Asian Journal of Computer Science and Technology
ISSN: 2249-0701 (P); 2583-7907 (E)
Vol.13 No.2, 2024, pp.48-55

DOI: https://doi.org/10.70112/ajcst-2024.13.2.4288

Enhancing DNS Performance with Efficient Cryptographic Algorithms: A Comparative Study of DoT Frameworks

Ohwo Onome Blaise[1*], Ajayi Wumi[2] and Udosen Alfred[3]
[1&2]Department of Computer Science, [3]Department of Software Engineering,
Babcock University, Ogun State, Nigeria
E-mail: ajayiw@babcock.edu.ng, udosena@babcock.edu.ng
*Corresponding Author: ohwoo@babcock.edu.ng
(Received 19 September 2024; Revised 14 October 2024, Accepted 8 November 2024; Available online 12 November 2024)

***Abstract* - The Domain Name System (DNS) is a critical component of the Internet, and its disruption can significantly affect service quality. To enhance security and protect user privacy, encrypted protocols, such as DNS over TLS (DoT), DNS over Quick UDP Internet Connections (DoQ), and DNS over HTTPS (DoH), have been introduced. This study evaluates the performance impact of different cryptographic algorithms within the DoT framework, focusing on how encryption influences DNS query performance and resolver efficiency. Performance evaluations were conducted using various cryptographic algorithms under different client load conditions. Metrics such as response rate, timeout rate, and resource utilization were analyzed to assess the impact of encryption on DNS recursive resolvers. The analysis revealed that the choice of encryption algorithm and client load significantly affect performance. Advanced Encryption Standard-Galois/Counter Mode (AES-GCM) 128 and ChaCha20-Poly1305 demonstrated superior performance, exhibiting higher response rates and lower timeout rates compared to AES-GCM 256. Organizations managing DNS infrastructure should monitor client loads and consider adopting efficient encryption algorithms, such as AES-GCM 128 or ChaCha20-Poly1305. These choices can optimize DNS recursive resolver performance while maintaining robust security in dynamic network environments.**
***Keywords:* Domain Name System (DNS), Encrypted Protocols, DNS over TLS (DoT), Cryptographic Algorithms, Performance Evaluation**

I. INTRODUCTION

The Domain Name System (DNS) plays an essential role in the functioning of the Internet and is crucial for its dependable and trustworthy operation. Consequently, any disruption in its functioning can significantly impact both the quality of service offered and the global Internet. Over time, numerous attempts have been made to compromise DNS security to launch various attacks against it [1].

The DNS recursive resolver lacks a critical security framework for ensuring data confidentiality, accessibility, and integrity [2]. To address these issues, new protocols, such as DNS over Transport Layer Security (DoT), DNS over Quick UDP Internet Connections (DoQ), and DNS over Hypertext Transfer Protocol Secure (DoH), have been developed [3]. DNS encryption is achieved by encrypting the content of requests and responses (between clients and recursive resolvers) using existing cryptographic methods in an upper-layer protocol. Encrypting DNS queries and responses between clients and resolvers enhances user privacy and helps defend against attacks.

As a result, several researchers [4]-[7] have investigated the impact of DNS encrypted transports on the end-user experience. These studies have shown that encrypted transports often lead to slower DNS queries due to connection and transport overhead, particularly in networks with suboptimal performance. To improve DNS query response times, it is crucial to understand the relative performance costs and benefits of the cryptographic algorithms used in DNS transport protocols.

Existing research has demonstrated that cryptography impacts performance when using Transport Layer Security (TLS). TLS employs both symmetric and asymmetric cryptographic primitives; however, asymmetric encryption consumes more memory. Symmetric cryptography includes ChaCha20-Poly1305, which is effective in software implementation, and Advanced Encryption Standard (AES), which is typically fast and efficient in hardware implementation. More importantly, the cryptographic operations of TLS can impose high CPU costs when processing DNS queries. This study aims to evaluate the performance of various cryptographic algorithms within the DoT framework under increasing client loads. Three cryptographic algorithms - AES-GCM 256, AES-GCM 128, and ChaCha20-Poly1305 - are investigated in this paper. Offline evaluations and performance analyses reveal insights into the efficiency and effectiveness of these algorithms in maintaining stability and accommodating increasing client demands.

A. Significance of the Study

This study is of significant importance as it aims to assess various cryptographic algorithms within the DNS over TLS (DoT) framework, focusing on managing an increasing influx of clients. This evaluation contributes to critical aspects of DNS security, thereby enhancing overall Internet functionality. The study examines the performance of algorithms such as AES-GCM 256, AES-GCM 128, and ChaCha20-Poly1305, providing insights into their effectiveness in securing DNS communications.

User privacy is a paramount concern in the digital landscape, and this study addresses it by analyzing the impact of cryptographic algorithms on encrypting DNS requests and responses. The findings highlight trade-offs between encryption strength, computational overhead, and DNS query response times, aiding in the identification of optimal algorithms that balance performance and privacy. Additionally, the study explores the CPU costs associated with cryptographic operations in DNS query processing. By evaluating different algorithms, it offers insights into their relative performance costs and benefits, thereby guiding the optimization of DNS performance.

This information is crucial for minimizing computational overhead while ensuring secure and reliable DNS operations. Furthermore, the study's findings provide valuable information for researchers, developers, and system administrators involved in DNS implementation decisions. It equips them with insights to make informed choices regarding the selection and configuration of cryptographic algorithms, ensuring stability, scalability, and effective management of growing client demands in DNS over TLS implementations.

II. MATERIALS AND METHODS

A. Review of Literature

DNS over TLS (DoT) aims to prevent on-path attackers from monitoring and altering victims' DNS queries and responses, although it remains unclear how much information can be extracted from DoT interactions through traffic analysis. One method proposed in [8] uses DoT fingerprinting to examine traffic and determine whether users have visited websites of potential interest to adversaries. With a false negative rate of less than 17% and a false positive rate of less than 0.5% when DNS packets are not padded, this method effectively identifies DoT activity for specific websites. Additionally, information leakage was shown to be possible even with padded DoT messages.

Over the first five months of 2021, this study monitored the adoption of DoH, DoT, and DoQ across three distinct global enterprises. The analysis, as detailed in [3], examined aggregate figures, requests per user, and traffic patterns to identify adoption trends. While the average Internet traffic associated with DoH, DoT, and DoQ increased in 2020, no statistically significant changes were observed during the first five months of 2021.

However, the study revealed a notable fourfold increase in accessible DoH servers. These findings suggest that connections to unfamiliar DoH servers may soon rise, potentially serving both beneficial and malicious purposes, despite the current absence of a marked uptick in encrypted DNS usage. Although DoT was introduced as a DNS protocol improvement in 2016, no comprehensive performance analysis has been conducted. A recent study by [4] evaluated the performance of DoT at the network edge, including its adoption rate, reliability, and response times compared to conventional DNS over UDP/53 (Do53), using 3.2k RIPE Atlas probes deployed in home networks. The survey found that local resolvers remain the primary proponents of DoT, while open resolvers are increasingly supporting it. However, DoT's reliability decreased as failure rates rose, and response times significantly increased. The majority of failures were attributed to timeouts caused by intermediate middleboxes along the network path that discard packets intended for port 853, the designated port for DoT.

The influence of Do53, DoT, and DoH on query response times and page load times across five distinct global orientations was assessed in [5]. The results showed that while DoH and DoT can surpass Do53 in page load speed, they frequently exhibit slower response times than Do53. During network degradation, significant packet loss and latency reduce the performance of all protocols.

T. Boettger *et al.,* [6] examined the DoH environment, focusing on its additional security overhead and its advantages over DoT by comparing it with other secure DNS protocols. The study also explored the effects of head-of-line blocking on DoT and DoH/1 performance, which partly explains why DoH/2 gained popularity more quickly than DoT.

According to B. Jonglez [7], DNS-over-TCP performance is comparable to Do53 when there are fewer clients, with only a 30% latency increase. However, as the number of clients grows, DNS-over-TCP performance deteriorates, stabilizing at a 75% latency increase. The performance profile of DNS-over-TCP is comparable to that of standard TCP, despite a 30-45% speed drop. As client numbers rise, both TCP and TLS suffer significant performance degradation, likely due to the kernel handling multiple concurrent TCP connections.

The existing UDP-based resolution method can take up to 5 seconds to detect a packet loss event, leading to significant delays caused by retransmission timeouts. B. Jonglez *et al.,* [9] investigated TCP- or TLS-based persistent DNS connections as a potential solution to this issue. Persistent connections were shown to significantly reduce worst-case latency. Large-scale platform experiments revealed that recursive DNS services can be effectively delivered over TCP and TLS with acceptable performance impacts when compared to UDP, provided standard software and sufficiently powerful hardware are used. However, switching to TCP or TLS increases the recursive resolver's load, particularly with a high number of concurrent connections.

DNS is a critical component of Internet infrastructure, making its security essential. Existing DNS security architectures, such as DoH, DoT, and DoQ, provide security services but face performance limitations and fail to fully protect DNS data against evolving cyber threats. The research in [10] evaluates the current state of DNS security and its inadequacies, proposing an Adaptive Security Architecture inspired by the immune response system. This approach addresses known threats and anticipates new ones through biodiversity-like defense mechanisms.

O. B. Ohwo *et al.,* [11] introduced an Adaptive Transport Layer Security Model (ad-TLSM) to address performance challenges in DoT, which encrypts communication between clients and recursive resolvers. The ad-TLSM incorporates real-time monitoring during the TLS handshake to track metrics such as throughput, CPU load, and cryptographic algorithm performance. This model adapts security dynamically based on client-server conditions. Performance evaluations showed that while AES-GCM 256 caused high CPU load, switching to ChaCha20 improved request handling by 15-25%. This adaptive approach reduces CPU strain and enhances overall performance, outperforming existing models in latency and quality of service.

M. S. Islam *et al.,* [12] explored various phishing attack types, including email spoofing, spear phishing, phone phishing, clone phishing, pharming, HTTP phishing, man-in-the-middle attacks, and fast-flux phishing. To combat these, the study developed a filtered website tool that identifies fraudulent links by analyzing website IPs, registration dates, and DNS records. Although the tool requires further refinement, it represents a significant step toward improving digital security.

Finally, P. Banu *et al.,* [13] addresses power consumption in wireless mobile devices, focusing on optimizing energy use through software-level cryptographic protocols. By measuring the energy consumption of these protocols, the research demonstrated that the proposed cryptographic protocol offers enhanced security while consuming significantly less energy compared to existing protocols. The findings suggest that the proposed scheme is simpler, more secure, and more efficient, making it a promising solution for battery-efficient mobile systems.

B. Methodology

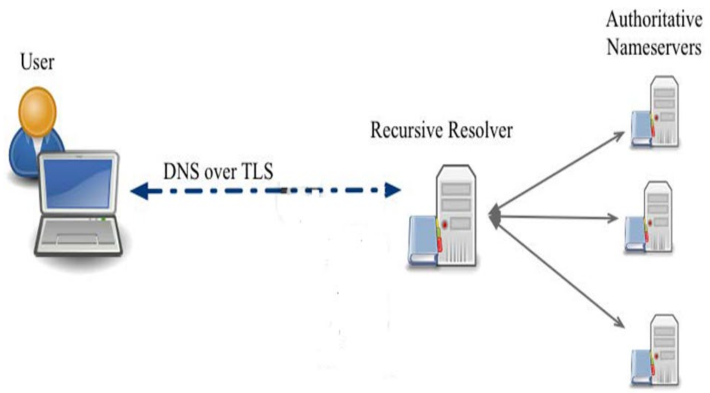

Fig. 1 Research Design (adopted from [8])

Fig. 1 illustrates a typical DoT session that occurs during the request-response cycle of a DNS server. To ensure the security of both requests and responses, TLS is employed at the TCP transport layer to encrypt the communication channel between the user and the DNS recursive resolver.

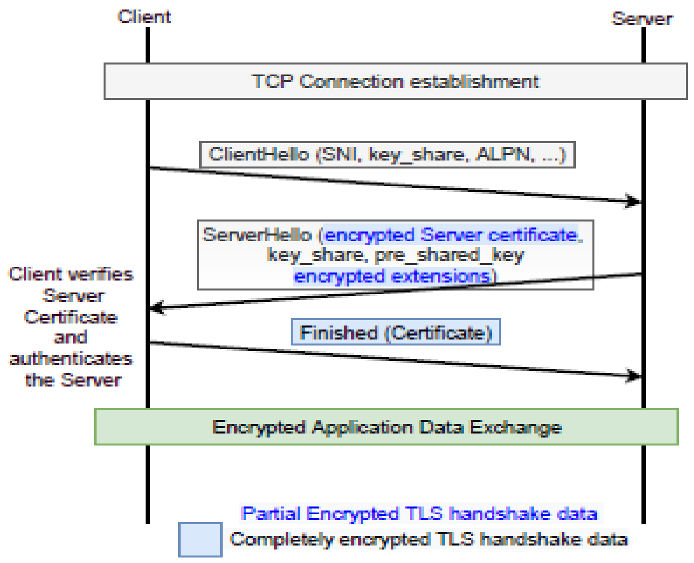
Client
Server
TCP Connection establishment
ClientHello (SNI, key_share, ALPN, ...)
ServerHello (encrypted Server certificate, key_share, pre_shared_key encrypted extensions)
Client verifies Server Certificate and authenticates the Server
Finished (Certificate)
Encrypted Application Data Exchange
Partial Encrypted TLS handshake data
Completely encrypted TLS handshake data

Fig. 2 A Typical TLS 1.3 Handshake (Adopted from [14])

The TLS 1.3 handshake procedure is illustrated in Fig. 2. To perform lookups, the client first connects via TCP to port TCP/853 of the DoT-enabled resolver. A TLS connection is then established to exchange cryptographic keys through the conventional TLS handshake process. The “key_share” and “pre_key_share” parameters in TLS 1.3 are used to encrypt the “ClientHello” handshake message. The “key_share” parameter facilitates the exchange of the endpoint’s public key share, enabling the generation of a secret key at the remote endpoint. Conversely, the “pre_key_share” parameter indicates the index of the encryption shared key currently in use from the list of negotiated shared keys.

Subsequently, the server’s security certificate is encrypted using TLS 1.3. Once the TLS session is successfully established, the client can perform TLS-encrypted DNS lookups through the DoT port TCP/853 on the resolver side. The TLS connections can remain open for subsequent DNS lookups, depending on client and server configurations, thereby reducing latency and avoiding additional TCP/TLS handshakes.

Transport Layer Security (TLS) version 1.3 offers a limited selection of cryptographic algorithms. By default, three cryptographic algorithms are available for establishing a secure connection: AES-GCM 128-bit, ChaCha20-Poly1305 256-bit, and AES-GCM 256-bit.

C. Performance Evaluation Parameters

The Evaluation Process:

1. *Security Algorithm Performance:* The throughput of common cryptographic algorithm implementations is evaluated using the same data size to analyze the security levels and performance overhead.
2. *Use-case Scenario:* The performance of the DNS recursive resolver is assessed using various cryptographic techniques and client-server configurations.

Client Activity for the Evaluation:

3. *Phase 1:* One session is initiated every 256 seconds using an HTTPerf script, resulting in a total of 10 sessions. Each session consists of 64 calls, separated by 4 seconds.
4. *Phase 2:* Over the first 500 seconds, 10 clients arrive every 3 seconds, resulting in an average request rate of 200 requests per second.
5. *Phase 3:* During the subsequent 250 seconds, client arrivals increase to 10 clients every 2.5 seconds, yielding an average request rate of 245 requests per second.
6. *Phase 4:* For the final 500 seconds, the request rate decreases to 200 requests per second.

III. RESULTS OF THE STUDY

The experiment was conducted on a Windows 10 computer running Apache 2.4, equipped with an Intel(R) Core(TM) i5-5300U CPU operating at 2.30 GHz and 16 GB of RAM. The Apache web server, renowned for its modular architecture, was employed to implement the DNS recursive resolver. Apache’s functionality is extended by modules, enabling the customization of client requests throughout the request-response cycle. The TLS module was integrated into Apache using OpenSSL 1.1. To ensure TLS session security, the SSLCipherSuite directive was utilized to specify a subset of security protocols for establishing secure sessions with clients.

A. Security Algorithm Performance Evaluation

A comprehensive assessment of cryptographic algorithm performance within the DNS over TLS (DoT) framework was conducted through offline evaluations. The objective of this evaluation was to demonstrate and analyze the efficacy and efficiency of the various cryptographic algorithms utilized in the DoT framework.

TABLE I AVERAGE NUMBER OF BYTES PER SECONDS

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

Table I illustrates the performance impact of the default TLS cryptographic algorithms when handling 532-byte file requests. The findings corroborate the assertion made in [7] that AES exhibits remarkable speed and efficiency, particularly when implemented in hardware. Additionally, they highlight the efficiency of ChaCha20 in software implementations.

Ten new clients are added every 2.5 seconds to initiate the client arrival rate, resulting in 240 requests per second. This cycle continues until the server reaches its overload level. The client load pattern then follows, with a 0.1-second delay between each subsequent wave of ten client arrivals. Notably, 90% confidence intervals ensure that response times remain below 2 milliseconds for all responses under 250 milliseconds and below 10 milliseconds for responses beyond this threshold.

The effect of client load on DNS recursive resolver performance is shown in Fig. 3, which demonstrates that AES-GCM 256 can efficiently process about 318 requests per second before overloading. The resolver is capable of handling an additional 15% to 25% of requests per second by using a different algorithm. Despite the final overload, AES-GCM 128 and ChaCha20 both outperformed AES-GCM 256 by about 15% and 25%, respectively, with response rates of 333 req/s and 343 req/s when compared under the same file size and client concurrency conditions.

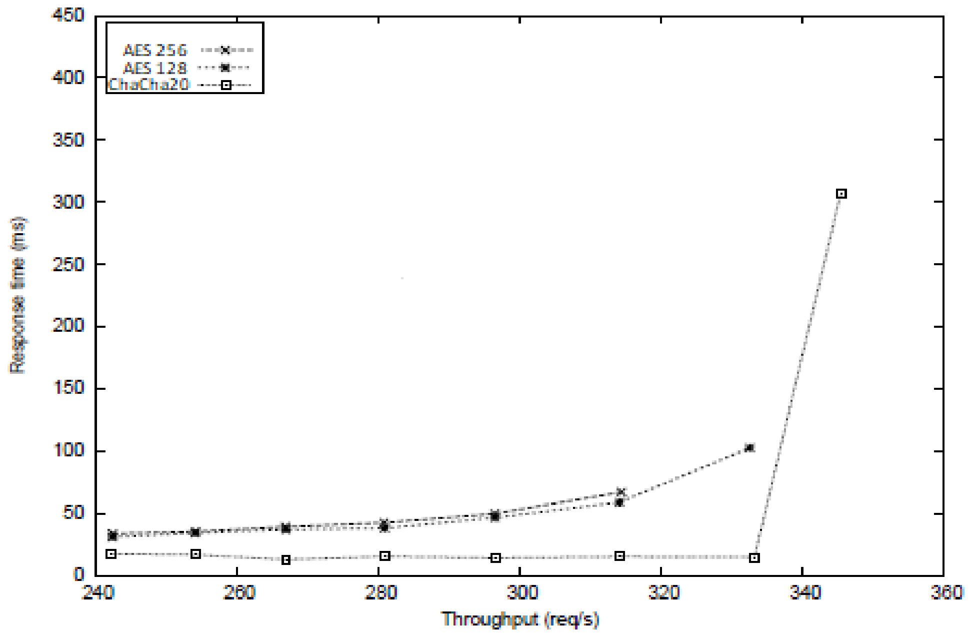

Fig. 3 Security Algorithm Performance

It is interesting to note that, even though cryptographic resources are prioritized, the DNS recursive resolver must allocate significantly more resources to other activities with each request. Nevertheless, it is clear that the selection of encryption technique is crucial, as it directly impacts the throughput of the DNS recursive resolver. A 15% to 25% increase in the number of supported clients could result from choosing a different security level.

B. Use-Case Scenario

In this use-case scenario, a DNS recursive resolver encounters an increasing number of clients. Three cryptographic methods for TLS are provided by the resolver: AES-GCM 128, AES-GCM 256, and ChaCha20-Poly1305. First, using each of the previously proposed cryptographic algorithms for security, we investigate how the DNS recursive resolver is affected by the increasing client demand. The average throughput recorded at 10-second intervals is represented by the throughput values shown in the figures. To determine the average number of timed-out requests, the assessment was repeated several times.

In the initial scenario, the utilization of AES-GCM 128 is considered appropriate for safeguarding the existing data. Clients expect a specific Quality of Service, as they prefer not to experience prolonged waits for the DNS recursive resolver response.

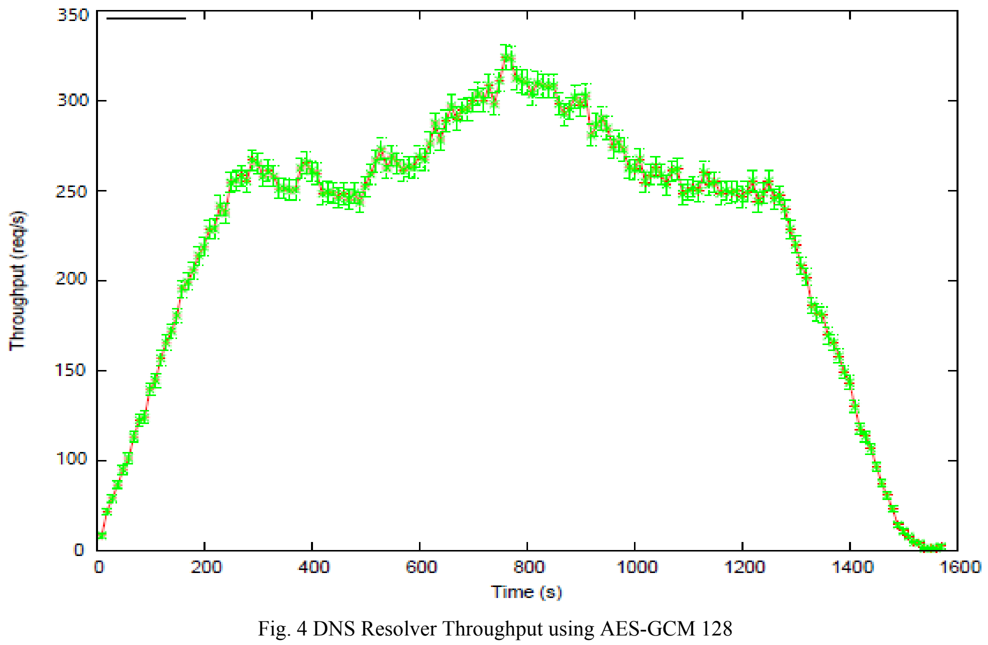

Fig. 4 DNS Resolver Throughput using AES-GCM 128

Fig. 4 provides a comprehensive overview of DNS recursive resolver throughput amidst varying client activities, specifically when employing AES-GCM 128 encryption. The illustration vividly portrays the resolver's ability to efficiently handle the influx of 10 clients at 3-second intervals for the initial 500 seconds, without any requests experiencing timeouts. This resilience persists even after repeated experimentation.

During the experiment, clients initiate departure from the system after 256 seconds upon successfully completing 64 requests each. Remarkably, the DNS recursive resolver maintains stability, achieving a throughput of 250 req/s. The subsequent phases of evaluation, depicted in Fig. 4, further demonstrate the resolver's adeptness at accommodating increasing client entries. The second phase registers a throughput of 340 req/s over 250 seconds, while the third phase maintains a throughput of 250 req/s for 500 seconds, aligning with anticipated behavior.

The comparison of DNS recursive resolver throughputs for AES-GCM 128 and ChaCha20-Poly1305 is noteworthy. Even with an increasing CPU load, the former shows equivalent performance. ChaCha20-Poly1305 has exceptional tolerance to a request rate of 343 requests per second within the 6-second client Quality of Service (QoS) restriction, as shown in Fig. 3, with no timeouts observed.

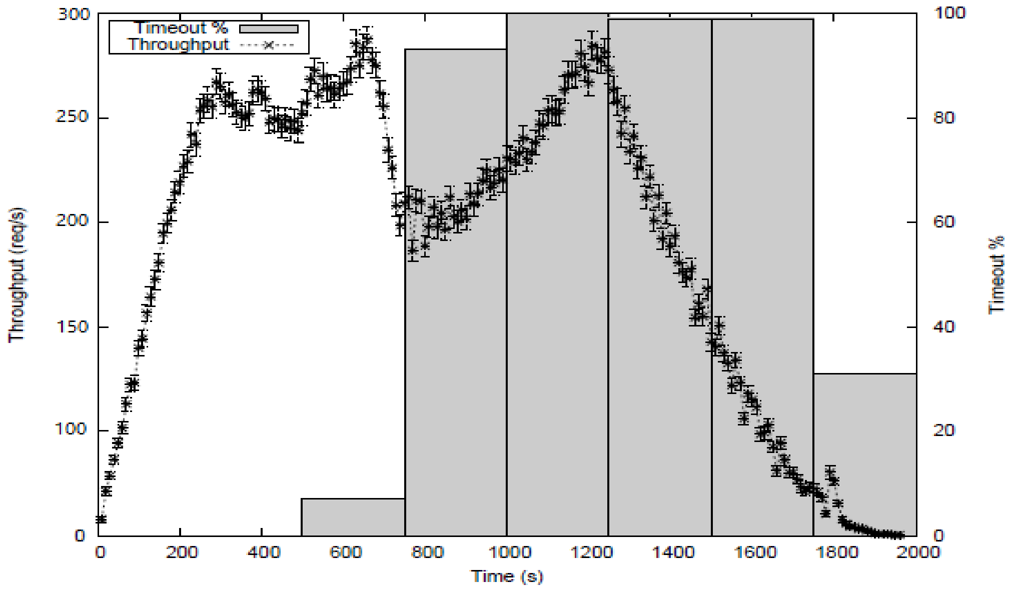

Fig. 5 DNS Resolver Throughput using AES-GCM 256

When AES-GCM 256 encryption is used, the DNS recursive resolver throughput is visually represented in Fig. 5, showing the impact of client activity. A percentage of timed-out requests within specific intervals is depicted in the bar graph. It highlights a significant finding by illustrating a situation in which the DNS recursive resolver struggles with growing client demand.

At the 500-second mark, an evident inflection point occurs as client demands surge beyond the capacity manageable by the DNS recursive resolver. This surge triggers the initiation of request timeouts. Notably, clients adhere to a sequential request pattern, dispatching the subsequent request only after awaiting the resolution of the preceding one. The extended response times contribute to a reduction in the number of queries per second, consequently diminishing the overall DNS recursive resolver throughput.

As the rate of client entries escalates, a corresponding increase in simultaneous sessions unfolds. This growth in simultaneous sessions contributes to an overall rise in the total request rate, culminating in the majority of requests experiencing timeouts. Even after the cessation of new client requests at the 1250-second mark and the completion of ongoing client sessions, a significant number of requests remain in a timed-out state.

Unaware of the timeouts, the server continues processing and holding back in an effort to fulfill pending client requests, which complicates the situation further. As a result, subsequent requests are not sent until the server has finished processing the current load, potentially leading to more timeouts. This use-case scenario illustrates how an increase in client arrivals might result in server overload and demonstrates how DNS recursive resolver performance deteriorates when AES-GCM 256 is used.

IV. DISCUSSION

The objective of this study was to evaluate the performance of different cryptographic algorithms within the DoT framework in managing a growing influx of clients. The study examined three cryptographic algorithms: AES-GCM 256, AES-GCM 128, and ChaCha20-Poly1305. Through offline evaluations and analysis of performance metrics, the findings shed light on the efficiency and effectiveness of these algorithms in maintaining stability and accommodating increasing client entries. It is clear from the results that the DNS recursive resolver's timeout rate and performance deterioration are significantly influenced by both client load and the encryption algorithm selection.

Firstly, the resolver's capacity to manage requests effectively is primarily determined by the client load. A greater number of timed-out requests arises as the resolver's performance declines with an increasing client load. This is particularly evident in scenarios where the client load surpasses the capacity manageable by the resolver, leading to a surge in request timeouts. The sequential request pattern followed by clients, who wait for the resolution of the preceding request before dispatching the subsequent one, further contributes to extended response times and reduced query rates.

Secondly, the choice of encryption algorithm also influences the performance degradation and timeout rate of the DNS recursive resolver. According to the results, AES-GCM 256 is less tolerant of client load than AES-GCM 128 and ChaCha20-Poly1305. However, the resolver shows the ability to handle more requests per second by using a different algorithm, such as AES-GCM 128 or ChaCha20-Poly1305. This implies that the resolver's throughput and capacity to effectively manage client load are directly impacted by the encryption algorithm used.

Additionally, the results demonstrate how effective AES-GCM 128 and ChaCha20-Poly1305 are compared to AES-GCM 256. Both AES-GCM 128 and ChaCha20-Poly1305 achieved greater response rates and outperformed AES-GCM 256 by roughly 15% and 25%, respectively, under similar file size and client concurrency conditions. This emphasizes the importance of selecting an appropriate encryption algorithm based on specific requirements and expected client load.

In conclusion, the findings discussed show that the DNS recursive resolver's performance degradation and timeout rate are greatly influenced by both client load and the encryption scheme selection.

V. CONCLUSION

In conclusion, this study delved into the assessment of cryptographic algorithms within the DoT framework, focusing on the management of an increasing influx of clients. The three cryptographic algorithms scrutinized - AES-GCM 256, AES-GCM 128, and ChaCha20-Poly1305 - were subjected to comprehensive offline evaluations and performance metric analyses, shedding light on their efficacy in maintaining stability and handling escalating client entries. The identified factors influencing the DNS recursive resolver's performance degradation and timeout rate were found to be twofold: client load and the choice of encryption algorithm. Firstly, the critical role of client load emerged, showcasing its direct correlation with the resolver's efficiency in managing requests. As the client load soared, the resolver's performance declined, resulting in a notable increase in timed-out requests. This trend became particularly pronounced when the client load exceeded the resolver's manageable capacity, leading to a surge in request timeouts. The sequential request pattern employed by clients, waiting for the resolution of the preceding request before initiating the next, further contributed to prolonged response times and diminished query rates.

Secondly, the choice of encryption algorithm surfaced as a pivotal factor influencing the performance dynamics of the DNS recursive resolver. Notably, AES-GCM 256 exhibited a lower tolerance for client load compared to its counterparts, AES-GCM 128 and ChaCha20-Poly1305. However, the study demonstrated that opting for an alternative algorithm, such as AES-GCM 128 or ChaCha20-Poly1305, enabled the resolver to manage additional requests per second. This underscored the direct impact of the encryption algorithm choice on the resolver's throughput and its ability to efficiently handle varying client loads.

Furthermore, the comparative analysis highlighted the superior efficiency of AES-GCM 128 and ChaCha20-Poly1305 relative to AES-GCM 256. Under equivalent conditions of file size and client concurrency, both AES-GCM 128 and ChaCha20-Poly1305 outperformed AES-GCM 256 by approximately 15% and 25%, respectively, achieving higher response rates. This underscores the critical importance of selecting an encryption algorithm aligned with specific requirements and anticipated client loads. In light of these findings, several recommendations can be put forth. Firstly, organizations and service providers should proactively monitor and manage client loads to prevent overload scenarios that can compromise resolver performance. Additionally, the choice of encryption algorithm should be a thoughtful consideration, with a preference for algorithms like AES-GCM 128 or ChaCha20-Poly1305, which demonstrated superior performance under varied conditions. Continuous monitoring, periodic evaluations, and potential adjustments in encryption algorithms can collectively contribute to the sustained optimal performance of DNS recursive resolvers in dynamic and evolving network environments.

REFERENCES

[1] B. Gupta, *Computer and Cyber Security: Principles, Algorithm, Applications, and Perspectives*. CRC Press, Taylor & Francis, 2018.

[2] K. Israry and F. William, "A demonstration of practical DNS attacks and their mitigation using DNSSEC," *Int. J. Wireless Networks and Broadband Technol.*, vol. 9, no. 1, pp. 58-78, 2020.

[3] S. García, K. Hynek, D. Vekshin, T. Čejka, and A. Wasicek, "Large scale measurement on the adoption of encrypted DNS," *ACM*, pp. 1-16, 2021.

[4] T. V. Doan, I. Tsareva, and V. Bajpai, "Measuring DNS over TLS from the edge: Adoption, reliability, and response times," *Int. Conf. Passive and Active Network Measurement*, 2021.

[5] A. Hounsel, K. Borgolte, P. Schmitt, and N. F. Jordan Holland, "Comparing the effects of DNS, DoT, and DoH on web performance," in *Proc. The Web Conf.*, 2020.

[6] T. Boettger, F. Cuadrado, G. Antichi, E. L. Fernandes, I. C. G. Tyson, and S. Uhlig, "An empirical study of the cost of DNS-over-HTTPS," in *IMC '19: ACM Internet Measurement Conf.*, 2019.

[7] A. Jonglez, “End-to-end mechanisms to improve latency in communication networks,” *Networking Internet Architecture*, pp. 1-137, 2021.

[8] R. Houser, Z. Li, C. Cotton, and H. Wang, “An investigation on information leakage of DNS over TLS,” in *The 15th Int. Conf. on Emerging Networking EXperiments and Technologies (CoNEXT ‘19)*, Orlando, FL, USA, 2019.

[9] A. Jonglez, S. Birbalta, and M. Heusse, “Poster: Persistent DNS connections for improved performance,” in *Networking 2019 - IFIP Networking 2019*, pp. 1-2, 2019.

[10] O. Alao, F. Y. Ayankoya, O. F. Ajayi, and O. B. Ohwo, “The need to improve DNS security architecture: An adaptive security approach,” *Inf. Dyn. Appl.*, vol. 2, no. 1, pp. 19-30, 2023.

[11] O. B. Ohwo, F. Y. Ayankoya, O. F. Ajayi, and D. O. Alao, “Advancing DNS performance through an adaptive transport layer security model (ad-TLSM),” *Ingénierie des Systèmes d’Information*, vol. 28, no. 3, pp. 777-790, 2023.

[12] M. S. Islam, M. Sajjad, M. M. Hasan, and M. S. I. Mazumder, “Phishing attack detecting system using DNS and IP filtering,” *Asian Journal of Computer Science and Technology*, vol. 12, no. 1, pp. 16-20, 2023.

[13] P. Banu and K. Kumar, “An experimental study on energy consumption of cryptographic algorithms for mobile hand-held devices,” *Asian Journal of Computer Science and Technology*, vol. 1, no. 1, pp. 91-97, 2012.

[14] V. S. Khandkar, M. K. Hanawal, and S. G. Kulkarni, “Challenges in adapting ECH in TLS for privacy enhancement over the internet,” pp. 1-9, 2022.