ANALYSIS OF DIFFERENT CIPHER SUITES ON THE PERFORMANCE OF TLS 1.3 WITH WIRELESS SENSOR NETWORKS

A Project Report submitted in partial fulfilment
Of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY
In
Electronics and Instrumentation Engineering

Submitted by

Aditya Tiwari
190932138

Under the guidance of

Dr.Yashwanth N
(External) Assistant Professor -Senior Scale
Department of Electronics & Communication Engineering,
MIT, Manipal MAHE, Manipal

&

Dr.Chinmay Rajhans
(Internal) Assistant Professor
Department of Instrumentation Control Engineering, MIT,
Manipal MAHE, Manipal

DEPARTMENT OF INSTRUMENTATION AND CONTROL ENGINEERING

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

NOVEMBER 2023

DEPARTMENT OF INSTRUMENTATION AND CONTROL ENGINEERING

MANIPAL INSTITUTE OF TECHNOLOGY

(A Constituent Institution of Manipal Academy of Higher Education)

MANIPAL – 576 104 (KARNATAKA), INDIA

Manipal

24/11/23

CERTIFICATE

This is to certify that the project titled **Analysis of different cipher suites on the performance of TLS 1.3 with WSNs** is a record of the bonafide work done by **Aditya Tiwari** (*Reg. No. 190932138*) submitted in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology (BTech) in **ELECTRONICS AND INSTRUMENTATION ENGINEERING** of Manipal Institute of Technology, Manipal, Karnataka, (A Constituent Institution of Manipal Academy of Higher Education), during the academic year 2020-2021.

Dr.Chinmay Rajhans
(Internal)
Assistant Professor
Department of Instrumentation &amp; Control Engineering, MIT, Manipal MAHE, Manipal

Prof. Dr. Shreesha C
Head of the Department
Dept. of Instrumentation and Control Engineering
MIT Manipal

&

Dr.Yashwanth N
(External)
Assistant Professor-Senior Scale
Department of Electronics & Communication Engineering,
MIT, Manipal MAHE, Manipal

ACKNOWLEDGMENT

First I would like to acknowledge Dr.Anil Rana(Director,MIT Manipal) for allowing me to go ahead with this project . Additionally would like to acknowledge the support of Dr.Yashwanth N(External Guide, Assistant Professor-Senior Scale, ECE Dept) & Dr.Chinmay Rajahns(Internal Guide, Assistant Professor, ICE Dept) for their help & support throughout the entirety of this project, without which this wouldn’t have come to fruition. I would also like to thanks for the support provided by Dr.Shreesha Chakkodi(HOD ICE Dept) for allowing me to do this project with ECE Dept. Also acknowledging the fact that without the support of lab staff of ECE dept especially of Microprocessor Lab this wouldn’t have been possible.

ABSTRACT

As Internet of Things technology expands to cover various facets of everyday life & research environments their security has never been more important. As IoT devices continue to expand exponentially attacks on them has been increasing at an ever-increasing pace. Current algorithms are not well suited to operate in such environments as they were never designed as such. But new standards & algorithms takes years to design & test, so many workarounds are used today. But these workarounds have limitations. In lieu of a NIST lightweight cryptography standard, industries & firms had to take matters in their own hands.

As such to test the efficacy of TLS 1.3 cipher suites if they are suitable to operate in a resource constrained environment, this work attempts to analyse the performance of TLS 1.3 cipher suites how they hold up currently against deprecated cipher suites & how they hold up against newer crypto algorithms such as the ASCON family, the winner of NIST's Lightweight Cryptography Standardization Project. As the current NIST guidelines don't cover for a resource constrained environments that WSNs operate in the approach is to test current TLS 1.3 cipher suites in these environments & measure their basic parameters & using those parameters as a baseline. That baseline will then be used to compare the performance of deprecated cipher suites &, if possible, with ASCON family to gauge how they perform in the same exact environment. Using this we can arrive at what factors need to be tweaked so that these cipher suites perform better in the constrained environment of the WSNs.

LIST OF FIGURES

LIST OF TABLES

Contents

CHAPTER 1
INTRODUCTION

The Internet of Things (IoT) is a community of uniquely diagnosed gadgets geared up with embedded software program for communique of temporary states and statistics, normally used to spark off actuators. Edge networking gadgets and protocols facilitate communique with a cloud server, wherein huge statistics from various gadgets is processed, aggregated, and analysed to tell enterprise decisions. IoT has revolutionized industries, agriculture, healthcare, and clever cities, gambling a pivotal function in statistics series and dissemination. Ensuring the safety of all entities inside an IoT community is of maximum significance because of the pervasive nature of statistics exchange. While modern IoT protocols depend upon IP protocols because the backbone, they may be mainly designed to function throughout a couple of layers, imparting safety at numerous degrees. This bankruptcy specializes in IoT protocols that cope with the safety demanding situations confronted in IoT networks. A key task lies withinside the loss of standardization all through manufacturing, which exposes hardware, software program, and statistics to numerous threats and assaults. IoT protocols need to address safety breaches on the cloud provider provider's web website online and cope with issues associated with statistics privateness, authentication, authorization, and believe control in disbursed heterogeneous surroundings[1].

A wi-fi sensor community (WSN) is a laptop community comprising a couple of sensor nodes disbursed throughout numerous locations. These sensor nodes are designed to display environmental activities which include temperature changes, sound, vibration, and extra. WSNs face giant demanding situations, such as node replication, node failure, packet loss, and the capacity for packet alteration through malicious attackers. To beautify statistics switch quotes and gain excessive throughput degrees in WSNs, numerous techniques had been advanced to save you or mitigate such assaults. However, only some of those techniques efficaciously examine the severity of community compromises with accuracy and efficiency. The adoption of wi-fi sensor networks has been pushed through various programs, starting from army surveillance and domestic hearthplace alarms to civilian makes use of which include environmental tracking, habitat observation, scientific programs, domestic automation, and site visitors control. Sensors deployed withinside the surroundings acquire statistics and integrate statistics from a couple of sources. Despite the convenience of deployment, flexibility, and cost-effectiveness of sensor networks, protective them from attackers who can inject fake measurements poses a giant task. For instance, an attacker ought to provoke a hearthplace to cause a fake alarm in a domestic hearthplace alarm system. Such malicious moves that manage or update measurements are referred to as malicious statistics injection. Event detection performs a essential function in tracking and figuring out incidents or activities going on withinside the surroundings[2].

WSNs and the speedy increase of IoT generation create capacity vulnerabilities in utility domain names wherein confidentiality, integrity, and availability (CIA) are critical. The integration and collaboration of WSNs with IoT introduce extra safety demanding situations and issues. Many WSN deployments in IoT arise in unattended antagonistic environments for amassing touchy statistics, making statistics leakage and alteration a giant difficulty for privateness and safety. Unauthorized customers might also additionally benefit get admission to the community, compromising the safety and confidentiality of the statistics. Moreover, useful resource confined IoT gadgets, which include sensor nodes, face barriers in strength consumption, reminiscence footprint, computing abilities, and speed. These barriers make it difficult to put into effect whole cryptographic algorithms, leaving WSNs at risk of numerous assaults and placing statistics safety at risk. Sacrificing the safety and privateness of consumer statistics to unauthorized customers isn't always an option, emphasizing the significance of enforcing strong safety measures.

To cope with safety troubles in WSNs, it's far frequently encouraged to rent light-weight safety schemes that limit technical overhead without affecting universal community performance. As the extent of touchy statistics will increase and will become extra prone to manipulation and transmission, the implementation of cryptographic safety will become essential. However, there's constrained statistics and definitive answers concerning light-weight cryptography. Nonetheless, light-weight cryptography may be carried out in software program and hardware, mainly tailor-made for useful resource-confined gadgets, to optimize useful resource utilization, computational time, strength consumption, and safety.

In addition to addressing safety issues all through statistics transmission, it's far essential to recollect statistics confidentiality, integrity, and authenticity whilst storing big quantities of statistics in disbursed environments. This bankruptcy explores the 3 pillars of safety supplied through wi-fi sensor networks withinside the context of statistics storage. Furthermore, it's far vital to observe that a giant part of a sensor's power in a wi-fi community is allotted to statistics processing in preference to statistics storage. Wireless sensor networks include battery-powered nodes geared up with statistics processing and radio communique components. Dynamic wi-fi sensor networks, not like static networks, facilitate extra particular management and cognizance through permitting the mobility of sensor nodes. Consequently, dynamic WSNs are swiftly followed in programs which include method tracking, mobility tracking, vehicle reputation checks, and fitness verification of livestock. Privacy stays a essential task in dynamic WSN programs, necessitating important safety requirements, such as hub authenticity, statistics-green key control, and anonymity and integrity of statistics all through node movements[3].

Previous works don’t point out the way to use TLS 1.3 cipher suites with confined environments like WSN networks. As TLS 1.3 is an entire new deployment of the TLS protocol, it brings new competencies to the leading edge which also can be deployed in confined surroundings. Many of the cipher suites have turn out to be deprecated which had been supported through TLS 1.3 and not offer the extent of safety they should. Cipher suites supported through TLS 1.3 aren't simplest extra green however additionally offer strong safety.

By checking out the numerous TLS 1.3 cipher suites which might be supported, deploying them in confined surroundings and the usage of that as a baseline after which deploying deprecated cipher suites of preceding generations of TLS, insights may be received on what makes the modern TLS era cipher suites strong and what similarly may be executed to make their deployment extra green. As a NIST cybersecurity framework doesn't exist for useful resource confined surroundings, so we cannot examine and discover wherein the lapse in information lies, this check throws mild directly to make those cipher suites deployment extra strong.

Objectives are as follows:

- Testing performance of supported TLS 1.3 cipher suites & using it as a baseline
- Testing performance of deprecated cipher suites no longer supported & compare them against the first test

Target specifications are as follows:

- Tools used
    1. NS3(simulation)
    2. Testssl.sh(data collection)
- Platform used:
    1. Linux

Project work schedule is as follows:

- April 2023: Literature Review
- May 2023: Formulation of Objectives
- June 2023: Implementation & Data Collection
- July 2023: Comparison of Data with Baseline
- August 2023: Result Analysis

Organization of the project report is as follows:

- Introduction
- Background Theory
- Methodology
- Result Analysis

CHAPTER 2
BACKGROUND THEORY

Wireless sensor networks are the number one recognition of assaults withinside the IoT domain. When a node withinside the belief layer is subjected to an attack, it can result in unfavourable behaviours together with unauthorized amendment of node records or not on time forwarding withinside the IoT device. In a few cases, a unmarried compromised node can hastily infect the complete IoT device, inflicting device paralysis. The contemporary malicious node detection device faces numerous challenges:

- Due to the presence of several micro-sensors withinside the wi-fi sensor network, detecting malicious nodes regularly needs big electricity resources.
- Certain malicious nodes interact in sports together with tampering with records packets or changing packet numbers, which complicates the identity of such nodes.
- Existing malicious node detection structures be afflicted by sluggish detection pace and restrained accuracy in figuring out malicious nodes[4].

The sensor layout poses 3 noteworthy challenges, which includes the hard venture of changing or recharging middleman batteries because of their heavy workload and the tricky nature of Wireless Sensor Networks (WSNs). It may be argued that the maximum pressing requirement at threshold is to increase the lifespan of sensing devices, which necessitates moves like prolonging the system's typical period via way of means of minimizing the criticality of focal factor utilization in WSNs. Empirical findings exhibit that records trade is successfully done via real consumption (EC), whilst unexpectedly, aspect records training famous low centrality. To deal with this, a tiered-enterprise technique became proposed to increase the durability of WSNs and decrease the reliance on massive sensors for records trade. Additionally, records safety poses a venture whilst transmitting records from the supply to the goal inside a WSN[5].

Elliptic curve cryptography (ECC) is a extensively diagnosed light-weight and strong cryptographic method significantly deployed in IoT networks because of its minimum reminiscence consumption. ECC operates inside a Gaussian discipline G and capabilities as a one-manner feature considering that computing its inverse is computationally infeasible. An elliptic curve is described over a discipline Fp (a, b)|p > 3, in which p is a top number. The curve is represented through a fixed of factors p(x, y) such that (x, yEFp(a, b)) satisfies the equation y^2 = x^3ax bmod(p) for 0 <= x <= p. The protection power of ECC relies upon at the computation of scalar or fixed-factor multiplication, with the scale of the curve governing the related computational costs. The ECC-primarily based totally cryptography adheres to the PKC requirements mentioned in ANSI X9.63 and IEEE P1363, which offer more suitable randomness and per-bit protection as compared to current PKC strategies like RSA and AES.

ECC reveals software in diverse IoT environments characterised through aid constraints and the deployment of self-sufficient devices.

The essential mathematics operations implemented in ECC are factor addition and factor doubling at some stage in multiplication. Point addition refers to the technique of including factors at the curve. Many strategies deploy the double-and-upload set of rules or its variations to generate a random sample of factors at the curve. Additionally, as shown in Figure 2.1, ECC encompasses diverse flavours together with Elliptic Curve Diffie-Hellman (ECDH), Elliptic Curve Diffie-Hellman (ECDDH), Elliptic Curve Discrete Logarithm Problem (ECDLP), Elliptic Curve Integrated Encryption Scheme (ECIES), Elliptic Curve Digital Signature Algorithm (ECDSA), Edwards curve Digital Signature Algorithm (EdDSA), and Elliptic Curve Menezes-Qu-Vanstone (ECMQV). Likewise, a numerous variety of curves is applied to cater to awesome protection necessities and curve performance. Notable examples consist of M-221, E-222, NISTP-224, Curve1174, Curve25519, BN (2,254), brainpoolP256t1, ANSSI FRP256v1, NIST P-256, secp256k1, E-382, M-383, Curve383187, brainpoolP38t1, NIST P-38, Curve 117, Ed 8-Goldilocks, M-511, and E-521. ECC is extensively embraced in several light-weight schemes because of its promise to generate strong keys with minimum size. In assessment to conventional encryption strategies like RSA, AES, and ElGamal, which make use of heavy-sized non-public or public keys, ECC-primarily based totally key technology is cost-powerful and gives the identical stage of key level required for extensive encryption and decryption processes[6].

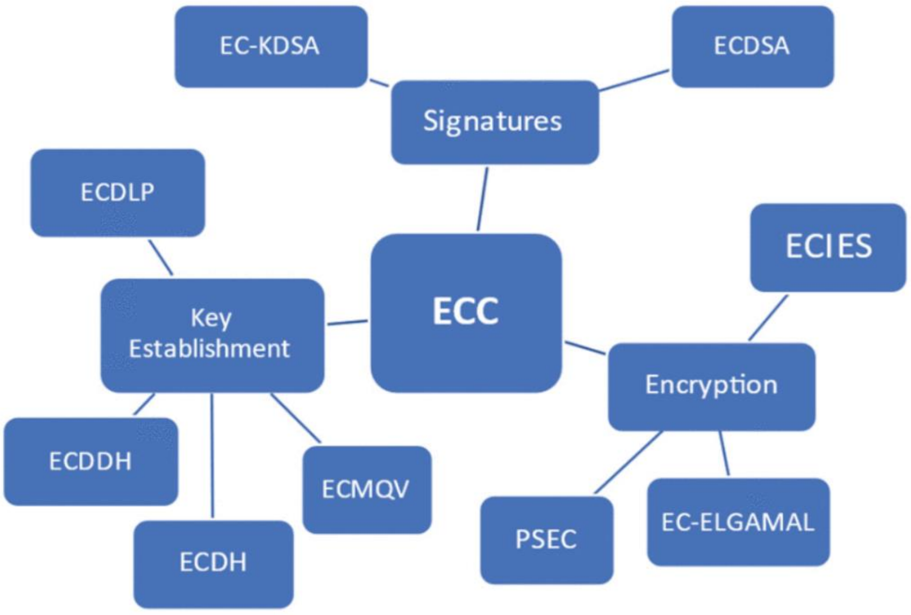


Figure 2.1 : LEC Cryptography Curve Classification[6]

As depicted in Figure 2.2, encryption algorithms can be divided into two most important categories: symmetric key encryption and asymmetric key encryption. Symmetric key cryptography makes use of a single key to encrypt and decrypt data, providing the blessings of safety and quite environment friendly processing. Its primary undertaking is the impervious sharing of the key between speaking parties. However, this predicament can be overcome by way of distributing the key in develop thru a depended-on intermediary. Symmetric key encryption technological know-how ensures confidentiality, statistics integrity, and authentication the usage of mainly authenticated encryption modes (AEAD). In contrast, uneven cryptography makes use of two separate key pairs: a personal key and an equal public key. It affords confidentiality and integrity by means of the use of the recipient's public key for encryption whilst authenticating the sender's non-public key, which is generally used to encrypt records as a digital signature. After receiving the data, the receiver decrypts the information with the sender's public key and then with its very own non-public key. The drawback of asymmetric encryption is its large key size, which will increase complexity and slows down processing. In the area of block ciphers, each encryption and decryption are carried out on blocks of constant measurement (typically sixty four bits or more), whilst stream ciphers manner enter factors one bit at a time. The simple cryptographic houses of obfuscation and diffusion delivered by way of Claude Shannon enhance cryptographic strength. Obfuscation includes maximizing the complexity of the affiliation between cipher and key, regularly the use of substitution (as considered in S-boxes). Diffusion, on the other hand, decomposes the statistical shape of the plaintext into a large section of the ciphertext by means of permutation. While stream ciphers in most cases use obfuscation, block ciphers include each obfuscation and confusion, which are normally less difficult in shape than their circulation cipher counterparts. Reversing the encryption system to decrypt the authentic textual content is especially challenging for block ciphers, whilst move ciphers use an XOR operation for encryption, which can be greater without problems restored to its unique form. In contrast, a hash feature works as a one-way mathematical transformation that converts statistics of unspecified size into a bit string (short string) of constant length, making the inversion computationally infeasible. Given these considerations, block cipher is desired overflow cipher for resource-constrained Internet of Things (IoT) devices. Block ciphers, adopts one of the following structural paradigms:

- Substitution-Permutation Network (SPN)
- Feistel Network (FN)
- General Feistel Network (GFN)
- Add-Rotate-XOR (ARX)
- Non-linear Feedback Shift Register (NLFSR)
- Hybrid

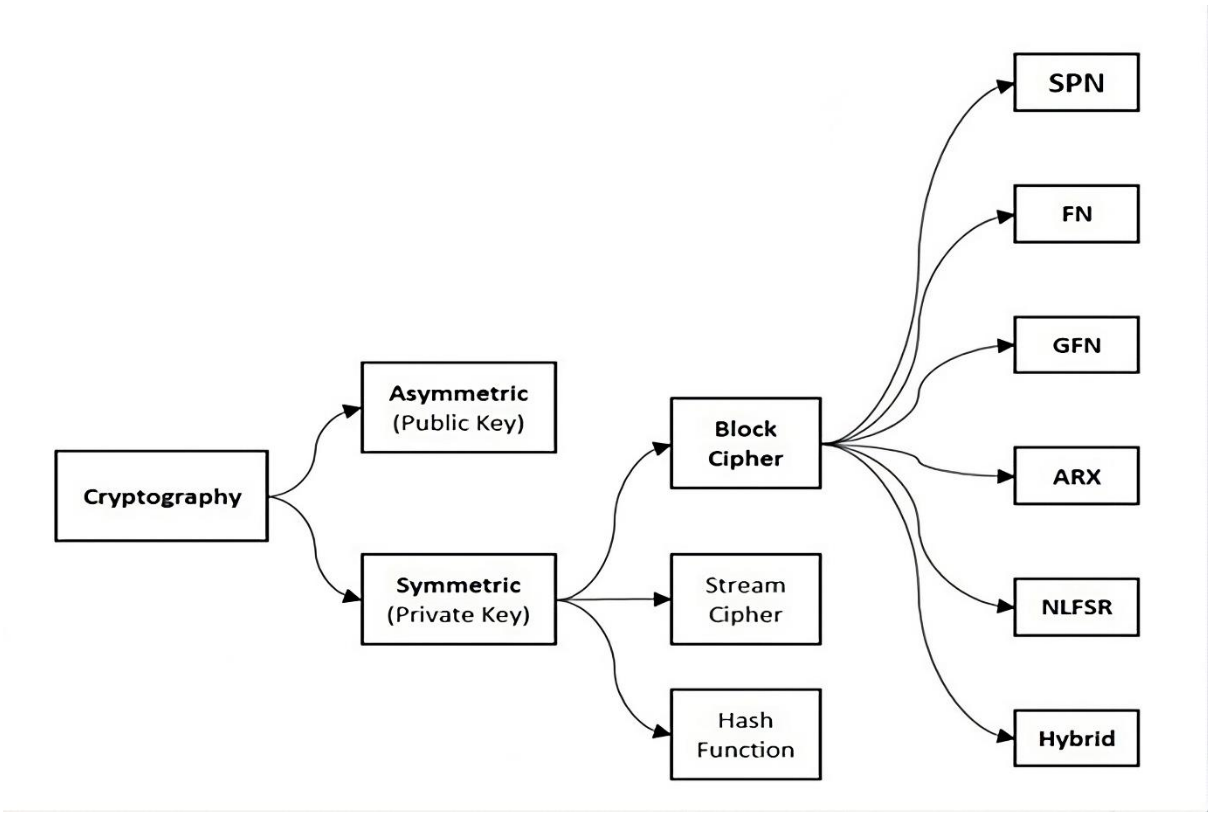


Figure 2.2 : Lightweight Cryptography Type Classification[7]

Block cipher constructions utilize various techniques to iteratively diffuse and confuse input data, with the predominant archetypes being substitution-permutation networks (SPNs), Feistel networks (FNs), and generalized Feistel networks (GFNs). In SPNs, substitutions via S-boxes and permutations through fixed tables alter data between rounds. FNs partition input into equal halves, diffusing one half per round and swapping the halves. GFNs generalize Feistel networks by dividing input into multiple sub-blocks, applying Feistel functions to sub-block pairs, and cyclic shifting the sub-blocks. While SPNs directly operate on the full input, FNs and GFNs provide diffusion through interleaved processing of input subsets. The selection of an optimal structure is dependent on the target platform and desired cryptographic properties. When judiciously constructed, these iterative blocked ciphers can effectively resist cryptanalysis.

The ARX construction encompasses encryption-decryption operations utilizing addition, rotation, and XOR functions, notably devoid of S-boxes. While the implementation of ARX is characterized by speed and compactness, it inherently possesses limitations in security properties when compared to SPN and Feistel ciphers[7].

The Nonlinear Feedback Shift Register (NLFSR) constitutes a fundamental component in both stream and block ciphers, harnessing a state that derives from a nonlinear feedback value associated with its prior state. Hybrid ciphers combine elements from any three of the types

(SPN, FN, GFN, ARX, NLFSR), or even amalgamate block and stream characteristics to enhance specific attributes, such as throughput, energy efficiency, or Generalized Entropy (GE), aligning with distinct application requisites. Among these structures, SPN and FN emerge as the most popular selections, predominantly due to their adaptability in implementation based on specific application demands. While Feistel structures are easily integrated into low-to-moderate power hardware, they often necessitate more rounds for function compared to SPN structures, a measure taken for safety considerations. When confronted with a decision between a reduced number of SPN function rounds and a higher count of Feistel function rounds, provided an equivalent level of security and comparable energy costs, opting for SPN functions may represent a more judicious choice[7].

Lightweight cryptography refers to the model of primitive cryptographic strategies with the goal of accomplishing required level safety whilst optimizing the required utilization and computational efficiency. The improvement of light-weight cryptographic strategies takes into consideration the hardware demands, computational complexity, reminiscence constraints, and barriers of IoT gadgets and Machine-to-Machine (M2M) networks. Lightweight cryptography is often followed in situations in which WSN nodes have low computational abilities and constrained inner memory capacity, as conventional encryption primitives aren't sufficient in such contexts. However, light-weight cryptography faces numerous challenges. To make sure adequate encryption in M2M networks, diverse authentication and information integrity techniques had been proposed, depending closely at the effectiveness of light-weight cryptography in securing IoT implementation models. Lightweight cryptography should own safety functions to offer sturdy safety to resource-restrained WSN nodes in M2M networks. Since M2M networks predominantly is based at the perception layer, safety issues should deal with essential functions which include information integrity, information confidentiality, node authentication, and information availability. Furthermore, superior self-reliant nodal devices like Machine-Type Communication (MTC), Cyber-Physical Systems (CPS), and Industrial Internet of Things (IIoT) function in large and greater complicated networks which include fitness tracking systems, self-reliant business workshops, and sensitive information sensing and tracking systems. In those situations, light-weight cryptography should showcase resilience towards current safety deployments together with Man-in-the-Middle (MITM), information spoofing, Man-at-the-End (MATE), forgery, tool replication, Denial-of-Service (DoS), and physical node manipulation. Moreover, light-weight cryptography should deal with vital safety functions which include collusion-resistance, making sure that an attacker with a couple of keys can't get right of entry to records if any unmarried key presents network nodal access. Forward secrecy ensures that encrypted keys continue to be uncompromised despite the addition of greater nodes and users, whilst backward secrecy guarantees that an adversary with information of a subset of keys can't find out preceding keys no matter the addition or elimination of nodes or computational network complexity. In the context of cloud, gateway, and principal nodal devices, firewall safety is likewise vital to disclaim undesirable get right of entry to requests. Furthermore, the cryptographic method deployed ought to allow stable network transit whilst minimizing energy consumption, computational overhead, and transmission costs[8].

Cryptography encryption strategies are usually taken into consideration for securing the network communication among IoT devices. However, because of the restrained computational ability and major memory limitations of IoT devices, traditional encryption strategies which includes AES or RSA won't be feasible. This problem is obvious in numerous IoT deployment situations, which includes:

Implantable Medical Devices (IMDs): IMDs with wi-fi connectivity for far flung affected person tracking require safety towards unauthorized system access and information transmission. However, using traditional cryptography might also additionally notably reduce the battery voltage supply of IMDs because of their restrained voltage supply.

Energy Harvesting Devices: Energy harvesting devices, regularly prepared with lower capacity computational chips, or included RFID tags, can't assist computationally extensive encryption schemes. To cope with the major resource constraints of IoT devices, more than one light-weight encryption schemes were proposed. These schemes deploy encryption via numerous means, which includes smaller key sizes, simplified key schedules, and using simple operations. However, a number of those design arrests their deployment situations with the aid of using on custom designed hardware or specialised software program.

As illustrated in figure 2.3, cipher overall performance evaluation includes a complete evaluation of various quintessential parameters, every taking part in a pivotal position in identifying the typical effectiveness and suitability of a cryptographic algorithm for precise software scenarios. One such parameter is the key length, which without delay affects the protection robustness of the cipher. Longer key lengths usually enhance the resistance to brute-force attacks, rendering them computationally infeasible. However, it is integral to strike a balance, as excessively lengthy key lengths can lead to greater computational overheads, impacting standard performance. Careful consideration of the key size is fundamental to meet the favoured safety stage whilst retaining an desirable overall performance trade-off.

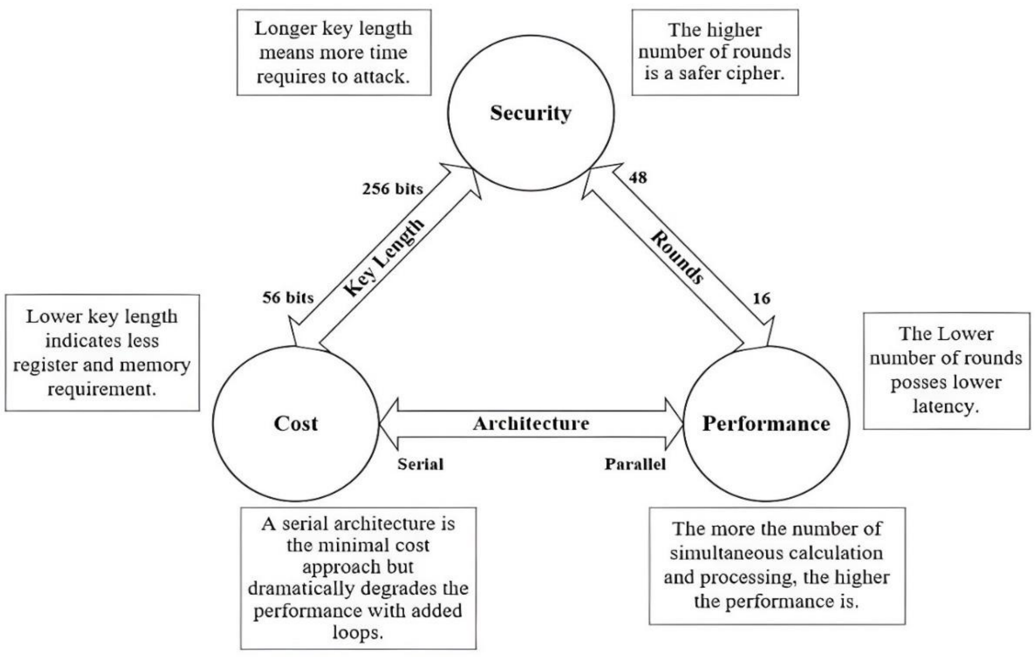

Longer key length means more time requires to attack.
Security
The higher number of rounds is a safer cipher.
256 bits
Key Length
56 bits
48
Rounds
16
Lower key length indicates less register and memory requirement.
Cost
Architecture
Performance
The Lower number of rounds posses lower latency.
Serial
Parallel
A serial architecture is the minimal cost approach but dramatically degrades the performance with added loops.
The more the number of simultaneous calculation and processing, the higher the performance is.

Figure 2.3 : Trade-off Triangle[8]

The variety of rounds in a cipher is every other critical thing influencing its performance. Additional rounds can bolster the safety margin by using introducing multiplied complexity and diffusion. Nevertheless, this can end result in extended computational requirements, probably impacting real-time purposes or resource-constrained environments. Achieving the optimum range of rounds is a imperative format challenge, the place too few rounds might also compromise security, and immoderate rounds may additionally lead to unacceptable overall performance overheads. Striking the proper stability between protection and overall performance necessitates thorough comparison and consideration of the unique use case and the preferred degree of protection.

Moreover, the chosen cipher structure performs a full-size position in dictating its overall performance characteristics. Different architectural designs, such as Feistel networks, Substitution-Permutation networks (SPNs), or Advanced Encryption Standard (AES) block ciphers, showcase various ranges of computational effectivity and ease of hardware implementation. The choice of an structure need to align with the meant application's overall performance necessities and the handy hardware resources.

While evaluating cipher performance, the factor of protection is of paramount importance. A cipher's robustness towards quite a number kinds of attacks, consisting of differential and linear cryptanalysis, determines its suitability for securing touchy data. Higher safety tiers regularly come at the price of improved computational complexity, which can affect standard performance. Striking the proper stability between protection and overall performance necessitates cautious consideration of the conceivable threats and the required degree of protection.

The cost element is critical in cipher evaluation, as it consists of no longer solely the computational overhead however additionally concerns of hardware implementation, strength consumption, and the doable want for specialised hardware accelerators. Cipher designs that require massive computational assets may additionally no longer be possible for resource-constrained gadgets or purposes with stringent electricity requirements. Assessing the cost implications throughout a number of dimensions is essential to make sure the chosen cipher aligns with the budgetary constraints of the implementation goal.

In summary, cipher overall performance assessment includes a subtle trade-off between key length, rounds, architecture, security, cost, and ordinary computational efficiency. Selecting an excellent stability amongst these parameters is necessary to format and install ciphers that meet the favoured protection desires whilst making sure compatibility with the unique application's overall performance and limited resource constraints[8].

Lightweight cryptography encryption algorithms were evolved particularly to offer safety for low resource IoT devices. These designs prioritize power limitations with the aid of using adopting smaller key sizes, counting on simple operations like XOR and AND, or using simplified key schedules. For instance, the PRESENT algorithm, present here is a light-weight block cipher primarily based totally on a easy Substitution-Permutation Network (SPN) with key sizes of eighty or 128 bits. The authors suggest KATAN, a hardware-orientated block cipher with a Feistel shape structure and a simplified key scheduling mechanism. Another instance is TWINE, presented here which makes use of a Type-2 Generalized Feistel Structure (GFS) to reap hardware performance at the same time with trade-off of minimizing hardware-orientated layout complexities.

Additionally, unique designs leverage custom designed hardware or specialised software program to facilitate their encryption schemes, as established in. However, those tactics might also additionally impose obstacles on their deployment situations. One fantastic encryption scheme, DyWCP, attracts idea from the one-time pad encryption method, acknowledged for its excessive degree of secrecy. DyWCP may be universally carried out to IoT programs with wi-fi connectivity, capitalizing at the wi-fi physical layer independently from the appliciation layer. Therefore, DyWCP enhances current light-weight cryptography algorithms and can be deployed alongside them to increase the general encryption grade of IoT devices[9].

The SSL/TLS protocol is broadly followed as a standard safety mechanism, and cipher suites play an imperative function in its implementation. Cipher suites embody four predominant components: key exchanging algorithms, authentication algorithms, encryption algorithms, and message authentication code (MAC) algorithms. Key exchanging algorithms, along with RSA, DH, ECDH, and ECDHE, facilitate steady key alternate transit among the sender and recipient (e.g., Client and Server). Authentication algorithms together with RSA, DSA, and ECDSA are deployed to confirm the authenticity of the sender and receiver. Encryption algorithms, together with AES and DES, are used to encrypt to be transmitted data among internet browsers and servers. The message authentication code algorithm, making use of MD5, SHA1, SHA256, SHA38 , and POLY1305, guarantees integrity constraints on each side.

SSL 3.0 delivered cipher suites like SSL_DH_RSA_EXPORT_WITH_DES40_CBC_SHA, SSL_DH_RSA_WITH_DES_CBC_SHA, and SSL_DHE_DSS_WITH_DES_CBC_SHA. Cipher Block Chaining (CBC) is the usually deployed method in those suites, no matter its vulnerability to attacks like POODLE. The weak point lies withinside the encrypted initialization vector (IV) used withinside the CBC chaining phase, applied with the DES algorithm, imparting an attacker with 256 combinations. TLS 1.0, an development over SSL 3.0, resolves the problems via way of means of changing the encryption approach for the initialization vector, making use of AES in place of DES. However, TLS 1.0 remains vulnerable to attacks like POODLE and BEAST. TLS 1.1 addresses those vulnerabilities via way of means of changing the implicit IV with an express IV, efficiently protecting in opposition to all CBC attacks. TLS 1.2, broadly followed in contemporary-day internet browsers and websites, employs elliptic curve cryptographic strategies for key exchanging and user, browser & server authentication. Integrity verification of messages is superior via way of means of changing MD5/SHA1 with SHA256, SHA38, and HMAC SHA256. AES_GCM and AES_CCM function the encryption algorithms in TLS 1.2, with potent cipher suites being TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 and TLS_ECHDE_RSA_WITH_ AES_256_GCM_SHA384.

Figure 2.4 shows Transport Layer Security (TLS) protocol has evolved through successive versions to optimize security, performance, and cryptographic agility, as depicted in the architectural overview. TLS originated from SSL 3.0 per RFC 2246, relying largely on RSA cryptosystems. TLS 1.0 represented an incremental advancement, still hindered by inherited vulnerabilities. TLS 1.1, specified in RFC 4346, enabled forward secrecy via ephemeral keying and bolstered confidentiality protections, while TLS 1.2 described in RFC 5246 incorporated authenticated encryption modes like GCM and CCM. The current TLS 1.3 standard defined in RFC 8446 emphasizes optimal security-latency trade-offs, leveraging symmetric cryptography and streamlined handshakes. Underlying each version is a robust handshake protocol, validating identities and initialization parameters, integrated with persistent storage protocols enabling session resumption. For HTTPS implementations, TLS operates in a layered model, with the handshake and record protocols providing key negotiation and transport encryption sandwiched between application and TCP layers. Through diligent cryptographic enhancements, the TLS ecosystem has continually adapted to emerging threats and use cases, cementing TLS as the de facto secure communications standard.

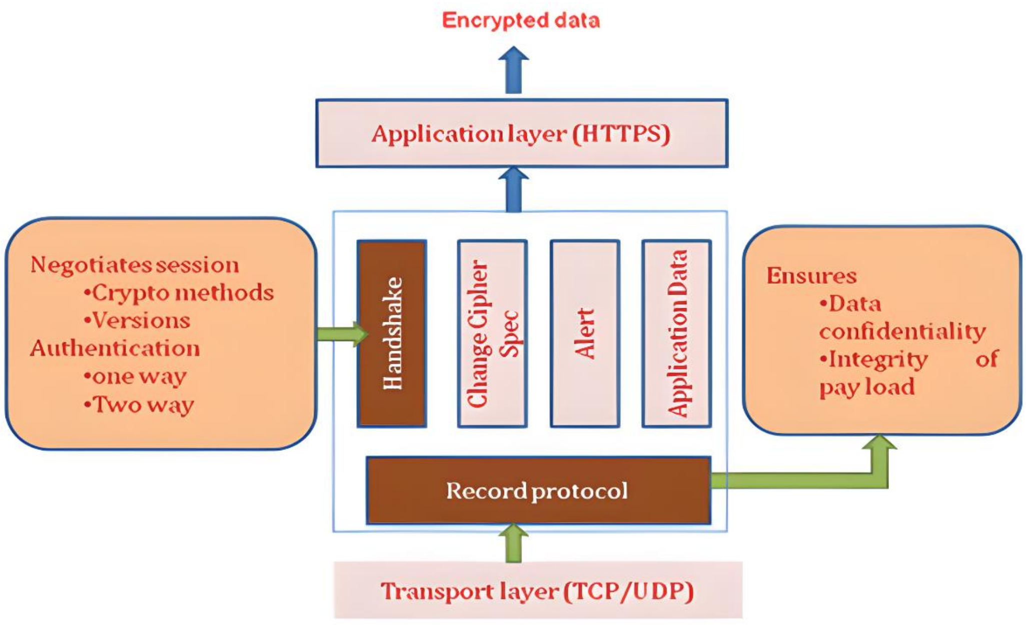
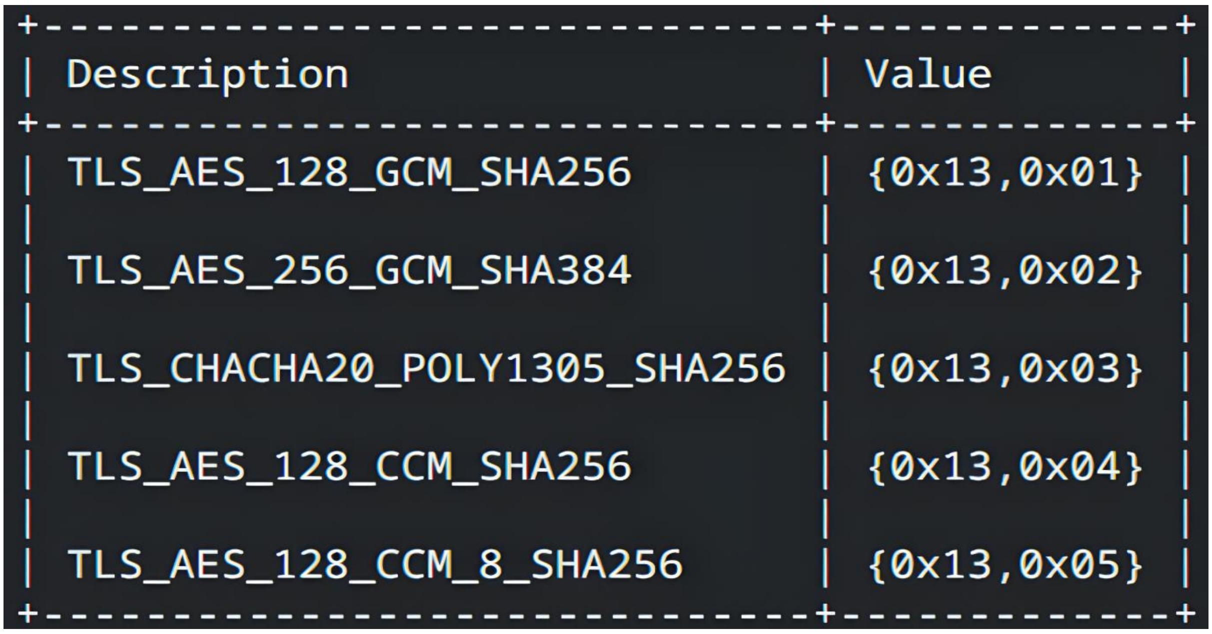


Figure 2.4 : TLS Cipher Suite Architecture[10]

TLS 1.3 remains withinside the draft level and gives improvements together with the choice of steady elliptic curves and secure primes. Older authentication schemes like MD5 and SHA22, in addition to key exchanging algorithms together with DH, ECDH, and ECDHE, had been removed. TLS 1.3 introduces the CHACHA20 stream cipher with POLY1305 as a unique authentication encryption method. For authentication purposes, the Edwards curve virtual signature algorithm (EdDSA) Ed25519 and Ed8 are deployed, even as Curve X25519 and Curve X448 function as key exchanging mechanisms. Additionally, TLS 1.3 reduces the Round Trip Time (RTT) 1 RTT for handshake negotiation among the internet server and browser, in comparison to two RTT in TLS 1.2. TLS 1.3 additionally introduces a zero RTT put off delay for next visits to the identical website. By using the AEAD system, which integrates key exchanging, authentication, and message authentication right into a unmarried handshake, the computation time period required inside the handshake system is reduced[11].

Analysis of Cipher Suites

Figure 2.5 shows Common TLS cipher suites like TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 and TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 combine key exchange, authentication, encryption, and message authentication algorithms. Early TLS relied heavily on RSA for ephemeral session key exchange between browsers and servers. RSA's modular exponential approach resists efficient factoring, especially for large prime numbers. To ensure sufficient security, a 2048-bit minimum key size was adopted, though larger 3072-bit keys better thwart cryptanalysis given sufficient computing power. However, extended RSA key sizes impose nontrivial memory burdens on constrained devices. Among potential RSA attacks, timing side-channel attacks pose considerable risks by exploiting cryptographic timing data. Hardening against timing attacks is critical for sustaining long-term RSA security. TLS has evolved its cipher suites to incorporate perfect forward secrecy through ephemeral key exchange using elliptic curve techniques like ECDHE. The migration to ECDHE mitigates multiple RSA limitations, providing efficient authenticated key establishment resilient to quantum computing advances. Carefully constructed cipher suites are essential for robust, modern TLS communications[11].



```
+--------------------------------+-------------+
| Description                    | Value       |
+--------------------------------+-------------+
| TLS_AES_128_GCM_SHA256         | {0x13,0x01} |
|                                |             |
| TLS_AES_256_GCM_SHA384         | {0x13,0x02} |
|                                |             |
| TLS_CHACHA20_POLY1305_SHA256   | {0x13,0x03} |
|                                |             |
| TLS_AES_128_CCM_SHA256         | {0x13,0x04} |
|                                |             |
| TLS_AES_128_CCM_8_SHA256       | {0x13,0x05} |
+--------------------------------+-------------+
```

Figure 2.5 : TLS Cipher Suites[11]

Message authentication is a critical mechanism that permits the recipient of a message to confirm its genesis from the claimed sender. One extensively deployed approach for such verification is the usage of message authentication codes (MACs), which append authentication tags to every message. However, it's far essential to observe that those tags introduce extra overhead to the packets. To meet the minimal safety standard of 128 bits put forward by NIST, a 16-byte tag requires to be added to the message. This poses demanding obstacles in resource-restricted environments, inclusive of commercial control systems, in which messages may be as small as a singular byte. In such cases, a substantial part of the packet is fed on via way of means of the authentication tag. The enlargement of packets in those environments will become complicated because of numerous factors, including (i) very small payloads, (ii) scarce bandwidth, (iii) power limitations, and (iv) reliability requirements[12].

There have been suggestions to apply TLS 1.3 in a resource constrained WSN environment, but they never extended to full cipher suites in the WSN environment. TLS 1.3 is the go to protocol that provides the necessary security cover, applying them in the WSNs will provide the same security cover for this environment as well. Cipher suites that are supported by the TLS 1.3 can fulfil the security requirements of the resource constrained nodes.

Cipher suites play a crucial and multifaceted function in the TLS 1.3 protocol, that's broadly used for encrypting network channels in online transactions, consisting of e-commerce and online banking. A cipher suite is a cautiously curated amalgam of cryptographic algorithms and protocols that together offer the essential mechanisms for setting up stable connections among clients and servers[13].

TLS 1.3 introduces huge improvements inside the blueprint and choice of cipher suites, aiming to fortify the safety and performance of the protocol. These suites embody key exchanging algorithms, authentication algorithms, encryption algorithms, and message authentication code (MAC) algorithms, all of which make contributions to the general security of the network channel[13].

Another of the notable enhancements in TLS 1.3 cipher suites is the adoption of safe elliptic curves and secure prime numbers. Elliptic curve cryptography (ECC) offers enhanced protection with shorter key lengths, making it an interesting choice for resource-constrained environments without compromising the confidentiality and integrity of the network channel. Secure prime numbers increase the probability of principal mathematical operations remaining strong against many cryptographic attacks[14].

In the context of cryptographic key exchange algorithms, TLS 1.3 actively advocates for the utilization of presently robust and dependable mechanisms. A deliberate phase-out strategy has been implemented for antiquated and notably less dependable cryptographic algorithms, among them, the likes of Diffie-Hellman (DH) and Elliptic Curve Diffie-Hellman (ECDH), within the cipher suite portfolio. In lieu of these legacy cryptographic techniques, TLS 1.3 positions itself on the foundation of more intricate cryptographic algorithms, notably exemplified by the Elliptic Curve Diffie-Hellman Ephemeral (ECDHE) method. ECDHE, by design, facilitates the realization of forward secrecy by creating unique session keys for each discrete client-server handshake. The underlying principle of forward secrecy furnishes a robust assurance: even in the face of a hypothetical compromise of long-term private cryptographic keys in the foreseeable future, previously encrypted communications remain invulnerable to prying eyes.

The landscape of authentication algorithms within the ambit of TLS 1.3 cipher suites has also been marked by significant enhancements. Cryptographically weaker algorithms, including the once-in-use MD5 and SHA-22, have been deliberately deprecated in response to heightened security concerns. These deprecated algorithms have made way for more robust counterparts, such as SHA-256 and SHA-384, which have been thoughtfully integrated into the cryptographic framework. Furthermore, the introduction of the Edwards curve digital signature algorithm (EdDSA) represents a pivotal innovation in the realm of authentication. EdDSA stands as an embodiment of streamlined yet stalwart authentication, proffering an array of advantages, encompassing more concise signature dimensions and expedited signature verification processes[14].

Encryption algorithms in TLS 1.3 cipher suites have witnessed enhancements as well. The broadly used Advanced Encryption Standard (AES) in Galois/Counter Mode (GCM) and Cipher Block Chaining (CBC) modes are advocated for steady records transmission. AES-GCM gives authenticated encryption, making sure each confidentiality and integrity of the transmitted records, whilst AES-CBC gives compatibility with older protocols and legacy systems.

In addition, bolster the integrity of the handshake, TLS 1.3 cipher suites include message authentication code (MAC) algorithms. Secure hash functions, together with SHA-256, SHA-38, and HMAC SHA-256, are deployed to generate MAC tags that affirm the authenticity and integrity of the transmitted messages. TLS 1.3 additionally emphasizes performance and decreased latency through optimizing the handshake process. It aims to reduce the Round Trip Time (RTT) required for organising a steady connection. By lowering the quantity of round trips involved, TLS 1.3 achieves quicker and secure handshakes, ensuing in advanced overall performance for on-line transactions.

The choice and configuration of suitable cipher suites in TLS 1.3 need to recollect the particular safety requirements and constraints of the node and environment. It is vital to make certain that the selected cipher suites align with the preferred degree of safety, overall performance considerations, and compatibility with the client and server implementations.

In summary, Transport Layer Security (TLS) version 1.3 cipher suites represent carefully constructed amalgamations of cryptographic primitives and protocols providing the foundation for robust security in online communications. Enhancements across key exchange, authentication, encryption, and message authentication algorithms, alongside an emphasis on minimizing latency, bolster the overall security and performance of TLS 1.3 cipher suites. The continual evolution of TLS cipher suites through progressive cryptographic upgrades plays a pivotal role in sustaining confidentiality and integrity protections for internet-based communications amidst an ever-changing threat landscape. Diligent cipher suite improvements ensure TLS offers a resilient trust anchor for the modern digital ecosystem across a myriad of applications and use cases[14].

CHAPTER 3
METHODOLOGY

TLS 1.3 cipher suites may be examined inside specialised simulators like NS3 and the use of specialised scripts inclusive of Testssl.sh for you to capture cryptographic parameters and network markers as well. Using those guidelines conclusions may be drawn to set up a baseline.

Testing methodology is as follows:

Cipher Suites the use of NS3 and Testssl.sh: Test Environment Setup:
a. Install NS3: Set up the NS3 community simulation framework on a Linux Machine Follow the set-up commands furnished through the NS3 documentation.

b. Install Testssl.sh: Download and configure the Testssl.sh tool, that's designed for trying out SSL/TLS encryption protocols. Ensure you've got the current version installed.

Test Scenario Definition:

a. Define the precise cipher suites to be examined: Determine the structure of cipher suites supported through TLS 1.3 which you need to evaluate. Select more than a few suites primarily based totally on protection, compatibility, and overall performance requirements.

Network Simulation Setup:

a. Build the network topology: Create a simulated network landscape in NS3 that replicates the meant configuration for trying out TLS 1.3 cipher suites.

b. Configure nodes and network scenario parameters: Set up the client nodes and server nodes with suitable IP addresses, port numbers, and conversation protocols required for TLS 1.3.

Test Execution:

a. Establish TLS 1.3 connections: Implement the vital code in NS3 to provoke TLS 1.3 connections among the patron and server nodes.

b. Enable logging: Enable logging mechanisms inside NS3 to seize parameter data at some point of the simulation process, together with handshake information, cipher suite negotiation, and any mistakes or warnings.

c. Monitor network's overall performance: Utilize NS3's tracking abilities to evaluate the network overall performance metrics inclusive of latency, throughput, and packet loss at some point of TLS 1.3 connections.

Analysing Test Results:

a. Extract applicable information: Extract the captured logs and overall performance metrics from NS3 for evaluation.

b. Evaluate cipher suite compatibility: Use Testssl.sh to carry out extra validation and evaluation at the supported cipher suites withinside the TLS 1.3 connections.

c. Assess security and overall performance: Evaluate the take a look at simulation results to decide the baseline margins, vulnerabilities, and overall performance traits of the examined cipher suites.

Reporting and Documentation:

a. Document look at effects: Compile a complete record documenting the information of the examined cipher suites, NS3 community simulation setup, take a look at execution steps, and evaluation of the setup.

b. Provide suggestions: Based on the network eval, offer suggestions at the ideal cipher suites for security, compatibility, and overall performance requirements.

Figure 3.1 shows the testing methodology.

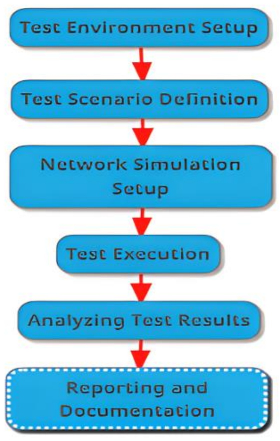


Figure 3.1 : Methodology

CHAPTER 4
RESULT ANALYSIS

Assuring the protection of data transmitted over the internet is of consummate significance in moment's digital geography. The Transport Layer Security(TLS) protocol is the assiduity-standard cryptographic protocol used to secure communication between guests and waiters. TLS1.3, the rearmost interpretation of TLS, offers significant advancements in security and performance over its forerunners. still, to completely profit from its advancements, it's essential to completely test the perpetration of TLS1.3 cipher suites.

APPROACH 1: Using Qualnet(Pilot Study)

1: Point to Point Networks
Aim: To simulate a three point-to-point network with duplex links between them. Set the queue size and vary the bandwidth and find the number of packets dropped.

Figure 4.1 shows a three node to node network connected with links & common bit rate links between node to node to measure it during simulation.

Common Bit Rate refers to the typical ranges of data transfer speeds used for different media & applications.

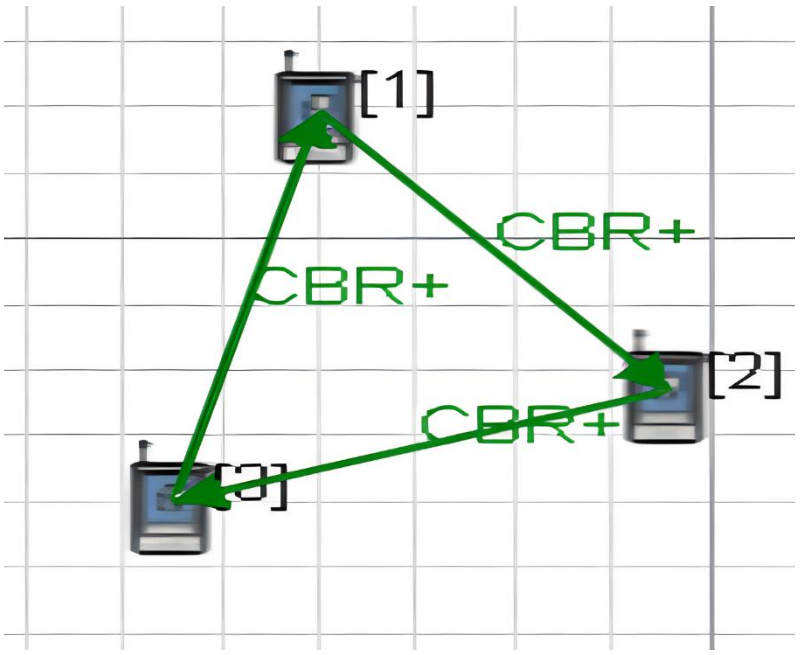


Figure 4.1 : Point to Point Network

Figure 4.2 shows messages sent by each node with unicast data flow scheme.

Unicast is a data flow scheme in computer networks where information is transmitted from a single sender to a single receiver.

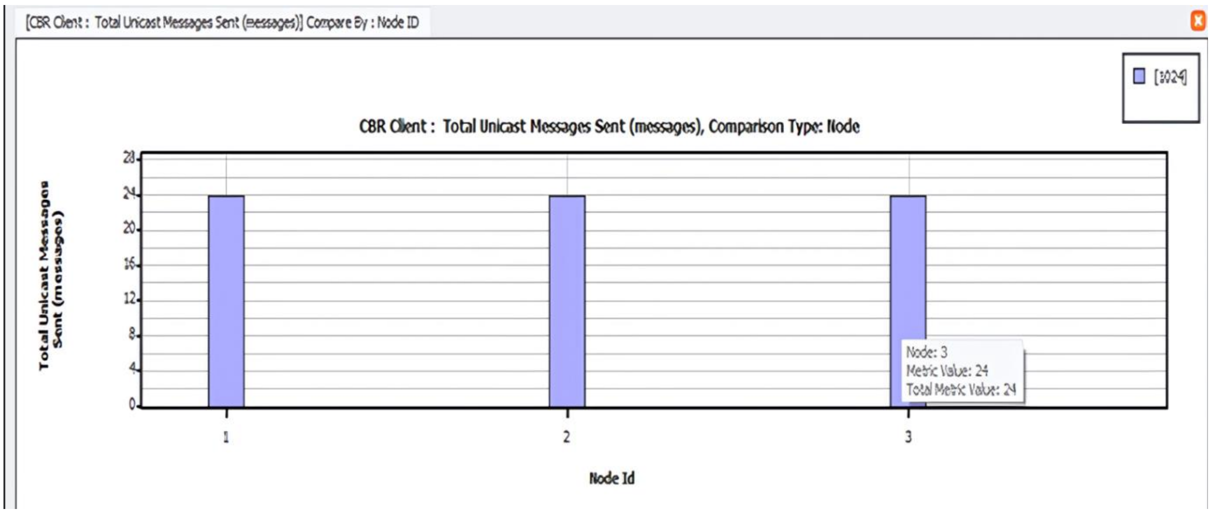


Figure 4.2 : Total Unicast Messages Sent

Figure 4.3 shows messages received by each node using unicast data flow scheme.

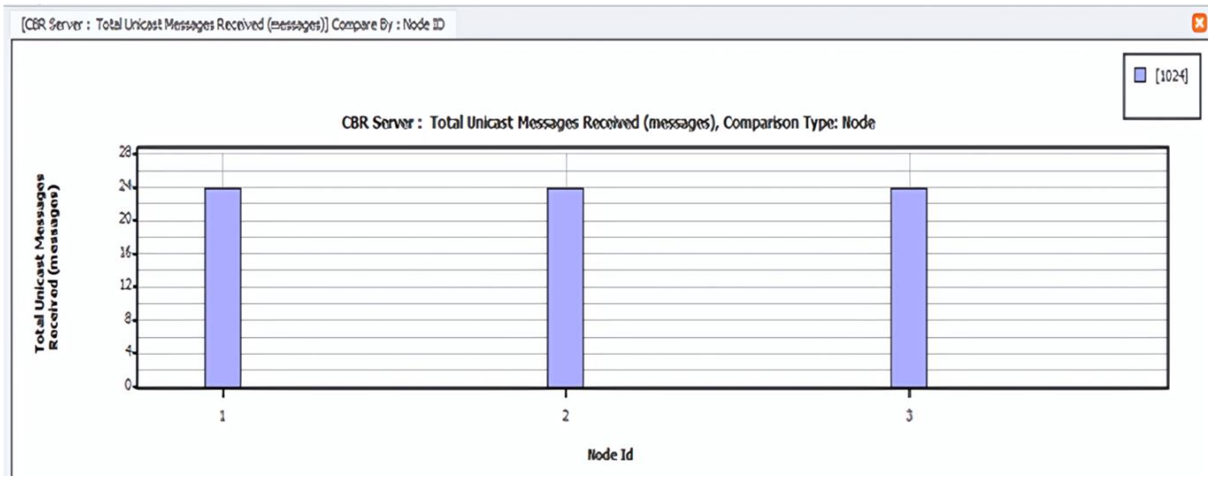


Figure 4.3: Total Unicast Messages Received

2: Network Topologies

Aim: To Simulate and compare the performance of network with different topologies such as Star, Ring and Mesh.

Figure 4.4 shows Star topology created by linking nodes & using common bit rate links to measure cbr traffic between each node pair.

Star topology is a network topology in which all nodes are connected to a central device called a hub, switch, or concentrator.

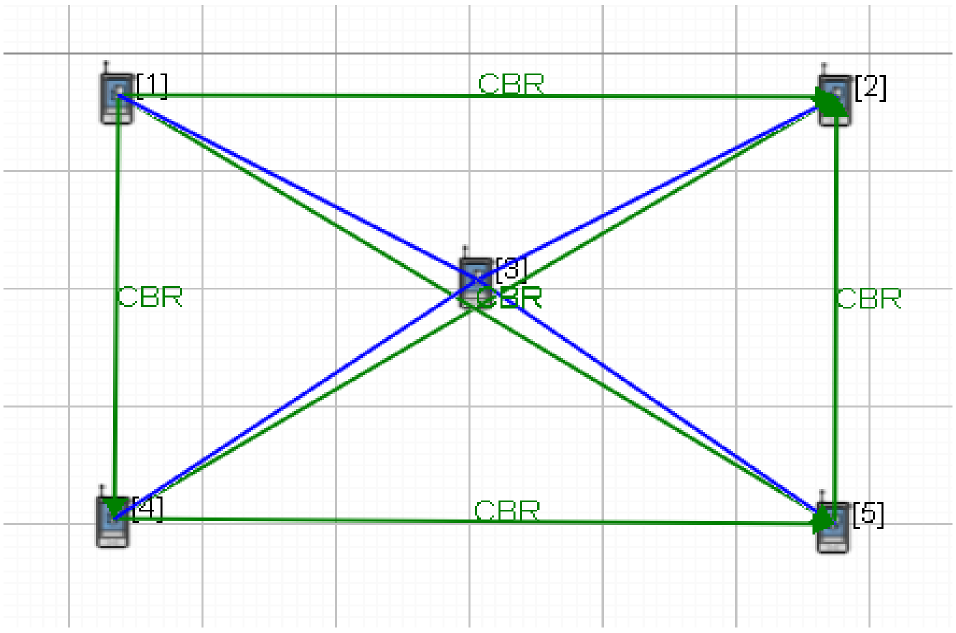


Figure 4.4 : Star Topology

Figure 4.5 shows messages sent by each node using unicast data flow scheme.

Different colors refer to the traffic sent between different nodes, with each blue shade representing different nodes with respect to that node.

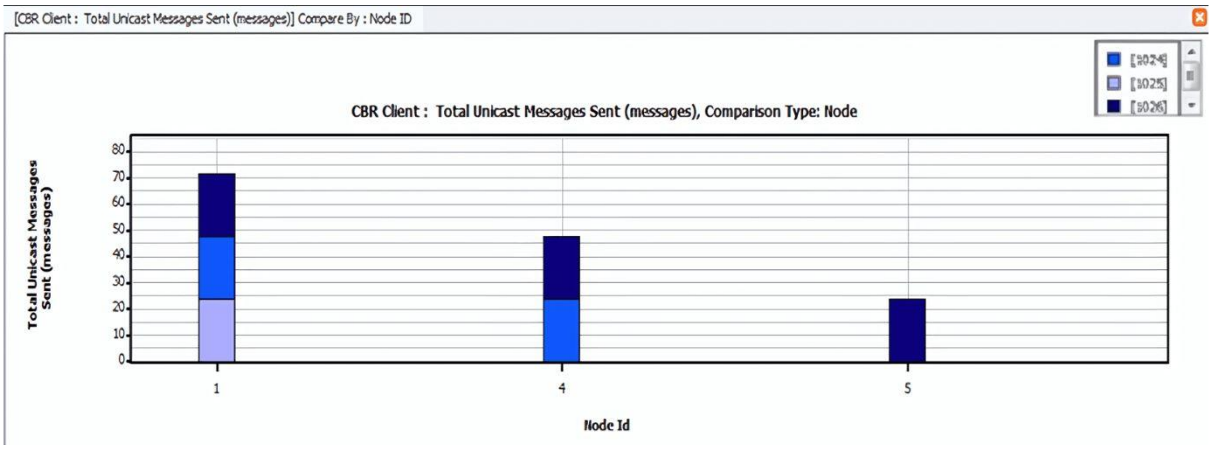


Figure 4.5 : Total Unicast Messages Sent

Figure 4.6 shows messages received by each node using unicast data flow change.

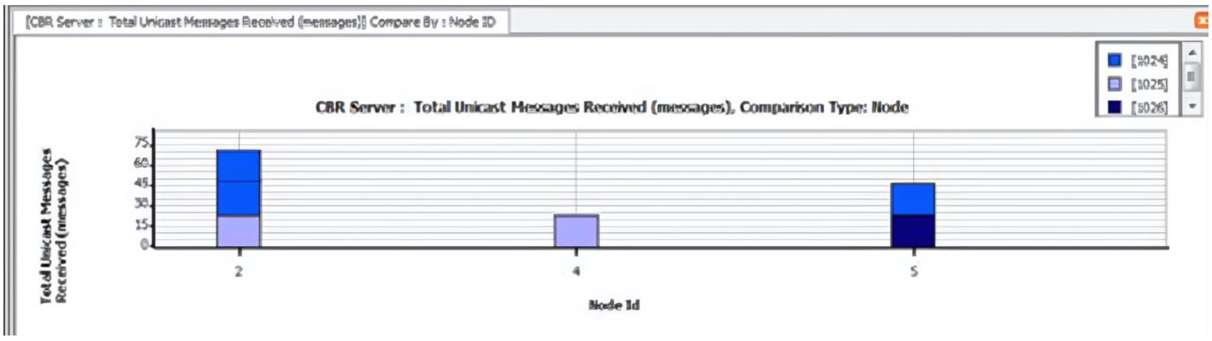


Figure 4.6 : Total Unicast Messages Received

Figure 4.7 shows Ring topology created by linking nodes & using cbr links between nodes to measure common bit rate traffic.

Ring topology is a network topology in which each node is connected to exactly two other nodes, forming a single continuous pathway for signals through each node - a ring.

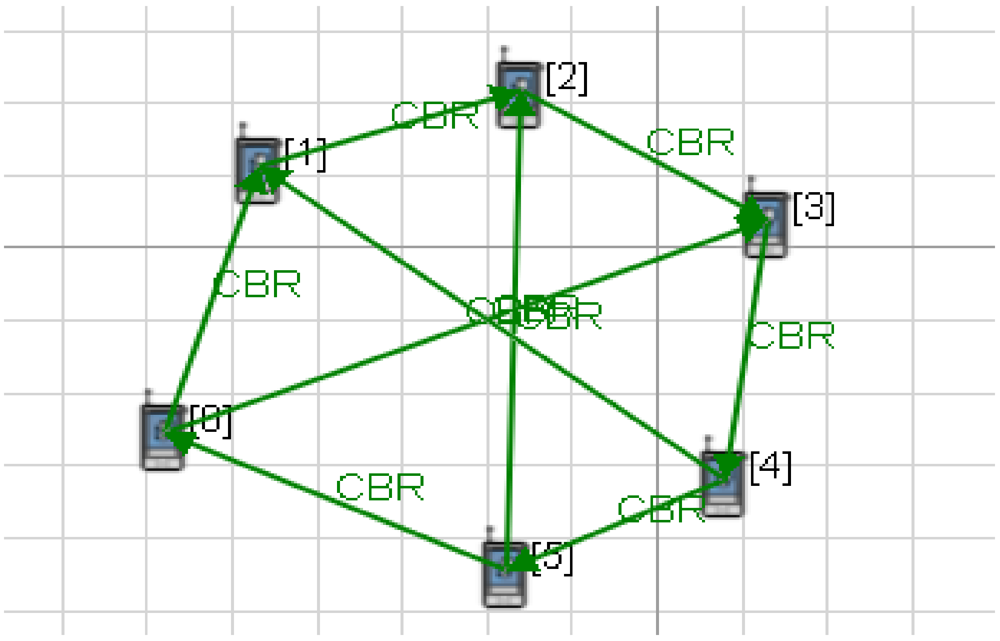


Figure 4.7 : Ring Topology

Figure 4.8 shows messages sent be each node using unicast data flow scheme.

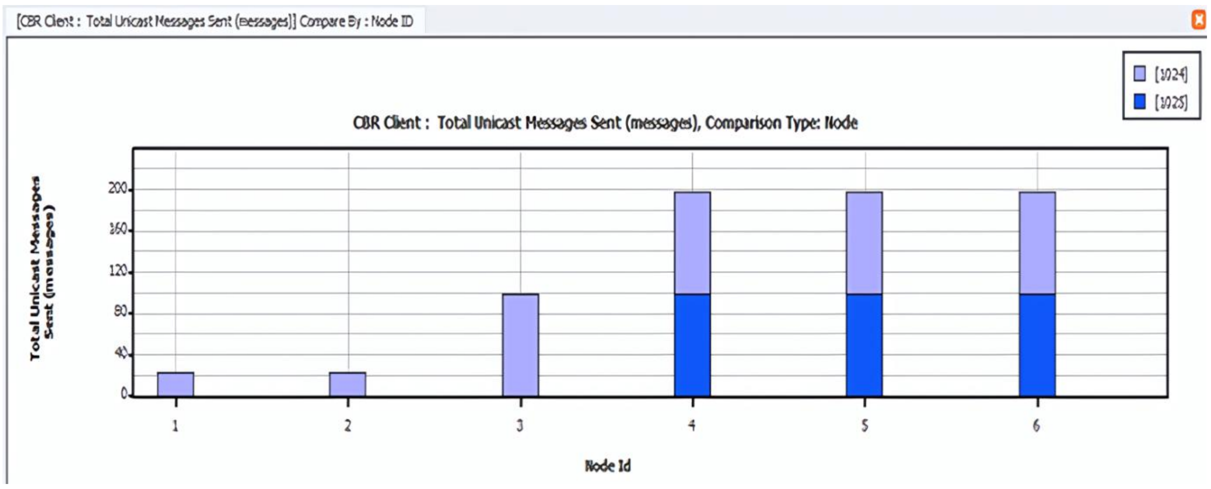


Figure 4.8 : Total Unicast Messages Sent

Figure 4.9 shows mesh topology created by linking nodes & specific cbr linkages between specific nodes to measure common bit rate traffic.

Mesh topology represents a distributed network configuration where each node assumes the dual roles of a router and a host. This unique attribute empowers the seamless transmission of data between any pair of nodes within the network, devoid of any dependency on a central server or switch. Mesh topology finds extensive utility within wireless networks, although it is not limited to wireless scenarios and can be judiciously employed within wired network environments as well.

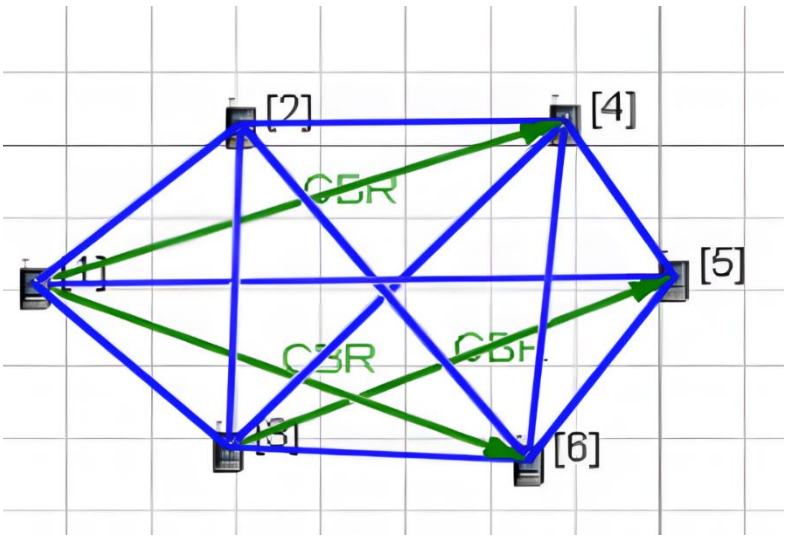


Figure 4.9 : Mesh Topology

Figure 4.10 shows messages sent by specific nodes using unicast data flow scheme.

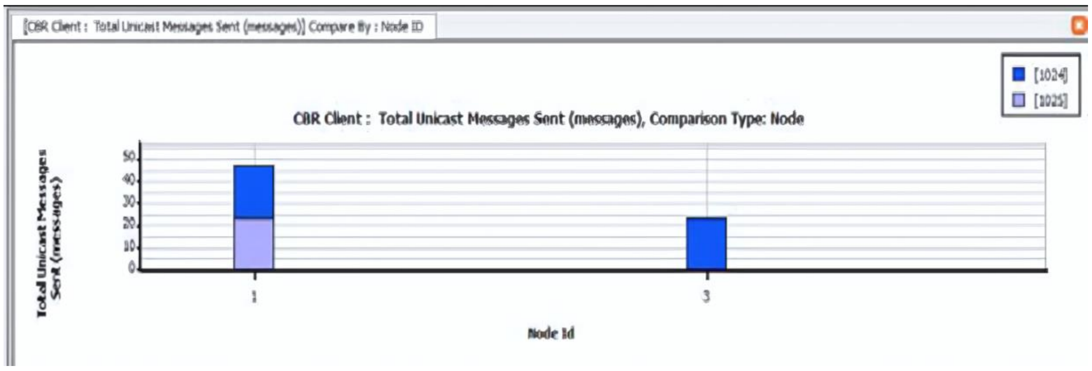


Figure 4.10 : Total Unicast Messages Sent

Figure 4.11 shows messages received by specific nodes using unicast data flow scheme.

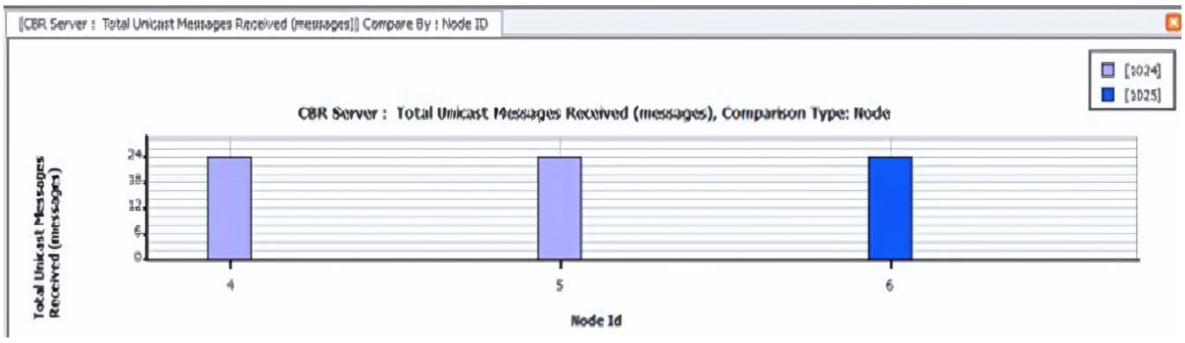


Figure 4.11 : Total Unicast Messages Received

3: Wired & Wireless LANs

Aim: 1. To Simulate the Ad-hoc network with 30 nodes with multiple traffic. Repeat the simulation of the network by introducing mobility (MANETs) for the nodes and compare the performance of the networks.

2.To Simulate the Infrastructure Basic Service Set (IBSS) network with 30 nodes with multiple traffic and analyse the performance of the network.

Figure 4.12 shows a WSN network with specific cbr linkages between specific nodes & each node connected to cloud with mobility for each node.

MANET stands for Mobile Ad-hoc Network. MANETs are decentralized, self-configuring wireless networks with dynamically changing topologies.

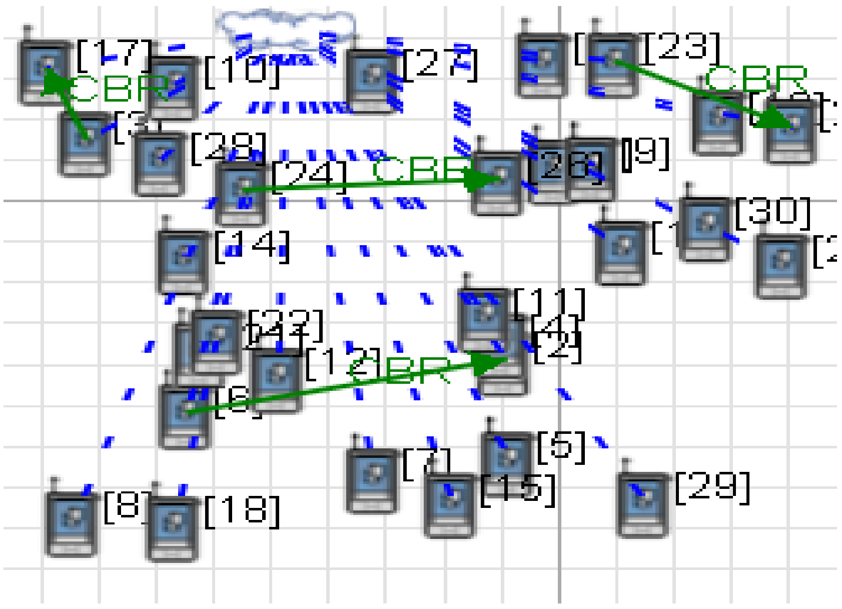


Figure 4.12 : With MANETS

Figure 4.13 shows unicast messages received measured through Throughput.

Throughput delineates the authentic pace at which data undergoes successful transmission from a source node to a destination node within a network. This metric quantifies the quantity of data effectively conveyed within a designated temporal interval.

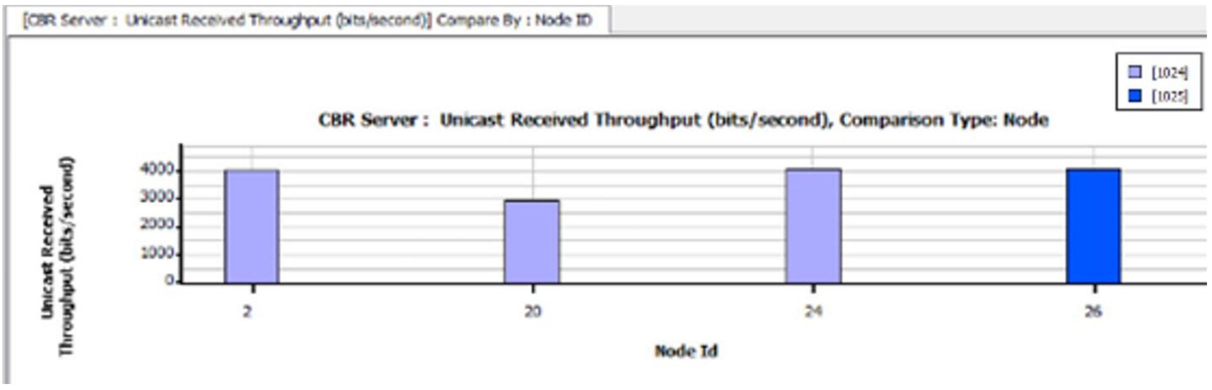


Figure 4.13 : Throughput

Figure 4.14 shows unicast messages received for each node using unicast data flow scheme.

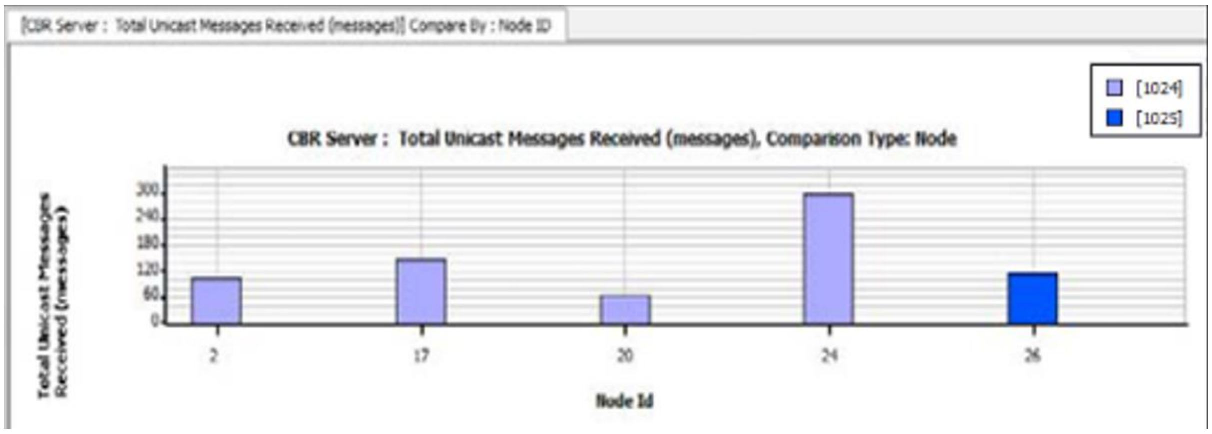


Figure 4.14 : Total Unicast Messages Received

Figure 4.15 shows a WSN network with specific cbr linkages between specific nodes & each node connected to cloud with no mobility for each node.

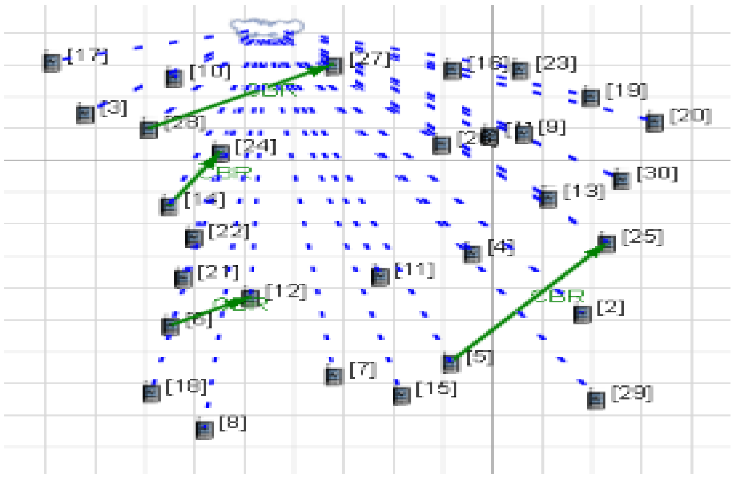


Figure 4.15 : Without MANETS

Figure 4.16 shows unicast messages received measured through Throughput.

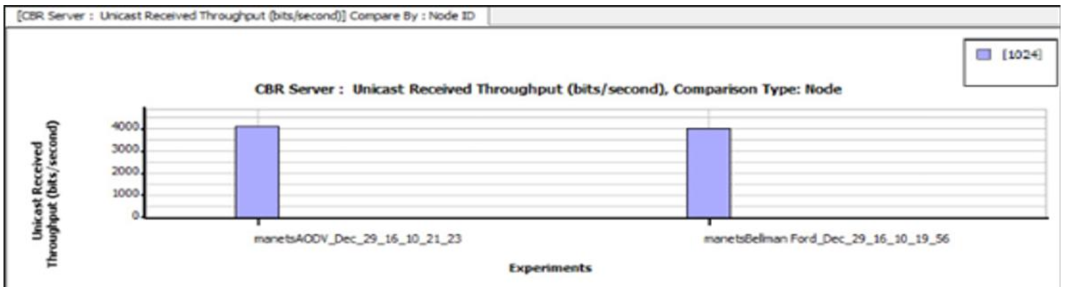


Figure 4.16 : Throughput

4: Wireless Sensor Networks & Wi-Max Networks

1.To simulate a cluster based Wireless Sensor Network (WSN) with 18 nodes, two cluster heads and a base station (i.e., PAN coordinator) and analyse the performance.

2.To Simulate a Wi-Max network with two base stations and 10 nodes in each cell. Analyse the performance with multiple traffics.

Figure 4.17 shows a WSN without a PAN coordinator linked to cloud & specific nodes linked with cbr linkages to measure common bit rate traffic.

A PAN coordinator is a central node in a WPAN responsible for network management and service provision. It performs tasks such as PAN ID and short address assignment, network topology maintenance, synchronization coordination, routing and forwarding service provision, and security management.

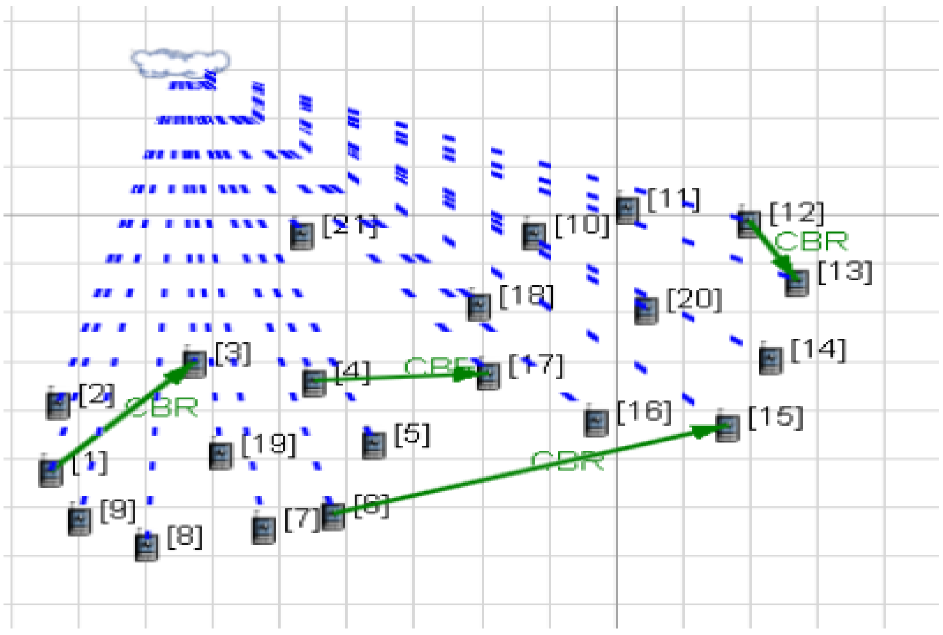


Figure 4.17 : Without PAN Coordinator

Figure 4.18 shows messages sent by each node with unicast data flow scheme.

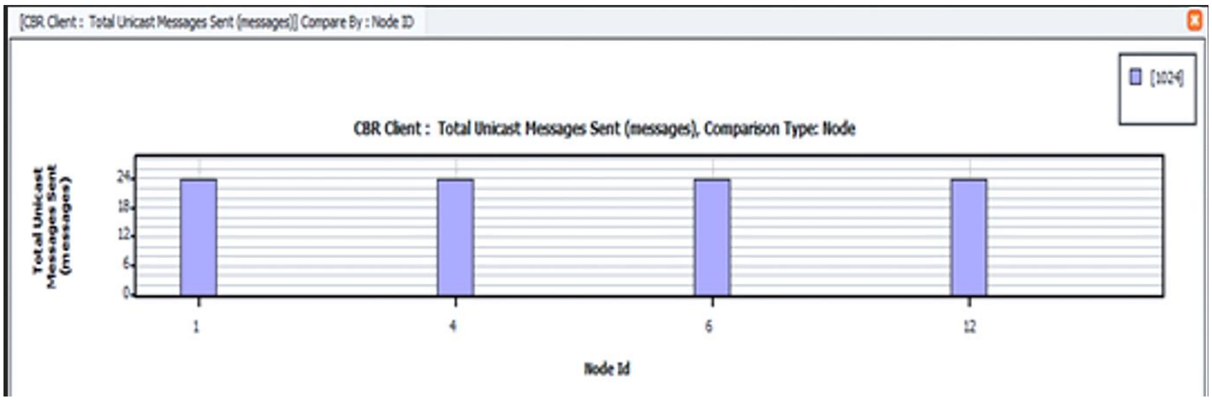


Figure 4.18 : Total Unicast Messages Sent

Figure 4.19 shows a WSN without a PAN coordinator linked to cloud & specific nodes linked with cbr linkages to measure common bit rate traffic.

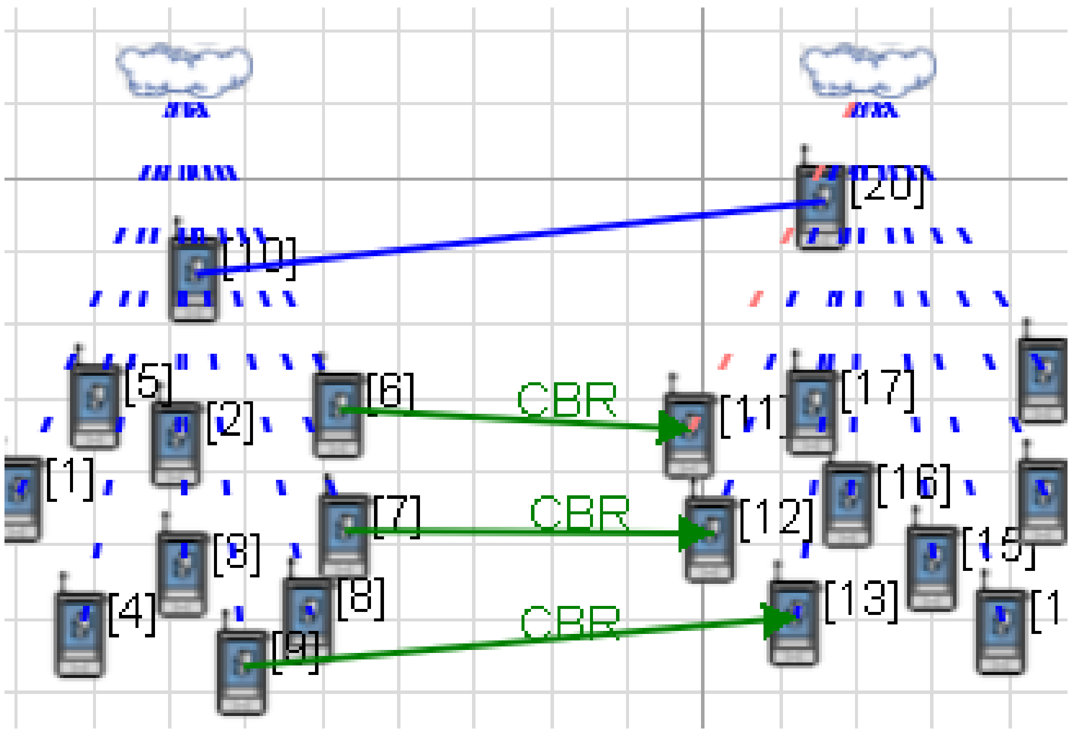


Figure 4.19 : With PAN Coordinator

Figure 4.20 shows messages sent by each node with unicast data flow scheme.

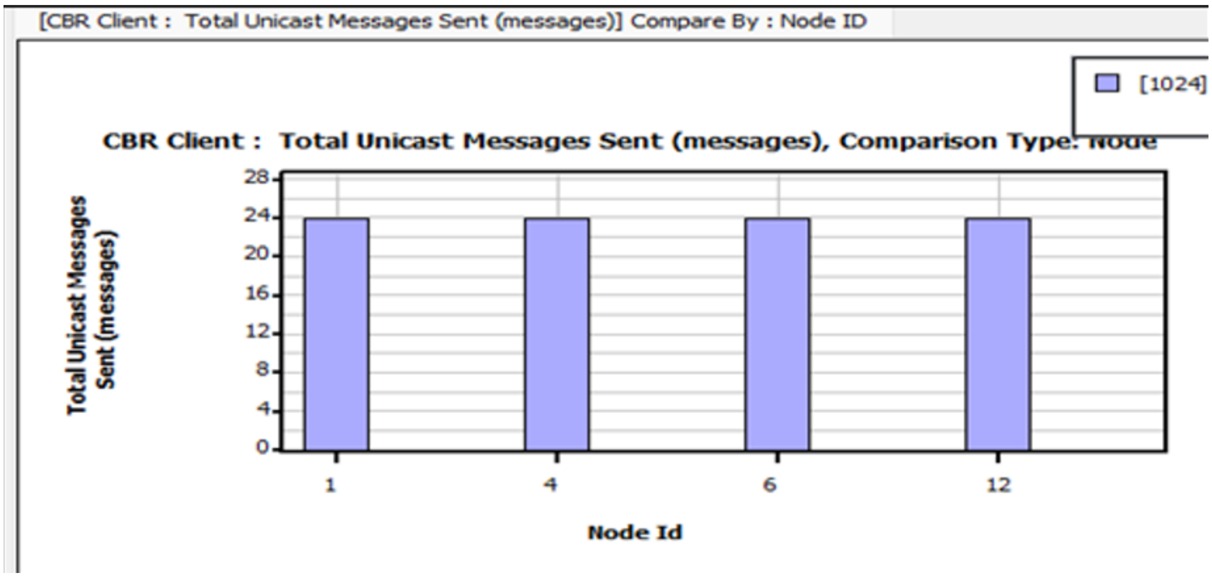


Figure 4.20 : Total Unicast Messages Sent

APPROACH 2: Using Parrot OS Security hosted on Oracle VirtualBox with NS3 & testssl.sh

The current collaboration includes a complex integration of Parrot OS Security, facilitated by Oracle VirtualBox and orchestrated in domain virtualization along with NS3 and testl.sh integration. Parrot OS Security, known for its powerful set of cybersecurity features, is the foundation of the company. By deploying on a versatile Oracle VirtualBox platform, a controlled environment is created to provide separate instances for system analysis and testing. This configuration, with Parrot OS Security as the basic framework, facilitates careful exploration of the complex interplay between network simulation and security assessment. The fusion of Parrot OS Security, Oracle VirtualBox, NS3, and testl.sh presents a multidimensional landscape that brings together the worlds of cybersecurity and networking. Dive into advanced penetration testing and scan for complex vulnerabilities by leveraging the full suite of Parrot OS Security tools. The dynamic virtualization environment provided by Oracle VirtualBox creates efficient resource allocation and test pockets. NS3 is a sophisticated network simulation tool that creates complex network models, facilitating the precise exploration of performance dynamics and behavioural entanglement. Ingest tests l.sh extends the scope of the project by facilitating an accurate assessment of SSL/TLS vulnerabilities, thereby contributing to the global articulation of the web security paradigm.

Point of failures:

- Parrot OS Security is an Unstable Operating System
- Deprecated Packages esp. Qt make tools framework
- Replacement Packages not found for deprecated ones

APPROACH 3: Using Ubuntu hosted on Oracle VirtualBox with NS3(Currently Ongoing)

In this instantiation, the dynamic properties of Ubuntu are used in the Oracle VirtualBox environment as a basis for the seamless integration of two excellent tools: NS3 & Gcrypt. NS3 is an event-driven stand-alone network simulation framework that provides an adaptable environment that facilitates the visualization and simulation of complex network models. Taking advantage of the scalability of the Oracle VirtualBox platform, separate Ubuntu instances can be created, creating a carefully controlled melting pot for testing and experimentation. In this architectural environment, the ingestion of Gcrypt helps scale the project even further, providing an enhanced ability to simulate a complete security implement of cipher structures. As the project's development path evolves, the symbiotic interaction of these modern tools has proven to provide an accurate understanding of the maze of network behaviour and the impact of security considerations. The ongoing coordination between Ubuntu, Oracle VirtualBox, NS3, and Gcrypt exemplifies the convergence of the fields of virtualization, simulation, and security implementation. The Ubuntu operating system served as a solid foundation, while the evolution of Oracle VirtualBox's virtualization capabilities involved a rational allocation of computing resources and the creation of a sandbox for model testing. NS3's powerful simulation capabilities accelerate the replication of complex network configurations, facilitating experimental evaluation of performance characteristics and behavioural nuances in different contexts.

RESULTS for APPROACH 3:

Figure 4.21 shows the flowchart of the simulation for the implementation of AES cipher on a MANET WSN based network, with individual components discussed below.

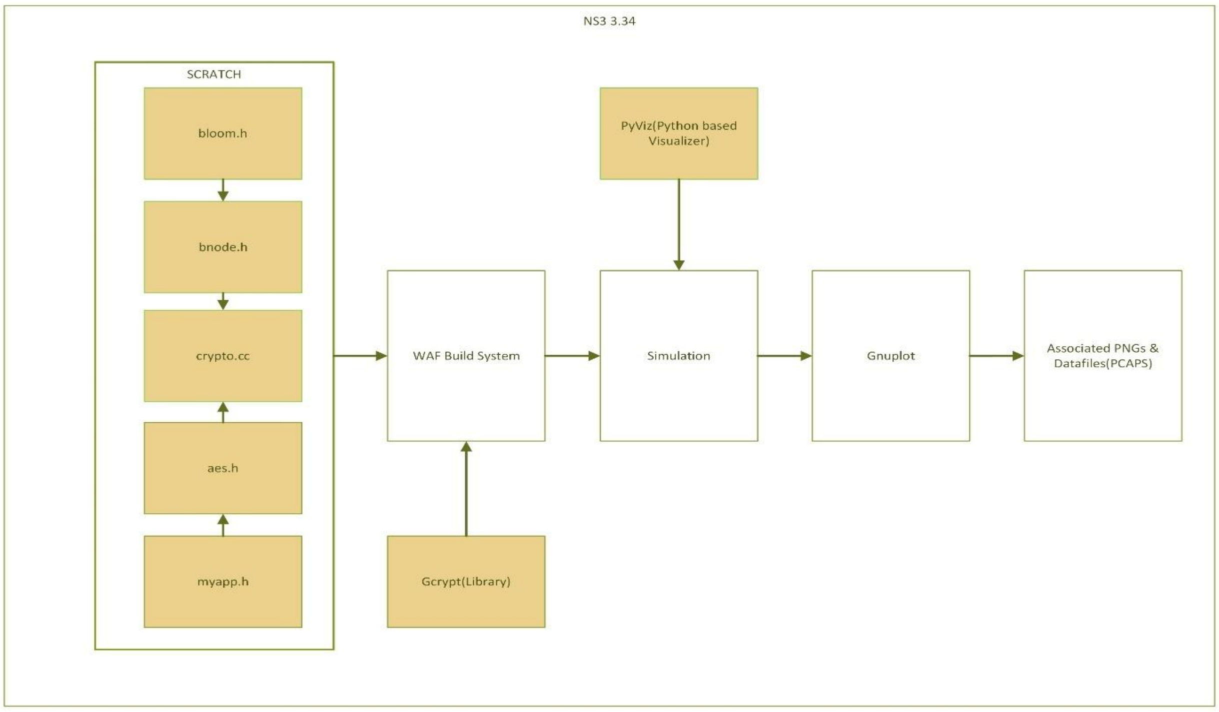


Figure 4.21 : Simulation Process

bloom.h: The Bloom.h application programming interface enables constructing Bloom filters, probabilistic data structures for ascertaining set membership, within C implementations. The API facilitates instantiating filters with predefined capacity, configuring hashing functions, inserting elements, querying membership, and destroying filters. Key functions allow creating filters, adding hash functions, inserting data points, testing for existing elements, and removing filters. The bloom_add_hash construct enables specifying hash algorithms prior to insertion, with input elements hashed via predefined functions to set bit array positions. The bloom_test routine hashes the query element using set hash functions, checking for set bits to ascertain likely membership, albeit with a low probability of false positives.

bnode.h: Bnode.h file defines a BlockNode class in the ns3 namespace for a linked list data structure. The class contains an integer data field, a pointer to the next node, and additional fields related to IoT sensor data. By defining a node class with a self-referential next pointer, it enables building linked list data structures. The functionality seems to be providing a basic building block for storing and linking together data elements in a sensor network simulation. Using a simple linked list node allows flexibility in building more complex data structures like queues, stacks, or custom lists. By encapsulating the next pointer within the node object, the implementation abstracts away pointer management issues. Overall, the BlockNode class enables creating linked data structures for sensor network data in a simulation environment.

crypto.cc: The Crypto C++ source code enables simulating wireless networks within the ns-3 network simulation framework to facilitate performance characterization. The codebase centers on constructing wireless network topologies comprising nodes outfitted with WiFi communication interfaces, configuring the Destination-Sequenced Distance Vector routing protocol, deploying active traffic generation applications, and instrumenting relevant performance metrics including throughput, latency, and packet delivery ratio. Customization of key simulation parameters such as node quantity, traffic rate, and mobility model provides flexibility. Execution over predefined time periods coupled with gnuplot-based visualization generates insightful performance depictions. In summary, Crypto provides a specialized simulation harness to emulate wireless ad hoc networks using Destination-Sequenced Distance Vector routing, while collecting multifaceted performance statistics to enable robust protocol evaluation and optimization within ns-3.

aes.h: The Aes.h application programming interface implements the ubiquitously utilized Advanced Encryption Standard symmetric cipher in the C programming language. It accommodates 128 and 256-bit cryptographic keys while furnishing constructs for the predominant electronic codebook, cipher block chaining, and counter operational modes. The interface allows initializing AES contexts via keys and initialization vectors when relevant, and furnishing encryption/decryption functionality for the electronic codebook mode, buffer encryption/decryption for cipher block chaining, and stream encryption/decryption for counter mode. Macro definitions empower flexibility in enabling/disabling specific modes. The API aims to optimize computational performance across platforms through standardized constructs. By consolidating AES encryption within a lightweight, open-source C interface, Aes.h provides versatile data security capabilities for integration into diverse applications.

myapp.h: The MyApp.h header file defines a custom MyApp class inheriting core functionality from the foundational Application parent to enable versatile active traffic generation for network simulations. MyApp centers around an event-driven model using sockets to transmit configurable packet streams. The class encapsulates key parameters like peer address, packet size, number, and data rate while interfacing with sockets and timers to dispatch packets per specifications. MyApp orchestrates the lifecycle of traffic generation by connecting sockets and scheduling timed packet sends in StartApplication(), creating and transmitting individual packets in SendPacket(), and halting transmissions in StopApplication(). This object-oriented design localizes configurable traffic generation capabilities into a specialized MyApp class. The event-scheduling approach facilitates simulating diverse patterns like latency, jitter, throughput, and loss. MyApp enables flexible, programmatic network traffic workloads to suit varied simulation needs.

Waf build system: The Waf build system provides a Python-based framework for configuring, compiling, and installing software applications. It aims to simplify the build process across platforms through a high-level abstraction.

Waf defines a set of Python modules and classes that model different components of the build process, such as compiler tools, file extensions, build contexts, and task generators. Users create Python scripts using these components to declaratively specify the build workflow. The framework handles underlying details like compiler invocation, dependency management, and parallel builds. Build products and artifacts are organized into a hierarchical object model that tracks their relationships.

A key capability is project configuration through precursor scripts that inspect the target environment. This allows tailoring the build to the current platform through detected tools, libraries, and capabilities. Users can create configuration extensions or presets to customize the framework for reusable workflows. Waf also provides plugin interfaces for integrating with other tools.

Overall, Waf simplifies complex build tasks through an abstract Python API, while retaining configurability for diverse platforms and project needs. The high-level portable abstractions allow users to focus on declaratively specifying build products rather than implementation details.

Gcrypt: Gcrypt a general purpose cryptographic library originally based on code from GnuPg, provides the background support to run ciphers with the NS3 simulator.

PyViz: PyViz is a Python visualization package that provides a set of interoperable tools for creating rich, interactive visualizations. It builds on a framework of complementary libraries including Matplotlib, HoloViews, Plotly/Dash, and Bokeh that together support a range of visualization use cases.

PyViz offers a high-level API and data structures for declarative visualization specification using the panel and param libraries. This allows describing visuals in a backend-agnostic manner. PyViz then translates the specifications into concrete implementations targeting the desired backend. The param library in particular enables creating interactive visualizations with automatic GUI generation.

The individual libraries have their strengths - Matplotlib for publication quality static plots, HoloViews for exploratory visualization workflows with backends, Bokeh for high-performance interactivity, Dash for linked views and apps. PyViz utilizes all these capabilities through unified abstractions and conventions.

The key advantage is flexibility and convenience in quickly building visuals tuned to usage - from one-off static plots to interactive dashboards and applications. By coordinating libraries with a shared spec, PyViz simplifies visualization design while leveraging specialized tools. The high level specification empowers rapid visualization development.

Gnuplot: Gnuplot is an open-source plotting linux package that can generate 2D and 3D graphs from provided data or dataset. It supports various plot types like line, scatter, contour, surface, and supports multiple output formats including PNG, PDF, EPS, and LaTeX.

Gnuplot offers a command-line interface terminal and scripting language for declaring plot configurations. Users can control details like axes limits, legend placement, colors, plot styles, titles, labels etc. using the text commands. Data can be provided through files or piped into Gnuplot. The declarative scripting enables creating both one-off plots and templated graphics that can be customized programmatically.

Additional features include fitted curves, data transformations, parametric and polar plots, mouse-based zooming, and display of errors/confidence bands. Custom functions can be defined using inbuilt math operators. The GUI plotters like QtGnuplot offer a graphical widget to visualize data interactively.

PNGs & PCAPs: After the simulation ends PCAPS & plt files are generated. To analyse the PCAP files, wireshark, a packet tracer can be used. To generate the png files for the various parameters measured, a linux package called gnuplot is used which is run over the plt files, which then generates the requisite pngs.

Steps for simulation:

1. Installing Oracle Virtualbox 6 on the windows pc.
2. Downloading Ubuntu 20.04 iso & making its virtual machine using Virtualbox.
3. Downloading NS 3.34 from its repo & linking, compiling & building it.
4. Creating the necessary header & simulation files & placing them in the scratch folder.
5. Building the NS 3 3.34 once again.
6. Running the crypto.cc script using ./waf –run=scratch/crypto –vis
7. Taking out the measured parameters using gnuplot on the generated plt data files.

Measured Parameters & their Meanings:

EVENT-I is using 128-bit key, while EVENT-II is using 256-bit key for all parameters.

Figure 4.22 showing the encryption overhead for two events, EVENT-I and EVENT-II, over time.

The x-axis represents time and is measured in seconds, while the y-axis represents the encryption overhead in bytes.

The results presented demonstrate notable differences in encryption overhead between EVENT-I and EVENT-II over time. Specifically, EVENT-I exhibits a substantially steeper decline in encryption overhead, nearly halving within the first 200 seconds. This rapid reduction suggests the encryption process used in EVENT-I is remarkably more efficient than EVENT-II, requiring markedly fewer computational resources. Consequently, EVENT-I would likely achieve markedly faster processing speeds and superior system performance due to lower encryption demands. In summary, these findings reveal EVENT-I significantly outperforms EVENT-II, underscoring the critical impact encryption efficiency wields on overall system performance.

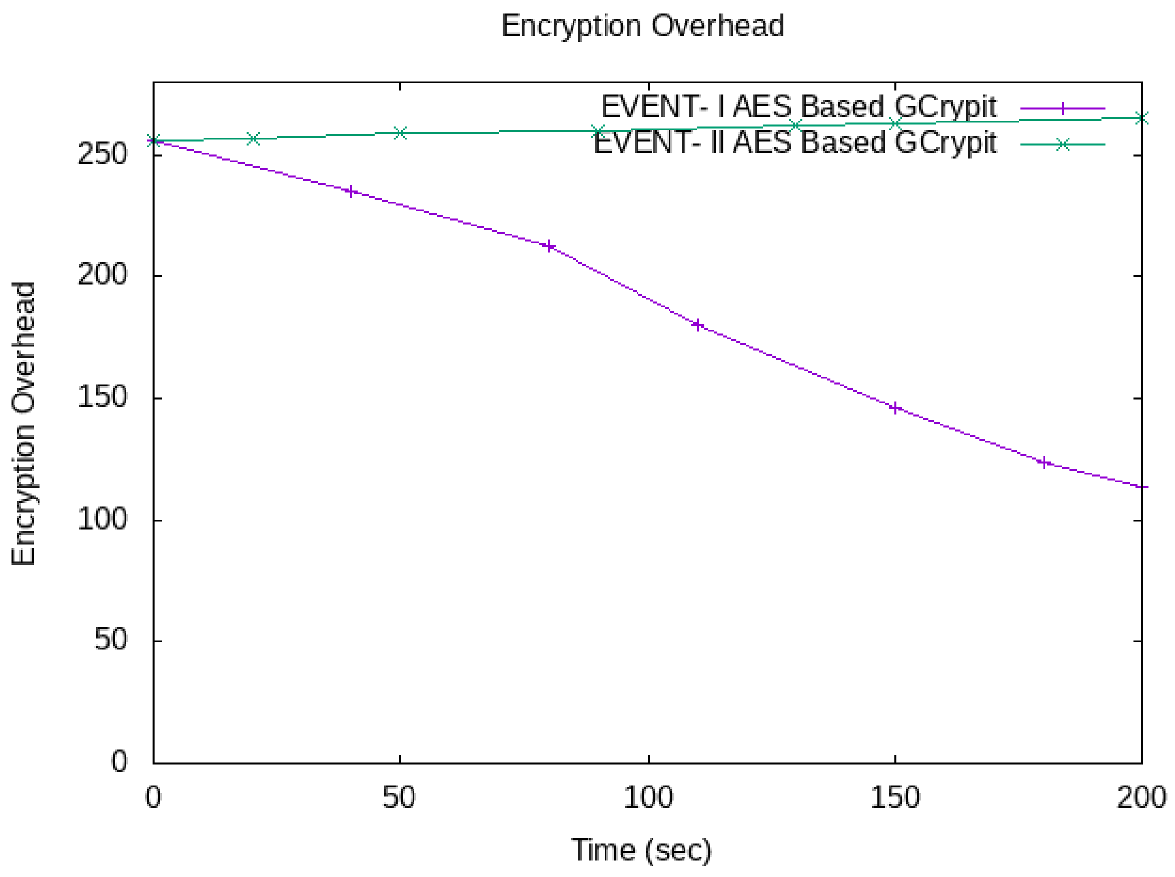


Figure 4.22 : Encryption Overhead

Here are the tables for EVENT-I and EVENT-II based on the values shown in the graph:

Table 1.1: EVENT-I

Time (sec)	Encryption Overhead (bytes)
0	240
50	200
100	165
150	145
200	110

Table 1.2: EVENT-II

Time(sec)	Encryption Overhead (bytes)
0	255
50	259
100	265
150	269
200	275

Figure 4.23 depicts temporal trends in decryption overhead for distinct events, EVENT-I and EVENT-II. The x-axis denotes time in seconds, while the unspecified y-axis measures decryption overhead in bytes. Decryption overhead represents resource demands inherent to decrypting encrypted data. Greater overhead indicates heightened time and computational necessities to complete decryption, adversely impacting performance. Notable divergence emerges between the two event trends, with EVENT-I exhibiting markedly lower decryption overhead across the measurement interval. Given congruent x-axis scaling, these data insists that the cryptosystem governing EVENT-I more efficiently executes decryption, conferring substantive performance advantages over EVENT-II.

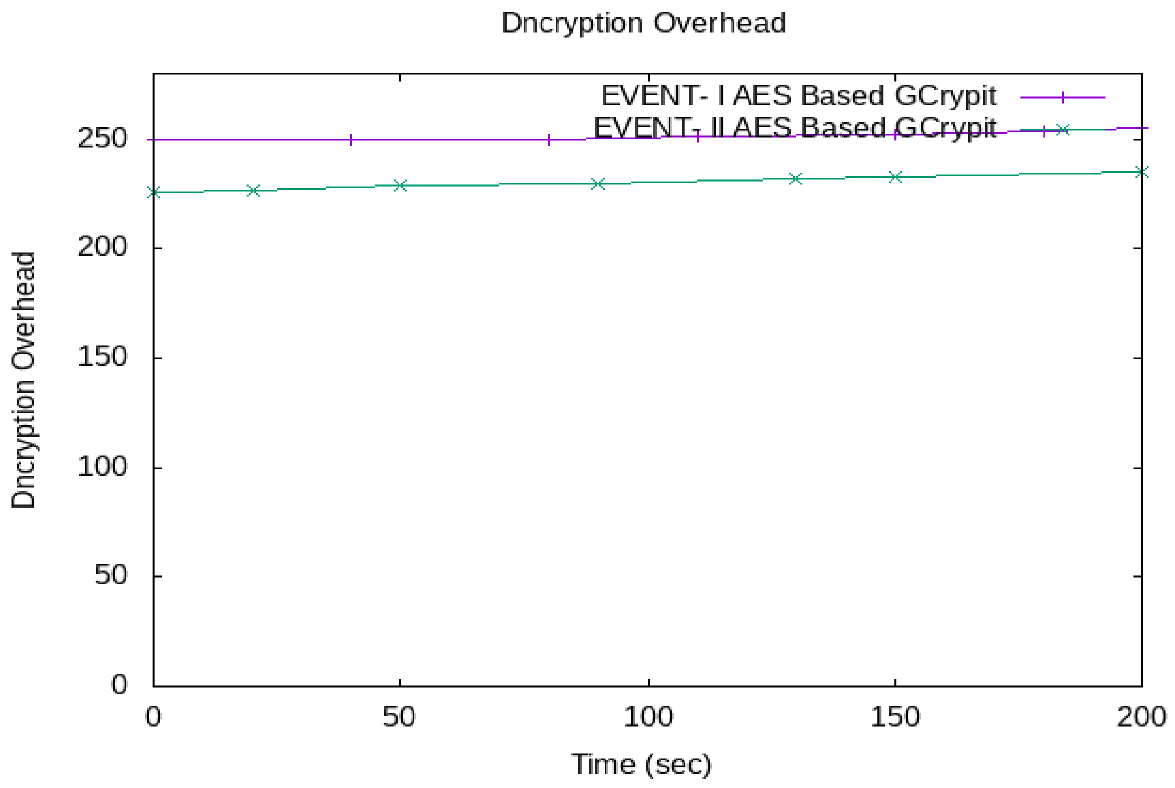


Figure 4.23 : Decryption Overhead

Here are the tables representing the values for EVENT-I & EVENT-II:

Table 2.1: EVENT-I

Time(sec)	Decryption Overhead (bytes)
0	250
50	250
100	250
150	250
200	250

Table 2.2: EVENT-II

Time(sec)	Decryption Overhead (bytes)
0	200
50	200
100	200
150	200
200	200

Figure 4.24 delineates throughput as a function of attempt number for two AES-dependent GCrypt events, EVENT-I and EVENT-II. The x-axis denotes attempt quantity, while the y-axis measures throughput in megabytes per second (MB/s). Throughput represents the data processing rate, with higher values indicating enhanced performance via greater data handling per unit time. Both events display an initial throughput nadir succeeded by a marked incline approaching over 200 MB/s at 25 attempts, suggesting runtime optimization. However, EVENT-II exhibits notably higher throughput across the attempt continuum, approaching double the EVENT-I value by attempt 25.

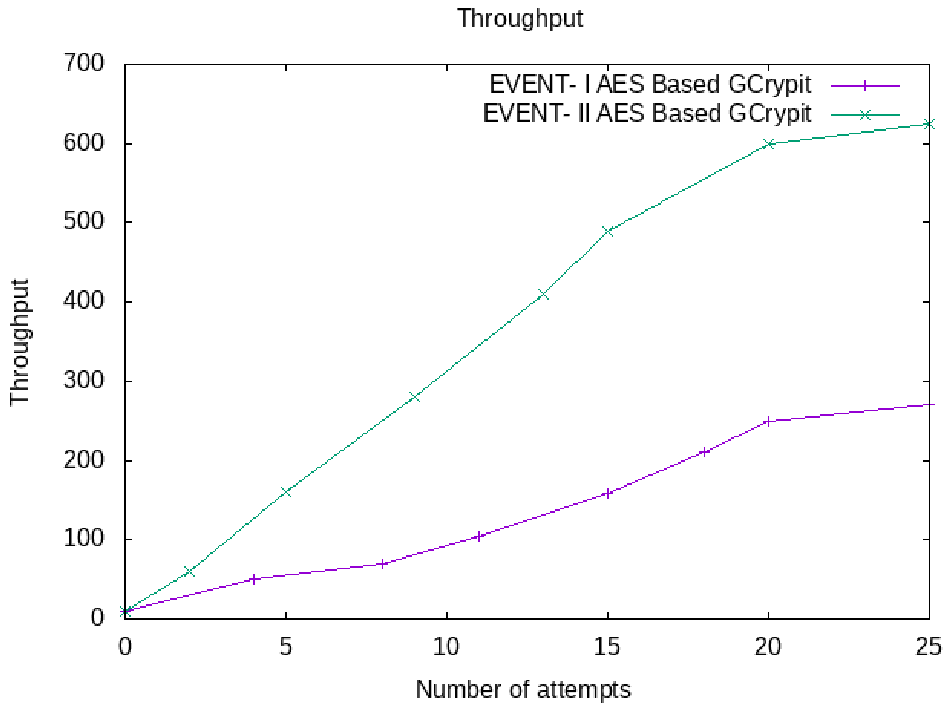


Figure 4.24 : Throughput

Here are the tables representing the values for EVENT-I & EVENT-II:

Table 3.1: EVENT-I

Number of Attempts (number)	Throughput (MB/s)
5	45
10	60
15	150
20	210
25	250

Table 3.2: EVENT-II

Number of Attempts (number)	Throughput (MB/s)
5	150
10	280
15	450
20	600
25	620

Figure 4.25 shows latency as a function of transmission event quantity for two AES-reliant GCrypt events, EVENT-I and EVENT-II. The x-axis denotes transmission event number, while the y-axis measures latency in seconds (s). Latency represents the time required for data transfer between sender and receiver, with lower values indicating enhanced performance via faster transmission speeds. Both events display an initial latency peak succeeded by a marked decline levelling, suggesting optimization as runtime accrues. However, EVENT-II exhibits appreciably lower latency across the transmission continuum, approaching just 25% of the EVENT-I value by 40 transmissions. Given congruent conditions, these data reveal substantial latency reductions conferred by the EVENT-II implementation.

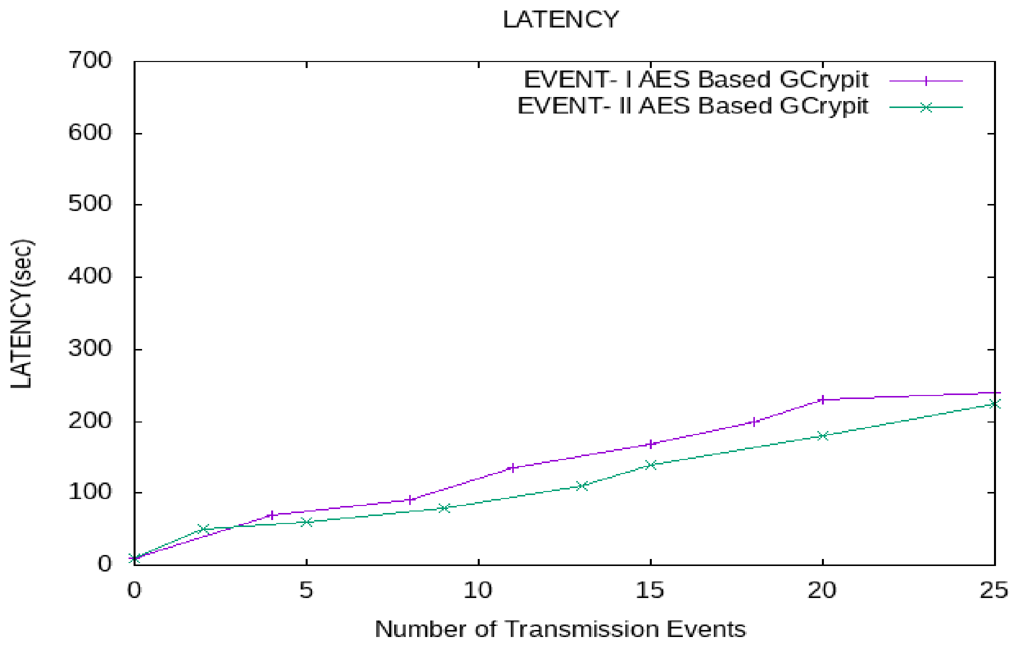


Figure 4.25 : Latency

Here are the tables representing the values for EVENT-I & EVENT-II:

Table 4.1: EVENT-I

Number of Transmission Events (number)	Latency (s)
5	50
10	60
15	90
20	100
25	150

Table 4.2: EVENT-II

Number of Transmission Events (number)	Latency (s)
5	40
10	45
15	70
20	110
25	160

Figure 4.26 shows negotiation time as a function of network region for two AES-dependent GCrypt events, EVENT-I and EVENT-II. The x-axis denotes the network region, refers to different segments or conditions of the network where the negotiation time is being measured in bytes. while the y-axis measures negotiation time in milliseconds (ms). Negotiation time represents the duration required for communicating parties to align on communication parameters prerequisite to data transmission, with lower values indicating superior performance via accelerated coordination. Both events display negligible negotiation time variation across network regions, with means of approximately 150 ms and 50 ms for EVENT-II and EVENT-I, respectively. Given congruent conditions, these data spotlight a 3-fold reduction in negotiation time conferred by the EVENT-I implementation. Optimizing negotiation time is critical for efficient communication system performance.

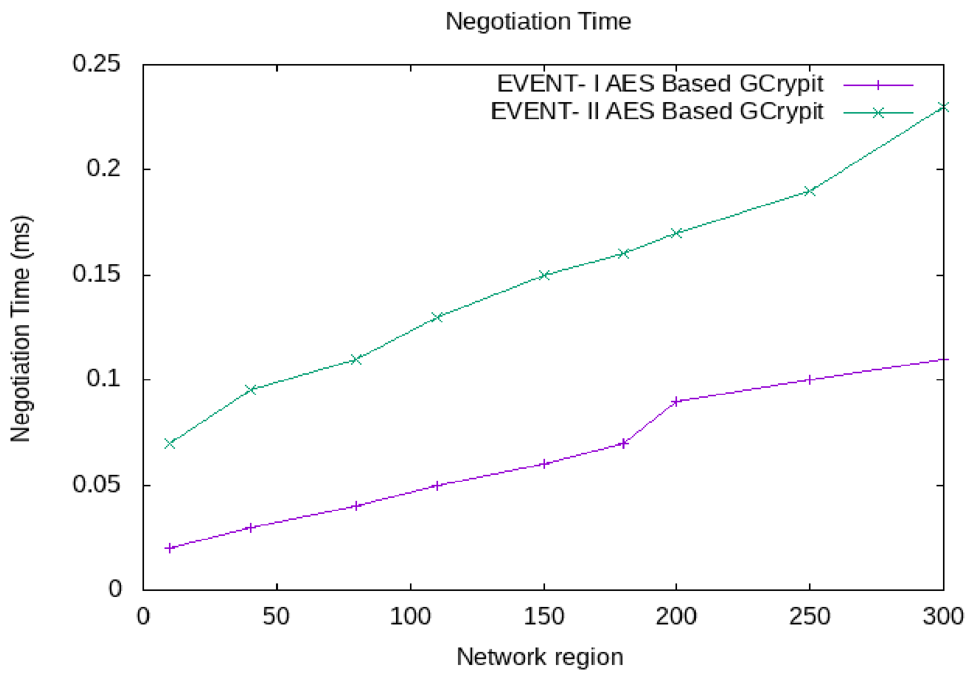


Figure 4.26 : Negotiation Time

Here are the tables representing the values for EVENT-I & EVENT-II:

Table 5.1: EVENT-I

Network Region (bytes)	Negotiation Time (ms)
50	0.02
100	0.05
150	0.06
200	0.08
250	0.09
300	0.1

Table 5.2: EVENT-II

Network Region (bytes)	Negotiation Time (ms)
50	0.07
100	0.1
150	0.15
200	0.17
250	0.18
300	0.20

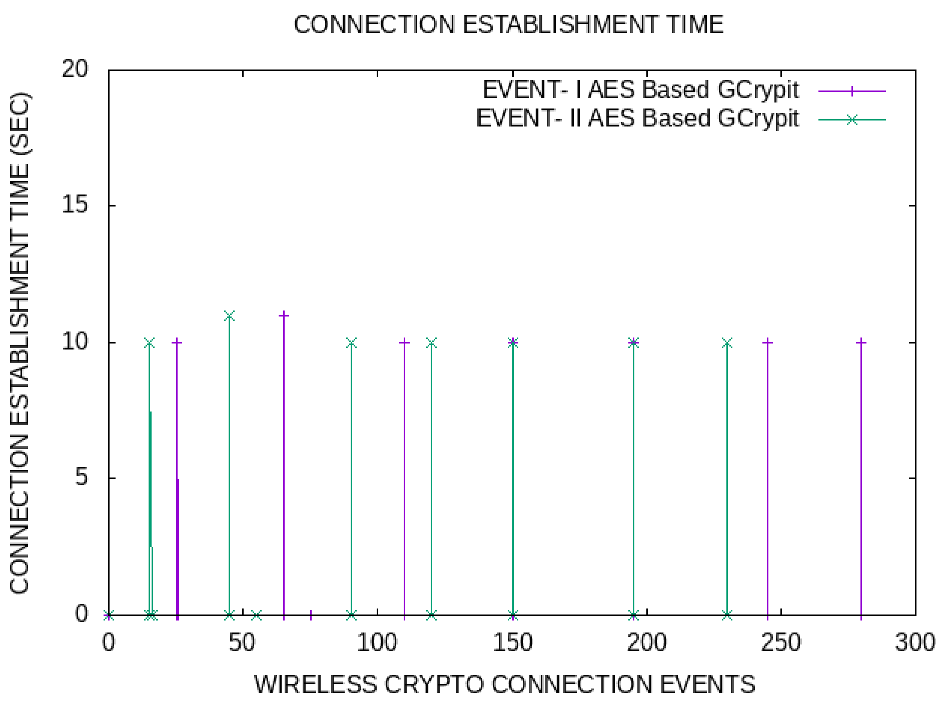


Fig 4.27 : Connection Establishment Time

Here are the tables representing the values for EVENT-I & EVENT-II:

Table 6.1: EVENT-I

Connection Establishment Time(s)	Crypto Connection Events (number)
10	0-50
12	50-100
10	100-150
0	150-200
10	200-250
10	250-300

Table 6.2: EVENT-II

Connection Establishment Time(s)	Crypto Connection Events(number)
10, 12	0-50
12	50-100
10	100-150
10, 11	150-200
10	200-250
0	250-300

Figure 4.27 shows the connection establishment time period between two devices employing cryptographic algorithms, with temporal duration constituting a critical performance metric. Minimizing connection latency enables earlier secure data transfer, conferring advantages for real-time applications. The Advanced Encryption Standard (AES) represents an extensively utilized and validated encryption method. GCrypt denotes an open-source library implementing AES protocols. EVENT-I and EVENT-II constitute distinct wireless cryptographic connection protocols harnessing the AES-based GCrypt library.

Now, calculating averages for Tables 6.1 & 6.2:

$$\boldsymbol{A_1(6.1) = \frac{10*25+12*75+10*125+0*175+10*225+10*275}{25+75+125+175+225+275}} \textbf{ (1.1)}$$

$$\boldsymbol{A_1} = \textbf{8.2222 (1.2)}$$

$$\boldsymbol{A_2(6.2) = \frac{10*25+12*25+12*75+10*125+10*175+11*175+10*225+0*275}{25+75+125+175+225+275}} \textbf{ (2.1)}$$

$$\boldsymbol{A_2} = \textbf{9.5833 (2.2)}$$

$$\boldsymbol{Ratio = \frac{A_2}{A_1}} \textbf{ (3.1)}$$

$$\boldsymbol{Ratio = \frac{9.5833}{8.2222}} \textbf{ (3.2)}$$

$$\boldsymbol{Ratio = 1.1655} \textbf{ (3.3)}$$

Hence, based on 3.3 EVENT-II demonstrates 1.1655X acceleration in connection establishment time which indicates that this AES cipher may confer substantive improvements in use contexts where latency figures prominently. Notably, EVENT-II demonstrates appreciably faster average connection establishment time. Given similar conditions, these results spotlight EVENT-II as a more optimized implementation of AES-dependent GCrypt handshakes. Furthermore, in conclusion the waiting time is also less for secure connections between the nodes in the WSN network.

CHAPTER 5
CONCLUSIONS AND FUTURE WORK

CONCLUSIONS

As shown by the above work,

- Using QualNet, we concluded that a discrete network simulator would be needed as the complexity of network simulating is less & so the overhead becomes manageable.
- As EVENT-I graph declines rapidly then EVENT-II, it means that EVEN-I would achieve faster processing speeds, meanwhile requiring fewer memory.
- EVENT-I achieves observationally lower decryption overhead across the timer interval.
- EVENT-II demonstrates markedly higher throughput as depicted by its steep slope.
- Latency incline for EVENT-II is lower, meaning offset between the two slopes is higher.
- Negotiation Time slope for EVENT-I is appreciably less, thus conclusion is that it is much more efficient.
- As EVENT-II demonstrates less time frame, for connection establishment & termination, it becomes the implementation of choice in those scenarios where latency remains an obstacle.

FUTURE WORK

- Expanding the scope of the work to include more cipher algorithms to demonstrate & appreciate their performance.
- Usage of a suitable library like Botan that supports TLS protocol so that overhead can be demonstrated.
- Demonstration of more key parameters to appreciate the importance of cipher algorithms & identify the best ones based on usage scenarios.

REFERENCES

[1] J. Cynthia, H. P. Sultana, M. Saroja, and J. Senthil, “Security protocols for IoT,” in *Studies in big data*, 2018, pp. 1–28. doi: 10.1007/978-3-030-01566-4_1.
[2] V. K. Kumar and P. S. Varma, "Malicious Data Injection Detection and Prediction in Wireless Sensor Network using Optimized Swarm Intelligence," 2022 IEEE 2nd International Conference on Mobile Networks and Wireless Communications (ICMNWC), Tumkur, Karnataka, India, 2022, pp. 1-5, doi: 10.1109/ICMNWC56175.2022.10031811.
[3]M. Ntebatseng, T. E. Mathonsi, D. Du Plessis and T. Muchenje, "An Enhanced Hybrid Algorithm for Wireless Sensor Networks Security in the Internet of Things," 2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME), Maldives, Maldives, 2022, pp. 1-9, doi: 10.1109/ICECCME55909.2022.9988649.
[4] Y. Guo, X. Tang and H. Sun, "A Malicious Node Detection Model for Wireless Sensor Networks Security Based on CHSA-MNDA Algorithm," *2022 IEEE International Conference on Communications Workshops (ICC Workshops)*, Seoul, Korea, Republic of, 2022, pp. 860-865, doi: 10.1109/ICCWorkshops53468.2022.9814635.
[5] B. Murugeshwari, S. A. Sabatini, L. Jose, S. Padmapriya, "Effective Data Aggregation in WSN for Enhanced Security and Data Privacy," *SSRG International Journal of Electrical and Electronics Engineering*, vol. 9, no. 11, pp. 1-10, 2022. *Crossref*, https://doi.org/10.14445/23488379/IJEEE-V9I11P101.
[6] S. Ullah, R. Z. Radzi, T. M. Yazdani, A. Alshehri and I. Khan, "Types of Lightweight Cryptographies in Current Developments for Resource Constrained Machine Type Communication Devices: Challenges and Opportunities," in *IEEE Access*, vol. 10, pp. 35589-35604, 2022, doi: 10.1109/ACCESS.2022.3160000.
[7] V. A. Thakor, M. A. Razzaque and M. R. A. Khandaker, "Lightweight Cryptography Algorithms for Resource-Constrained IoT Devices: A Review, Comparison and Research Opportunities," in *IEEE Access*, vol. 9, pp. 28177-28193, 2021, doi: 10.1109/ACCESS.2021.3052867.
[8] L. Gashi, A. Luma and A. Aliu, "A comprehensive review of cybersecurity perspective for Wireless Sensor Networks," 2022 International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), Ankara, Turkey, 2022, pp. 392-395, doi: 10.1109/ISMSIT56059.2022.9932788.
[9] S. Bi, T. Hou, T. Wang, Y. Liu, Z. Lu, and Q. Pei. 2022. DyWCP: Dynamic and Lightweight Data-Channel Coupling towards Confidentiality in IoT Security. In Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '22). Association for Computing Machinery, New York, NY, USA, 222–232. https://doi.org/10.1145/3507657.3528550
[10] B. Arunkumar and G. Kousalya, "Secure and light weight elliptic curve cipher suites in ssl/tls," *Computer Systems Science and Engineering*, vol. 40, no.1, pp. 179–190, 2022.

[11] R. Muhammad et al. “Current Lightweight Cryptography Protocols in Smart City IoT Networks: A Survey.” *ArXiv* abs/2010.00852 (2020): n. pag.

[12] A. Belova, T. Bergs, M. Bodenbenner, A. Bührig-Polaczek, M. Dahlmanns, I. Kunze, M. Kröger, S. Geisler, M. Henze, D. Lütticke, B. Montavon, P. Niemietz, Ortjohann L, M. Rudack, R. Schmitt, U. Vroomen, K. Wehrle and M. Zeng (2023). Evolving the Digital Industrial Infrastructure for Production: Steps Taken and the Road Ahead. Internet of Production. 10.1007/978-3-030-98062-7_2-1. (1-25).

[13] K. Tsantikidou, N. Sklavos, Hardware Limitations of Lightweight Cryptographic Designs for IoT in Healthcare. *Cryptography* **2022**, *6*, 45. https://doi.org/10.3390/cryptography6030045

[14] E. Rescorla. 2018. RFC 8446: The Transport Layer Security (TLS) Protocol Version 1.3. RFC Editor, USA.

PROJECT DETAILS

Student Details			
Student Name	**Aditya Tiwari**		
Register Number	190932138	Section / Roll No	26
Email Address	adityatiwari1208@gmail.com	Phone No (M)	9140630672
Project Details			
Project Title	**Analysis of different Cipher Suites on the performance of TLS 1.3 with WSNs**		
Project Duration	4 months	Date of reporting	1 May 2023
Expected date of completion of project	29 September 2023		
Organization Details			
Organization Name	**Manipal Institute of Technology, MAHE, Manipal**		
Full postal address with pin code	Tiger Circle Road, Madhav Nagar, Manipal, Karnataka 576104		
Website address	manipal.edu		
Supervisor Details			
Supervisor Name	**Dr.Chimay Rajhans**		
Designation	**Assistant Professor**		
Full contact address with pin code	Dept. of Instrumentation & Control Engineering, Manipal Institute of Technology, Manipal – 576 104 (Karnataka State), INDIA		
Email address	chinmay.rajhans@manipal.edu	Phone No (M)	9920961789
Internal Guide Details			
Faculty Name	**Dr.Chinmay Rajhans (Internal) Assistant Professor**		
Full contact address with pin code	Dept. of Instrumentation & Control Engineering, Manipal Institute of Technology, Manipal – 576 104 (Karnataka State), INDIA		
Email address	chinmay.rajhans@manipal.edu		
Co- Guide Details(if any)			
Faculty Name	**Dr.Yashwanth N (External) Assistant Professor-Senior Scale**		
Full contact address with pin code	Dept. of Electronics & Commnications Engineering, Manipal Institute of Technology, Manipal – 576 104 (Karnataka State), INDIA		
Email address	yashwanth.n@manipal.edu		