operations, these Precomputed Lookup Tables help in maintaining lower power consumption.

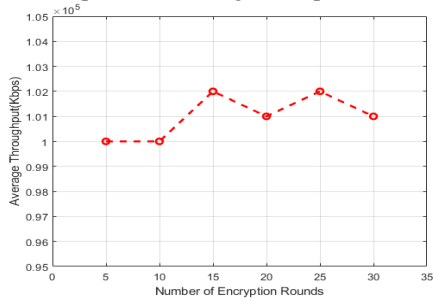

Fig.8: Average throughput of the suggested model

Figure 8 depicts the suggested model's average throughput. The suggested model achieves an average throughput value of 1.00000 Kbps when the number of encryption rounds is 5 and also attains an average throughput value of 1.03000 Kbps when the encryption rounds are 25. By optimizing the number of rounds based on system conditions using dynamic round adjustment, the average throughput is maintained at an optimal level, balancing security needs and system performance.

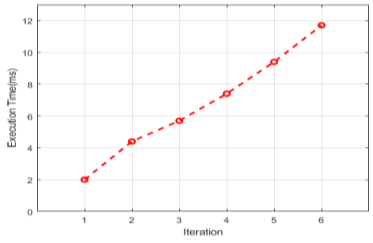

Fig.9: Execution time of the suggested model

The execution time of the suggested model is illustrated in the above figure 9. When the iterations are 6 and 1, the suggested method achieves the execution time of 11.8 ms, and 2 ms respectively. The proposed model's execution time expands with the number of iterations. The dynamic round adjustment algorithm reduces execution time by having logic constantly check system parameters and decide the ideal amount of encryption rounds in real time.

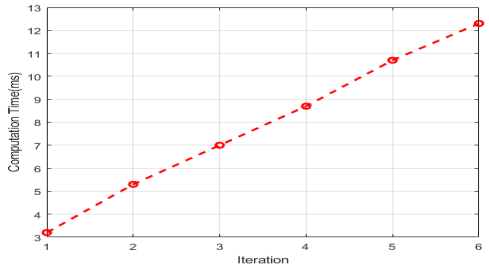

Fig.10: Computation time of the suggested model

The suggested model's computation time is represented in Figure 10. The computation time increases linearly with each iteration, starting from around 3.1 ms in the first iteration to 12.4 ms in the sixth iteration. Precomputed lookup tables and parallel processing are significantly reducing computation times by eliminating needless calculations and accelerating encryption and decryption processes.

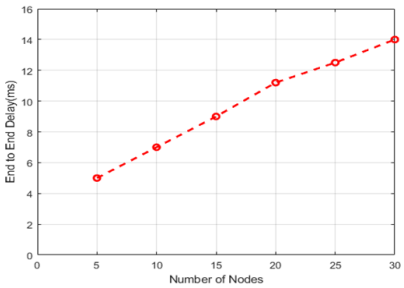

Fig.11: End-to-end delay of the suggested model

The suggested model's end-to-end delay is shown in Figure 11. The mode's end-to-end delay increases as the number of nodes rises. When the number of nodes is 5 the suggested approach attains a delay of 5 ms, also when the number of rounds is 30 the suggested approach attains a delay of 14 ms respectively. The suggested ASX-EM's ability to control end-to-end latency over a range of node densities emphasizes its applicability for scalable smart city applications, ensuring safe and effective communication even as the network increases.

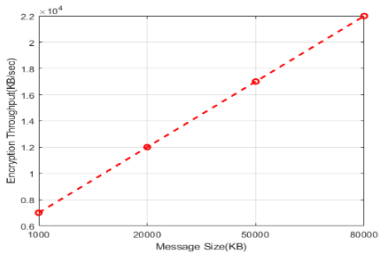

Fig.12: Encryption throughput of the suggested model

Figure 12 shows the encryption throughput across various iterations. The proposed model achieves 22000KB/sec and 6000KB/sec encryption throughput, respectively, at 80000KB and 1000KB message sizes. The encryption throughput of the proposed model rises with message size. The proposed ASX-EM maintains a high level in terms of encryption throughput, which indicates that the optimizations, such as Bit-Slice Processing, effectively sustain throughput performance without degradation over multiple iterations.

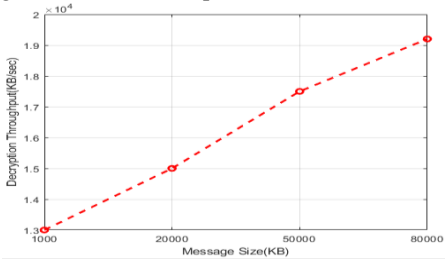

Fig.13: Decryption throughput of the suggested model

Figure 13 shows the decryption throughput across various iterations. When the message size is 80000KB and 1000KB, the suggested approach achieves a decryption throughput of 19200KB/sec and 13000KB/sec, respectively. The proposed ASX-EM maintains a high level in terms of decryption throughput, which indicates that the optimizations, such as Bit-Slice Processing, effectively sustain throughput performance without degradation over multiple iterations.

C. Comparative analysis of the proposed model

In this section, a detailed explanation of the effectiveness of the suggested technique and the achieved outcome were explained. According to the evaluation, the following metrics have been considered: encryption throughput, decryption throughput, End-to-End delay, encryption time, decryption time, throughput/area, latency, and execution time**.**

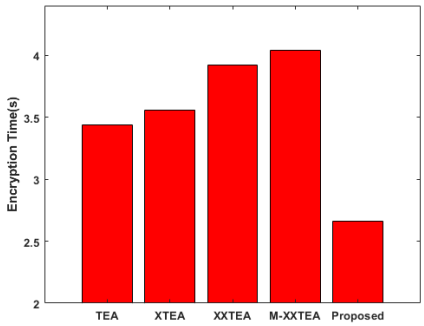

Fig.14: Comparison of encryption time

A comparison of the encryption time of the suggested model with existing models at 50000 KB message size is shown in Figure 14. The encryption time is used to measure how quickly plaintext data can be converted into ciphertext, which is essential for ensuring efficient and real-time secure communication in smart city applications. The existing models [16] such as TEA, XTEA, XXTEA, and M-XXTEA attain an encryption time of 3.43s, 3.55s, 3.91s, and 4.03s, Whereas the proposed model achieves an encryption time of 2.7s. Compared to previous approaches, the proposed model has less encryption time.

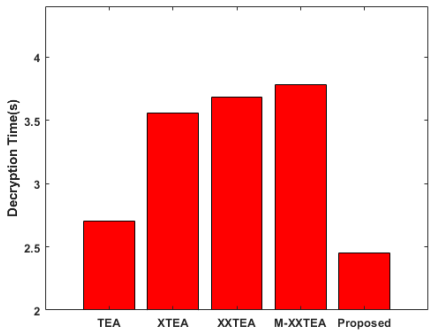

Fig.15: Comparison of decryption time

A comparison of the decryption time of the suggested model with existing models at 50000 KB message size is shown in

Figure 15. The decryption time is essential for assessing how quickly an encrypted message can be converted back to its original form, which is particularly crucial for real-time data access in smart city applications. The existing models [16] such as TEA, XTEA, XXTEA, and M-XXTEA attain a decryption time of 2.7s, 3.57s, 3.66s, and 3.78s, Whereas the proposed model achieves a decryption time of 2.4s. Compared with all the above existing models the proposed model attains a low decryption time.

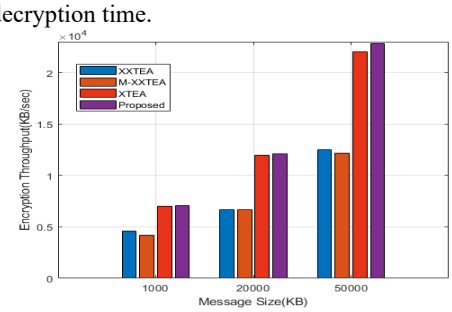

Fig.16: Comparison of encryption throughput

Figure 16 illustrates the encryption throughput of the suggested model with the existing model [16]. Encryption throughput measures the rate at which data is encrypted per unit of time, typically in KB/sec or Mbps. It is crucial for smart city applications where large volumes of data need to be securely processed in real-time. In encryption throughput, the proposed model demonstrates superior performance, especially for larger message sizes (50,000 KB), reaching around 22,000 KB/sec compared to existing models such as 13,000 KB/sec for XXTEA, 12,500 KB/sec for M-XXTEA, and 20,000 KB/sec for XTEA respectively.

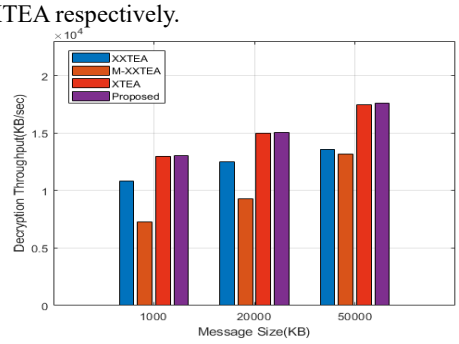

Fig.17: Comparison of decryption throughput

Figure 17 illustrates the decryption throughput of the suggested model with the existing model [16]. Decryption throughput measures how efficiently an encrypted message can be converted back to its original form per unit of time. This is crucial in smart city applications, where real-time data access is essential for traffic control, environmental monitoring, and public safety. The suggested model outperforms existing models in terms of decryption throughput, particularly for larger message sizes (50,000 KB), reaching roughly 17600 KB/sec compared to 13,800 KB/sec for XXTEA, 13,000 KB/sec for M-XXTEA, and 17,500 KB/sec for XTEA.

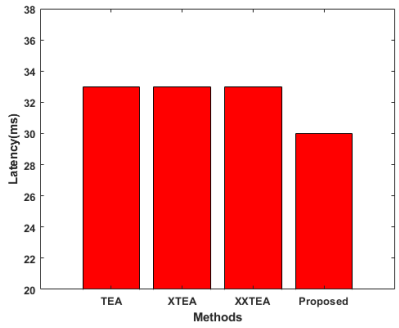

Fig.18: Comparison of latency of the suggested model

A latency comparison between the suggested model and the current models is shown in Figure 18. Latency measures the time delay between the input of a data packet and its corresponding output after encryption or decryption. The latency for TEA, XTEA, and XXTEA [18] in the existing models is 33ms, 33ms, and 33ms, respectively. The lowest latency of the suggested model is 30 ms when compared to the existing models.

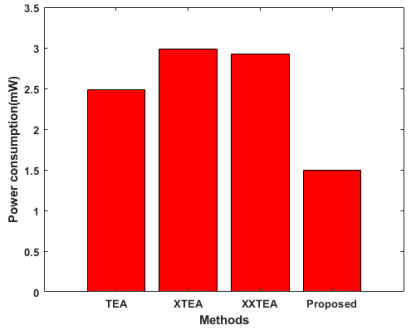

Fig.19: Comparison of power consumption

The power consumption comparison of the suggested model with the existing models is represented in Figure 19. Power consumption is used to evaluate the energy efficiency of the encryption model, which is crucial for embedded microcontroller-based SDN nodes in smart cities. The various existing [18] models including TEA, XTEA, and XXTEA attain a power consumption value of 2.5mW, 3mW, and 2.92mW respectively. Compared with existing models the suggested model achieves a low power consumption of 1.5mW.

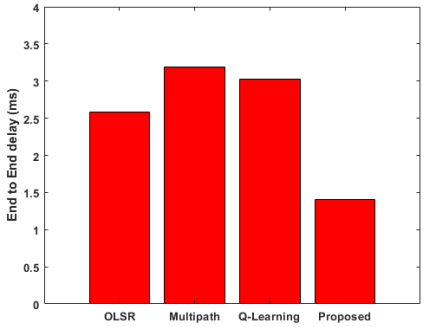

Fig.20: Comparison of end-to-end delay

The comparison of the suggested model's End-to-End Delay with existing models is shown in Figure 20. End-to-end delay measures the total time taken for a data packet to travel from the source to the destination, including encryption, transmission, processing, and decryption delays. The existing models [27] such as OLSR, Multipath, and Q-Learning are attaining an end-to-end delay value of 42ms, 18ms, and 15ms. Compared with existing models the suggested model achieves the lowest end-to-end delay of 14ms.

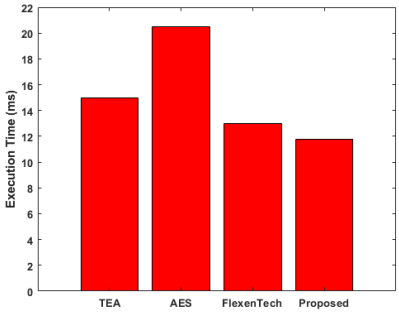

Fig.21: Comparison of execution time

A comparison of the execution time of the suggested model with various existing models is depicted in Figure 21. Execution time measures the total time required for the encryption and decryption processes to complete. This metric is crucial for embedded microcontroller-based SDN nodes in smart cities, as they have limited computational resources and operate in real-time environments. Various existing models [26] such as TEA, AES, and FlexenTech have an execution time of 15ms, 20.5ms, and 13ms, conversely, the suggested mode attains an execution time of 11.8ms. Compared with existing models the suggested model attains the lowest execution time.

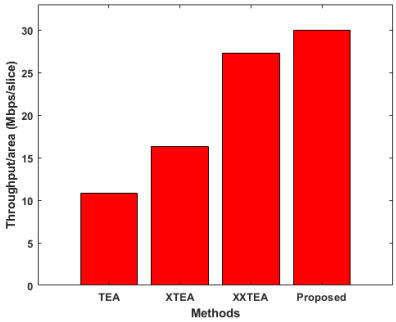

Fig.22: Comparison of throughput/area

Figure 22 illustrates the comparison of the throughput/area of the suggested model with the existing models. Throughput measures the rate at which data is successfully processed (encrypted or decrypted) over time. This metric is crucial for embedded microcontroller-based SDN nodes in smart cities, as they process large volumes of real-time data from connected devices like traffic management systems, surveillance cameras, and environmental sensors. The existing models [18] such as TEA, XTEA, and XXTEA attain a throughput/area value of 10.87 Mbps/slice, 16.27 Mbps/slice, and 27.30 Mbps/slice. Compared with existing models the suggested model attains the highest throughput/area value of 30 Mbps/slice.

TABLE I
COMPARISON OF THE PROPOSED MODEL WITH OTHER EXISTING APPROACHES

Metric	Proposed Model	TEA	XTEA	XXTEA	M-XXTEA
Encryption Time (s)	2.7	3.43	3.55	3.91	4.03
Decryption Time (s)	2.4	2.7	3.57	3.66	3.78
Encryption Throughput (KB/sec)	22,000	-	20,000	13,000	12,500
Decryption Throughput (KB/sec)	17,600	-	17,500	13,800	13,000
Latency (ms)	30	33	33	33	-
Power Consumption (mW)	1.5	2.5	3	2.92	-
Throughput/ Area (Mbps/slice)	30	10.87	16.27	27.30	-

Table 1 proposed model outperforms TEA, XTEA, XXTEA, and M-XXTEA in encryption and decryption time, achieving the fastest execution of 2.7s and 2.6s, respectively. It also demonstrates superior encryption and decryption throughput of 22,000 KB/sec and 17,600 KB/sec while maintaining the lowest power consumption of 1.5 mW. Additionally, the proposed model achieves the highest throughput per area of 30 Mbps/slice, highlighting its efficiency in resource utilization.

TABLE II
COMPARISON OF THE PROPOSED MODEL END-TO-END DELAY AND EXECUTION TIME WITH OTHER EXISTING APPROACHES

Metric	Proposed Model	TEA	OLSR	Multipath	Q-Learning	AES	Flexen Tech
End-to-End Delay (ms)	14	-	42	18	15		
Execution Time (ms)	11.8	15	-	-	-	20.5	13

Table 2 shows the proposed model achieves the lowest end-to-end delay of 14 ms compared to OLSR of 42 ms, Multipath of 18 ms, and Q-Learning of 15 ms, ensuring faster data transmission. It also outperforms TEA and AES in execution time, completing tasks in 11.8 ms, which is faster than TEA of 15 ms and AES of 20.5 ms. These results highlight the proposed model's efficiency in both latency and computational performance.

Overall, in the results section, the proposed model is compared to existing models, and the performance is explained using graphs. This shows that the technique that is used in the novelty Dynamic XTEA Optimization and Secure Key Management for Embedded Microcontroller-based SDN for smart cities has comparatively higher encryption throughput and decryption throughput, and low decryption time, encryption time, and execution time than the previous techniques that are taken for the comparison.

V. CONCLUSION

In conclusion, the proposed ASX-EM tackles major security and performance concerns for SDN nodes in smart cities. By using CAKE-SPV, the suggested approach successfully mitigates POA vulnerabilities while providing secure communication between nodes. Moreover, the ARPP method addresses limitations about chip space, power consumption, and processing time to optimize XTEA for 8-bit microcontrollers. Efficient encryption and decryption operations are made possible by the Dynamic Round Adjustment method and Bit-Slice Processing with Precomputed Lookup Tables, which balance security needs with efficiency. These combined strategies make ASX-EM a highly efficient and secure encryption method suitable for the constrained environments of smart city SDN nodes. Compared with existing models TEA, XTEA, XXTEA, AES and M-XXTEA the proposed model achieves a high encryption throughput of 22,000 KB/sec, decryption throughput of 17600 KB/sec, and low execution time of 11.8ms, power consumption of 1.5mW, encryption time of 2.7s and decryption time of 2.6s. The proposed solution not only mitigates prevalent security risks but also ensures the smooth and efficient operation of smart city infrastructures, paving the way for future advancements in secure and efficient embedded systems.

REFERENCES

[1] A. Al Hayajneh, M. Z. A. Bhuiyan, and I. McAndrew, "Improving internet of things (IoT) security with software-defined networking (SDN)," *Computers*, vol. 9, no. 1, p. 8, 2020.

[2] W. Iqbal, H. Abbas, P. Deng, J. Wan, B. Rauf, Y. Abbas, and I. Rashid, "ALAM: Anonymous lightweight authentication mechanism for SDN enabled smart homes," *Journal of Network and Computer Applications*, p. 103 672, 2023.

[3] S. Atiewi, A. Al-Rahayfeh, M. Almiani, S. Yussof, O. Alfandi, A. Abugabah, and Y. Jararweh, "Scalable and secure big data IoT system based on multifactor authentication and lightweight cryptography," *IEEE Access*, vol. 8, pp. 113 498–113 511, 2020.

[4] N. Anand, and M. A. Saifulla, "EN-LAKP: Lightweight Authentication and Key Agreement Protocol for Emerging Networks," *IEEE Access*, vol. 11, pp. 28 645–28 657, 2023.

[5] A. Ragab, G. Selim, A. Wahdan, and A. Madani, "Robust hybrid lightweight cryptosystem for protecting IoT smart devices. In Security, Privacy, and Anonymity in Computation, Communication, and Storage: SpaCCS 2019 International Workshops, Atlanta, GA, USA, July, 2019, Proceedings," *Springer International Publishing*, vol. 12, no. 14–17, pp. 5–19, 2019.

[6] N. N. Josbert, M. Wei, W. Ping, and A. Rafiq, "A look into smart factory for Industrial IoT driven by SDN technology: A comprehensive survey of taxonomy, architectures, issues and future research orientations," *Journal of King Saud University-Computer and Information Sciences*, p. 102 069, 2024.

[7] L. Vishwakarma, A. Nahar, and D. Das, "Lbsv: Lightweight blockchain security protocol for secure storage and communication in sdn-enabled iov," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 6, pp. 5983–5994, 2022.

[8] S.K. Shah, R. Sharma, and N. Shukla, "Data Security in IoT Networks using Software-Defined Networking: A Review. In 2022 IEEE World Conference on Applied Intelligence and Computing (AIC)," *IEEE*, pp. 909–913, 2022 June.

[9] Q. Zhou, J. Yu, and D. Li, "A dynamic and lightweight framework to secure source addresses in the SDN-based networks," *Computer Networks*, vol. 193, p. 108 075, 2021.

[10] S. Majhi, and P. Mitra, "Lightweight Cryptographic Techniques in 5G Software-Defined Internet of Things Networking. In Lightweight Cryptographic Techniques and Cybersecurity Approaches," IntechOpen, 2022.

[11] M. Rana, Q. Mamun, and R. Islam, "Current lightweight cryptography protocols in smart city IoT networks: a survey," *arXiv preprint arXiv:2010.00852*, (2020).

[12] I. S. Olimov, and X. I. Ibrohimov, "Analysis of Lightweight Cryptographic Algorithms. Golden Brain," vol. 1, no. 18, pp. 189–197, 2023.

[13] A. Diro, H. Reda, N. Chilamkurti, A. Mahmood, N. Zaman, and Y. Nam, "Lightweight authenticated-encryption scheme for internet of things based on publish-subscribe communication," *IEEE Access*, vol. 8, pp. 60 539–60 551, 2020.

[14] I. Laassar, and M. Y. Hadi, "Intrusion detection systems for internet of thing based big data: a review," *International Journal of Reconfigurable and Embedded Systems*, vol. 12, no. 1, p. 87, 2023.

[15] J. O. Olaide, "Internet of Things Security: Encryption Capacity Comparison for IoT Based on Arduino Devices," 2020.

[16] A. A. M. Ragab, A. Madani, A. M. Wahdan, and G. M. Selim, "Design, analysis, and implementation of a new lightweight block cipher for protecting IoT smart devices," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–18, 2023.

[17] M. Nagarajan, M. Rajappa, Y. Teekaraman, R. Kuppusamy, and A. R. Thelkar, "Research Article Renovated XTEA Encoder Architecture-Based Lightweight Mutual Authentication Protocol for RFID and Green Wireless Sensor Network Applications," 2022.

[18] Z. Mishra, and B. Acharya, "High throughput novel architectures of TEA family for high speed IoT and RFID applications," *Network*, vol. 61, p. 102 906, 2021.

[19] N. Khute, Z. Mishra, A.S. Rajput, and O. Parmar, "Optimized Hardware Implementation of XXTEA-192 for Resource Constrained Applications. In 2022 IEEE International Conference on Current Development in Engineering and Technology (CCET)," *IEEE*, pp. 1–6, 2022 December.

[20] D. Z. Fawwaz, S. H. Chung, C. W. Ahn, and W. S. Kim, "Optimal distributed MQTT broker and services placement for SDN-edge based smart city architecture," *Sensors*, vol. 22, no. 9, p. 3431, 2022.

[21] S. K. Keshari, V. Kansal, and S. Kumar, "A cluster based intelligent method to manage load of controllers in SDN-IoT networks for smart cities," *Scalable Computing: Practice and Experience*, vol. 22, no. 2, pp. 247–257, 2021.

[22] R. Anusha, and V. Shastrimath, "FID-MA XTEA: Cost-Effective RFID- Mutual Authentication Design Using XTEA Security on FPGA Platform," *International Journal of Electronics and Telecommunications*, vol. 67, 2021.

[23] W. Chen, S. Xiao, L. Liu, X. Jiang, and Z. Tang, "A DDoS attacks traceback scheme for SDN-based smart city," *Computers & Electrical Engineering*, vol. 81, p. 106 503, 2020.

[24] A. A. A. Abdulkadhim, A. S. Mahmood, and M. R. Ghanim, "Block of Data Encryption Using the Modified XTEA Algorithm," *Ingénierie des Systèmes d'Information*, vol. 29, no. 3, 2024.

[25] A. A. M. Ragab, A. Madani, A. M. Wahdan, and G. M. Selim, "Hybrid cryptosystems for protecting IoT smart devices with comparative analysis and evaluation. In Proceedings of the Future Technologies Conference (FTC) 2019," *Springer International Publishing*, vol. 1, pp. 862–876, 2020.

[26] O. A. Khashan, R. Ahmad, and N. M. Khafajah, "An automated lightweight encryption scheme for secure and energy-efficient communication in wireless sensor networks," *Ad Hoc Networks*, vol. 115, p. 102 448, 2021.

[27] L. El-Garoui, S. Pierre, and S. Chamberland, "A new SDN-based routing protocol for improving delay in smart city environments," *Smart Cities*, vol. 3, no. 3, pp. 1004–1021, 2020.

Sunil Kumar Shah is curently PhD scholar in Electronics Engineering at Amity University Madhya Pradesh, Gwalior, India. He did B. Tech from The Institution of Engineers (India) and M. Tech from Gyan Ganga Institute of Technology and Sciences, Jabalpur, India, in Embedded Systems andVLSI Design. His area of specialization includes IoT systems and VLSI Design for digital circuits.

Raghavendra Sharma received his B. Tech in Electrical Engineering from Dayalbagh University Agra, M. Tech in Electronic Design & Technology from Central Government Institute CEDTI, Aurangabad (MS), and PhD from Dayalbagh University, Agra. He is working as a professor and head at the Department of Electronics & Communication Engineering, Amity University Madhya Pradesh, Gwalior. He is a Fellow of the Institution of Electronics and Telecommunication Engineers (IETE), and the Institution of Engineers (IEI), and a member of many professional bodies such as IEEE, IETI (USA), and IET, etc. He has filed sixteen patents, out of which fifteen are published. His current research interests include Digital Signal Processing, Image Processing, VLSI Design, Antenna Design, and Soft Computing. He can be contacted at e-mail: rsharma3@gwa.amity.edu.

Neeraj Shukla received his PhD in Computer Science and Engineering from MANIT Bhopal in 2014. He has over two decades of academic and research experience, having served as Professor at Gyan Ganga Institute of Technology and Sciences, Jabalpur, and previously at Gyan Ganga College of Technology and Shri Ram Institute of Technology. His research interests include Image Processing, Internet of Things (IoT), Machine Learning, Network Security, Wireless Sensor Networks, and Big Data applications.