



A Multi-Layered Defense AES-GCM and Dragonfly-ECC Based Protocol for Mitigating Spoofing Attacks in DNS Systems

Sura Aljassim¹, Mohammed Abdulridha Hussain¹, Zaid Ameen Abduljabbar^{1,2}, Hamid Ali Abed AL-Asadi¹, Vincent Omollo Nyangaresi^{3,4}, Ali Hasan Ali^{5,6,7}, Abdulla J. Y. Aldarwish¹

¹Department of Computer Science, College of Education for Pure Sciences, University of Basrah, Basrah 61004, Iraq

²Shenzhen Institute, Huazhong University of Science and Technology, Shenzhen 518000, China

³Department of Computer Science and Software Engineering, Jaramogi Oginga Odinga University of Science and Technology, Bondo 40601, Kenya

⁴Department of Applied Electronics, Saveetha School of Engineering, SIMATS, Chennai 602105, Tamil Nadu, India

⁵Department of Mathematics, College of Education for Pure Sciences, University of Basrah, Basrah 61004, Iraq

⁶Technical Engineering College, Al-Ayen University, Thi-Qar 64001, Iraq

⁷Institute of Mathematics, University of Debrecen, Pf. 400, H-4002 Debrecen, Hungary

E-mails: sura.samer@uobasrah.edu.iq mohammed.abdulridha@uobasrah.edu.iq (corresponding author) zaid.ameen@uobasrah.edu.iq hamid.abed@uobasrah.edu.iq vnyangaresi@jooust.ac.ke ali.hasan@science.unideb.hu abdullajas@uobasrah.edu.iq

Abstract: Domain Name System (DNS) is the way that Internet domain names are located and translated into IP addresses. The security architecture of the original design suffers from a lack of strength, as it was designed without built-in security. Among the most pervasive threats is DNS spoofing, in which attackers inject fraudulent responses that redirect users to malicious websites, thereby compromising both security and data integrity. This threat necessitates a defense mechanism that embeds protection directly within the DNS communication workflow. We propose a multi-layered defense approach that protects the communication channel between server and client without needing alteration to the network infrastructure. It combines the Dragonfly PAKE protocol with AES-GCM authenticated encryption to secure DNS responses. Practical experiments through a virtual platform showed that the proposed method is able to thwart attacks better than DoH and DNSSEC. The novelty of this work lies in combining Dragonfly-ECC key exchange with AES-256-GCM encryption in a protocol designed for DNS

response protection, while employing HKDF-based nonce derivation that avoids transmitting nonces over the network. The security analysis based on SVO logic verified the integrity and reliability of the protocol.

Keywords: AES-GCM, SVO logic, DNS security, PAKE, Spoofing attack.

1. Introduction

Computer networks are the foundation of today's digital infrastructure, with billions of devices worldwide connected, providing fast interfaces for communication and data exchange. Computer networks are operated by protocols that govern the exchange of data between multiple layers, ranging from the OSI model's physical layer to TCP/IP application layers [1, 2]. As personal users grow increasingly dependent on distributed cloud services, cyberattacks have increased in frequency and sophistication to make networks relatively vulnerable, particularly with the rapid development of network technology in recent years [3, 2]. This has made network security and DNS especially important since it is the bedrock of modern networks [4].

DNS is a fundamental internet infrastructure responsible for translating domain names into IP addresses [5, 6]. The DNS was introduced in the early 1980s to simplify IP address handling by providing name-based access that leverages a non-numeric format and prioritizes speed over security, which led to its exploitation by multiple attacks throughout these decades [7, 8].

DNS spoofing and cache poisoning attacks are the two most important threats in DNS by changing the system reply, leading users to untrusted sites or malicious websites [9, 10, 3]. Various security tools have been proposed to address these threats, such as DNSSEC, DNS over HTTP (DoH), and DNS over TLS (DoT) [11, 12]. Nevertheless, these solutions have several challenges, including the increasing response time from computational overload and infrastructure problems that prevent efficient deployment of such systems [13, 4].

Due to the constant change of cyber-attacks and the transition from attacking traditional network layers toward exploiting logical vulnerabilities and the application layer [5], There is a strong need to find integrated security mechanisms that target those multiple layers [6, 7]. Although the WPA3 protocol is deployed in modern wireless environments and offers robust protection with Dragonfly [8], it doesn't secure the DNS layer, which can be exploited by DNS spoofing attacks against WPA3 networks [14, 7].

This research presents an integrated security system operating at the multi-layer level, combining a secure key exchange mechanism using Dragonfly-ECC and

an integrated authenticator with AES-256-GCM encryption. It is marked by not requiring modifications to the network infrastructure and provides comprehensive protection from the wireless authentication layer to domain name translation. The contributions of this work are:

- 1) Develop an inbuilt security system to secure domain names against attacks, bridging the divide between wireless and application layer security (Sections 4 and 5)
- 2) Attaining a good trade-off among performance, security, and latency, enabling the system to operate in high-performance environments (Section 6).
- 3) Verification of secure protocols under the condition of SVO to achieve both mathematical and confidentiality assurance (Section 7).
- 4) Designing a nonce derivation such that HMAC-based Key Derivation Function (HKDF) is used for novelty and secrecy enablement without network transmission, which is disparate from the common practices (Sections 2 and 4).

This article aims to address the trade-off between security and efficiency in wireless DNS systems by combining authenticated encryption with faster response times. The cryptographic prerequisites and techniques are introduced in the next section.

2. Technical background and threat model

2.1. Dragonfly-ECC Protocol (Password-Authenticated Key Exchange – PAKE)

This standard protocol, which is implemented using elliptic curve methods, was formally described by RFC 7664 and incorporated into the WPA3 standard [9].

It enables the establishment of a shared session key without the need for secret keys transferring each other in the network [10]. The protocol operates through five steps:

- 1) Both parties initially agree on a secret password for the DNS server and the client, using the SECP256R1 elliptic curve and a public generation point G [11].
- 2) Commit step: Each of the participants chooses a random secret (server b , client a). These private keys are then used to compute public keys according to the mathematical equations $A = a.G$, $B = b.G$. This generates a commit message using the password and random value. The commit message is then sent on the public channel A_c [12].
- 3) Secret key derivation step: Once the public keys are received, each party will derive the shared key by using ECC equations. This means both the client and the server end up with the same key, without ever transmitting it over the network. Moreover, solving logarithmic equations is a challenging issue in the sense that the

private key of a legitimate entity (as it is deployed with each participation) cannot be guessed by an attacker, and this increases the strength [13, 11].

4) Authentication and confirmation: Derived confirmation messages are exchanged to guarantee both sides have derived a similar session key [14].

5) Once the session is set up and the key is matched on both server and client sides, a and b are immediately removed as depicted in Fig. 1. But the session key persists as long as the session is on, being part of the encryption process in the AES-256-GCM Algorithm.

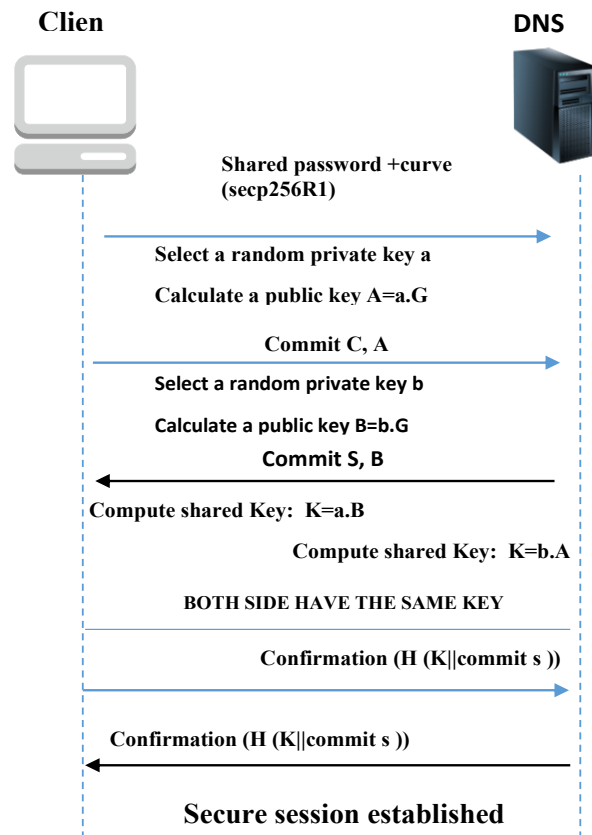


Fig. 1. Dragonfly-ECC Algorithm

2.2. AES-GCM (Advanced Encryption Standard-Galois/Counter Mode)

It is a symmetric encryption algorithm that combines encryption and authentication in a single operation and has been standardized in (NIST SP 800-38D). It operates at high speed, making it suitable for fast responses, as with DNS [15]. Its use is pervasive in many current network protocols, including TLS 1.3 and IPsec [16]. It utilizes a session key (K_{shared}), which is the encryption key, generated from Dragonfly, as no extra secret material transformations other than nonce generation

are needed [15]. It simplifies key management and reduces associated overhead [17]. The encryption and authentication can both happen in one step [17]. The operation of the algorithm relies on two critical components apart from encryption: (a) deriving a unique nonce for each DNS query which is calculated by both receivers according to RFC5869, i.e., $\text{nonce} = \text{HKDF}(\text{K_shared}, \text{counter}, 12 \text{ bytes})$, and it never departs over the network; (b) authentication GCM tag, where the tag travels in parallel with the ciphertext over the network [18].

2.2.1. Nonce

The proposed system uses nonces to ensure both non-reusability and confidentiality. The nonce is derived as $\text{nonce} = \text{HKDF-Expand}(\text{session key}, \text{counter}, 12 \text{ bytes})$ [18]. The session key is the starting seed for the creation of a Nonce and is associated with an increasing counter on every DNS request [19]. The size is guaranteed to be 12 bytes for use with AES-GCM (which specifies the nonce size to be 96 bits). The nonce is not transmitted over the network and is obtained via Standardized Key/Context Separation via HKDF [18, 20].

2.2.2. GCM-tag

The authentication tag of GCM mode using the field-terminating hash function (Galois Hash) applied to the encryption output of the nonce value [21] $\text{Tag} = \text{GHASH}(H, C) \oplus E(K, N)$, where: $H = \text{AESK}(0, 128)$ is hash key; C is encrypted plaintext [21].

2.3. Tools

The experimental setup uses VMware to generate a virtual environment that consists of the following components:

DNS server: A virtual machine was configured to generate and send encrypted responses within the test environment.

Client/Victim: A virtual machine was configured within a client environment to receive DNS queries, test responses, and verify system security.

Attacker: A virtual attacker machine was configured to execute attacks and simulate spoofing/poisoning scenarios.

Python and utilized libraries: The system uses Python and a set of libraries to perform operations. Based on the defined implementation setup, the anticipated threat landscape is described to clarify the system security objective.

2.4. Threat model

This work assumes an active attacker operating on the local network who eavesdrops on communications between the client and DNS server and, because it has exploited its position between the two parties, can perform ARP spoofing attacks as well as send responses with DNS spoofing before the legitimate response or try to intercept, modify, or forge packets. The attacker, however, is not able to decrypt the AES-256-GCM encryption, derive the shared secret from public keys, and generate a valid GCM authentication tag without the secret key. The discussions in this paper do not include the direct compromise of endpoints or physical attacks.

3. Related works

Previous work on encryption, authentication, and vulnerability disclosure is described in this section, which has been addressed by the proposed system.

Yu et al. [22] proposed the DNS-Sensor system, a distributed sensor network that continuously monitors DNS cache records by comparing them against authoritative data to detect cache poisoning anomalies in real time. While the system demonstrates accurate detection, it operates as a post-entry monitoring layer and does not provide cryptographic protection at the point of response generation.

Blaise, Wumi and Alfred [17] analyzed the impacts of symmetric encryption algorithms on the performance of encrypted DNS protocols for DNS over TLS and DNS over QUIC frameworks. The study systematically compared well-known AEAD encryption algorithms like AES-GCM and ChaCha20 Poly1305, confirming the hardware acceleration advantage of AES-GCM. Their analysis remained at the transport layer and did not address application-layer authenticated encryption or dynamic key exchange integration.

Heftrig, Shulman and Waidner [13] exposed a critical DNSSEC vulnerability where off-path attackers exploit IP fragmentation and algorithm downgrade to disable authentication. Results showed that 7756 of 43,181 domains were vulnerable, with an overall attack success rate of 18.84% on disabling DNSSEC. This finding demonstrates that signature-based integrity verification alone is insufficient to guarantee DNS security.

Heftrig et al. [23] revealed a harmful algorithmic complexity attack named KeyTrap targeting DNSSEC infrastructure. The experimental results showed that sending responses with complicated key strings causes name resolvers to fall into computational loops with 100% CPU utilization, triggering a single-packet denial of

service. This protocol-level flaw cannot be resolved through implementation patches alone.

Bellare and Tackmann [24] developed a security analysis of the AES-GCM algorithm for the TLS 1.3 protocol. They proved through mathematics that when the nonce is used more than once, it leads to full security compromise, since the adversary can infer the authentication key and forge authenticated messages. Their analysis motivated the deterministic nonce derivation approach adopted in the proposed system through HKDF, ensuring non-reuse without network transmission.

Vanhoef [25] performed a comprehensive security evaluation of the Dragonfly protocol in WPA3, highlighting several issues collectively known as the Dragonblood vulnerability. They showed how timing side channels can leak information, allowing downgrade attacks by forcing weak ECC combinations. While Dragonblood addresses wireless authentication layer vulnerabilities, the proposed system extends Dragonfly protection to DNS through application-layer authenticated encryption with HKDF-based nonce derivation.

Dawood et al. [26] discussed security issues and implementation limitations of DoH. The study revealed that strong encryption provided by DoH can be exploited to hide malware activities, and highlighted the computational overhead affecting network efficiency in resource-constrained environments. These limitations underscore the need for lightweight authenticated encryption approaches, like AES-256-GCM.

Patil and Shyamala [27] conducted a comprehensive survey reviewing 225 studies on securing the Social Internet of Things covering the period from 2014 to 2025. Their taxonomy-driven analysis mapped security requirements, attack types, and defense mechanisms across protocol layers, while evaluating the integration of blockchain, AI/ML, and edge computing for scalable and decentralized trust management. The survey highlights that lightweight dynamic trusted models and privacy-preserving service discovery remain open challenges, reinforcing the need for cryptographic solutions that operate efficiently on resource-constrained devices, a design objective shared by the system proposed in this paper.

Rotună, Sacala and Alexandru [28] examined the use of machine learning to identify botnet-generated DNS queries by analyzing patterns in query logs. The classifiers showed reasonable accuracy in separating normal traffic from suspicious resolution requests, though the system functions as a detection layer that flags problems after they have already entered the network rather than blocking them through cryptographic means at the point of response generation.

Luo [29] proposed a blockchain-based Moving Target Defense system employing End Hopping with the PBFT consensus protocol to dynamically rotate

communication endpoints, providing proactive defense against flooding attacks. Their approach does not address DNS-layer spoofing or response-level encryption.

Amirkhanova et al. [30] designed a lightweight end-to-end AES-256-GCM encrypted pipeline for IIoT deployments on ESP32 and Raspberry Pi over MQTT, demonstrating practical authenticated encryption on resource-constrained hardware without performance degradation.

Abu Al-Haija, Alohalay and Odeh [31] designed a two-stage classifier that separates normal DoH traffic from malicious flows attempting to abuse encrypted DNS channels for covert data transfer. Their experiments confirmed that DoH encryption creates a monitoring blind spot that attackers exploit, reinforcing the need for application-layer authentication rather than transport-layer encryption alone.

Kundu et al. [32] proposed a formal post-quantum cryptographic framework for layered network protocols, proving that a single quantum-safe layer suffices for payload confidentiality while authentication requires full-stack migration. Their analysis does not address DNS-specific response protection or lightweight key exchange.

The comparative analysis presented in Table 1 reveals several critical gaps in existing DNS security research, while individual cryptographic primitives such as AES-GCM and Elliptic Curve Cryptography have demonstrated strong security properties in isolation, no existing study integrates password-authenticated key exchange with authenticated symmetric encryption specifically for DNS response protection. Current solutions such as DNSSEC remain vulnerable to fragmentation-based downgrade attacks and algorithmic complexity exploits, indicating that digital signature-based integrity verification alone is insufficient to guarantee DNS security. Encryption-only approaches like DoH introduce substantial computational overhead and latency while failing to address replay or response forgery at the application layer. None of the reviewed studies provides formal verification of their protocol security claims through established formal logics such as SVO or BAN. These identified gaps collectively motivate the design of the proposed system, which combines Dragonfly-ECC password-authenticated key exchange with AES-256-GCM authenticated encryption, incorporates deterministic nonce derivation via HKDF to prevent replay attacks, and provides formal security verification through SVO logic to establish provable guarantees of authentication, integrity, and key security.

Table 1. Summary of the main characteristics and limitations of the reviewed studies

Source	Year	Method / Algorithm Used	Strengths	Limitation
Bellare and Tackmann [24]	2016	AES-GCM security analysis for TLS 1.3	Formal proof that nonce reuse in AES-GCM leads to full authentication compromise	Analysis focuses on multi-user TLS settings; does not propose a DNS-specific nonce management solution
Vanhoef [25]	2019	Dragonfly (WPA3)	Identifies implementation weaknesses	Covers the wireless layer only; does not extend Dragonfly protection to DNS or application-layer encryption
Dawood et al. [26]	2023	DNS over HTTPS (DoH)	Strong privacy protection through encryption	Encryption exploitable to conceal malware; high computational overhead in constrained environments
Heftrig, Shulman and Waidner [13]	2023	DNSSEC analysis (fragmentation-based exploit)	Reveals a critical DNSSEC vulnerability	Signature-based integrity verification alone is insufficient; no authenticated encryption proposed
Yu et al. [22]	2025	DNS-Sensor distributed real-time cache poisoning detection	Accurate real-time detection of cache poisoning anomalies using distributed sensor comparison	Detection only after cache entry; no cryptographic prevention at point of response generation
Abu Al-Haija, Alohalay and Odeh [31]	2023	Two-stage hybrid ML classifier for malicious DoH detection	Effective separation of benign and malicious DoH flows	Focuses on traffic classification rather than cryptographic prevention
Heftrig et al. [23]	2024	DNSSEC/ signature verification analysis	Identifies a severe algorithmic DoS vulnerability	Protocol-level DoS flaw in DNSSEC; cannot be resolved through implementation patches alone
Blaise, Wumi and Alfred [17]	2024	AES-GCM & ChaCha20-Poly1305 with DoT/DoQ	Good performance, balanced security and latency, practical evaluation	Transport-layer analysis only; no application-layer key exchange addressed
Patil and Shyamala [27]	2025	Taxonomy-driven survey (Blockchain, AI/ML, Edge Computing)	Comprehensive mapping of security across protocol layers, evaluates decentralized trust management	Lightweight dynamic trusted models and privacy-preserving discovery remain open challenges
Rotun, Sacala and Alexandru [28]	2025	Machine Learning (DNS query log analysis)	Reasonable accuracy in identifying botnet-generated queries and separating normal from suspicious traffic	Functions as a detection layer post-entry rather than blocking attacks at the point of response
Luo [29]	2026	Blockchain-based MTD (End Hopping + PBFT consensus protocol)	Eliminates single point of failure; provides proactive defense against flooding attacks	Focused on flooding attacks; does not address DNS-layer spoofing or response-level encryption
Amirkhanova et al. [30]	2026	AES-256-GCM E2E encryption pipeline (ESP32 + MQTT + Raspberry Pi)	Practical authenticated encryption on resource-constrained IIoT hardware without performance loss	Pre-shared key provisioning only; no dynamic key exchange mechanism
Kundu, Chakraborty and Kompella [32]	2026	Formal PQC composition framework (lattice algebra across network layers)	Proves confidentiality and authentication compose via dual lattice operators; demonstrates one Q-Safe layer suffices for payload protection	General cross-layer analysis; does not address DNS-specific authenticated responses or lightweight PAKE for constrained environments

4. Proposed secure DNS system

In this paper, we present a DNS protection system that implements secure key exchange using the Dragonfly-ECC protocol with authenticated encryption using the AES-256 GCM Symmetric Algorithm [32]. The system guards against man-in-the-middle, cache poisoning, DNS spoofing, and replay attacks. The designed system operates at the application layer and concentrates on secure DNS responses by authenticated encryption with dynamic session key creation.

4.1. System operational phases

Phase 1. Key generation. The client and server use the Dragonfly protocol to generate a shared key (K_{shared}). This key is used to encrypt all responses during the session.

Key exchange. Establishing trust between the client and server is a fundamental part of each integrated security system. Therefore, this research adopts the Dragonfly protocol, based on Elliptic Curves (ECC), to ensure the generation of dynamic session.

Keys. Here, we review the mathematical and logical steps for generating these keys and how to verify the identities of connected parties.

Initialization. Both the client and server agree on common parameters before connecting. These parameters are:

Elliptic-curve (SECP 256R1 – 256 bit);

Generator point on the curve (G);

Shared password.

1. Commit. This stage authenticates both parties using a shared password

(Client). The client selects a random private key (a) to calculate the public key according to the equation $A = a.G$. Then, it generates a message $\text{Commit}_C = H(\text{password} \parallel \text{NC})$, where NC is a random number. Finally, it sends ($\text{Commit} - C, A$) to the server.

(Server). The server selects a random private key (b) to calculate the public key according to the equation $B = b.G$, then it generates a message $\text{commit}_S = H(\text{password} \parallel \text{NS})$, where NS is a random number. Finally sends (Commit_S, B) to the client.

2. Deriving the shared key (K_{shared})

The client calculates $K_{\text{shared}} = a.B = a.(b.G)$. The server calculates $K_{\text{shared}} = b.A = b.(a.G)$

Due to the properties of elliptic curves, $a.B = b.A = (a.b).G$; this feature ensures that both parties receive identical keys without the need to transfer private keys, as shown in Fig. 2.

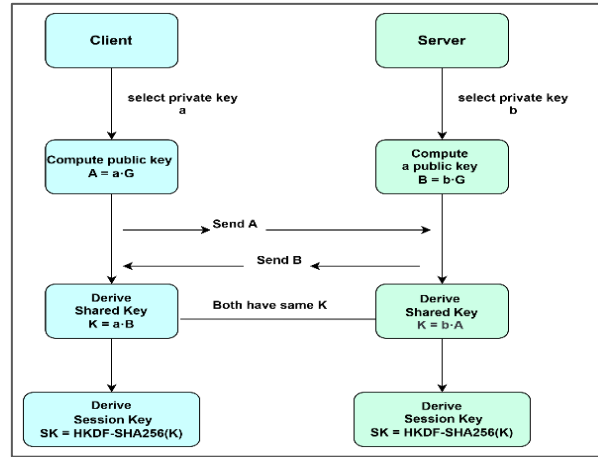


Fig. 2. Key generation and exchange

3. Confirmation to ensure that both parties derive the same key:

The client sends. $\text{Confirm_C} = H(K_{\text{shared}} \parallel \text{Commit_C} \parallel \text{Commit_S})$

The server sends. $\text{Confirm_S} = H(K_{\text{shared}} \parallel \text{Commit_S} \parallel \text{Commit_C})$

Each party verifies the received confirmation message. This step is to ensure that both parties have derived an initial session key before proceeding with encrypted communication.

NS-NC. Two random numbers are generated by the client and server, respectively, during the key exchange phase

Key lifecycle. Keys (a, b) are automatically deleted after the shared key is derived, ensuring perfect forward secrecy. The shared key remains valid for the duration of the session and is deleted after the session ends. This lifecycle management ensures that the session key is short-lived and used exclusively within a single session of communication.

Phase 2. Secure connection. After generating the shared key, the client sends an unencrypted DNS query on port 53. This design maintains compatibility with standard DNS operation, while the corresponding DNS responses are protected using authenticated encryption. The server processes the query and responds with the requested IP address, but encrypts it using the AES-GCM algorithm with the session key derived in Phase 1. The encrypted response includes a GCM tag for authentication, a timestamp for recency, and a nonce with a counter used to generate it.

Port 53 is explicitly designated for receiving DNS queries, while port 5454 is for sending and receiving encrypted responses only. Two-port methods are employed to isolate encrypted traffic from traditional traffic, as shown in Fig. 3.

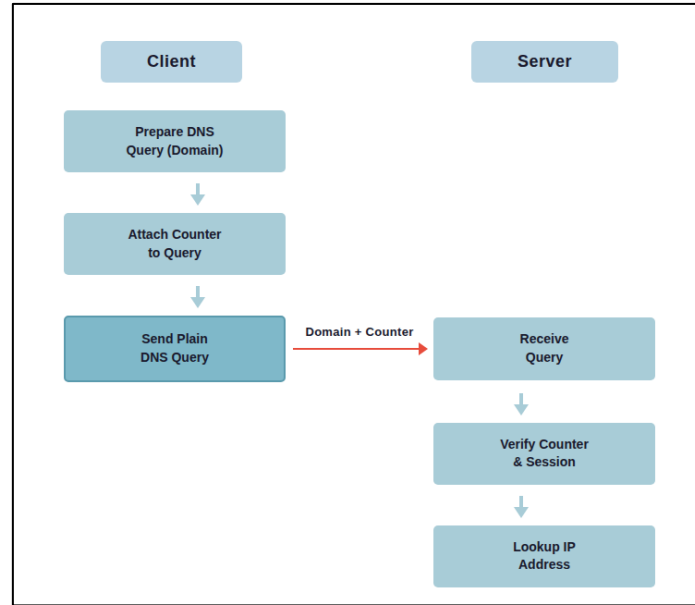


Fig. 3. Encrypted transmission DNS query

Phase 3. Response verification. The client receives the encrypted response and derives the nonce using the same counter.

Receiving the encrypted packet via port 5454, which includes a packet of encrypted data for the requested IP address, GCM authentication tag, a counter for nonce derivation, and a timestamp.

Verifying the recency of the response: The client matches the time provided with the response with the current time within a predetermined timeframe. If this timeframe is exceeded, the response is rejected.

After completing the previous two steps, the client derives the nonce value using the HKDF SHA256 algorithm based on the shared key and the counter value. This derivation ensures the generation of the same nonce value for both the server and the client without the need to transfer it across the network. It checks the nonce against previously used values stored during the same session. If it is repeated, the request is rejected immediately. The nonce values are deleted as soon as the session ends.

Then the GCM tag is recalculated by the client using the same inputs: the encrypted data, the shared key, and the nonce. The recalculated tag is then compared with the received tag. Any discrepancy indicates data modification or an attempted forgery, and the response is immediately rejected.

After all previous verification steps are completed, the encrypted data is decrypted using the AES-256-GCM algorithm to retrieve the requested IP address. The final stages of DNS response processing are shown in Fig. 4.

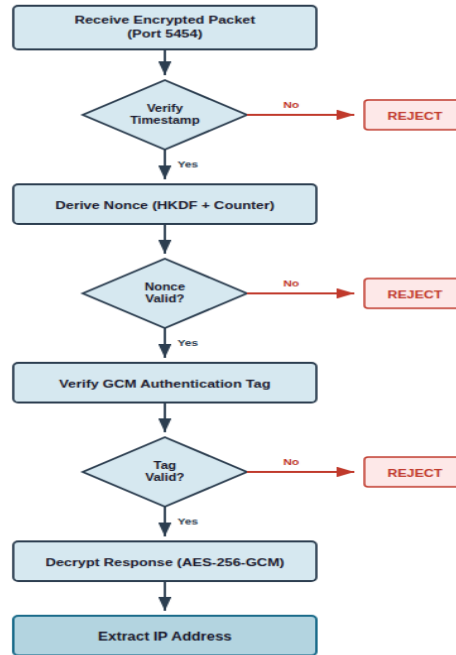


Fig. 4. Secure DNS response verification flow

Once all verification steps are finished and the requested IP address is obtained, the system updates the session status by increasing the counter in preparation for the next query, and preventing the reuse of the same values.

5. System architecture and implementation

The proposed system consists of three main components deployed within a VMware-based virtual environment designed to emulate a controlled DNS communication scenario.

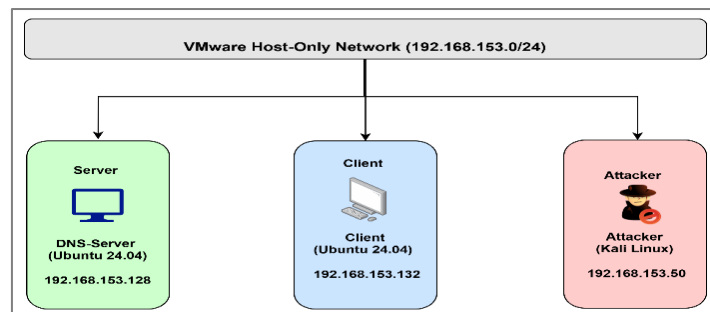


Fig. 5. Virtual network architecture for DNS security testing responses

A) DNS Server

- Operating System: Ubuntu 20.04
- Function: Processing DNS queries and sending encrypted IP addresses to the client
- Port: 53 for receiving unencrypted queries, and port 5454 for sending encrypted responses.

B) Client

- Operating System: Ubuntu 20.04
- Function: Sends DNS queries to the server and receives encrypted responses
- Generates cryptographic keys and performs response verification

C) Attacker

- Operating system: Kali Linux
- Function: Simulates an adversarial node to evaluate system behaviour under hostile network conditions.
- Tools: Ettercap, Scapy, ARP Spoofing, as shown in Fig. 5.

Implementation details. Security overview: The system implements multiple layers of security to secure the DNS communication.

The first is Perfect Forward Secrecy, where private keys are deleted immediately when K_{shared} is generated, and intercepting one session does not compromise others.

The second is Protection from MITM: The attacker cannot calculate a valid commit without the password, and they cannot derive K_{shared} from public keys A and B alone.

The proposed design offers a comprehensive set of security features: confidentiality through encryption using the AES-256-GCM Algorithm, authentication and data integrity via tagging, recency via timestamp, nonce replay resistance via time-stamping, and resistance to forgery and spoofing due to the inability to generate a valid response without K_{shared} . This results in the tradeoff between security and performance, making it adapted to future applications in practical scenarios.

6. Results

The practical value of the research is summarized through the results of practical tests conducted on the virtual environment, and comparing them with DNSSEC and DoH systems, focusing on response time and resource consumption. The research results are in Table 2.

Table 2. Performance comparison of DNS security systems under stress conditions

System	Average response time	Queries per 1 s	Security level
Plain DNS	0.930	1075.27	None
DNSSEC	52.00	19.23	Integrity only
DoH	263.00	3.80	Full Encryption + Authentication
Proposed system	44.12	22.67	Full Encryption + Authentication

The proposed system provides full encryption and authentication comparable to DoH, while achieving significantly lower response latency. Compared to DNSSEC, the system offers both encryption and integrity verification rather than integrity alone.

The system produces an average response time of 44.12 ms (or 22.67 queries per 1 s), which is 1.18 times faster than DNSSEC and 5.96 times faster than DoH. This is attributable to its session-based design (one handshake), and the overhead is amortized across several queries.

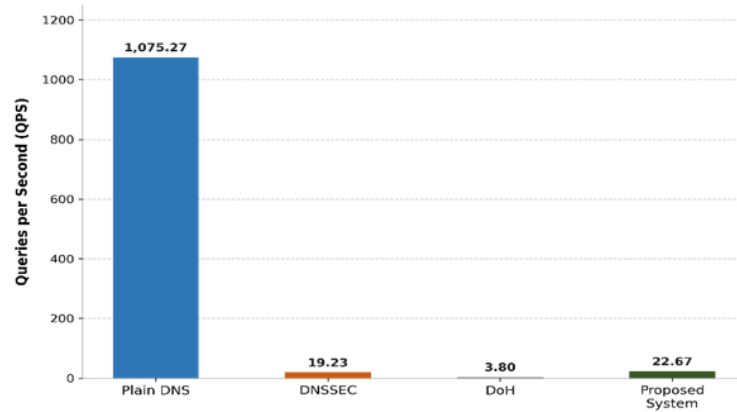


Fig. 6. Throughput

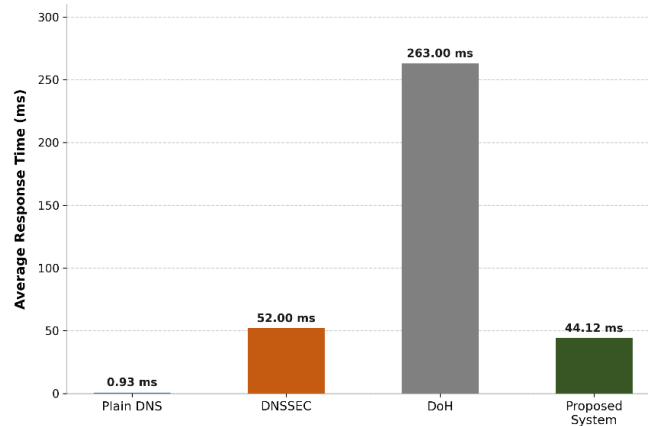


Fig 7. Response time

6.1. Scalability under concurrent load

The per-query encryption cost held below 5 ms across all load levels with a standard deviation of 0.37 ms, confirming that AES-256-GCM processing does not degrade under concurrency. The handshake cost rises with client count but is incurred once per session, making its amortized impact negligible over multiple queries. All systems were evaluated under identical 60-second stress test conditions.

Table 3. Proposed system performance under concurrent load (no attack)

Metric	100 clients	500 clients	1000 clients	Mean	STD/DV
Success (%)	100	100	100	100	0
QPS	258.00	273.80	246.70	259.50	± 13.68
AVG/HS (ms)	23.5	117.11	235.54	125.38	± 106.12
AVG/Q (ms)	4.43	4.07	4.80	4.43	± 0.37
Avg total (ms)	27.42	121.18	239.54	129.38	± 106.15

Table 4. Resource utilization under stress test

Metric	Plain DNS	DNSSEC	DoH	Proposed system
Queries per 1 s (QPS) throughput	1075.27	19.23	3.80	22.67
CPU utilization (%)	0.5	7.4	1.8	1.95
Memory usage (MB)	7.33	3.58	3.59	7.34
User time (s)	28.88	31.28	14.28	1.98
System time (s)	17.64	16.97	3.16	0.79
Total CPU time (s)	46.52	48.25	17.44	2.77
Context switches	31.978	36.401	5.743	2.438

From the results in Table 4, plain DNS can reach the maximum request rate since it does not involve any encryption or security check, while there is more processing time and CPU usage for handling the DOH, which adopts encryption and DNSSEC related to digital signatures. The proposed system preserved balanced performance by employing one-time key exchange plus symmetric encryption; it consumes less CPU time to perform Dragonfly key exchange for a session, and follows that up with lightweight operation of data processing using the AES-GCM when querying users' data than full processing utilizing asymmetric one. In terms of memory consumption, the proposed system exhibited memory consumption comparable to conventional DNS, indicating its efficient design and low software overhead during the defined stress-test period.

6.2. DNS Query and response size comparison

The total overhead of 420 bytes (140-byte query and 280-byte response) is a calculated trade-off to ensure multi-layered security through AES-256-GCM encryption, 16-byte GCM tags, and session metadata. This introduces a marginal

increase in packet size compared to unencrypted DNS. It provides a superior security posture against spoofing and replay attacks that standard protocols like DNSSEC or DoH do not fully mitigate. Statistical stability in Table 3 was verified through 1,000 controlled query cycles in a virtualized environment to ensure reliability. Thus, the system achieves an optimal balance, providing high-level security within an acceptable operational latency.

Table 5. DNS query and response size analysis

System	Query size (Bytes)	Response size (Bytes)	Total overhead
Plain DNS	50	100	150
DNSSEC	150	450	600
DOH	200	500	700
Proposed system	140	280	420

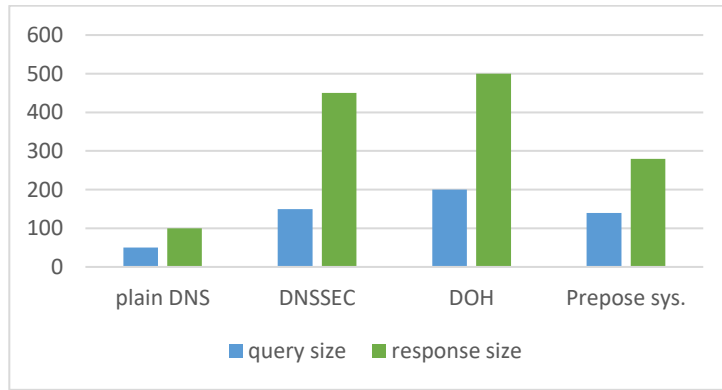


Fig. 8. Query size / Response size

In addition to empirical verification, formal analysis is carried out in order to justify the established security claims of the protocol.

7. Formal security verification using SVO logic

7.1. Introduction

The security properties of the proposed protocol are formally verified using SVO logic, a formal reasoning framework designed to analyse authentication and key exchange protocols. SVO logic is selected for this analysis because it provides native support for authenticated key exchange protocols, symmetric encryption with derived nonces, and session key freshness verification, all of which are central components of the proposed system. The proposed system involves two main communicating entities, the Client (C) and the DNS Server (S).

7.2. Notation and definitions

The following notation is used throughout the formal analysis. Each symbol corresponds to a specific logical construct within SVO logic that maps directly to a component of the proposed DNS security protocol:

- P, Q denote the communicating principals.
- $P \models M$ indicates that principal P believes M .
- $P \ni M$ means that principal P possesses M .
- $\#(M)$ denotes freshness.
- $P \leftrightarrow_k Q$ signifies a shared session key.
- $P \Rightarrow M$ indicates jurisdiction.
- $P \triangleleft M$ means receiving a message.
- $P \sim M$ indicates that P once said M .

7.3. Assumptions

The following assumptions define the initial conditions under which the protocol operates. A specific design decision in the proposed system justifies each assumption:

- A1: $C \ni \text{password}, S \ni \text{password}$
- A2: $C \models \#(N_C), S \models \#(N_S)$
- A3: $C \ni K, S \ni K$
- A4: $C \models (S \Rightarrow IP)$
- A5: $C \models \text{sync}(\text{counter}), S \models \text{sync}(\text{counter})$.

7.4. Inference rules

The following inference rules are adopted from SVO logic and applied to analyse the security properties of the proposed protocol.

Rule 1. Message Meaning Rule (MMR). If a principal believes that a key is shared with another principal, and receives a message encrypted with that key, then it concludes that the other principal once said that message.

Formal representation. $P \models (P \leftrightarrow_k Q), P \triangleleft \{X\}_k \rightarrow P \models Q \mid \sim X$.

Application. When the client C receives a DNS response encrypted with K_{shared} and successfully verifies the GCM authentication tag, it concludes that the server S is the originator of that message.

Rule 2. Freshness Propagation Rule (FPR). If a component of a message is fresh, then the entire message is considered fresh.

Formal representation. $\#(X), X \subset Y \rightarrow \#(Y)$.

Application. Since the nonce derived via HKDF using a counter is fresh for each query, the entire DNS response containing that nonce is also fresh, ensuring resistance against replay attacks.

Rule 3. Jurisdiction Rule (JR). If a principal trusts another principal regarding a statement and believes that the latter believes that statement, then it accepts the statement as true.

Formal representation. $P \models (Q \Rightarrow X), \quad P \models Q \models X \rightarrow P \models X$

Application. Since the client C trusts the server S regarding DNS resolution, and verifies that S believes the correctness of the IP mapping through authenticated encryption, C accepts the resolved IP address as authentic.

7.5. Protocol steps

The protocol execution is expressed as a sequence of SVO statements corresponding to the three operational phases defined in Section 4.

Phase 1. Key exchange

- Step 1.** $C \triangleleft \text{Commit_C}$ (Client generates commit message)
- Step 2.** $S \triangleleft \text{Commit_C}$ (Server receives client commit)
- Step 3.** $S \triangleleft \text{Commit_S}$ (Server generates commit message)
- Step 4.** $C \triangleleft \text{Commit_S}$ (Client receives server commit)
- Step 5.** $C \ni K$ (Client derives the session key)
- Step 6.** $S \ni K$ (Server derives the session key)

Phase 2. DNS query

- Step 7.** $C \ni \text{Domain, Counter}$
- Step 8.** $C \triangleleft \text{DQ} = (\text{Domain, Counter})$
- Step 9.** $S \triangleleft \text{DQ}$

Phase 3. Encrypted response

- Step 10.** S derives nonce = HKDF (K, Counter, 12)
- Step 11.** $S \ni \{\text{IP, Timestamp}\}_k, \text{GCM_tag, Counter}$
- Step 12.** $S \triangleleft \text{DR} = (\{\text{IP, Timestamp}\}_k, \text{GCM_tag, Counter})$
- Step 13.** $C \triangleleft \text{DR}$
- Step 14.** C derives nonce = HKDF (K, Counter, 12)
- Step 15.** C verifies GCM_tag and Timestamp
- Step 16.** C decrypts $\{\text{IP, Timestamp}\}_k$.

7.6. Formal derivation of security beliefs

This subsection demonstrates how the inference rules are applied to the protocol steps to derive the target security beliefs.

Derivation 1. Mutual Key Agreement. From A3, both C and S possess K. Since NC is fresh (A2), applying FPR yields $\#(K)$. By Steps 1-4, the commit exchange verifies both parties used the same password. Applying MMR: $C \models (C \leftrightarrow_k S)$ and $S \models (S \leftrightarrow_k C)$.

Derivation 2. Query Authentication. From Steps 7-9, S receives DQ from C over the authenticated channel (Derivation 1). The counter is fresh (A5), and applying FPR extends freshness to the query. Therefore: $S \models C \sim (\text{Domain}, \text{Counter})$ and $S \models \#(DQ)$.

Derivation 3. Response Authenticity. From Steps 10-16, C receives DR and verifies the GCM tag. Applying MMR: $C \models S \sim (\text{IP}, \text{Timestamp})$. The counter is fresh (A5), so by FPR: $\#(DR)$. Finally, applying JR with A4: $C \models \text{IP}$.

7.7. Verified security goals

The formal derivation establishes four security properties:

G1 (Key Agreement): $C \leftrightarrow_k S$ both derive the same session key through Dragonfly-ECC (Derivation 1).

G2 (Key Security): K is known only to C and S, never transmitted; private keys are deleted after derivation (A3).

G3 (Response Authenticity): $C \models (S \Rightarrow \text{IP})$, only the legitimate server can produce valid GCM-authenticated responses (Derivation 3).

G4 (Replay Prevention): $\#(\text{nonce})$ per query counter-based HKDF derivation binds each response to a unique query (Derivations 2, 3).

8. Informal security analysis

Claim 1. The secure DNS proposed system is proven to be immune against the DNS spoofing attack, as the client only accepts the responses that are encrypted with a correct session key (k_{shared}) and an authenticated tag.

Justification. If an attacker could submit a forged DNS response, for instance, they would not possess the session key generated during the Dragonfly key exchange. If they don't have that key, they can't produce a valid AES-256-GCM authentication tag; the client just rejects any response it's not capable of verifying.

Claim 2. The proposed system prevents a replay attack.

Justification. First, an increment counter is used, starting from an initial value. On both sides, the nonce for every response is computed by the system. The timestamp is a fixed-duration epoch associated with every response. If a duplicate is detected, the system rejects the response immediately.

Claim 3. The proposed system prevents a Man-in-the-Middle Attack MITM.

Justification. The system operates at the application layer, rendering intercepted packets computationally infeasible to decrypt or modify without the session key.

Proof: Even if the attacker has successfully performed an ARP spoofing attack, they cannot read or change the intercepted packets since protection is application-level and occurs regardless of attacking methodologies. The packet is secured at all times since it is encrypted. And it cannot be read without the secret key. Any tampering is discarded without a valid authentication tag, as shown in Fig. 6.

Claim 4. The system has a strict policy that prevents the storage of any unencrypted, unauthenticated, or expired responses, and the cache contains only trusted data.

Justification. The proposed system enforces strict response validation before caching. Each response that fails GCM tag verification, timestamp checking, or nonce validation is immediately rejected without being stored. Consequently, only cryptographically authenticated and fresh responses are accepted, maintaining cache coherence against illegitimate resource record poisoning.

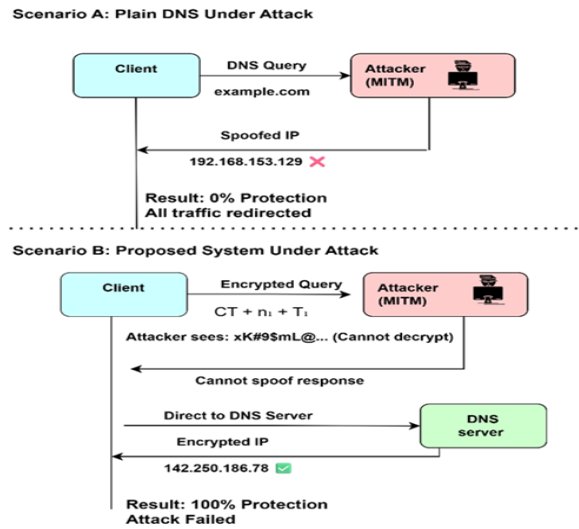


Fig. 9. MITM attack scenario

9. Limitations

Although the proposed system demonstrates strong security properties and practical performance, several limitations should be acknowledged that define the current scope of this work:

1. The evaluation was performed in a controlled virtual environment that may not fully reflect the scale and variability of production DNS infrastructure.
2. The current implementation has been tested with a limited number of concurrent clients, and behavior under extremely high query volumes remains unexplored.
3. The system has not yet been evaluated on resource-constrained devices such as IoT endpoints where memory and processing power are restricted.

10. Future work

Building on the identified limitations, several directions are planned to extend the applicability and robustness of the proposed system:

1. Evaluating the system on real, geographically distributed DNS infrastructure.
2. Replace ECC with a quantum computing-resistant algorithm (ML-KEM).
3. Adding an AI layer to detect new unknown attacks.

11. Conclusion

In this study, we proposed a cryptographic defense system to address DNS spoofing vulnerabilities based on encrypting responses as well as protecting them from tampering during transmission through the network. The results showed that the mechanism could successfully deter impersonation, spoofing, and man-in-the-middle attacks with low computation overhead, without significantly degrading response time. On top of that, the session system and symmetric encryption lightened this load and struck a balance between speed and security. This is still a noticeable increase in memory usage compared to conventional systems, but well within reason, especially since it does not grow with the number of requests. The main contribution of this research is the integration of Dragonfly-ECC password-authenticated key exchange with AES-256-GCM authenticated encryption within a single protocol designed for DNS response security, further distinguished by HKDF-based nonce derivation that eliminates nonce transmission over the network. The proposed system achieves both encryption and authentication while maintaining a competitive response time compared to existing approaches. Additionally, the formal verification through SVO logic provides a structured basis for validating the protocol security properties. In summary, the proposed system contributes a practical and flexible way to improve DNS security and demonstrates potential for deployment in emerging network environments.

References

1. Alzoubi, Y. I., A. Aljaafreh. Blockchain-Fog Computing Integration Applications: A Systematic Review. – Cybernetics and Information Technologies, Vol. **23**, 2023, No 1, pp. 3-37.
2. Hussain, M. A., H. Jin, Z. A. Hussien, Z. A. Abduljabbar, S. H. Abbdal, A. Ibrahim. Enc-DNS-HTTP: Utilising DNS Infrastructure to Secure Web Browsing. – Security and Communication Networks, 2017. DOI: 10.1155/2017/9479476.
3. Blancaflor, E. B., J. O. Duldulao, J. V. E. Espeño, G. S. M. Patag, M. A. T. Menor, G. L. Intal. Advanced Phishing Techniques: Analyzing Adversary-in-the-Middle and Browser-in-the-Browser Attacks in Modern Cybersecurity. – Cybernetics and Information Technologies, Vol. **25**, 2025, No 1, pp. 55-77.
4. Zhrav, D. S., M. A. Hussain, Z. A. Abduljabbar, V. O. Nyangaresi, A. J. Y. Aldarwish. Chronological Review of MITM Attacks: Challenges, Solutions and Recommendations. – In: Proc. of Computer Science on-line Conference, Springer, 2025, pp. 202-220. DOI: 10.1007/978-3-032-03406-9_13.
5. Btoush, A., A. Abadleh, A. A. Alkasasbeh et al. The Intrusion Detection and Recovery of Deauthentication Frame in WPA3 SAE. – Research Square (Preprint), 2024. DOI: 10.21203/rs.3.rs-4856594/v1.
6. Hussain, M. A., H. Jin, Z. A. Hussien, Z. A. Abduljabbar, S. H. Abbdal, A. Ibrahim. DNS Protection against Spoofing and Poisoning Attacks. – In: Proc. of International Conference on Information Science and Control Engineering, 2016, pp. 1308-1312. DOI: 10.1109/ICISCE.2016.279.
7. Hussain, M. A., et al. Web Application Database Protection from SQLIA Using Permutation Encoding. – In: Proc. of ACM International Conference Proceedings Series, 2021, pp. 13-21. DOI: 10.1145/3459955.3460594.
8. Iyengar, N. C. S. N., G. Ganapathy. Trilateral Trust-Based Defense Mechanism against DDoS Attacks in Cloud Computing Environment. – Cybernetics and Information Technologies, Vol. **15**, 2015, No 2, pp. 119-140.
9. Lancrenon, J., M. Škrobot. On the Provable Security of the Dragonfly Protocol. – In: Information Security (ISC'2015), Lecture Notes in Computer Science, Vol. **9290**, Springer, 2015. DOI: 10.1007/978-3-319-23318-5_14.
10. Clarke, D., F. Hao. Cryptanalysis of the Dragonfly Key Exchange Protocol. – IET Information Security, Vol. **8**, 2014, pp. 283-289. DOI: 10.1049/iet-ifs.2013.0081.
11. Alharbi, E., N. Alsulami, O. Batarfi. An Enhanced Dragonfly Key Exchange Protocol against Offline Dictionary Attack. – Journal of Information Security, 2015, pp. 69-81. DOI: 10.4236/jis.2015.62008.
12. Braga, D. D. A., N. Kulatova, M. Sabt, P. Fouque, K. Bhargavan. From Dragondoom to Dragonstar: Side-Channel Attacks and Formally Verified Implementation of WPA3 Dragonfly Handshake. – In: Proc. of IEEE European Symposium on Security and Privacy (EuroS&P), 2023, pp. 707-723. DOI: 10.1109/EuroSP57164.2023.00048.
13. Heftrig, E., H. Shulman, M. Waidner. Off-Path DNSSEC Downgrade Attacks. – In: ACM SIGCOMM Conference, 2023, pp. 1120-1122. DOI: 10.1145/3603269.3610840.
14. Harkins, D. Dragonfly Key Exchange. – RFC 7664, Internet Engineering Task Force, 2015. DOI: 10.17487/RFC7664.

15. Barker, E. Recommendation for Pairwise Key-Establishment Schemes Using Discrete Logarithm Cryptography. – NIST Special Publication 800-56A, 2018. DOI: 10.6028/NIST.SP.800-56Ar3.
16. Krawczyk, H., P. Eronen. HMAC-Based Extract-and-Expand Key Derivation Function (HKDF). – RFC 5869, 2010. DOI: 10.17487/RFC5869.
17. Blaise, O. O., A. Wumi, U. Alfred. Enhancing DNS Performance with Efficient Cryptographic Algorithms: A Comparative Study of DoT Frameworks. – Asian Journal of Computer Science and Technology, Vol. **13**, 2024, No 2, pp. 48-55. DOI: 10.70112/ajcst-2024.13.2.4288.
18. Padmavathi, G., A. Sabitha Banu. Hybrid Detection and Mitigation of DNS Protocol MITM Attack Based on Firefly Algorithm with Elliptical Curve Cryptography. – Engineering and Technology for Public Health, Vol. **8**, 2022, No 4. DOI: 10.4108/eetpht.v9i1.3177.
19. Mattsson, J. P. Collision-Based Attacks on Block Cipher Modes: Exploiting Collisions and their Absence. – Cryptology ePrint Archive, 2024.
20. Maksutov, A. A., I. A. Cherepanov, M. S. Alekseev. Detection and Prevention of DNS Spoofing Attacks. – In: Proc. of Siberian Symposium on Data Science and Engineering, 2017, pp. 84-87. DOI: 10.1109/SSDSE.2017.8071970.
21. Iwata, T., K. Ohashi, K. Minematsu. Breaking and Repairing GCM Security Proofs. – In: Proc. of Annual Cryptology Conference, Springer, 2012, pp. 31-49. DOI: 10.1007/978-3-642-32009-5_3.
22. Yu, H., X. Yuchi, X. Yang, H. Li, X. Yang, W. Wang. DNS-Sensor: A Sensor-Driven Architecture for Real-Time DNS Cache Poisoning Detection and Mitigation. – Sensors, Vol. **25**, 2025, 6884. DOI: 10.3390/s25226884.
23. Heftrig, E., H. Shulman, N. Vogel, M. Waidner. The Harder You Try, the Harder You Fail: The KeyTrap Denial-of-Service Algorithmic Complexity Attacks on DNSSEC. – In: Proc. of ACM SIGSAC Conference on Computer and Communications Security (CCS'24), 2024, pp. 497-510. DOI: 10.1145/3658644.3670389.
24. Bellare, M., B. Tackmann. The Multi-User Security of Authenticated Encryption: AES-GCM in TLS 1.3. – In: Advances in Cryptology (CRYPTO 2016), LNCS. Vol. **9814**. Springer, 2016, pp. 247-276. DOI: 10.1007/978-3-662-53018-4_10.
25. Vanhoef, M. Dragonblood: Analyzing the Dragonfly Handshake of WPA3 and EAP-pwd. – In: Proc. of IEEE Symposium on Security and Privacy, 2020, pp. 517-533. DOI: 10.1109/SP40000.2020.00031.
26. Dawood, M., et al. The Impact of Domain Name Server (DNS) over HTTPS on Cybersecurity: Limitations, Challenges, and Detection Techniques. – Computers, Materials and Continua, Vol. **80**, 2024, No 3, pp. 4513-4542. DOI: 10.32604/cmc.2024.050049.
27. Patil, D. A., G. Shyamala. A Comprehensive Survey on Securing the Social Internet of Things: Protocols, Threat Mitigation, Technological Integrations, Tools, and Performance Metrics. – Scientific Reports, Vol. **15**, 2025, 40190. DOI: 10.1038/s41598-025-23865-4.
28. Rotuna, C. I., I. S. Sacala, A. Alexandru. Towards Proactive Domain Name Security: An Adaptive System for .ro Domains Reputation Analysis. – Future Internet, Vol. **17**, 2025, No 10, 478. DOI: 10.3390/fi17100478.
29. Luo, S. Blockchain-Enabled Decentralized End Hopping for Proactive Network Defence. – Telecom, Vol. **7**, 2026, No 2, 28. DOI: 10.3390/telecom7020028.

30. Amirkhanova, G., S. Ismailov, A. Amirkhanov, S. Adilzhanova. A Lightweight End-to-End Encrypted Data Pipeline for IIoT. – Information, Vol. 17, 2026, No 1, 33. DOI: 10.3390/info17010033.
31. Abu Al-Haija, Q., M. Alohalay, A. Odeh. A Lightweight Double-Stage Scheme to Identify Malicious DNS over HTTPS Traffic Using a Hybrid Learning Approach. – Sensors, Vol. 23, 2023, No 7, 3489. DOI: 10.3390/s23073489.
32. Kundu, A., V. Chakraborty, R. Kompella. Post-Quantum Cryptographic Analysis of Message Transformations across the Network Stack. – arXiv Preprint, 2026. DOI: 10.48550/arXiv.2604.08480.
33. Agrawal, N. K., S. Alam, H. Raghav. OSI Model: The Basic Structure of Network Communication. – International Journal of Recent Technology and Engineering, Vol. 9, 2021, No 5, pp. 66-69. DOI: 10.35940/ijrte.D4991.019521.

*Received: 19.03.2026; First Revision 12.05.2026; Second Revision 20.05.2026;
Accepted: 25.05.2026*

© 2026 by the authors. Licensee IICT-BAS, Sofia, Bulgaria. This article is an open access article distributed under the terms and conditions of the **Creative Commons Attribution-NonCommercial-NoDerivatives** license.