IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Ingénierie des Systèmes d'Information
Vol. 28, No. 3, June, 2023, pp. 777-790
Journal homepage: http://iieta.org/journals/isi

Advancing DNS Performance Through an Adaptive Transport Layer Security Model (ad-TLSM)

Onome B. Ohwo*, Folasade Y. Ayankoya, Oluwabukola F. Ajayi, Daniel O. Alao

Department of Computer Science, Babcock University, Ilishan-Remo 121103, Nigeria

Corresponding Author Email: ohwoo@babcock.edu.ng

https://doi.org/10.18280/isi.280329

Received: 6 March 2023
Accepted: 5 June 2023

Keywords:
domain name system, adaptive security, quality of service, DNS security, security architecture, cybersecurity, performance issue

ABSTRACT

The present study endeavors to enhance DNS over TLS performance via the development of an Adaptive Transport Layer Security Model (ad-TLSM). DNS over TLS, which employs TLS encryption to safeguard communication between clients and DNS recursive resolvers, suffers from performance issues that pose significant challenges. In response to these issues, the ad-TLSM has been designed to boost DNS performance by integrating a monitoring mechanism for real-time observation of the DNS recursive resolver. During the TLS handshake, crucial data, including throughput, CPU load, and the active cryptographic algorithm, are meticulously monitored and documented. This data forms the foundation for an adaptive strategy, which facilitates intelligent security adaptation during runtime, based on the prevailing conditions between the client and the server at the time of secure connection establishment. The performance evaluation of the ad-TLSM demonstrated that the DNS recursive resolver experiences excessive load while employing AES-GCM 256. However, it was found capable of managing an additional 15%-25% requests per second when ChaCha20 was implemented. These findings led to the formation of an adaptive strategy that effectively alleviates CPU load by adjusting the security level, thereby ameliorating the overall performance. In summary, the ad-TLSM surpasses existing models in latency performance and can be employed to improve performance, while satisfying quality of service constraints. This research represents a significant step towards the development of more efficient and secure DNS services.

1. INTRODUCTION

In the early stages of the Internet, navigation proved challenging, with messages manually transmitted from one computer to another [1]. This process necessitated a comprehensive understanding of the Internet's architecture from source to destination. However, the introduction of the Berkeley Internet Name Domain (BIND) program in 1984 at the University of California, Berkeley, revolutionized Internet navigation by establishing a decentralized mechanism for naming Internet-connected nodes based on hierarchical records [1]. This innovation eliminated the need for each node to maintain a complete routing database and introduced the concept of mapping data in the namespace to an IP address.

Today, two primary namespaces are utilized by the Internet: the Internet Protocol (IP) address spaces and the domain name space [2]. Whereas an IP address serves as a numerical label for each device on a computer network using the Internet Protocol for communication, the Domain Name System (DNS) performs translation services between itself and the address spaces, maintaining the domain name hierarchy [2]. The DNS provides a distributed, fault-tolerant global directory service, vital for Internet operations. By delegating domain name assignment and mapping those names to Internet resources to authoritative name servers for each domain, the DNS effectively circumvents the need for a single, large centralized database.

The DNS protocol specifically articulates data structures and data transmission exchanges. A host's domain name is assembled from individual group names, comprising strings separated by dots. The highest authority is the root domain (Top Level Domains (TLDs)), which is subdivided into Generic Top-Level Domains (gTLDs) (e.g., edu, com, net, and mil) and Country Code Top-Level Domains (ccTLDs) (e.g., .ng, .se, .us, .ca) [2]. As such, the DNS is instrumental for the reliable and trustworthy operation of the Internet, with disruptions in its operation potentially causing significant impact on provided services and the global Internet at large.

Regrettably, breaches of DNS security have been attempted over the years, resulting in various attacks [3]. The existing DNS recursive resolver lacks adequate security mechanisms for data confidentiality, availability, and integrity, making it susceptible to hackers and attackers who could falsify DNS records and redirect genuine users to malicious domains [4]. To mitigate these challenges, new protocols such as DNS over Hypertext Transfer Protocol Secure (DoH), DNS over Transport Layer Security (DoT), and DNS over Quick UDP Internet Connections (DoQ) have been introduced [5].

DNS encryption, typically achieved through the encryption of the content of queries and responses (between clients and recursive resolvers) using cryptographic techniques in an upper layer protocol, has the potential to maintain user privacy against attacks. However, the introduction of encrypted transports incurs new performance costs, including overhead associated with Transmission Control Protocol (TCP) and TLS connection establishment, and additional application-

layer overhead [6]. These performance costs were not well understood initially [6]. Therefore, numerous researchers have probed into how encrypted transports for DNS impact the end-user experience [7-10]. Their findings suggest that DNS queries are generally slower with encrypted transports, and these protocols begin to experience difficulties on networks with sub-optimal performance due to their connection and transport overhead.

The relative costs and benefits of a particular DNS transport protocol and its implementation for DNS query response times are heavily influenced by the underlying network conditions. Therefore, the features and ideologies of adaptive security, a new architectural approach, warrant consideration. Adaptive security techniques, akin to risk management, strive to manage risk and meet the required Service Level Agreement (SLA). They aim to circumvent the impact and degree of potential threats in a timely manner [11].

The implementation of an adaptive security approach can be achieved using currently available technologies [11]. Besides upholding SLAs, adaptive security seeks to maintain integrity, foster trustworthiness, and provide assurance, inspiring confidence in data and processing resources, ensuring trustworthiness, reliability, availability, and operation within satisfactory parameters. What distinguishes adaptive security architecture from existing advanced practices is its design to guard against identified threats and anticipate unidentified threats in a fashion resembling the human immune-response system.

Given the inherent performance issues and the need to ensure complete security service, it is necessary to improve DNS security by deploying an adaptive solution. This solution should enhance DNS performance with an increasing client base and satisfy diverse usage patterns.

1.1 Statement of the problem

The Domain Name System (DNS) is integral to the functionality of the Internet, offering global distributed directory services. However, it has been found that the DNS recursive resolver lacks adequate security mechanisms for data confidentiality, availability, and integrity. Several security measures, including DNS over TLS (DoT), DNS over HTTPS (DoH), and DNS over QUIC (DoQ), have been developed to secure communications with the DNS recursive resolver. While these techniques have indeed bolstered DNS security, they have also introduced significant performance costs, with overall failure rates fluctuating between 1.3% and 39.4%.

Further, research has revealed that the use of cryptography in Transport Layer Security (TLS) can negatively impact performance. Both asymmetric and symmetric cryptographic primitives are employed by TLS, with the former requiring more memory. Symmetric cryptography involves the use of the Advanced Encryption Standard (AES) – typically fast and efficient in hardware implementation – and ChaCha20-Poly1305, which excels in software implementation. More importantly, DNS query processing may necessitate high CPU usage due to the cryptographic operations performed by TLS.

Previous efforts to address this performance problem have utilized various techniques such as an authoritative DNS server (ADNS) and Private DNS over TLS (PDOT). While the ADNS approach improved DNS performance, this was only true for policies on resource records with smaller authoritative Time-To-Live (ATTL). Larger ATTL values resulted in performance issues. On the other hand, PDOT focused on privacy, taking performance into account. However, applications needing functionality not available within the Trusted Execution Environment (TEE) had to switch to the non-Trusted Execution Environment, leading to overhead associated with TEE Call-in/Call-out. Despite addressing privacy concerns, performance issues persist.

Given that current security techniques do not provide satisfactory performance, this study aims to develop an Adaptive Transport Layer Security Model (ad-TLSM) to enhance DNS throughput.

1.2 Justification of the study

Considering the biased and abstract factors that influence security decisions, threats to the system remain a significant concern. Therefore, it's vital to continue efforts to mitigate these threats by enhancing DNS security, using Adaptive TLS to provide optimum security while maintaining Quality of Service (QoS) constraints.

A specific constraint involves the effective management of available DNS resources without causing congestion or violating client QoS constraints. A security measure capable of utilizing available DNS resources to maintain high security levels offers the potential for timely and fine-grained security control.

2. LITERATURE REVIEW

2.1 The domain name system security

DNS security, a strategy aimed at safeguarding the DNS infrastructure from cyberattacks, seeks to maintain its robust and efficient performance. A successful DNS security strategy incorporates a blend of overlapping defenses, which may include the implementation of redundant DNS servers, the application of security protocols such as DNSSEC, and the insistence on comprehensive DNS logging. As with many Internet protocols, the DNS infrastructure was not originally designed with an emphasis on security, resulting in several inherent design limitations. These limitations, when coupled with technological advancements, render DNS servers vulnerable to a wide spectrum of attacks such as Denial of Service, spoofing, amplification, and private data interception.

Given that DNS forms a crucial component of the majority of web requests, it presents an attractive target for attackers. DNS attacks are often executed in conjunction with other cyberattacks, thereby diverting the attention of security teams from the primary target. It is imperative for organizations to swiftly neutralize DNS attacks to avoid being overly preoccupied, thereby leaving them vulnerable to simultaneous attacks from other vectors.

Privacy constitutes another significant issue within DNS security. The lack of encryption for DNS queries, even when the client uses a DNS resolver that does not log their activities, means that these queries traverse the Internet in plaintext. This lack of privacy not only jeopardizes security but also, in certain contexts, infringes upon human rights. The visibility of DNS queries simplifies the task for governments seeking to censor the Internet and for attackers aiming to monitor users' online activities.

Research conducted by Wessels [12] estimated that there are approximately 11.7 million public DNS servers on the Internet. Of these, around 52% permit arbitrary queries due to

improper configuration, while approximately 33% allow denial of service attacks or cache poisoning attack, given that their authoritative name servers reside on the same network.

Additionally, the sophistication of attacks targeting DNS has increased, complicating detection and control processes. For instance, the Fast Flux attack swiftly modifies DNS information about the domain to delay or evade detection. Similarly, the conficker worm attacks [13], also known as SQL Slammer, leverage domain names to make network attacks resilient against detection and shutdown. To assist in identifying attacked domains, the Internet Corporation for Assigned Names and Numbers (ICANN) has compiled a list of domains that could potentially be used in such attacks across each Top-Level Domain.

2.2 Adaptive security

Security threats are counter-productive to the functionality, performance, availability, and integrity of Information Technology systems. The goal is to decrease possible security threats to the level wherein Service Level Agreements (SLAs) can still be satisfied; as well as the risk management concept. By timely prevention of an attack, adaptive security attempts to decrease the effect and degree of possible threats.

2.2.1 Characteristics of an adaptive security

de Castro and Timmis [14] identified some characteristics of adaptive security useful in Information Technology systems to include preventing attacks, contain the impact of attacks, and timely responds to attacks. Other characteristics includes:

1) **Self-identity:** This involve separating and removing what does not belong in line with the governing security policy. This comprises of support for systems communication and information exchange on attacks and threats, preventive measures, security guidelines and policies, and trust relations amongst 3rd party systems.
2) **Diversity:** This shows itself via diverse control mechanisms such as compartmentalization. This can be achieved through operating system (OS) virtualization or Trusted Platform Module (TPM)-based hardware trust anchors.
3) **Autonomy:** There are different components that controls the security system function autonomously to prevent attacks and threats. This is required for security and integrity control devices to autonomously function in responding to attacks and threats.
4) **Multi-layered:** This can be likened to the idea of "defense-in-depth", where in a properly designed security architecture preserves and employs multiple security measures to subdue the hazard of a compromised single measure.
5) **Resilience:** The effectiveness of a security system can be decreased by various factors. By maintaining a level of resilience, it can continuously recognize and prevent attacks in spite of reduced capacity.
6) **Anomaly detection:** This involves supporting the ability to detect and prevent any abnormal behavior or known threats automatically.

2.2.2 Adaptive security capabilities

New approaches in information security techniques have tried to mimic adaptive system so as to be able to fine-tune to continuously developing and varying security threats. The core of Adaptive Security is to serve as the immune system of a system. This is realized by developing an Adaptive Security aimed at containing active attacks and neutralizing possible threat vectors [15]. Adaptive security is defined based on four security abilities:

1) **Preventive capability:** A set of rules, guidelines and policies that prevents an attack from being successful. Thus, information is protected from illegal alteration, ruin, or exposure, whether unintentionally or deliberately.
2) **Detective capabilities:** These are controls (including logging of events) intended to recognize attacks, that have eluded the preventive procedures, and decrease the attack magnification. Thus, this provides an outlook into malicious actions, violation and attacks.
3) **Retrospective capabilities:** These provide a way of reducing the attack area, the attack rate and recovery time. Thus, this provides the procedures necessary to take suitable action in responding to diverse cybersecurity events.
4) **Predictive capabilities:** These allows attack predictions, security trends analysis and changing to a proactive security from reactive security. Thus, security intelligence is achieved from internal and external events monitoring to recognize attackers, their purposes and approaches before the appearance of the attacks.

2.3 Transport layer security

Transport Layer Security (TLS) is used to secure communications via a consistent transport protocol (such as TCP/IP) between a web server and client using a cryptographic protocol. This allows client-server applications to communicate via a public network, while preventing messages from eavesdropping, altering, and counterfeiting. TLS provides the following security characteristics such as confidentiality, integrity, authentication and non-repudiation of messages. The TLS protocol goals are extensibility, cryptographic security, interoperability and efficiency. There are two main components of TLS protocol: **Handshake protocol** and **Record protocol**. Through the handshake, to meet the mentioned security characteristics, algorithms are selected based on availability to both the client and server. This is generally known as TLS negotiation, with the ensuing secured connection called a session. Also, an established TLS session can be renegotiated at the decision of the client or server [16]. Figure 1 depicts the logical description of Transport Layer Security (TLS) architecture.

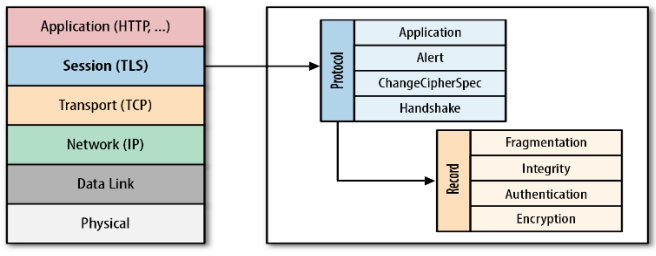

Figure 1. Transport layer security architecture [17]

2.3.1 DNS over TLS

This study looks to investigate Domain Name System security such as DNS over TLS (DoT) to find some fact that helps to analyze the problems and capitalizing on them to develop the proposed solution. When initiating a TLS handshake, the client and the DNS resolver negotiate and

agree upon a cryptographic algorithm to be used for encrypting and decrypting the data exchanged during the communication. This algorithm is typically selected from a set of available options supported by both the client and the DNS resolver. Figure 2 shows the typical TLS 1.3 handshake process.

During lookups, the client establishes a TCP connection to a selected DoT port TCP/853 on the DoT-enabled resolver. Next, a TLS connection will be established via a typical TLS handshake process to exchange their cryptographic keys. TLS 1.3 uses the "key_share" and "pre_key_share" parameters in the "ClientHello" handshake message for encryption purposes. The "key_share" parameter is used to exchange the end-point's public key share required to generate secret key at the remote end-point. The "pre_key_share" parameter specifies the index of the presently used shared key for encryption in the list of negotiated shared keys. Then, TLS 1.3 encrypts the server security certificate. Upon successful establishment of the TLS session, the client is able to perform TLS-encrypted DNS lookups through the DoT port TCP/853 on the resolver side. Depending on the configurations of clients and servers, the TLS connections may remain open for further DNS lookups, reducing latency (that is, preventing additional TCP/TLS handshakes for subsequent requests).

The cryptographic algorithm plays a vital role in ensuring the confidentiality, integrity, and authenticity of the DNS query transactions. It determines how the data is encrypted, decrypted, and authenticated, providing protection against eavesdropping, tampering, and spoofing attacks. By using TLS to secure DNS query transactions, sensitive information, such as the domain names being queried, is encrypted and protected from unauthorized access. This helps to prevent malicious actors from intercepting and manipulating DNS queries, thereby safeguarding the privacy and integrity of the communication. TLS is employed in DoT to secure DNS query transactions by selecting an appropriate cryptographic algorithm that ensures the confidentiality, integrity, and authenticity of the exchanged data.

Though DoT is a viable approach for DNS encryption, it faces several performance challenges that may hinder its usage, such as high failure rate due to timeouts (that is, no response within 5 seconds), head-of-line blocking, and computational overhead [7-10].

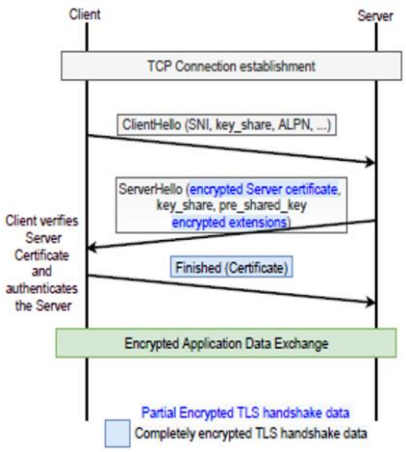

Figure 2. A typical TLS 1.3 handshake [18]

2.3.2 Security services of cryptography

The essential security objectives of cryptography are to offer the accompanying vital services. According to Willian [19], these vital services are:

1) **Confidentiality:** It is a security service that keeps the data from an unapproved individual. Cryptography ensures that DNS data remains confidential by encrypting it. With encryption, sensitive information, such as the domain names being queried, is transformed into ciphertext that is unreadable without the corresponding decryption key. This prevents unauthorized parties from accessing and understanding the contents of DNS queries, protecting the privacy of users.
2) **Data integrity:** It is a security service that deals with perceiving any alteration to the information. Cryptographic algorithms, such as message authentication codes (MACs) and digital signatures, ensure the integrity of DNS data. MACs provide a way to verify that the data has not been tampered with during transmission, as any modifications to the data would result in an invalid MAC. Digital signatures, on the other hand, provide a mechanism for verifying the authenticity and integrity of DNS responses, ensuring that they have not been altered by malicious actors.
3) **Authentication:** This gives the identity of the message originator. Cryptography enables the verification of the authenticity of DNS data. Digital signatures, for instance, can be used to verify that a DNS response has been generated by a trusted DNS server and has not been tampered with during transit. This prevents attackers from impersonating DNS servers and providing false or malicious responses.
4) **Non-repudiation:** It is a security service which guarantees an entity cannot reject the ownership of a prior action. Cryptographic mechanisms, such as digital signatures, provide non-repudiation, which means that a party cannot deny their involvement in generating a particular DNS response. This is important in DNS security as it ensures accountability and prevents malicious actors from denying their actions.

2.3.3 Review of related works

It is unknown how much information may be gleaned via traffic analysis on DoT communications, despite the fact that DoT is meant to stop on-path adversaries from observing and manipulating the victims' DNS requests and responses. A DoT fingerprinting technique was proposed by Houser et al. [20] to examine DoT traffic and identify whether a user has visited websites that are of interest to adversaries. When DNS messages are not padded, the suggested approach can detect DoT traffic for websites with a false negative rate of less than 17% and a false positive rate of less than 0.5%. Furthermore, it was demonstrated that even when DoT messages are padded, information leakage is still feasible.

For five months at the start of 2021, this study tracked the adoption of DoH (DNS over HTTPS), DoT (DNS over TLS), and DoQ (DNS over QUIC) by three separate enterprises with worldwide reach. García et al. [5] analyzed the overall numbers, requests made per user, and traffic seasonality in order to determine the potential adoption trends. It was concluded that, despite increasing in 2020, there was statistically substantial evidence that the average volume of Internet traffic for DoH, DoT, and DoQ remained constant throughout the first five months of 2021. However, we discovered that the number of DoH servers that are available for use has increased by a factor of 4. These findings indicate that although the volume of encrypted DNS is not now

increasing, there may soon be an increase in connections to unknown DoH servers for both good and bad intentions.

Although DNS over TLS (DoT) was established as an addition to the DNS protocol in 2016, little research has been done on how it performs. Research by Doan et al. [7] used 3.2k RIPE Atlas probes installed in home networks to quantify DoT from the edge and compare its adoption, dependability, and response times to DNS via UDP/53 (Do53). It was found that open resolvers are becoming more supportive of DoT. DoT is still only supported by regional resolvers. However, the reliability of DoT decreased while failure rates rose. Response times, according to DoT, are getting longer. Most failures occur due to timeout, that is no response within 5 seconds, which was suspected to be as a result of the intervening middleboxes on the path that blackhole the connections by dropping packets destined for port 853.

Using Transport Layer Security (TLS) to secure DNS communication has become popular recently. But at least two significant problems continue: (1) How can DNS-over-TLS endpoints be authenticated by clients in a scalable and extendable way? (2) How can clients be confident that endpoints will act as expected? A revolutionary Private DNS-over-TLS (PDOT) architecture was proposed by Nakatsuka et al. [21]. A DNS Recursive Resolver (Rec Res) that works in a Trusted Execution Environment (TEC) is part of PDOT. The study offered an open-source PDOT proof-of-concept implementation and empirically showed that its throughput and latency matched those of the well-known Unbound DNS-over-TLS resolver. The functionality that is available to code that runs within them is constrained, which presented the following major difficulties throughout the design process of PDoT. It also has a little quantity of memory. And applications must move to the non-TEE side if they need functionality that is not provided by the TEE.

Security has been largely handled by Transport Layer Security (TLS). However, the initial handshakes of vanilla TLS send information about the sort of service being accessed in plain-text, possibly disclosing user behavior and jeopardizing privacy. The "Encrypted ClientHello" (ECH), is a TLS 1.3 extension that Khandkar et al. [18] suggested to address the privacy concerns in TLS 1.3 by masking all of the information that may potentially disclose the service type. This study showed that the Encrypted Client Hellos (ECH) TLS 1.3 enhancement does not deliver the desired privacy. This is partly due to the fact that many services continue to use TLS 1.2, whilst ECH only supports TLS 1.3. The limited switchover to TLS 1.3+ECH can fall short of protecting against malicious attacks that throttling/blocking particular internet services, as well as failing to fulfill the stated goals of privacy and anonymity.

The impact of Do53, DoT, and DoH on query response times and page load times was measured, in the study by Hounsel et al. [8], from five different worldwide perspectives. This study discovered that although while DoH and DoT response times are often higher than Do53, both protocols can outperform Do53 in terms of how quickly pages load. However, significant packet loss and latency are introduced when network conditions deteriorate.

Böttger et al. [9] examined the DNS-over-HTTPS environment in this study, paying particular attention to the cost of the added security. And to demonstrate the gains DoH offers over its predecessor, DoT, they examined various secure DNS protocols. It was then determined that head-of-line-blocking affects DoT and DoH/1. This difference in behavior may (at least partially) explain why DoH/2.0 gained traction more quickly than DoT.

According to research by Jonglez [10], DNS-over-TCP performance with few clients is comparable to DNS-over-UDP with only a 30% lag. Performance of DNS-over-TCP decreases as the number of clients rises and stabilizes at a 75% slowness. The performance profile for DoT is comparable to TCP, although there is a 30% to 45% speed impact. However, performance suffers noticeably as the number of clients rises for both TCP and TLS. This was thought to be a result of the kernel's need to manage a large number of TCP connections concurrently.

In this research, Shang and Wills [22] presented a novel approach called an authoritative DNS server (ADNS), that can piggyback resolutions for future queries as part of the response message for an initial query. This exploits the relationships among domain names to improve the cache hit rate for a local DNS server. The approach improves the cache hit rate as well as reducing the total queries and responses. Trace-based simulations show more than 50% of cache misses can be reduced in the best case while straightforward policies, using frequency and relevancy data for an ADNS, reduce cache misses by 25-40% and DNS traffic by 20-35%. However, these percentages improve if focused on the resource records policy with smaller authoritative Time-To-Lives.

Khandkar and Hanawal [23] presented a method called Encrypted TLS/SSL Handshake, to mask the server host identity by encrypting the Server Name Indicator (SNI). This simple method completes the SSL/TLS connection establishment over two handshakes-the first handshake establishes a secure channel without sharing SNI information, and the second handshake shares the encrypted SNI. This method makes it mandatory for fronting servers to always accept the handshake request without the SNI and respond with a valid SSL certificate. However, specific changes in the handshake parameter setting limit the operational viability of the solution.

In this paper, the present state of DNS security architecture was evaluated, and saw clearly that existing DNS security architectures are insufficient to secure DNS data transiting over the network; considering the growing cybersecurity landscape. On this note, Alao et al. [24] propose the need and adoption of a security architecture named Adaptive Security Architecture. Adaptive Security Architecture is devised to guard against identified threats, and anticipate unidentified threats in a manner similar to the immune-response system of human. Basically, mimicking nature’s biodiversity as the fundamental means of effective attack responses. Finally, we conclude by an analysis to prove the need to improve DNS security architecture.

Hounsel et al. [25] studied the performance of encrypted DNS protocols and conventional DNS from thousands of home networks. They found that clients do not have to trade DNS performance for privacy. For certain resolvers, DoT was able to perform faster than DNS in median response times, even as latency increased. Also, there was significant variation in DoH performance across recursive resolvers. Based on these results, it was recommended that DNS clients (such as, web browsers) should periodically conduct simple latency and response time measurements to determine which protocol and resolver a client should use. However, no single DNS protocol nor resolver performed the best for all clients.

Emerging protocols such as DNS-over-HTTPS (DoH) and DNS-over-TLS (DoT) improve the privacy of DNS queries

and responses. While this trend towards encryption is positive, deployment of these protocols has in some cases resulted in further centralization of the DNS, which introduces new challenges. In particular, centralization has consequences for performance, privacy, and availability. Towards this goal of increased de-centralization and improved flexibility, Hounsel et al. [26] presents the design and implementation of a refactored DNS resolver architecture that allows for de-centralized name resolution, preserving the benefits of encrypted DNS while satisfying other desirable properties, including performance and privacy. The researchers argued for a re-decentralization of the DNS, considering users may prefer one distribution strategy over another. They explored various alternative strategies for resolving and distributing encrypted DNS queries. However, this research provides only a starting point as a proof-of-concept.

In the DNS resolution process, packet losses and ensuing retransmission timeouts induce marked latencies: the current UDP-based resolution process takes up to 5 seconds to detect a loss event. Jonglez et al. [27] explored persistent DNS connections based on TCP or TLS as a possible solution to this problem. Experimentation showed that persistent DNS connections significantly reduces worst-case latency. Thus, a large-scale platform was leveraged to study the performance impact of TCP/TLS on recursive resolvers. The results showed that off-the-shelf software and reasonably powerful hardware can effectively provide recursive DNS service over TCP and TLS, with a manageable performance hit compared to UDP. However, switching to TCP or TLS has an impact on the load of the recursive resolver, which is significant, especially with a large number of concurrent connections.

2.3.4 Limitations of related works

From the literatures reviewed, two research works focused on improving performance of DNS, by piggyback resolutions for future queries as part of the response message for an initial query, and a DNS Recursive Resolver (Rec Res) that operates within a Trusted Execution Environment (TEE) respectively. The outcome of the research works was encouraging but are limited by the additional performance overhead. The first research used an authoritative DNS server to improving DNS performance; however, this is only true for the policies on resource records with smaller authoritative Time-To-Live (ATTL). This means with larger ATTL, performance issues arise. Also, the last research used a novel Private DNS-over-TLS (PDoT) architecture. However, applications requiring functionality that is not available within the TEE must switch to the non-TEE side, this introduces TEE Call-in/Call-out Overhead.

It can be concluded that attempt at improving performance has resulted in additional performance issues. Therefore, the primary purpose of this research is to develop an adaptive Transport Layer Security Model (ad-TLSM) that adapts security in light of DNS contextual features to provide optimum performance; whilst preserving Quality of Service (QoS) constraints. The task is to effectively manage available DNS resources without congestion or breaching client's QoS constraints.

3. METHODOLOGY

This section presents a description of the methodology used to satisfy the objective of this research work. These includes several processes, procedures and architectural structures adopted within the research.

Figure 3 represents a typical DNS over TLS session during a DNS server request-response processing cycle. It uses a Transport Layer Security (TLS) layer under the Transmission Control Protocol (TCP) transport layer to encrypt the communication channel between the user and DNS recursive resolver thus securing queries and responses.

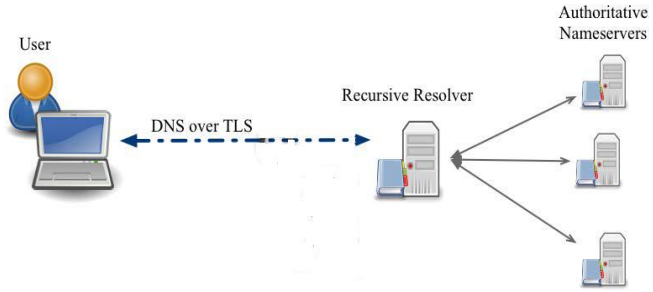

Figure 3. Research design [20]

3.1 Proposed adaptive TLS model (ad-TLSM)

The proposed adaptive TLS method incorporates a feedback loop mechanism to enhance security adaptation. It relies on measuring contextual features such as logging, monitoring, error detection, and information security to make informed security decisions. These measurements are compared with offline data to determine the impact of security adaptation. In the case of DNS recursive resolver, a monitoring mechanism is used to measure contextual features like throughput and CPU load of the DNS recursive resolver during the TLS handshake. This information forms the basis for an adaptation strategy that enables smart security decisions based on real-time client-server conditions. Figure 4 shows the ad-TLSM handshake process.

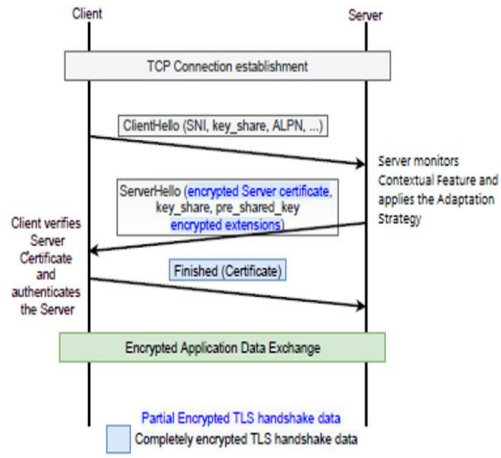

Figure 4. The ad-TLSM handshake

The adaptive TLS handshake involves the client generating a "ClientHello" request, the server monitoring contextual features, and responding with a partially encrypted "ServerHello" message. If successful, a TLS-encrypted DNS lookup can be performed. Through this tight coupling of security and runtime monitoring, the proposed solution ensures improved performance of the DNS recursive resolver. By monitoring throughput and encryption and applying this information to an adaptation strategy, the system can balance security and performance based on workload and capacity.

Regular monitoring and adaptation ensure efficient and effective system operation.

The primary objective of the ad-TLSM is to enable security adaptation in systems where all measurable contextual features, such as logging, monitoring, error detection and correction, and information security, are present. By leveraging these contextual features, the method aims to dynamically adjust security measures at runtime, ensuring optimum performance while preserving Quality of Service (QoS) constraints. The concepts and components include:

1) **Feedback Loop:** The ad-TLSM incorporates a feedback loop mechanism, where the impact of security adaptation serves as a response to the monitoring component. This connection between contextual features and security adaptation enables runtime automated security actions to be taken, based on real-time data analysis and comparison.
2) **Runtime Monitoring:** The ad-TLSM includes a monitoring mechanism that allows for the real-time monitoring of the DNS recursive resolver during the TLS handshake. Contextual features such as throughput and CPU load are measured and recorded to provide a basis for the adaptation strategy.
3) **Adaptation Strategy:** The adaptation strategy utilizes the monitored contextual features to make smart security decisions at runtime, based on the present client-server conditions during the establishment of a secure connection. This strategy aims to balance the need for security with other factors such as performance and resource utilization. Figure 5 shows the process flow diagram of the ad-TLSM.

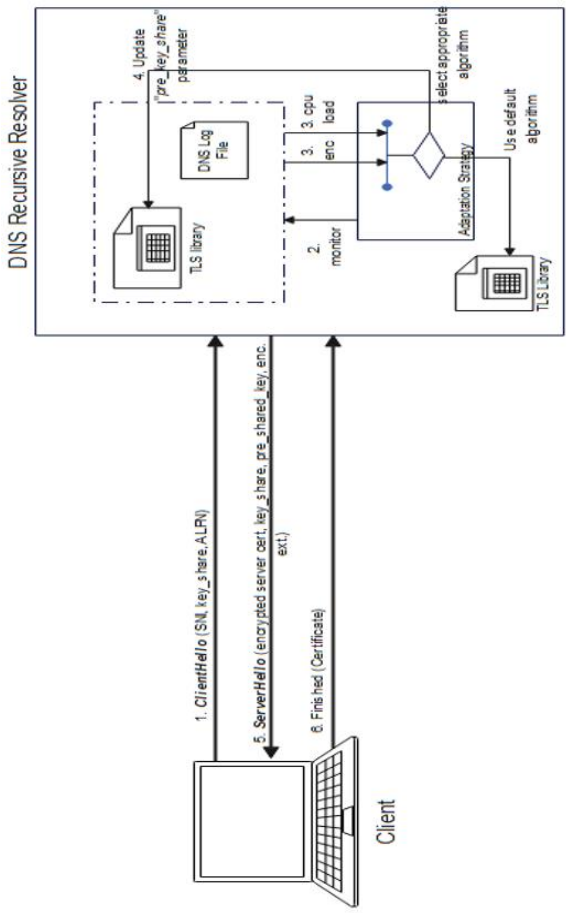

Figure 5. The process flow diagram of the ad-TLSM

The ad-TLSM follows a step-by-step procedure to dynamically adjust security measures based on real-time monitoring and contextual features. The client creates a TCP connection to the DoT-enabled resolver's TCP/853 port during lookups. The ad-TLSM will then be used to establish a TLS connection as follows:

1. The client creates "ClientHello" requests by using the data from the content-public server's URL. This data, which is sent in plain text, contains the server's name (SNI) that the client wants to connect to.
2. The server's monitors the contextual feature (such as, throughput and CPU load) using the Log file and the presently used cryptographic algorithm in the list of negotiated shared keys provided by the "pre_key_share" parameter in the TLS library.
3. This information is then applied to the Adaptation strategy for smart security decision.
4. If the request is approved, the "pre_key_share" parameter is updated.
5. The server replies with a "ServerHello" message that is only partially encrypted and contains the server certificate and other details pertaining to encryption. To authenticate the server connection, the client validates the hostname in the server certificate it receives.
6. Finally, after the TLS session has been successfully established, the client can do TLS-encrypted DNS lookups through the DoT port TCP/853 on the resolver side.

This permits the tight coupling of security with runtime monitoring of the DNS recursive resolver, allowing security adaptation as the environment or security requirements varies. Note that, should step 2 fail, it reverts back to using the default cryptographic algorithm for encryption. This ensures improved performance of the DNS recursive resolver using the ad-TLSM.

3.2 Security-performance evaluation parameter

A security-performance evaluation will be conducted on the ad-TLSM to demonstrate the practical utility in adapting security and performance at runtime. The evaluation will follow this process:

1) Security Algorithm Performance: The performance overhead and security level of typical cryptographic algorithm implementations will be investigated by measuring their throughput when subjected to same data size.
2) Use-case Scenario: To examine how the DNS recursive resolver performs utilizing different cryptographic algorithm under varying client-server conditions.
3) Based on step 2, the indirect monitoring adaptation strategy was used. This strategy relies on examining the DNS recursive resolver throughput indirectly, by recognizing that increased throughput suggests increased CPU load.
4) Such understandings will offer the foundation for a security-performance adaptation strategy. This will all the more essentially use the accessible DNS recursive resolver assets to further improve security without over-burdening the DNS recursive resolver and upsetting client QoS requirements. This grants clever security transformation at runtime in view of current client-server conditions. Then, the proposed solution will be compared to existing DNS over TLS (DoT) technique using the throughput and latency metrics.

3.2.1 Evaluation metrics

A security-performance evaluation will be conducted on the ad-TLSM to demonstrate the practical utility in adapting security and performance at runtime. The evaluation will follow this process:

1) **Throughput:** This refers to the number of DNS requests that can be processed per second by a DNS recursive resolver. It is generally measured in requests per second (req/s). The higher the throughput, the more efficiently the DNS server can handle incoming requests, resulting in faster response times and a better user experience. For example, a basic DNS recursive resolver running on a low-end server with limited processing power and network bandwidth may have a throughput of a few hundred requests per second. In contrast, a high-performance DNS recursive resolver running on a powerful server with multiple CPUs and high-speed network connections can handle thousands or even millions of requests per second.
2) **Latency:** This refers to the time it takes for a DNS query to receive a response from the DNS recursive resolver. It is measured in seconds and directly impacts the speed and responsiveness of website access. If a DNS query experiences high latency, it means that the resolver took a longer time to provide a response, which can result in slower website load times or even complete inability to access a website if the DNS lookup times out.

In conclusion, the ad-TLSM presents a systematic approach to enhance DNS security while considering runtime monitoring and adaptation. The objective of this methodology is to dynamically adjust security measures based on real-time contextual features of the DNS recursive resolver. By monitoring factors such as throughput and CPU load, the adaptation strategy intelligently adapts security measures during the TLS handshake process.

The significance of this methodology lies in its ability to balance security and performance, preserving Quality of Service (QoS) constraints. By dynamically selecting cryptographic algorithms and making security-related adaptations, the proposed method ensures optimum security while improving DNS performance.

The outcomes of applying this methodology have shown promising results. The evaluation demonstrated that by adapting the security measures, such as utilizing different cryptographic algorithms, the DNS recursive resolver could handle a higher number of requests per second. This translates to improved performance and reduced CPU load.

The implications of the applied methodology are far-reaching. It addresses the limitations of existing security measures in DNS over TLS and provides a framework for adapting security at runtime. This methodology enables systems to respond effectively to varying client-server conditions, enhancing both security and performance. The ad-TLSM offers a structured and systematic approach to optimize DNS security. By incorporating real-time monitoring, adaptation strategies, and runtime security actions, it ensures a fine balance between security and performance, ultimately improving the overall effectiveness of DNS systems.

4. IMPLEMENTATION AND EVALUATION

4.1 Implementation

The experiments were conducted on a Windows 10 system with an Intel Core i5 processor, 16GB RAM, and Apache 2.4. OpenSSL 1.1.1 was utilized by both Apache's TLS module and the proposed solution module. This investigation focuses on the design of the Adaptive Transport Layer Security Model (ad-TLSM) for DNS recursive resolver. Apache's flexibility as an open-source web server makes it a suitable platform for implementing the ad-TLSM design. The ad-TLSM allows for runtime security adaptation based on the contextual features of the DNS recursive resolver. The modular architecture of Apache enables the integration of modules, such as the TLS module, which configures TLS session security using the SSL Cipher Suite directive. Similarly, the proposed solution acts as a module that can be added to existing Apache installations without requiring any modifications to Apache or TLS code. By leveraging these features, security can be adapted at runtime, providing flexibility and enhancing the overall security of DNS sessions. Figure 6 shows the ad-TLSM.

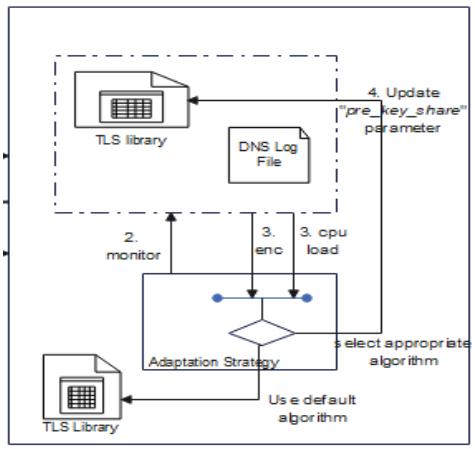

Figure 6. The ad-TLSM

TLS Handshake: To securely connect to a server, a hostname (*www.example.com*) and a port (*433*) is required. Based on the implementation, the certificate authority was also added. For instance:

openssl s_client-connect www.example.com: 443-CAfile c:/cacert.pem

Figure 7 to Figure 9 show the TLS handshake using AES-GCM 256, ChaCha20 and AES-GCM 128 respectively.

Once you type the command, diagnostic output is displayed followed by an input prompt. Because when interacting with a server, a request is submitted. Now, the TLS communication layer is working, as request is sent to the server and response received. For a better look at the certificate, Figure 6 to Figure 8 show a self-explanatory information about the TLS Handshake using the available cryptographic algorithm respectively:

The vital information is the protocol version (TLSv1.3) and cipher suite utilized (TLS_AES_256_GCM_SHA 384). It also showed the server has allotted a session ID and a TLS session ticket**.**

The security policy in the proposed solution consists of condition and SSL Cipher Suite pairings that trigger the renegotiation of the client's security session if the requirements are met and the current session security is not dependent on the chosen SSL Cipher Suite. The conditions, created using the Require directive, are complex Boolean expressions utilizing CGI, Apache, and TLS variables. These conditions are checked in order, and the first matching condition is selected. The Require env provider allows access control based on the

existence of an environment variable. By using mod setenvif directives, environment variables can be set based on client request characteristics like User-Agent or HTTP request header fields. This flexibility enables the module to be applied in various applications and data settings at different levels of abstraction.

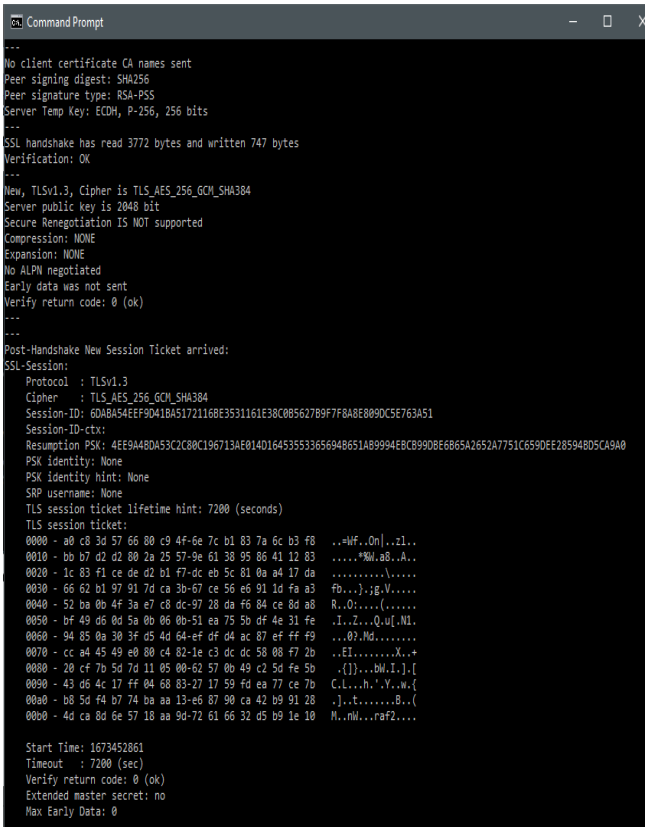

Figure 7. TLS handshake using AES-GCM 256

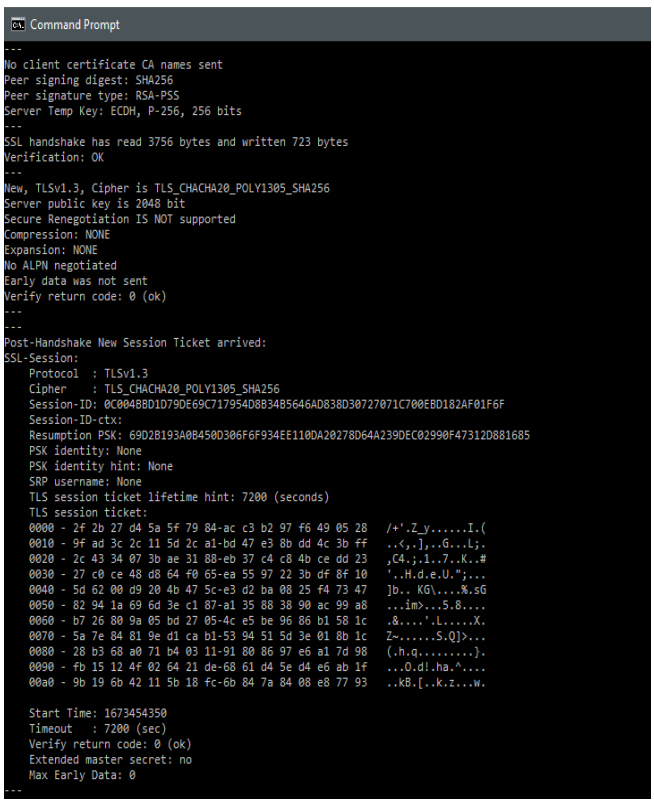

Figure 8. TLS handshake using CHACHA20

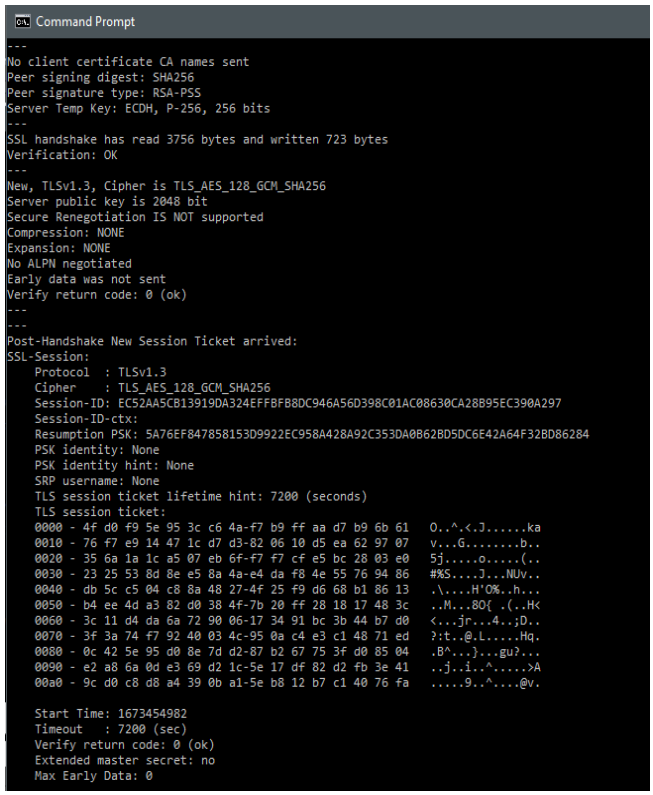

Figure 9. TLS handshake using AES-GCM 128

The ad-TLSM registers hooks with Apache, seamlessly integrating with existing TLS hooks and assuming control of the TLS renegotiation function. The TLS's SSL Cipher Suite directive is stored for future application when the Security Policy is empty or none of its requirements match the client's request. This ensures that existing or previously applied security remains in control of session security.

A load generator was used to reproduce multiple client requests to a server. New user sessions are created with the server at a specific rate (λ), according to a Poisson process with some mean. Multiple requests (λ_n) can be made by users within each session, according to a typical request rate based on a Poisson process with some mean. Mean user arrival and request rates can be altered during an experimentation, to simulate genuine user load. Each user's session maintains an exclusive security state. During a session or its creation, exclusive user information stored permits the server to adjust the security for users. Client load was produced utilizing HTTPerf and Autobench.

For the ad-TLSM evaluation, the client activity is as follows:

1) This HTTPerf script generates a total of 10 sessions at a rate of 1 session per 256 seconds. Each session consists of 64 calls that are spaced out by 4 seconds.
2) During the first 500seconds, 10 clients arrive every 3seconds (average request rate at 200req/s).
3) Then, the next 250 seconds, client's arrival increases at 10 clients every 2.5 seconds (245req/s)
4) And then, reduces to 200req/s for the last 500 seconds.

Furthermore, HTTPerf will use HTTP version 1.0, this requires new TCP connection per request. Also, no reuse of the TLS session ids, so the TLS handshake occurs for each connection. To automate the client workload generation process, The Autobench script repeatedly runs HTTPerf against the host, requesting more connections per second with each run, and extracts the important information from the output, delivering a CSV or TSV format file that can be

directly loaded into a spreadsheet for analysis or graphing.

4.2 Evaluation

This section demonstrates the adaptation of security and performance to achieve the objectives of the proposed solution. The Adaptive Transport Layer Security Model (ad-TLSM) design was implemented to maximize security by effectively utilizing available processing resources while considering client Quality of Service (QoS) requirements. Monitoring components collect data on system resources and encryption, which is then used by the adaptation strategy to determine the performance gains of security adaptations based on the gathered information. The ad-TLSM module enables successful runtime security adaptation, ensuring the effective adjustment of security measures.

4.2.1 Security algorithm performance

Table 1 show the default TLS cryptography algorithms performance effect, when 532B file requests are made. This supports the statement [10] that AES is typically very fast and efficient when implemented in hardware and ChaCha20 is efficient in software implementation.

Client arrival rate starts with a set of 10 new clients each 2.5 seconds (240req/s), until the server is overloaded, the client load pattern is followed, with a 0.1 second wait between each group of 10 client arrivals. 90% confidence intervals are <2ms for all response times under 250ms and under 10ms for all response times beyond 250ms.

Table 1. Average number of bytes per second

Cryptographic algorithm	Throughput (kB/s)
ChaCha20-Poly1305	112000
AES-GCM 128	36000
AES-GCM 256	26500

Figure 10 illustrates that as the client load increases, the DNS recursive resolver can handle approximately 318req/s with AES-GCM 256 before becoming overloaded. By using a different algorithm, it can respond to 15%-25% more requests per second. AES-GCM 128 and ChaCha20 outperformed AES-GCM 256 by approximately 15% and 25% respectively, achieving 333req/s and 343req/s with the same file size and client concurrency level. While the DNS recursive resolver allocates more resources to tasks other than cryptography, the choice of encryption algorithm significantly impacts its throughput. Therefore, using a different security level can increase the number of supported clients by 15%-25%.

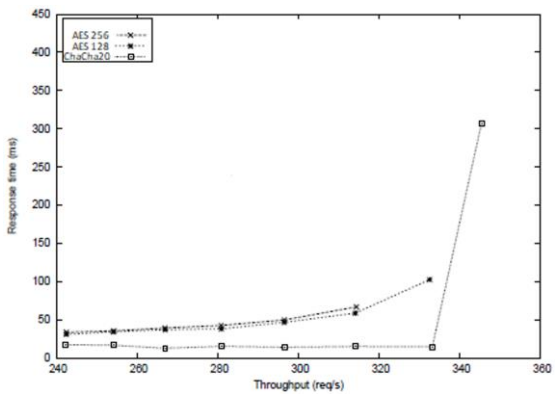

Figure 10. Security algorithm performance

4.2.2 Use-case scenario

In the given use-case, we examine the impact of increasing client entries on a DNS recursive resolver using three cryptography algorithms for TLS: AES-GCM 128, AES-GCM 256, and ChaCha20-Poly1305. We demonstrate the effect of these algorithms on the resolver's performance and compare it to the proposed solution's enhancement in security level, considering client Quality of Service (QoS) requirements. The figures display average throughput values over 10-second intervals, while the average number of timed out requests was determined through repeated evaluations.

Based on this scenario, AES-GCM 128 is deemed suitable for protecting the data, ensuring a satisfactory level of QoS for clients who expect timely responses from the resolver.

Figure 11 depicts the DNS recursive resolver throughput when AES-GCM 128 encryption is used. It demonstrates that the resolver can handle client load without any timed-out requests. The experiment was repeated multiple times with consistent results. The graph shows that 10 clients arrive at intervals of 3 seconds during the first 500 seconds, and start leaving the system after 256 seconds. The resolver's throughput stabilizes at 250 requests per second. The subsequent evaluation phases show an increase in client entry as expected. Despite a higher CPU load, the throughput for ChaCha20-Poly1305 is comparable to AES-GCM 128. As shown in Figure 10, ChaCha20-Poly1305 can handle a request rate of 343 requests per second within the 6-second client QoS constraint, with no reported time-outs.

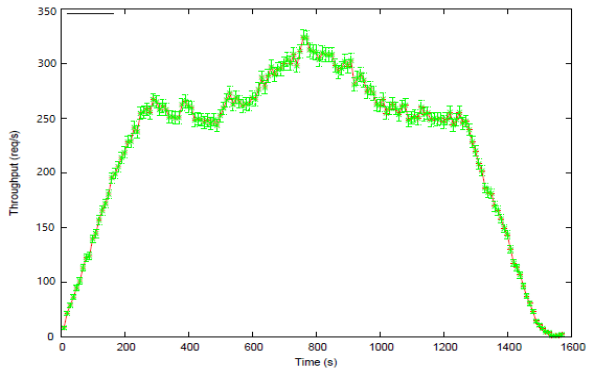

Figure 11. DNS recursive resolver throughput using AES-GCM 128

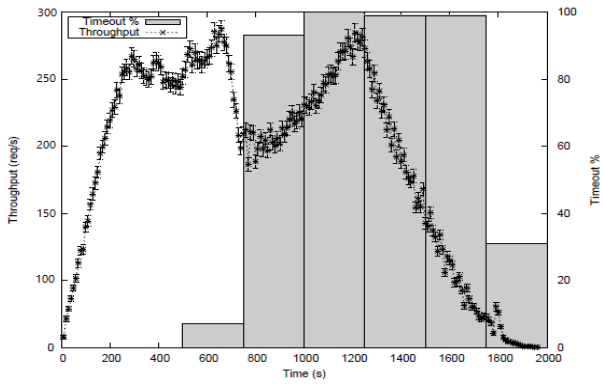

Figure 12. DNS recursive resolver throughput using AES-GCM 256

Figure 12 illustrates the impact of client activities on the throughput of a DNS recursive resolver using AES-GCM 256 encryption. The bar graph shows the percentage of timed-out

requests in each interval. As the client load increases beyond the resolver's capacity, requests start timing out at 500 seconds. With longer response times, clients send fewer queries per second, resulting in a decrease in resolver throughput. The slow request rate leads to extended sessions and an increase in simultaneous sessions as more clients enter. This increase in simultaneous sessions further elevates the total request rate, causing nearly all requests to time out. Even after the arrival of new client requests ceases and current sessions finish, many requests still time out due to the server's ongoing load. New requests are delayed until the server clears its load, potentially leading to additional timeouts. This use-case scenario highlights the performance degradation that occurs in a non-adaptive DNS recursive resolver server when faced with a spike in client arrivals.

4.2.3 Adaptation strategy

This section will illustrate how the ad-TLSM can be utilized to adapt security and performance utilizing available system resources to adapt security with regards to client QoS constraints. In light of the client load patterns, indirect monitoring adaptation strategy was considered. This approach relies on indirectly monitoring of the DNS recursive resolver throughput by identifying that upsurge in throughput infers increased DNS recursive resolver CPU load. Also, CPU load is monitored in the background by the Log file and the information can be utilized by the proposed solution. Thus, effectively utilize available resources, requires the need to know how much resource is free, how much resource is needed or released when the security is adapted.

Figure 13 depicts the performance of the DNS recursive resolver under different levels of encryption, with an average session duration of 256 seconds and varying client arrival rates. The graph demonstrates that the DNS recursive resolver becomes overloaded at different request rates depending on the encryption used. AES-GCM 256 overloads at 230req/s, while ChaCha20-Poly1305 and AES-GCM 128 overload at 296req/s. These findings align with the outcomes in Figure 10 and provide additional insights. It is important to note that a DNS recursive resolver with a CPU load over 90% using AES-GCM 128 should not switch to AES-GCM 256, as it would result in overload. Similar considerations apply to ChaCha20-Poly1305 to AES-GCM 128 (over 70%) and ChaCha20-Poly1305 to AES-GCM 256 (less than 50%). Lowering the security level becomes practical when the CPU utilization reaches 100%. Based on these insights, a security policy was formulated and presented in Table 2. This policy ensures efficient utilization of system resources to enhance security while avoiding DNS recursive resolver overload and maintaining QoS constraints.

By comparing the current CPU load to values in Table 2, this policy determines if the security level needs to be adapted. The current CPU load is evaluated against each value in the security policy reference table. If the CPU load is lower/higher than a specific value, the security may be adapted to the corresponding algorithm.

Table 2. The ad-TLSM security policy reference

Cryptographic Algorithm	AES 256	ChaCha 20	AES 128
AES 256	X	0.7	0.9
ChaCha20	0.5	X	0.9
AES-GCM 128	0.5	0.7	X

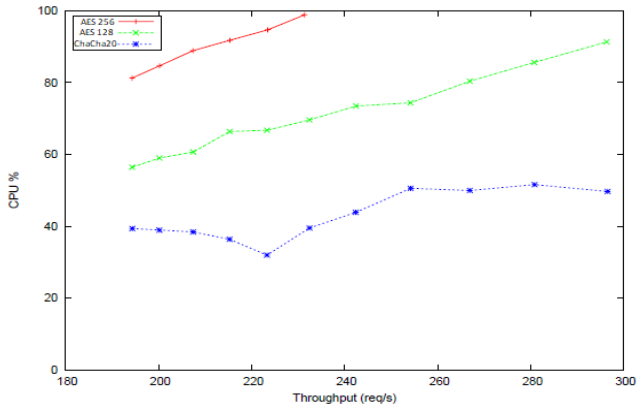

Figure 13. CPU utilization under different client loads

To achieve the indirect monitoring adaptation strategy, the following procedure was utilized:

```
set cpu_load_low=0.5
set cpu_load_meduim=0.7
set cpu_load_high=0.9
set load, crypto_algo, N
for (i=0; i<=N, i++) {
if (load<cpu_load_low)
{
set crypto_algo=1
        }
else if (load>=cpu_load_low && load<=cpu_load_medium)
{
set crypto_algo=0
        }
else if (load>=cpu_load_medium && load<=cpu_load_high)
{
set crypto_algo=0
        }
}
return crypto_algo
end procedure
```

For example, if the current security is ChaCha20-Poly1305, the CPU load is compared to values in AES-GCM 128 or AES-GCM 256. If the CPU load is 0.5, AES-GCM 256 is selected, or if the CPU load is 0.9, AES-GCM 128 is selected. Similarly, if the current security is AES-GCM 256, the CPU load is compared to values in ChaCha20-Poly1305 or AES-GCM 128. If the CPU load is 0.9, AES-GCM 128 is selected, or if the CPU load is 0.7, ChaCha20-Poly1305 is selected. Lastly, if the current security is AES-GCM 128, the CPU load is compared to values in AES-GCM 256 or ChaCha20-Poly1305. If the CPU load is less than 0.5, AES-GCM 256 is selected, or if the CPU load is 0.7, ChaCha20-Poly1305 is selected.

4.2.4 Evaluation of the ad-TLSM

Every 10 seconds the average CPU load is documented over the preceding 10 seconds and adapts the proposed solution security in light of the indirect monitoring in Subsection 4.2.3.

By utilizing the ad-TLSM, Figure 14 demonstrates effective maximization of security throughout most of the evaluation duration. New clients arriving before 460 seconds and after 720 seconds received security levels that exceeded recommendations while maintaining client Quality of Service (QoS) constraints. The configuration depicted in Figure 14 allowed the DNS recursive resolver to handle client loads successfully. The CPU load steadily decreased as security was

adapted to ChaCha20 and then to AES-GCM 128. This occurred twice in a short interval, at 450 seconds and 460 seconds, preventing overloading of the DNS recursive resolver. Once the CPU load reached a safe level below 50% (as shown in Table 3.), AES-GCM 256 and then ChaCha20 were employed for security, observed at 720 seconds and 750 seconds. However, the CPU load began increasing after approximately 780 seconds, prompting the adaptation of security to AES-GCM 128.

Table 3 and Figure 14 demonstrate the adaptation of security measures. While the monitoring mechanism may require additional processing time to gather and analyze data on network conditions, the overhead is minimal due to its efficient and lightweight design. The benefits of the ad-TLSM outweigh the potential impact on throughput, as it enables automatic adjustment to network changes, improving performance and security management. To evaluate the ad-TLSM's performance, we compared its throughput and latency values with those of PDOT [21] using a similar client load pattern, as shown in Table 4. Table 3 shows the ad-TLSM security adaptation.

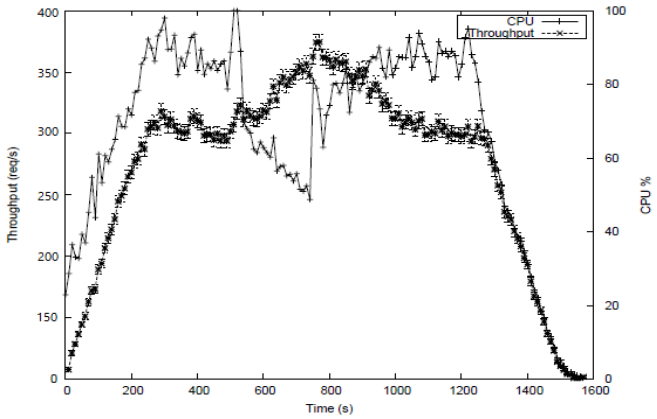

Figure 14. DNS recursive resolver throughput using ad-TLSM

Table 3. The ad-TLSM security adaptation

Adaptation	Time (seconds)	Throughput (Req/s) (approx.)
AES 256	0	-
ChaCha20	450	318
AES 128	460	321
AES 256	720	348
ChaCha20	750	350
AES 128	780	375

Table 4. The ad-TLSM against PDoT based on throughput and latency

Models	Time (seconds)	Throughput (Req/s) (approx.)
PDOT [21]	700	0.3
Ad-TLSM	700	0.2

Table 4 shows the ad-TLSM against PDoT based on throughput and latency.

The evaluation of the ad-TLSM method demonstrated its superior performance compared to other methods. However, the CPU capacity directly affects the performance of both the ad-TLSM and the DNS recursive resolver. Higher CPU capacity enables more efficient handling of incoming requests, resulting in faster response times and improved overall performance. The ad-TLSM also shows similar throughput values to other methods, indicating efficient handling of requests and a better user experience. Furthermore, the ad-TLSM exhibits better latency values, which can be attributed to its effective adaptation strategy. The satisfactory throughput and latency values of the ad-TLSM are presented in Table 4.

4.3 Security analysis

The section examines the potential risks and vulnerabilities associated with a malicious operator and network adversary in the context of the research on monitoring CPU load of a DNS recursive resolver and applying an adaptation strategy for enhancing performance. This section explores the importance of selecting appropriate encryption algorithms, the need for an adaptive security strategy, the impact on client QoS constraints, and the measures to mitigate overload attacks. It highlights the significance of protecting against malicious actors and ensuring secure communication to safeguard the DNS recursive resolver from potential threats and breaches. In terms of security implications related to a malicious operator and network adversary, the research and evaluation discussed several important factors:

1) **Encryption algorithm selection:** The choice of encryption algorithm can have significant security implications when dealing with a malicious operator or network adversary. The evaluation showed that different encryption algorithms have varying levels of performance and efficiency. It is important to choose an algorithm that provides strong security against potential attacks, as a malicious operator or network adversary may attempt to exploit vulnerabilities in weaker algorithms.
2) **Adaptive security strategy:** The research proposed the ad-TLSM strategy to adapt security and performance based on system resources and client QoS requirements. By monitoring the CPU load and dynamically adjusting the security level, the ad-TLSM aims to optimize security without overloading the DNS recursive resolver. This is crucial in defending against malicious operators or network adversaries who may attempt to overload the resolver to disrupt its functionality.
3) **Client QoS constraints:** The research also emphasized the importance of maintaining client QoS (Quality of Service) constraints while adapting security. This ensures that clients receive timely responses and are not negatively impacted by security adaptations. By considering client QoS requirements, the ad-TLSM strategy aims to strike a balance between security and performance, providing protection against malicious actors while ensuring a satisfactory user experience.
4) **Mitigating overload attacks:** A malicious operator or network adversary may attempt to overload the DNS recursive resolver by sending a high volume of requests. This can lead to denial-of-service (DoS) attacks and disrupt the resolver's normal operation. The ad-TLSM strategy, by monitoring CPU load and adapting security accordingly, helps mitigate the risk of overload attacks by ensuring that the resolver does not become overwhelmed and can handle the incoming traffic effectively.
5) **Secure communication:** The use of strong encryption algorithms, such as ChaCha20-Poly1305 and AES-GCM, helps protect against potential attacks from malicious operators or network adversaries. These algorithms provide confidentiality, integrity, and authenticity of the communication, making it difficult for adversaries to

intercept or tamper with the data exchanged between clients and the DNS recursive resolver.

6) **Privacy:** By utilizing encryption algorithms and adaptation strategies, the research aims to prevent unauthorized access to user information, thereby safeguarding privacy. The implementation of robust encryption protocols helps in preventing eavesdropping, data interception, and unauthorized data tampering, thereby enhancing the overall privacy of users utilizing the DNS recursive resolver. Additionally, the research highlights the significance of selecting encryption algorithms that offer strong privacy guarantees, ensuring that user data remains confidential and protected from potential privacy breaches.

4.4 Limitations of the study

Despite the limitations of this research, we believe that our findings provide valuable insights into the impact of DNS on user experience and how different DNS stakeholders can improve it. Achieving a perfect balance between security and performance is challenging, especially considering that our measurements were conducted on the Windows operating system, which may affect the results due to the networking stack and algorithm parameters. However, we expect our findings to be applicable to other operating systems as well, considering the highly optimized nature of networking stacks. Additionally, it is important to acknowledge that the direct monitoring approach assumes file size as a direct indicator of DNS recursive resolver workload, which may not always hold true.

5. CONCLUSION

The research developed an Adaptive Transport Layer Security Model (ad-TLSM) for enhancing DNS performance by adapting security, without breaching QoS constraint. The ad-TLSM was developed, implemented and evaluated. The Adaption strategies used are based on an Indirect Monitoring of contextual features. Evaluation results indicated that the ad-TLSM could be adopted and used were security and performance is required in communication.

The results of this study suggest that the methodology used to achieve high throughput while maintaining low latency in DNS security could be applied to other forms of network environment. Using additional contextual features, such as latency, Round-Trip-Time (RTT), would improve the security and performance of the network environment. Furthermore, it will be important to conduct field evaluations and obtain information from local stakeholders on how to improve the adaptation strategy, particularly in developing nations with cyber-security deficiencies.

The research identifies the high failure rate in DNS performance in DoT as a research gap. It then developed an ad-TLSM to compensate for the identified inadequacy in DoT. Additionally, the inclusion of an Adaptation strategy helped to provide better results, qualifying the ad-TLSM for use in secure and confidential communication in places where performance is important. Finally, it gives academics a chance to do further research to enhance the findings already made in this study.

Further studies could focus on the implementation and testing of the ad-TLSM scalability and how adaptive security can be achieved with existing security protocols and techniques.

REFERENCES

[1] Larry, L. The past, present and future of DNS security. Security Intelligence. https://securityintelligence.com/the-past-present-and-future-of-dns-security/, accessed on Dec. 22, 2017.

[2] Kabelova, A., Dostalek, L. (2006). DNS in action: A detailed and practical guide to DNS implementation, configuration, and administration. Packt Publishing Ltd.

[3] Gupta, B.B. (2018). Computer and Cyber Security: Principles, Algorithm, Applications, and Perspectives. CRC Press.

[4] Khan, I., Farrelly, W., Curran, K. (2020). A demonstration of practical DNS attacks and their mitigation using DNSSEC. International Journal of Wireless Networks and Broadband Technologies (IJWNBT), 9(1): 56-78. https://doi.org/10.4018/IJWNBT.2020010104

[5] García, S., Hynek, K., Vekshin, D., Čejka, T., Wasicek, A. (2021). Large scale measurement on the adoption of encrypted DNS. arXiv Preprint arXiv: 2107.04436. https://doi.org/10.48550/arXiv.2107.04436

[6] Lyu, M., Gharakheili, H.H., Sivaraman, V. (2022). A survey on DNS encryption: Current development, malware misuse, and inference techniques. ACM Computing Surveys, 55(8): 1-28. https://doi.org/10.1145/3547331

[7] Doan, T.V., Tsareva, I., Bajpai, V. (2021). Measuring DNS over TLS from the edge: adoption, reliability, and response times. In Passive and Active Measurement: 22nd International Conference, PAM 2021, Virtual Event, March 29-April 1, Springer International Publishing. Proceedings, 22: 192-209. https://doi.org/10.1007/978-3-030-72582-2_12

[8] Hounsel, A., Borgolte, K., Schmitt, P., Holland, J., Feamster, N. (2020). Comparing the effects of DNS, DoT, and DoH on web performance. In Proceedings of The Web Conference, 2020: 562-572. https://doi.org/10.1145/3366423.3380139

[9] Böttger, T., Cuadrado, F., Antichi, G., Fernandes, E.L., Tyson, G., Castro, I., Uhlig, S. (2019). An empirical study of the cost of DNS-over-https. In Proceedings of the Internet Measurement Conference, pp. 15-21. https://doi.org/10.1145/3355369.3355575

[10] Jonglez, B. (2020). End-to-end mechanisms to improve latency in communication networks. Doctoral Dissertation, Université Grenoble Alpes.

[11] Joel, W. (2008). Designing an adaptive security architecture. Sun BluePrints™ Online, 1-19.

[12] Wessels, D. (2004). A recent DNS survey. DNS-OARC.

[13] Piscitello, D. (2010). Conficker summary and review. Technical Report.

[14] de Castro, L.N., Timmis, J. (2002). Artificial immune systems: A new computational intelligence approach. Springer Science & Business Media.

[15] Srihari, H.S., Koundinya, A.K., Srinivasan, G.N. (2018). Generalized adaptive security for computer systems. In Proceedings of the World Congress on Engineering and Computer Science, Vol. 1.

[16] Vahab, P. (2010). Notes on transport layer security.

Computer Science Department, University of California, 1-6.
[17] Hao, Y. (2017). Transport layer security (TLS)-Transport layer security performance testing. Xena Networks, 1-13.
[18] Khandkar, V.S., Hanawal, M.K., Kulkarni, S.G. (2022). Challenges in adapting ECH in TLS for privacy enhancement over the internet. arXiv Preprint arXiv: 2207.01841. https://doi.org/10.48550/arXiv.2207.01841
[19] Willian, S. (2005). Cryptography and network security: Principles and practice (4th ed.). Prentice.
[20] Houser, R., Li, Z., Cotton, C., Wang, H. (2019). An investigation on information leakage of DNS over TLS. In Proceedings of the 15th International Conference on Emerging Networking Experiments and Technologies, pp. 123-137. https://doi.org/10.1145/3359989.3365429
[21] Nakatsuka, Y., Paverd, A., Tsudik, G. (2021). PDOT: Private DNS-over-TLS with TEE support. Digital Threats: Research and Practice, 2(1): 1-22. https://doi.org/10.1145/3431171
[22] Shang, H., Wills, C.E. (2006). Piggybacking related domain names to improve DNS performance. Computer Networks, 50(11): 1733-1748. https://doi.org/10.1016/j.comnet.2005.06.016
[23] Khandkar, V.S., Hanawal, M.K. (2021). Masking host identity on internet: Encrypted TLS/SSL handshake. arXiv Preprint arXiv: 2101.04556. https://doi.org/10.48550/arXiv.2101.04556
[24] Alao, D.O., Ayankoya, F.Y., Ajayi, O.F., Ohwo, O.B. (2023). The need to improve DNS security architecture: An adaptive security approach. Information Dynamics and Applications, 2(1): 19-30. https://doi.org/10.56578/ida020103
[25] Hounsel, A., Schmitt, P., Borgolte, K., Feamster, N. (2021). Can encrypted DNS be fast? In Passive and Active Measurement: 22nd International Conference, PAM 2021, Virtual Event, March 29-April 1, Springer International Publishing, 22: 444-459. https://doi.org/10.1007/978-3-030-72582-2_26
[26] Hounsel, A., Schmitt, P., Borgolte, K., Feamster, N. (2021). Encryption without centralization: Distributing DNS queries across recursive resolvers. In Proceedings of the Applied Networking Research Workshop, pp. 62-68. https://doi.org/10.1145/3472305.3472318
[27] Jonglez, B., Birbalta, S., Heusse, M. (2019). Poster: Persistent DNS connections for improved performance. In 2019 IFIP Networking Conference. IEEE, pp. 1-2. https://doi.org/10.23919/IFIPNetworking46909.2019.8999394