TOWARDS SECURING THE DNS RESOLUTION PROCESS

by

Ali Sadeghi Jahromi

A thesis submitted to the Office of Graduate Studies
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

Carleton University
Ottawa, Ontario

May, 2025

Contents

List of Tables

List of Figures

Abstract

Essentially every interaction on the Internet is preceded by at least one DNS resolution; therefore, the security and privacy of the DNS resolution process directly impact all entities that rely on this process. This thesis pursues five objectives regarding the security and privacy of DNS resolution.

First, a comprehensive threat model of the DNS resolution process is developed to systematically identify existing threats. Based on this, a set of security and privacy properties intended to mitigate these threats effectively is defined. Second, these properties are used to construct a comparative evaluation framework, by which 11 previously proposed secure DNS schemes are evaluated to assess their security, privacy, and availability guarantees.

Third, after analyzing the weaknesses of these pre-existing schemes, a novel secure DNS scheme, DNSSEC+, is proposed and designed to enhance the security and privacy of interactions between recursive resolvers and authoritative nameservers. DNSSEC+ aims to provide stronger security and privacy properties than previously proposed schemes in this stage while maintaining comparable performance overhead.

Fourth, to examine whether the performance of DNSSEC+ may impact potential adoption and its practical deployment, we developed a prototype implementation of DNSSEC+ and carried out a comparative performance evaluation versus other secure DNS schemes. Finally, to establish confidence in its security and privacy properties, a symbolic model of DNSSEC+ is developed, and its properties are formally verified.

Acknowledgements

First and foremost, I would like to thank my supervisor Dr. AbdelRahman Abdou, who initially accepted me as a Master's student and, through his exceptional guidance, encouraged and supported my transition to the PhD program after I completed my Master's degree. His mentorship has been instrumental in shaping both my academic and personal growth. Without his unwavering support (both financial and academic) my journey would have taken a different course. I am profoundly grateful to have had the opportunity to learn under his guidance.

I am also sincerely thankful to Dr. Paul C. Van Oorschot, my co-supervisor, whose mentorship throughout my PhD program has been invaluable. His insights, constructive feedback, and attention to details have helped me grow into a more thoughtful and rigorous researcher, resulting in the development of stronger critical thinking skills.

Finally, I extend my appreciation to the members of the Carleton Security Research Labs (CISL and CCSL) labs for their engaging discussions, valuable feedback, and generous support, all of which contributed meaningfully to the development of my work.

I would also like to thank the members of my PhD committee: Professor Aanjhan Ranganathan (Northeastern University), Professor Guy-Vincent Jourdan (University of Ottawa), Professor Ashraf Matrawy (Carleton University), and Professor David Barrera (Carleton University) for their careful reading of my thesis and for their insightful comments and valuable feedback.

In addition, I would like to thank my parents for their unwavering encouragement and lifelong support. Their belief in me made the achievement of this milestone possible. I am also grateful to my friends, who became my family in Ottawa. Their companionship during both challenging and joyful times provided emotional strength and balance throughout this journey.

Ali Sadeghi Jahromi
May, 2025.

List of Acronyms

AD Authenticated Data

ANS Authoritative NameServer

ARP Address Resolution Protocol

AS Autonomous System

BGP Border Gateway Protocol

CA Certificate Authority

CDN Content Delivery Network

CRL Certificate Revocation List

DDoS Distributed Denial-of-Service

DGA Domain Generation Algorithm

DH Diffie-Hellman

DHT Distributed Hash Table

DNS Domain Name System

DNSSEC DNS Security Extensions

DoDTLS DNS-over-DTLS

DoH DNS-over-HTTPS

DoQ DNS-over-QUIC

DoS Denial-of-Service

DoT DNS-over-TLS

DoTor DNS-over-Tor

DS Delegation Signer

ECC Elliptic-Curve Cryptography

ECDH Elliptic-Curve Diffie-Hellman

ECH Encrypted Client Hello

ECS EDNS Client Subnet

EDNS Extension Mechanisms for DNS

ENS Ethereum Name Service

ESNI Encrypted Server Name Indication

FTP File Transfer Protocol

GNS GNU Name System

IETF Internet Engineering Task Force

IoT Internet of Things

IP Internet Protocol

ISP Internet Service Provider

KDF Key Derivation Function

KSK Key Signing Key

MAC Message Authentication Code

MTU Maximum Transmission Unit

NSEC Next Secure

NTP Network Time Protocol

NXDOMAIN Non-Existent Domain

ODNS Oblivious DNS

OOB Out-of-Band

PII Personally Identifiable Information

PKI Public-Key Infrastructure

RD Recursion Desired

RFC Request For Comments

RRSIG Resource Record Signature

SEND Secure Neighbor Discovery

SLD Second-Level Domain

SNI Server Name Indication

TTL Time to Live

TLD Top-Level Domain

TLS Transport Layer Security

TVP Time-Variant Parameters

URI Uniform Resource Identifier

VRF Verifiable Random Function

WG Working Group

WSEC DNS Wildcard SECure DNS

ZSK Zone Signing Key

Chapter 1

Introduction

The Domain Name System (DNS) [119, 120] was introduced in 1987 with the primary goal of translating human-readable domain names into numerical Internet Protocol (IP) addresses. Although DNS was originally proposed as a decentralized and scalable solution intended to be able to accommodate the growth of the Internet, security and privacy were not among the goals of *Vanilla DNS* (the original design) [119, 120]. As a result, attackers have exploited the lack of security and privacy in Vanilla DNS, mounting attacks such as surveillance and information gathering [63, 70], web censorship [16], and injecting false DNS responses that lead to DNS cache poisoning [74, 98]. These attacks can compromise the security and privacy of the hundreds of billions of DNS queries resolved over the Internet each day.[1] Additionally, the absence of adequate security measures in DNS has jeopardized the security and privacy of other protocols and ecosystems built upon it [43], including the Internet of Things (IoT) ecosystem [10], the domain validation process [30], Network Time Protocol (NTP) [95], and applications such as email [43].

For more than three decades, researchers have focused on addressing the security and privacy weaknesses of Vanilla DNS in an effort to mitigate both active and passive DNS-resolution-related attacks. The majority of the proposed solutions aim to augment Vanilla DNS by adding security or privacy properties [11, 46, 80, 86]. These improvements generally provide security and privacy properties in specific sections of the DNS resolution process (*e.g.,* DNS-over-TLS [86] between stub and recursive resolvers), while leaving other parts (*e.g.,* recursive resolver to nameserver interactions) vulnerable. Moreover, several of these (*e.g.,* DNSCurve [46] and DNS Security Extensions (DNSSEC) [34]) encounter deployment challenges that hinder their widespread adoption. Some other solutions aim to provide alternative name systems that require fundamental modifications to the currently established name system (*e.g.,* GNU Name System (GNS) [167] and NameCoin [97]). The limited adoption of schemes requiring fundamental changes to the existing DNS infrastructure can be attributed to several factors. These include the impracticality of implementing radical modifications to the deployed DNS infrastructure, the lack of

[1] `https://vercara.com/resources/2023-dns-traffic-and-trends-analysis`

incremental deployability and backward compatibility of such schemes, and the reluctance of current Internet infrastructure providers to adopt untested approaches and abandon the well-established and operational Vanilla DNS [97, 167]. Therefore, rather than fundamentally modifying the DNS resolution process, augmenting its security and privacy properties presents a more feasible approach to improving the overall security of DNS resolution.

Most of the proposed security enhancements to DNS focus on the last mile, *i.e.,* the interactions between clients and recursive resolvers [48, 80, 86, 89]. Some of these schemes, such as DNS-over-TLS (DoT) [86] and DNS-over-HTTPS (DoH) [80] have been adopted and endorsed by industry stakeholders, such as those who control major web browsers [32] and operating systems.[2] Only a few schemes, for example DNSSEC [11] and DNSCurve [25, 46], have been proposed to secure the communication between recursive resolvers and Authoritative nameservers (ANSes). None of these has been widely adopted [34, 168]; thus, the majority of the interactions between recursive resolvers and ANSes remain vulnerable to active (*e.g.,* cache poisoning) and passive (*e.g.,* surveillance) attacks.

In order to investigate the security and privacy of secure DNS schemes that have been proposed to augment DNS resolution process, one approach is to analyze existing DNS security surveys and related literature. However, these tend to either focus on specific threats, such as DNS-based botnet evasion [113] and DNS-based data exfiltration [125], or provide a broad overview of the entire DNS infrastructure, as in the surveys by Khormali *et al.* [99] and Zou *et al.* [177]. As a result, while offering valuable insights into specific attacks and broad DNS-related issues, their primary objective is not to provide a comprehensive evaluation of the threats to the DNS resolution process or the advantages and limitations of the schemes proposed to augment its security and privacy.

In this thesis, we conduct a comprehensive survey of schemes designed to enhance the security and privacy of the DNS resolution process. Through a systematic analysis, we identify and categorize the security, privacy, and availability threats associated with this process and develop a detailed threat model to elucidate the threats of the DNS resolution process.

Based on the identified threats, we then define a set of properties that we argue

[2] `https://techcommunity.microsoft.com/t5/networking-blog/windows-will-improve-user-privacy-with-dns-over-https/ba-p/1014229`

are essential to the DNS resolution process to effectively mitigate the security and privacy threats of this process. These properties serve as the foundation for developing a framework to analyze the properties of proposed schemes aimed at enhancing the security and privacy of the DNS resolution process and are intended to help the advancement of future designs. We then use this framework to comparatively evaluate the properties offered by 11 previously proposed DNS schemes designed to enhance the security and privacy of the DNS resolution process.

After identifying the threats and required properties of the DNS resolution process and conducting this comparative evaluation, we propose a novel DNS scheme, DNSSEC+, designed to augment the real-time security and privacy of interactions between recursive resolvers and ANSes within the DNS resolution process. Next, we implement a prototype of DNSSEC+, and evaluate its performance in comparison to existing schemes, including DNSSEC [11, 12, 13] and TLS-based [86] schemes. Our evaluation indicates that DNSSEC+ has performance comparable to most previously proposed schemes in terms of total resolution latency and CPU utilization, and significantly outperforms DoT in these metrics.

Finally, we developed a symbolic model of DNSSEC+ in the Tamarin protocol verifier syntax. Under standard assumptions, the model is used to formally verify the security and privacy properties of the DNS resolution process with the root zone when using DNSSEC+, which is then extended to the subordinate zones. Additionally, the impact of key and entity compromises on the security and privacy properties of DNSSEC+ is formally analyzed, providing insights on the robustness of DNSSEC+ under various threat scenarios.

1.1 Thesis Scope

The operation of the DNS infrastructure on the Internet depends on multiple entities and processes, including name resolution, domain registration, and transfer of zone files. Our primary focus in this thesis is the name resolution process, which involves a set of interactions between DNS resolvers (stub and recursive) and nameservers through which a client resolves a query to its corresponding DNS record. In this section, we define the scope of the research areas explored in this thesis by explicitly defining the subjects and aspects included within its scope and the ones that fall outside scope.

1.1.1 A Systematic Evaluation of DNS Resolution Process

Since the primary focus of Chapter 3 is the name resolution process, this chapter aims to identify the security and privacy threats associated with this process while excluding domain registration procedures, zone transfers, and host-based attacks. *Domain registration* refers to the process by which a domain registrant acquires ownership of a specific domain by licensing an available domain from a registrar. *Zone transfer* is the mechanism through which a zone file (data) is copied from one name server to another. Host-based attacks range from a host compromised by malware (*e.g.,* a recursive resolver) to manipulations carried out by a malicious insider. For example, in host-based attacks, an adversary can change OS configurations such as the `Hosts.txt` file, primary DNS server (*e.g.,* modify network configuration files), or IP/domain-name associations on the DNS caches or databases to falsify DNS resolution. Another example of a host-based attack that targets the authenticity of network configuration on a client involves using a malicious Dynamic Host Configuration Protocol (DHCP) server to configure bogus DNS servers on a client's machine or orchestrate false responses to the client queries. All such host-based attacks are excluded from the scope of our systematic analysis in Chapter 3.

Following the identification and extraction of threats within the DNS resolution process, a comprehensive threat model is developed in Section 3.1. This model can be used as an objective tool by industry stakeholders and researchers for evaluating the security strengths and weaknesses of their deployed DNS resolution process and proposed secure DNS schemes, respectively. After developing a threat model, the identified threats are employed to define security, privacy, and availability properties that a DNS scheme can offer to mitigate them. The focus of this part is explicitly limited to security, privacy, and availability properties, while usability and deployability properties fall outside the scope of Section 3.2.

Next, we develop a comprehensive and objective evaluation framework using the properties defined in Section 3.2. The scope of the properties evaluated by this framework is limited to the threats and properties from Sections 3.1 and 3.2. This framework is then applied to assess the security and privacy properties provided by a number of previously proposed secure DNS schemes. Since the identified threats and properties are derived from the lack of security and privacy in the name resolution process of Vanilla DNS, the evaluation framework focuses on schemes designed to augment security or privacy of this process. Schemes that introduce radical changes

to the DNS resolution process (*e.g.,* Namecoin [97]) are not suitable for evaluation using this framework and have therefore been excluded from this assessment.

1.1.2 Proposing and Evaluating an Enhanced DNS Scheme

After identifying threats to the DNS resolution process and defining required properties to mitigate the threats in Chapter 3, in Chapter 4, DNSSEC+ is proposed as an enhanced DNS scheme that aims to augment the security and privacy between recursive resolvers and nameservers. The initial scope of DNSSEC+ is delimited to the communication between recursive resolvers and nameservers, and DNSSEC+ alone does not secure the entire DNS resolution process. In Chapter 7, we briefly discuss how to use DNSSEC+ with different Public Key Infrastructures (PKIs) to secure communication between clients and recursive resolvers separately to secure the remaining part of the DNS resolution process. The underlying rationale is that DNS resolution is a two-stage process, with each stage executed separately and requiring distinct security and privacy properties. Furthermore, comprehensive solutions that require fundamental changes to the original two-stage DNS resolution process or significant modifications to the existing name system architecture often encounter substantial barriers to widespread adoption [38, 97, 167].

Throughout the design of DNSSEC+, while the primary objectives were to enhance security and privacy, secondary considerations, such as avoiding barriers to deployment and minimizing performance degradation that could affect end-user experience, which were not the focus of the systematic analysis in Chapter 3, were also taken into account. In Chapter 5, a prototype of DNSSEC+ was developed as a proof of concept. This also enabled performance evaluation and comparative assessment with the previously proposed secure DNS schemes. While the prototype accurately demonstrates the functionality of DNSSEC+ and provides valuable insights into its performance, it is intended mainly for proof-of-concept and for evaluation purposes and would require further development for deployment in live DNS environments.

1.1.3 Formal Analysis of DNSSEC+

In Chapter 6, we proceed with the formal verification of the security and privacy properties defined for DNSSEC+. The primary objective of formal verification tools

is to generate proofs that verify these properties for all possible protocol behaviors of the defined symbolic model. For this purpose, we use the Tamarin prover [117], a symbolic formal verification tool, with the results thereby contingent on the tool's capabilities, inherent limitations, and the specified adversary model. In symbolic verification, cryptographic primitives are assumed to be idealized, error-free constructs that cannot be compromised, making the soundness of the generated proofs contingent on the security of the cryptographic primitives that instantiate the design. Furthermore, as DNSSEC+ is designed to secure interaction between recursive resolvers and ANSes, the formal verification confirms the specified properties strictly within this scope of the communication. As a further limitation, formal verification tools may have inherent or undiscovered flaws, which can also affect the reliability of the generated proofs, and discovering them is beyond the scope of this thesis.

1.2 Motivation

Essentially all Internet users, mobile phones [5], IoT devices [10], and embedded systems connected to the Internet rely on DNS for name resolution before initiating communication with their intended services. Therefore, security or privacy weaknesses of the DNS resolution process can compromise the security and privacy of all these Internet-connected entities. The long-standing absence of adequate security and privacy properties in the DNS resolution process and the lack of a systematic threat model motivate our development of a comprehensive threat model for the DNS resolution process. Our aim is to elucidate security and privacy vulnerabilities that can be exploited within the DNS resolution process by passive and active adversaries.

Since the introduction of Vanilla DNS [119, 120], various security extensions [11, 12, 13] and enhancements [25, 48, 80, 86] have been proposed to augment the security and privacy of the DNS resolution process. Each of these proposed schemes provides a set of security and privacy properties in a part of DNS resolution process. However, to our knowledge, there has previously been no objective method for evaluating the provided properties by these schemes. This suggests a need for a systematization of existing knowledge on DNS security to elucidate the security strengths and weaknesses of the proposed schemes and improve future secure DNS scheme designs and motivates the development of a framework for systematically

evaluating the security of previously proposed DNS schemes. We hope that it is also of use in evaluating future DNS designs.

Upon investigating existing secure DNS schemes proposed to enhance the security of interactions between recursive resolvers and ANSes, we observe that none of these has achieved widespread adoption over the Internet. For example, while DNSSEC, introduced nearly twenty years ago, has been predominantly accepted by the root and Top-level Domains (TLDs), Chung *et al.* [34] reported that, as of 2017, only 1% of domains had implemented DNSSEC, with significant mismanagement observed in their deployments. Additionally, recent deployment statistics for DNSSEC indicate that less than 4% of domains within the .com TLD have implemented DNSSEC signatures in 2025.[3] Moreover, DNSSEC only provides message authentication in this stage of DNS resolution process, without providing additional security or privacy properties. DNSCurve [25, 46], a promising alternative to secure the same stage of the DNS resolution process as DNSSEC, has to date failed to achieve adoption at the root level due to security concerns related to its key management method and PKI [145, 168].

Thus, the lack of a widely adopted secure DNS scheme in this stage of DNS resolution process indicates the need for a secure DNS scheme for secure DNS resolution. This motivates our design and proposal of DNSSEC+, a novel secure DNS scheme between recursive resolvers and ANSes, aiming to augment the security and privacy of DNS resolution while maintaining a comparable performance to that of less secure DNS schemes. By introducing a secure DNS scheme in Stage 2, we aim to protect the DNS resolution path, thereby enabling clients to securely obtain the correct IP address of their intended service points. While DNS is necessary for correct address resolution, it is not sufficient on its own for end-to-end secure communication of clients with their intended service points. Additional mechanisms such as secure routing protocols and infrastructures like the web PKI and TLS are required to ensure both the integrity of packet delivery and the confidentiality of subsequent client-service interactions. Therefore, secure DNS resolution is an essential component in achieving the end goal of enabling a client to interact with its intended service endpoint.

In the design of complex security protocols, subtle flaws may exist that compromise the security and privacy properties intended by those protocols. For example,

[3] `https://www.statdns.com/` (January 2025)

17 years after the introduction of the Needham-Schroeder protocol [128], Lowe [108] discovered a previously undetected attack that had been inherent in the protocol since its initial proposal. Thus, a formal analysis of security protocols helps us build confidence that they meet the properties asserted in protocol specifications for given assumptions regarding the adversary. In this thesis, following the design and prototype implementation of DNSSEC+, we conduct a formal verification of the security and privacy properties in Chapter 6. Using the Tamarin Prover, we establish these properties over a symbolic model of DNSSEC+.

1.3 Research Questions

The primary research questions addressed in this thesis are as follows.

RQ1. What are the primary security and privacy threats facing the DNS resolution process; and, what security, privacy, and availability properties are sufficient to effectively mitigate these threats?

RQ2. What type of evaluation framework could be developed to objectively evaluate the security and privacy of both existing and future DNS schemes designed to enhance the security or privacy of the DNS resolution process?

RQ3. How can an enhanced Stage 2 DNS resolution scheme be designed and developed to augment the security and privacy in Stage 2 of the DNS resolution process?

1.4 Contributions

The main contributions of this thesis address the aforementioned research questions and can be categorized into four main areas, as outlined below.

Threat Modeling (Sections 3.1 and 3.2): Through a systematic informal analysis of the DNS resolution process, we develop a comprehensive threat model and attack taxonomy of existing network-based threats within this process. In this model, we classify the threats into three primary categories: security, privacy, and availability. We then define 14 security and privacy properties that we assert as requirements to mitigate the identified threats.

Evaluation Framework (Section 3.3): The properties that we assert as requirements serve as our foundation for the development of a framework for systematic evaluation of secure DNS schemes. The result is an objective framework for evaluating the security, privacy, and availability properties of schemes proposed to enhance the DNS resolution process. This framework is then used to evaluate 11 schemes surveyed in our literature review.

Proposal and Design of DNSSEC+ (Chapter 4): Following the evaluation of previously proposed DNS schemes and learning from their benefits and shortcomings, we design and propose DNSSEC+, an enhanced secure DNS scheme to augment the interactions between recursive resolvers and ANSes. This proposes a novel solution for the issue of long-term private key replication on nameserver instances in the DNS context, which is a similar problem to the delegation problem in TLS. Additionally, we specify key distribution methods and a trust model suitable for the specific requirements of the zones within the DNS hierarchy. Our scheme aims to augment real-time security and privacy properties that mitigate the identified threats for this stage of DNS resolution.

Evaluation and Formal Verification (Chapters 5 and 6): We implement a prototype of DNSSEC+ and evaluate its performance metrics in comparison with those of the other secure DNS schemes. Subsequently, we develop a symbolic model of DNSSEC+, formally define its security and privacy properties, and use the Tamarin Prover [117], to formally verify these properties. We then incorporate the compromise of cryptographic keys and involved entities in DNSSEC+ into the symbolic model to analyze and demonstrate the impact of such compromises on the modeled security and privacy properties.

1.5 Organization

The remainder of this thesis is organized as follows. Chapter 2 presents background on the DNS resolution process and secure delegation of authorization; a comprehensive literature review of the techniques and schemes proposed since the introduction of Vanilla DNS to enhance the security and privacy of the DNS resolution process; and an overview of the symbolic modeling of secure protocols and the formal definition of their security and privacy properties in the Tamarin protocol verifier.

Chapter 3 provides a comprehensive threat model and attack taxonomy of the DNS resolution process. Based on the identified threats, we propose security and privacy properties aimed at mitigating these threats, and using the defined properties, develop an objective evaluation framework for comparative evaluation of secure DNS schemes. This framework is then employed to comparatively evaluate 11 DNS schemes that have been proposed to enhance the DNS resolution process by augmenting its security or privacy properties.

In Chapter 4, we propose and explain the design goals of DNSSEC+, our new DNS scheme that augments real-time security and privacy in the interaction between resolvers and ANSes. Chapter 5 discusses a prototype implementation of this scheme and evaluates its performance in terms of time efficiency and CPU utilization, comparing it against other existing secure DNS schemes.

Chapter 6 develops a symbolic model of DNSSEC+, along with its defined security and privacy properties, using the Tamarin prover's syntax [117]. We then formally prove that these properties hold for all the possible behaviors of the protocol in the defined model. Additionally, we model the compromise of involved entities and cryptographic keys within DNSSEC+ to analyze their impact on the security and privacy properties of the protocol.

In Chapter 7, we review the research questions addressed within this thesis, discuss the open research challenges, propose potential future research directions, and present concluding remarks.

1.6 Related Publications

Chapters 2 and 3 contain material that is currently being prepared for submission as a Systematization of Knowledge (SoK) paper [94].

- **Ali Sadeghi Jahromi**, AbdelRahman Abdou, Paul C. van Oorschot. "SoK: An evaluation framework for secure DNS schemes"

The content of Chapters 4 and 5 has been submitted for publication in a conference and is currently undergoing peer review. A preprint of the work is available on arXiv [92].

- **Ali Sadeghi Jahromi**, AbdelRahman Abdou, Paul C. van Oorschot.

"DNSSEC+: An enhanced DNS scheme motivated by benefits and pitfalls of DNSSEC", Aug 2024. Technical report available at: `https://arxiv.org/abs/2408.00968`

The work presented in Chapter 6 has been submitted to a conference for publication and is currently under peer review [93].

- **Ali Sadeghi Jahromi**, AbdelRahman Abdou, Paul C. van Oorschot. "Formal security analysis of DNSSEC+"

Chapter 2

Literature Review and Context

This chapter begins by presenting the required background on DNS infrastructure and the DNS resolution process. Subsequently, we explain the PKI and trust model concepts, which are used as foundations for augmenting security and privacy of the DNS resolution process. Following this, the chapter provides a summary and a review of the provided properties by secure DNS schemes that have been proposed to enhance the security of DNS resolution in different stages, along with simpler techniques aimed at improving the security and privacy of DNS resolution. We then provide a brief overview of a few related schemes that require fundamental changes to the existing DNS infrastructure. Next, the chapter reviews the delegation of authorization and various delegation methods employed in different Internet protocols. Finally, we provide an overview of symbolic analysis and the process by which formal verification of security protocols can be carried out using the Tamarin prover.

2.1 Domain Name System

The DNS was designed as a distributed naming system to replace the static `HOSTS.txt` file, with the objective of ensuring consistency and scalability [119, 120]. To resolve a DNS record, a DNS resolver typically engages in a sequential interaction with the nameservers within the DNS hierarchy in order to obtain the final response. As illustrated in the right-hand part of Figure 2.1, the naming system is organized in a reverse-tree structure, consisting of hierarchical zones beginning from the root. Each zone typically has two (primary and secondary) or more ANSes responsible for storing the DNS records within the authoritative scope of that zone and serving the incoming queries. The hierarchy of DNS starts from the root zone at the apex through the Top Level Domains (TLDs) (*e.g.,* .com), Second Level Domains (SLDs), and other subordinate zones. The DNS resolution starts when an application on a client (*e.g.,* web browser) issues a DNS query, and the query is passed on to the stub resolver of the client. The stub resolver then forwards the query to a recursive resolver (Step *1*). Then, in Steps *2* through *n-1*, the recursive resolver traverses the DNS hierarchy and queries ANSes of the zones until it obtains the IPv4 address (if

an A record was requested) associated with the queried domain name or receives an error. Eventually, the recursive resolver sends back the final response to the client (Step n).

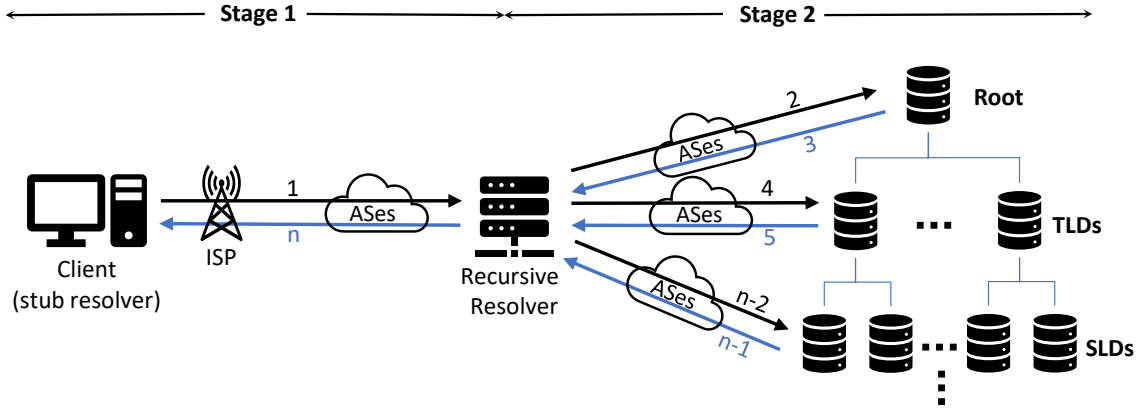


Figure 2.1: Complete DNS resolution process (divided into two stages): Pre-recursive-resolver (Stage-1) and post-recursive-resolver (Stage-2).

Herein, the communication between the client and recursive resolver is defined as Stage 1, and the communication between the recursive resolver and ANSes as Stage 2. The recursive resolver in a name resolution process can reside in the client's local network, be part of the Internet Service Provider (ISP), or be a remote public recursive resolver (*e.g.,* Cloudflare's recursive resolver at 1.1.1.1). As illustrated in Figure 2.1, in this thesis we assume that the client is configured to use a remote (public) recursive resolver. Thus, the client's local network, ISP, and other Autonomous Systems (ASes) reside as intermediate entities between the client and the recursive resolver. The paths to the remote resolvers contain the local network and ISP as the intermediate entities. Additionally, the path to these remote resolvers passes through ASes other than the ISP between the stub- and recursive-resolver.[1] Moreover, due to the reliability and lower costs of public resolvers, their use by users and applications has increased over time [122, 139].

2.1.1 Extension Mechanisms for DNS

Extension Mechanisms for DNS (EDNS) [41] was introduced as a standard to expand the functionality of the DNS protocol while ensuring backward compatibility.

[1]The case of a remote resolver is considered the most difficult case from a defensive perspective, as it has a larger attack surface compared to the two other scenarios (*i.e.,* local or ISP resolvers).

Initially proposed in RFC 2671 [165] and later updated to RFC 6891 [41], EDNS addresses the limitations of Vanilla DNS [119, 120], specifically the fixed maximum response size of 512 bytes for UDP responses. In Vanilla DNS [119], messages exceeding this limit were truncated and needed to be re-transmitted over TCP, which posed challenges for use cases requiring larger response sizes, such as DNSSEC [11, 12, 13] and other protocols that require large DNS records.

EDNS enables the transmission of larger responses over UDP, with support for sizes up to 4096 bytes, reducing the need for TCP fallback and improving performance. Additionally, EDNS introduced extended message header fields and label types, enabling the inclusion of new options and flags. These extensions enabled adding advanced features in DNS, including DNSSEC for response authentication, the Client Subnet extension for network-proximity-based query resolution [36] (see Section 2.1.2), and DNS Cookies for mitigating off-path spoofed responses [57].

2.1.2 EDNS Client Subnet

Typically, ANSes only see the recursive resolvers' IP addresses, instead of clients'. Content Delivery Networks (CDNs) aim to minimize latency by connecting clients to the closest CDN edge servers. The functionality of CDNs requires access to a client's IP address in order to provide the IP address of the closest CDN edge server to the respective client. EDNS Client Subnet (ECS) is a DNS extension that facilitates including a client's IP address or subnet inside the DNS queries [36]. Although the specification of ECS suggests truncating IPv4 and IPv6 to 24 and 56 bits, respectively, to protect clients' privacy, this is not specified as a requirement of the protocol, and clients can send their 32/128-bit IPv4/v6 addresses alongside their queries [36]. Additionally, even if truncation of a client's IP address is applied, /24 IPv4 or /56 IPv6 addresses still often reveal client-related information (*e.g.,* country, city, or organization). Thus, a scheme that truncates ECS may still disclose client-related information that can be used by attackers (*e.g.,* for selective cache poisoning [101] or censorship at country- or organization-level).

If a client includes ECS in queries, when an ECS-supporting intermediate recursive resolver receives a DNS query, it will forward the ECS as part of its queries to all the traversed ANSes. Therefore, all ANSes from the root and TLDs, down to the name server that is authoritative for the domain name, can view the ECS in the resolver's queries. As a result, the client's IP address, and consequently its

geographic location, will be disclosed to the queried ANSes.

2.1.3 Surveys on DNS Security

An early security survey of DNS was done in 2005 by Ollmann [130], in which he studied six attack vector categories that can target five entity groups in the DNS infrastructure to carry out pharming. *pharming* is an attack that involves injecting false responses into DNS queries (we provide a detailed discussion and categorization of pharming techniques in Section 3.1). In contrast to our systematic evaluation (Section 3.1), which focuses on an extensive threat model of DNS resolution process, Ollmann's work focuses on the attack vectors that specifically enable pharming. Zou *et al.* [177] conducted a relatively short survey with DNS threats categorized into five groups, and then compared eight alternative DNS schemes and DNS enhancement proposals that can address the threats. In contrast, our systematic evaluation in Chapter 3 develops a more comprehensive threat model that specifically focuses on network-related threats of DNS resolution, which includes replay attacks, DNS-based censorship, and other attack categories and instances. Additionally, their evaluation of surveyed schemes had only three parameters related to security and privacy aspects, while our evaluation focuses on security, privacy, and availability properties of DNS schemes with 14 parameters.

The security vulnerabilities and deployability obstacles of DNSSEC have been thoroughly analyzed and demonstrated in several research contributions [25, 34, 37]. Anagnostopoulos *et al.* [6] conducted a detailed investigation and comparison of security, deployability, and performance of two Stage 2 DNS schemes, namely DNSSEC and DNSCurve. Compared to our work in Chapter 3, which focuses on the weaknesses and required properties in the entire name resolution path, these studies focus on a limited number of schemes with scheme-specific threat models. For example, as they focus on Stage 2 schemes, they exclude Stage 1 threats and the DoS attacks other than reflection amplification.

In another comprehensive survey, Khormali *et al.* [99] studied DNS threats and vulnerabilities, DNS research methods, and the impact of DNS entities' (*i.e.,* clients, resolvers, name servers, and hosting providers) security on the DNS ecosystem. Their approach is identifying broad threats and vulnerabilities by surveying previous literature. In contrast, in our work, we define our threat model with a specific focus on network-based threats, attack instances, and techniques, and then use the

threat model as a foundation to evaluate threats and vulnerabilities of previously proposed schemes that augment security or privacy of the DNS resolution process. For example, we cover replay attacks and traffic correlation attacks that can be used to infer queried domain names in our threat model, while these attacks are not among their investigated threats.

Among other DNS-related surveys that focus on specific threats and defense approaches. Torabi *et al.* [160] investigated systems designed to passively monitor and analyze (*e.g.,* using ML models) DNS traffic to detect DNS-related threats (*e.g.,* fast-flux domains, malicious domains). They also proposed their own DNS monitoring system that detects malicious DNS-related behaviors at a near real-time speed. Zhauniarovich *et al.* [175] conducted a comprehensive survey analyzing malicious domain detection methods, highlighting their differences and similarities. Lu *et al.* [110] surveyed a number of secure Stage 1 schemes and investigated the adoption rates and performance overhead of DoT [86] and DoH [80] over the Internet.

Several studies explore DNS abuse by botnets for intra-botnet communication, evading detection, or data exfiltration. Lyu *et al.* [112] surveyed encryption-based DNS schemes, as well as the misuse of encrypted DNS schemes by botnets for command and control (C&C) communication and data exfiltration. Additionally, they surveyed encrypted DNS traffic classification techniques for DNS traffic isolation and classification of malicious DNS traffic. Using DNS for evading detection through DNS-based fast-flux was studied by Mahmoud *et al.* [113]. In order to evade the detection of botnet C&C domains, botnets often employ Domain Generation Algorithms (DGAs) to generate a large number of random domain names, subsequently employing a part of these domains for their C&C operations. Researchers have analyzed the use of DGAs by botnets [134] and developed systems (*e.g.,* Notos [8] and Pleiades [9]) for detecting such maliciously generated domains. The abuse of DNS for evading detection and facilitating intra-botnet communications falls out of the scope of our systematic evaluation in Chapter 3.

2.1.4 Public-Key Infrastructure and Trust Model

Similar to other secure Internet protocols, such as HTTPS (web) or S/MIME (email), there is a need for a PKI in a secure DNS scheme to establish a structured system for ensuring security or privacy of DNS queries and responses. In the secure DNS context, a PKI can be defined as a set of technologies, entities, policies, and proce-

dures according to which the public keys are managed and used by DNS software. Managing keys can include creation, distribution, usage, and revocation of public keys that are used within a DNS scheme. As part of PKI for a DNS scheme in Stage 2, we define a trust model by which DNS resolver applications recognize public keys of ANSes as valid (trusted; valid trust anchors) and then employ those keys as defined by the DNS scheme specification to cryptographically protect transmitted DNS messages. For example, the web trust model can be represented as a "forest of hierarchical trees," wherein each tree depicts a CA distinct from the others [161]. Similarly, in Stage 1, the stub resolvers of the secure DNS schemes must establish trust in the public keys of the recursive resolvers they use by relying on a trust model. Additionally, secure DNS schemes in Stage 1 must rely on a PKI to cryptographically protect or verify transmitted DNS messages. Some secure DNS schemes, including DoT [86] and DoH [80], rely on pre-existing PKIs [91], such as the web PKI, whereas others, such as DNSSEC, introduce a dedicated PKI specifically designed for the DNS context.

2.2 Initial Review and Analysis of Secure DNS Schemes

Various DNS schemes have been proposed to enhance the security or privacy of the DNS resolution process. Some are designed to enhance DNS security in Stage 1, while others are intended for Stage 2, and a subset provide security in both stages. In this section, we provide a comprehensive survey of the proposed DNS schemes aimed at providing security or privacy in Stage 1 followed by the schemes that enhance Stage 2. Additionally, we review partial techniques that are not comprehensive or fully-developed schemes but are enhancements to address specific security or privacy weaknesses of Vanilla DNS.

Secure DNS Schemes in Stage 1: The following DNS schemes are proposed to augment security or privacy of the communication between stub resolvers and recursive resolvers (*i.e.,* Stage 1).

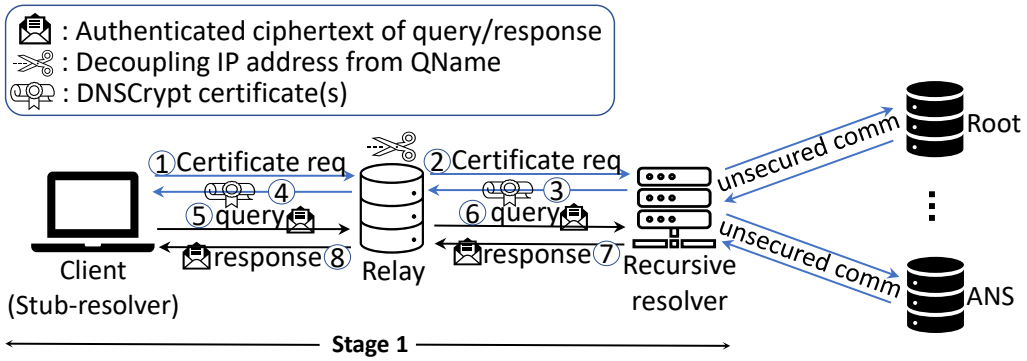


Figure 2.2: **DNSCrypt*V3**: Using authenticated encryption in Stage 1, alongside the use of relay servers to enhance client privacy against resolvers.

2.2.1 (Stage 1) DNSCrypt*V3:

DNSCrypt was introduced as an open-source solution aimed at enhancing the security and privacy of DNS messages in Stage 1 [47, 48]. However, the lack of standardization has impeded the adoption of DNSCrypt. Regular DNS queries and responses in Stage 1 are transmitted as authenticated and encrypted messages through implemented software on clients (*e.g., dnscrypt-proxy*)[2] and recursive resolvers (*e.g., dnscrypt-wrapper*).[3] DNSCrypt uses Elliptic Curve Cryptography (ECC) for establishing symmetric keys and transferring the authenticated ciphertext of queries and responses at the application layer.

As Figure 2.2 shows, in Steps 1 and 2, the client queries a DNSCrypt resolver for its certificates. DNSCrypt does not rely on the Internet's PKI, and the client must know the resolver's name and signing public key to verify the certificates and authenticate the resolver [47]. A recursive resolver's public key and its other properties, such as IP address, port number, and provider name, are encoded in the form of *DNS stamps* and embedded in the client-side software. DNSCrypt certificates are encoded as TXT DNS records, and have a specific structure as described in its specification [47]. The resolver returns one or more signed certificates as response (Steps 3 and 4), and the client verifies the certificates using a previously distributed public key of the resolver.

The client picks the valid certificate with the highest serial number among the supported ones. Then, the client uses the public key and other cryptographic param-

[2] `https://github.com/DNSCrypt/dnscrypt-proxy`
[3] `https://github.com/cofyc/dnscrypt-wrapper`

eters from the selected certificate to generate an authenticated and encrypted query based on NaCl cryptographic library [24], then sends the query alongside its own public key to the resolver (Steps 5 and 6). The resolver decrypts the received query, traverses the DNS hierarchy to obtain the response, and then sends an encrypted and authenticated response back to the client (Steps 7 and 8).

In DNSCrypt version 3, sending anonymized queries is added to the protocol. As Figure 2.2 shows, DNSCrypt uses an intermediate relay server between the client and recursive resolver to hide the IP address of the client from the resolver [48]. As a result, the relay server does not have access to the queried domain name, and the recursive resolver does not have access to the client's IP address, ensuring the client's privacy and anonymity with a low overhead [48].

In practice, a list of public relay servers is maintained by community volunteers.[4] If the owner of the relay is the same as the owner of the resolver (or if they collude), this one-layer anonymization technique is compromised [147]. Also, if ECS is included in the client's DNS queries, the anonymization provided by this method is rendered ineffective.

The key exchange in DNSCrypt is performed using Curve25519 ECC and hSalsa20 hash function [47]. The Salsa20 or ChaCha20 stream cipher is used with Poly1305 for authenticated encryption and transferring messages with confidentiality, integrity, and authenticity [47]. Additionally, DNSCrypt requires a 24-byte nonce as part of the authenticated ciphertexts, which can provide anti-replay means by adding uniqueness to DNS messages. To increase privacy by obfuscating packet sizes, DNSCrypt uses ISO/IEC 7816-4 format for padding DNS messages to a multiple of 64 bytes prior to encryption [47]. DNSCrypt runs over port 443, both TCP and UDP. However, due to its distinguishable characteristics, DNSCrypt's traffic is not completely merged with the web [131]. Therefore, the traffic of DNSCrypt can simply get isolated by analysis of its distinguishable characteristics [131].

By employing Poly1305 message authentication code, DNSCrypt mitigates false response injections in Stage 1. Using different keys per client-resolver communication prevents replay of messages among clients, and the nonces included per message mitigate replay of an old message in the same session to the same client. Analogous to other Stage 1 schemes, for DNSCrypt, we assume that a simple padding scheme with encryption cannot resist traffic analysis attacks and an adversary is able to

[4] `https://github.com/DNSCrypt/dnscrypt-resolvers/blob/master/v3/relays.md`

obtain the domain names of the web pages a user visits (*e.g.,* using ML-based techniques [150]). Hence, DNSCrypt only partially mitigates eavesdropping attacks. As a result, with an additional effort, censoring agents in Stage 1 can obtain the queried domain name by traffic analysis and censor based on the payload content. Thus, DNSCrypt is only partially resilient against payload-based censorship techniques.

The default port number for DNSCrypt is 443. Blocking access to only the DNS servers via port 443 blocking is not possible (would disrupt all HTTPS connections). However, isolating [131] and blocking access to DNSCrypt resolvers is feasible by its traffic characteristics; thus, DNSCrypt traffic is not concealed, and its traffic and servers can be easily identified. DNSCrypt is not resilient against DoS attacks targeting resolvers, as if a DNSCrypt resolver supports the UDP transport protocol (recommended in the specification), an adversary can send spoofed requests and force the resolver to expend computing resources to resolve arbitrary DNS requests. Intermediate relay servers, positioned between clients and recursive resolvers, serve to hide client IP addresses from recursive resolvers. This mechanism is effective when other methods, such as ECS (see Section 2.1.2), do not expose the client's IP address to the resolver.

The PKI in DNSCrypt uses X.509 certificates received from the resolvers. However, it does not fully depend on web PKI Certificate Authorities (CAs). Instead, it utilizes *DNS stamps*, which are transmitted through Out-of-Band (OOB) channels to validate resolver certificates.

2.2.2 (Stage 1) Strict DNS-over-TLS (DoT)

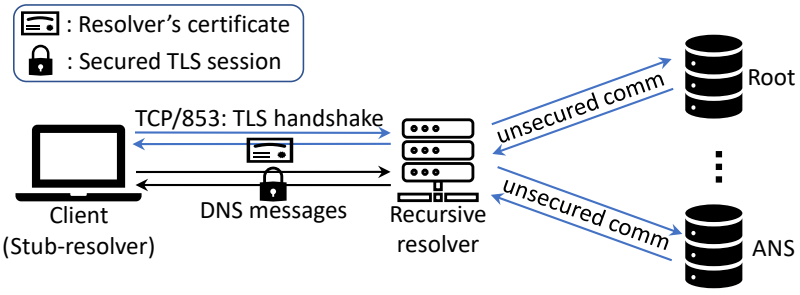


Figure 2.3: **DoT**: Sending DNS queries over TLS in Stage 1.

DoT was proposed to securely transfer DNS messages in Stage 1 using Transport Layer Security (TLS) [86, 176]. Figure 2.3 depicts DoT's working procedure. A client

initiates a TLS handshake over TCP port 853 with a DoT resolver and verifies the resolver's certificate using the web PKI trust anchors [91] (similar to the PKI used by browsers for TLS) or trust anchors from an OOB channel. Trust anchors might vary based on stub resolver configurations or the client's operating system [136]. Similar to HTTPS, DoT uses X.509 certificates to bind the public key of resolvers to their owner. Relying on the web PKI exposes DoT to inherit PKI-related vulnerabilities (*e.g.,* compromised CAs), as well as security weaknesses within any external trust anchors upon which it relies (when using OOB channels). After completion of the TLS handshake, the client and resolver use a secure TLS session with a shared secret to transfer DNS messages [86]. DoT provides two privacy modes: *strict* and *opportunistic*. In the *opportunistic* privacy mode, when DoT is supported by both ends but the resolver is not trusted by the client, the client uses DoT to encrypt queries, ensuring confidentiality and privacy without requiring authentication or trust in the resolver [86]. If DoT is not supported, the client falls back to Vanilla DNS. In the *strict* mode, successful authentication and encryption are mandatory to resolve a domain name. In the presence of an active attacker, the *opportunistic* mode may mislead clients about the provided security due to the possibility of downgrading to insecure Vanilla DNS.

When DoT is used with persistent TLS connections, it adds negligible latency compared to Vanilla DNS [110]. However, when connections are reused, the increased number of open connections can exhaust the resolver resources. Regarding anonymity, DoT does not provide any means to hide the client's IP address from resolvers. The TLS-based entity and message authentication offer resilience against false response injection attacks. Freshly generated session keys per handshake prevent replay from one session to the next, while per-record nonces prevent replay of messages within the same session, hence DoT is resilient against resolver replay attacks.

DoT suggests using the EDNS(0) padding option to pad the DNS messages with a variable number of octets (*e.g.,* filled with 0x00) and make them more resilient against traffic analysis and side-channel leaks in Stage 1. However, recent literature shows that encryption and padding techniques in DoT are not strong enough to resist ML-based traffic analysis, which can reveal the domain name visited by a user [85, 150]. As a result, the web pages visited by users will be revealed or classified, and name resolution can be censored based on that information. Thus,

DoT is not completely resilient against eavesdropping as the queried domain names can be inferred from the encrypted traffic and this information can be used for domain-name based censorship. The accuracy and effectiveness of such attacks vary based on the trained models and selected traffic features.

TCP prevents spoofed queries from being processed at the application layer without completing the handshake at the transport layer. Additionally, several mitigation measures have been proposed to improve the resistance of TCP to SYN flooding attacks [60]. Therefore, DoT provides means for mitigating DoS attacks targeting resolvers. However, DoT servers can be identified and subjected to censorship if they use the designated port number (853). Additionally, DoT messages are distinguishable based on their port number and thus can be censored completely. The majority of DoT resolvers on the Internet rely on and benefit from the established web PKI [91]. Consequently, their certificates are issued by web CAs, which employ standard cryptographic fields and log the certificates in Certificate Transparency (CT) logs [91].

2.2.3 (Stage 1) DNS-over-HTTPS (DoH)

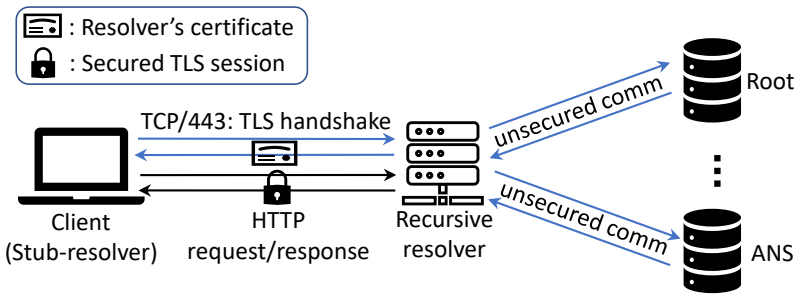


Figure 2.4: **DoH**: Sending DNS queries over HTTPS using URIs and HTTP methods.

DoH encodes DNS messages as HTTP requests/responses and sends them over TLS using the format specified by HTTP Uniform Resource Identifier (URI) templates (*e.g.,* https://dns.server.com/dns-query?) [80]. As DoH works on top of TLS, it provides many of the same properties as strict DoT in Stage 1.

The domain names of recursive resolvers are associated with their public keys through X.509 certificates often issued by the CAs of the web PKI. As such, similar to DoT, DoH inherits any limitations and vulnerabilities of the web PKI and trust

anchors (*e.g.,* web PKI CAs or private CAs) upon which it is dependent. However, unlike DoT, DoH does not support an opportunistic mode, meaning it does not fall back to an unencrypted connection when certificate validation fails. If certificate validation fails, name resolution fails. Thus, due to the absence of opportunistic privacy mode in DoH, resolvers must be explicitly authenticated. Moreover, Lu *et al.* [110] did not identify any DoH resolvers with invalid certificates in their measurements, as opportunistic privacy mode is not applicable in DoH when certificates are invalid. Also, DoH uses the same port (443) as HTTPS, merging DNS traffic with web traffic, and its traffic is thus less distinguishable [80, 110]. However, based on traffic characteristics, isolating DoH traffic from other web traffic is feasible, but it requires additional effort, and the accuracy of traffic isolation relies on various factors such as trained Machine Learning (ML) models and selected features [163].

Similar to DNSCrypt conducting traffic analysis on DoH traffic is costly for intermediate entities, as they need to initially separate the DoH traffic from the web before employing traffic analysis techniques on the isolated DoH traffic. Thus, DoH traffic is partially concealed, and traffic analysis of DoH is not straightforward. Similarly, distinguishing and censoring DoH servers requires specific methods to differentiate DoH servers from general HTTP servers. Due to its integration with the HTTPS protocol, DoH exhibits partial resistance to server censorship.

DoH specification suggests HTTP/2 compression, in addition to EDNS(0) DNS padding, to enhance the privacy of transmitted messages and make DoH traffic robust against traffic analysis and side-channel leaks [80, 116]. However, similar to DoT, Siby *et al.* [150] showed that simple padding schemes do not make a scheme resilient to ML-based traffic analysis attacks, but it makes traffic analysis more costly and less accurate. As DoH works on top of TLS, its remaining properties are similar to those of DoT.

2.2.4 (Stage 1) Strict DNS-over-QUIC (DoQ)

DoQ uses the QUIC transport-layer protocol [90, 159] and has been proposed as a general-purpose solution to enhance the security and efficiency of the DNS name resolution process in both stages, and zone transfers [89].[5] As Figure 2.5 illustrates, the DoQ workflow is similar to DoT with a faster handshake in DoQ by combining TCP

[5]The scope herein is Stage 1 use of DoQ.

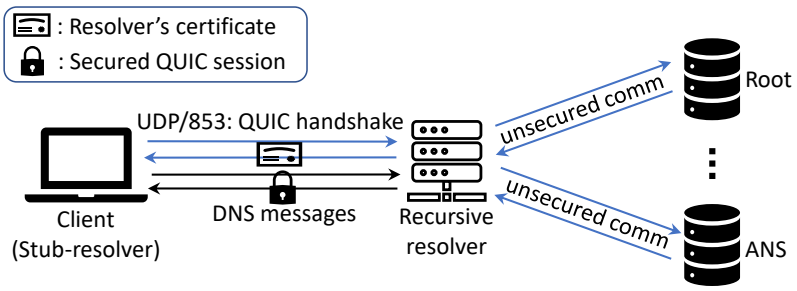


Figure 2.5: **DoQ**: Sending DNS over QUIC transport protocol.

and TLS handshakes in a single QUIC handshake. DoQ improves name resolution performance and solves some known issues in TCP (*e.g.,* head-of-line-blocking [28]) and UDP (*e.g.,* IP fragmentation [74]). In 2022, over 1200 DoQ resolvers were discovered over the Internet [102]. DoQ uses the designated port UDP/853; thus, its traffic can be distinguished, and access to DoQ servers can be blocked. However, the specification also recommends that using port UDP/443 can be beneficial by merging DoQ traffic with HTTP/3, and thus prevent port-based blocking [89].

DoQ provides similar security properties, usage profiles, and resolver authentication methods to DoT [49, 89]. Similarly to DoT, herein we analyze DoQ with the *strict* usage profile.

As the specification recommends, EDNS(0) padding or QUIC packet padding must be used prior to encryption of messages to mitigate traffic analysis attacks [89]. Padding at the QUIC packet level is preferred over EDNS(0) padding, as it provides better performance with the same level of protection [89]. To the best of our knowledge, there is no study of traffic analysis attacks to reveal the queried domain names in DoQ. However, for web traffic, QUIC padding does not eliminate fingerprinting attacks, and other application-layer defenses (*e.g.,* injecting cover traffic) are shown to be more effective [149].

Similar to DoT and DoH, we assume that with simple padding strategies, DoQ is presumed vulnerable to traffic analysis attacks despite the use of basic padding techniques. Such attacks can expose queried domain names, meaning the confidentiality provided by DoQ is incomplete. Moreover, the use of a designated port number makes DoQ traffic identifiable, reducing its capacity to fully conceal DNS communications. Consequently, DoQ messages and servers are not entirely resistant to censorship. The remaining properties of DoQ remain similar to those of DoT.

2.2.5 (Stage 1) DNS-over-DTLS (DoDTLS)

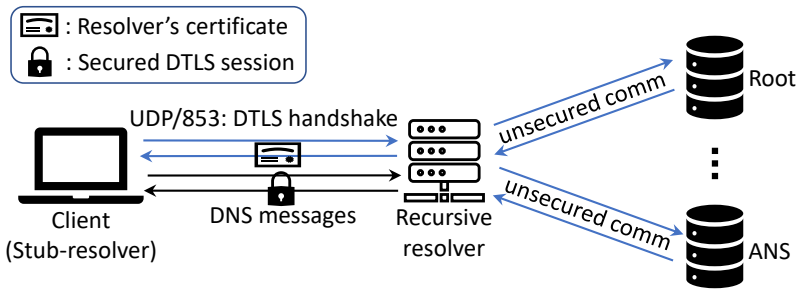


Figure 2.6: **DoDTLS**: Sending DNS over DTLS transport protocol.

As proposed in Request For Comments (RFC) 8094, DoDTLS uses Datagram Transport Layer Security (DTLS) to secure Stage 1 DNS messages [140]. DoDTLS uses UDP with port 853 to address head-of-line blocking and provide improved performance compared to DoT. DTLS has comparable security properties to TLS, with one of the primary differences being the handshake, which in DTLS contains extra header fields, cookies, and retransmission timers for error handling and DoS attack prevention [142]. DoDTLS authentication mechanisms are similar to those of DoT; hence, it has the same PKI vulnerabilities and challenges as DoT.

Similar to DoT, EDNS(0) padding could be implemented in DoDTLS. However, encryption and simple padding schemes do not significantly improve the privacy of the visited domains in TLS-based alternatives [150]. To the best of our knowledge, there are no real-world implementations of DoDTLS. Here, we assume that the similar padding scheme to DoT is not resilient to traffic analysis attacks, and thus does not provide complete confidentiality. A client must complete the DTLS handshake with the recursive resolver before sending a DNS query in the strict DoDTLS privacy profile. Consequently, the query is not processed at the application layer on the recursive resolver prior to the completion of the handshake. Thus, this method is resistant to DoS attacks and provides resilience against DoS attacks targeting DoDTLS resolvers. The remaining properties of DoDTLS are the same as DoT.

2.2.6 (Stage 1) DNS-over-Tor (DoTor)

DoTor can be implemented in different ways, such as a hidden resolver or a regular public resolver over the Internet queried through Tor network. In April 2018, Cloud-

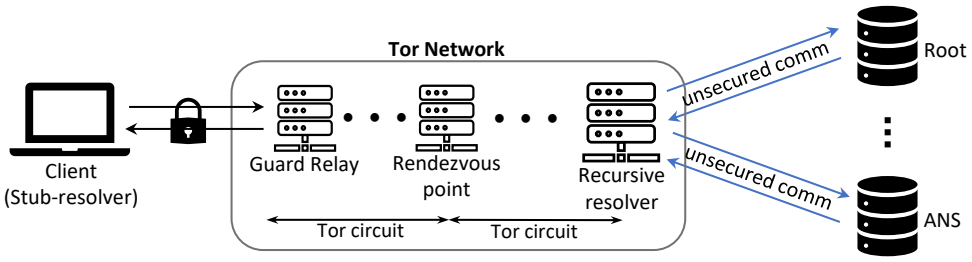


Figure 2.7: **DNS-over-Tor**: Based on Cloudflare's hidden resolver structure [144].

flare implemented a hidden resolver for DoTor [144].[6] Tor is vulnerable to a variety of attacks [69, 124, 135, 171]. One example is traffic correlation, where incoming and exit traffic of the Tor network can be analyzed to deanonymize clients [69, 124]. Bad relays [135] are another problem which can perform different types of manipulation, redirection, or eavesdropping to damage anonymity or manipulate a client's traffic [135, 171].

DoTor resolver can be implemented as a '.onion' service that can mitigate specific Tor correlation attacks, such as DefecTor [69] attack, where the adversaries analyze the exit relay name resolution for correlation attacks [144]. By using a '.onion' (hidden) recursive resolver, no name resolution takes place at the exit nodes. Instead, a Distributed Hash Table (DHT) lookup occurs within the Tor network, which mitigates the DefecTor attack, a technique that can reveal the recursive resolver used by a client [69].

Figure 2.7 illustrates the DoTor working procedure. First, the client establishes a secure session with the hidden resolver through the onion routers, including a rendezvous point [52]. TLS is used to establish authenticated encrypted sessions between the nodes. Then, the DNS traffic gets encrypted with the established keys and is sent through the Tor network to the hidden resolver. The entry node to the Tor network that the client directly communicates with is called Guard-relay. Upon receiving a query, the hidden resolver traverses the hierarchy of DNS using Vanilla DNS and sends back the response through the secured session to the client.

DoTor hides DNS messages using layers of encryption, enhancing the client's privacy from the recursive resolver and Stage 1 intermediate entities [52]. As a result, ISPs are unable to identify DNS traffic or correlate it with other traffic in order to reveal requested domain names. Additionally, Siby *et al.* [150] demonstrated

[6]Cloudflare's DoTor resolver supports DoT, DoH, and Vanilla-DNS over Tor.

that compared to DoT or DoH, DoTor is a more resilient scheme against ML-based traffic analysis attacks that reveal the pages visited by users. However, layers of encryption and TLS communications between relays can significantly increase the time and size overhead compared to alternatives, and Tor naturally sacrifices latency for security and anonymity. Depending on the user/application, the security benefits of sending DNS messages over Tor can outweigh the overhead that DoTor causes [123]. The hidden resolvers have some degree of protection against DoS attacks as they are placed behind the Tor network, benefiting from the distributed and load-balanced network. Thus, DoTor hidden resolvers are resilient against DoS attacks. Additionally, as Tor only supports TCP for internal communication, Tor nodes can implement SYN cookies and rate-limiting to mitigate DoS attacks.[7]

Among the schemes reviewed, DoTor is the only scheme offering complete confidentiality by encapsulating messages within encrypted, fixed-size Tor cells [150]. This approach effectively renders DNS traffic indistinguishable, ensuring confidentiality and concealing DNS traffic. Furthermore, no specific port number has been assigned to Tor communication, thus port 443 can be used to combine Tor with web traffic. Tor intermediate nodes hide a client's IP address from the recursive resolver, providing anonymity for the client by hiding its IP address. Additionally, using short-term ephemeral keys in Tor circuit communications mitigates replay attacks between different sessions. Also, TLS per-record nonces can mitigate replay of messages in the same session—so DoTor resolutions are resilient against replay attacks.

Although censoring governments cannot directly censor the recursive resolvers in DoTor, they can restrict access to the resolvers by blocking all Tor-related traffic (*e.g.,* using deep packet inspection to detect Tor-related traffic or blocking access to Tor-Related nodes) [1]. As DoTor relies on the Tor network as its underlying protocol, its resolvers are susceptible to censorship through measures targeting Tor itself. Cloudflare removes all included ECS from DNS queries [147], and thus in this implementation, DoTor also hides client IP addresses from both resolvers and the entities in Stage 2.

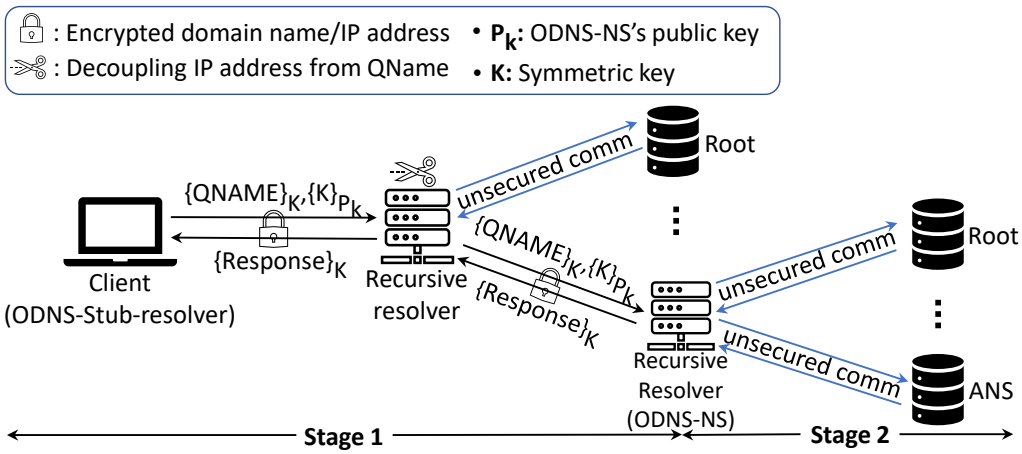


Figure 2.8: **Oblivious DNS**: Adding confidentiality and anonymity to DNS messages [147].

2.2.7 (Stage 1) Oblivious DNS (ODNS)

Schmitt *et al.* [147] proposed ODNS to add confidentiality and preserve clients' privacy from recursive resolvers by decoupling a client's IP address from the queried domain name, while resulting in a negligible page load-time overhead. An ODNS nameserver will not have access to a client's IP address, and intermediate entities in Stage 1 including the recursive resolver, which acts as a relay in Stage 1, will not have access to the client's queried domain name. The encrypted DNS query is forwarded by the recursive resolver to the ODNS nameserver (ODNS-NS), which is responsible for decrypting the query and traversing the DNS hierarchy to obtain the response.

Before name resolution starts, to get ODNS-NS's public key P_k, a client must send a special query to the recursive resolver, and the recursive resolver uses the anycast address to receive the nearest ODNS-NS's public key included in the EDNS extensions of the response message. Initially, the client obtains the public key of the ODNS-NS. Then, as Figure 2.8 illustrates, the client encrypts domain name in the DNS query with a symmetric key K, and then encrypts K with P_k [147]. The query and the encrypted symmetric key '$\{K\}_{P_k}$' are sent to the recursive resolver, which passes the DNS query to the ODNS-NS. The ODNS-NS that receives the query observes the recursive resolver's IP address (which acts as a relay) as the source IP address of the query. At this point, the ODNS-NS traverses the hierarchy

[7] `https://support.torproject.org/abuse/`

of the DNS to resolve the queried domain name, encrypts the final response with the client's symmetric key K, and sends it back to the client [147].

In Stage 1, the intermediate entities between the client and recursive resolver can see the client's IP address but not the plaintext query, and the entities between the recursive resolver and ODNS-NS neither know the query nor the client's IP address. ODNS removes ECS from the DNS queries in the stub resolvers to preserve client's privacy [147]. Thus, even if the client adds its IP address to the queries in the ECS, an ideal ODNS stub resolver forwards the query without it. In this case, ODNS completely hides the clients identity in Stage 2 by removing ECS and also hides client IP addresses from the ODNS-NS using a one-layer proxy. However, as ODNS is proposed for Stage 1, it does not offer any additional properties beyond hiding client IP address (by removing ECS) in Stage 2. As the integrity and availability of DNS requests are not among the goals of ODNS, it does not provide related properties to these goals.

2.2.8 (Stage 1) CGA-TSIG

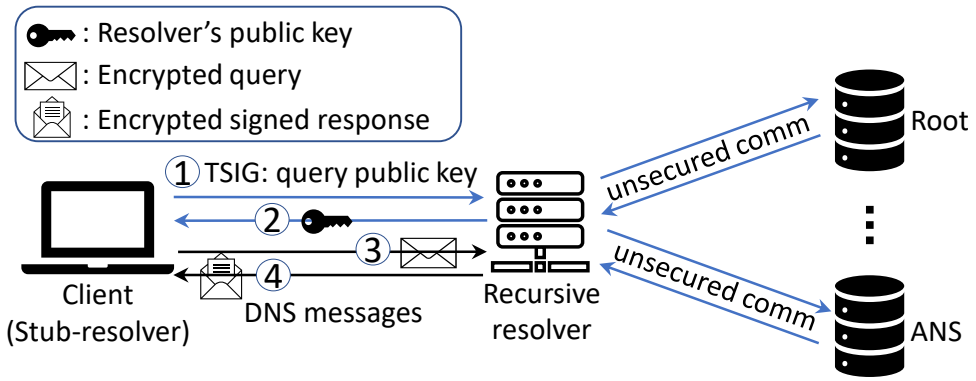


Figure 2.9: **CGA-TSIGe**: Enhancing name resolution by encrypting queries and encrypting and signing responses.

Cryptographically Generated Addresses (CGA) [19] and Transaction Signature (TSIG) [59] are the two fundamental components of the CGA-TSIG protocol. CGAs are a specific type of IPv6 addresses, whose interface identifiers (last 64 bits) are derived by hashing a signing public key along with other parameters. This type of address generation establishes a binding between the IPv6 address and a public signature key, allowing recipients to independently compute the hash and verify the

binding between the IP address and the public key. As outlined in its specifications, CGA operates without reliance on external PKI or CAs. Originally, CGAs were proposed to secure messages within the Secure Neighbor Discovery (SEND) protocol by signing messages with signature keys linked to IPv6 addresses, thus providing message authentication.

TSIG, the second component of CGA-TSIG, provides authentication for DNS processes, including dynamic updates, zone transfers, and recursive resolver responses. It employs a shared key and a keyed hash function to provide both authentication and message integrity. However, TSIG's specification requires that the shared key be distributed via an OOB mechanism, without defining a specific method for this key exchange.

The integration of CGA and TSIG in CGA-TSIG enables a mechanism for the authentication and secure transfer of DNS messages. By linking public keys to IP addresses through CGAs and utilizing them within the TSIG protocol, CGA-TSIG achieves a specific type of "*authentication*." Specifically, authentication here refers to the recipient's ability to verify that a public key is bound to a specific IP address. While this is sufficient for providing security in the IP-layer protocols (*e.g.,* SEND), it does not provide means for verifying identity of the entity behind the IP address, which is required in application-layer protocols like DNS [31]. In the context of DNS, additional mechanisms are needed for clients to ensure the resolvers they query are operated by their trusted entities (*e.g.,* Cloudflare).

As demonstrated in Figure 2.9, name resolution in CGA-TSIG begins with the client querying the resolver to obtain its public key (Step 1). Upon receiving the resolver's public key and its associated signature (Step 2), the client verifies the binding between the public key and the resolver's IP address. Subsequently, the client generates a symmetric key, encrypts it with the resolver's public key, and encrypts the DNS message with the generated symmetric key. Then, the client sends the encrypted key along and the encrypted query with a hash of the message for integrity (Step 3). The resolver decrypts the symmetric key using its private key, processes the query, and checks its integrity. After obtaining the response by querying the DNS hierarchy, the resolver encrypts the response with the symmetric key from the query, signs the encrypted message, and sends it to the client (Step 4). The client decrypts the response using the symmetric key and verifies the included signature.

Although CGA-TSIGe provides message authentication and confidentiality to DNS resolution in Stage 1, the authentication is limited to verifying the binding between the IP address and public signing key that was used for generating the IP address, which is insufficient in the DNS context. This limitation highlights the need for additional mechanisms to trust the entity behind a resolver's IP address (*e.g.,* CAs in the web PKI or other PKI).

Secure DNS Schemes in Stage 2: Aside from schemes that to enhance Stage 1, in this part, we provide a detailed survey of two schemes that have been proposed to enhance the security or privacy of Stage 2.

2.2.9 (Stage 2) DNSSEC

A decade after proposal of DNS RFCs [119, 120], the initial version of DNSSEC was standardized [58] and later updated to its current version defined in RFCs 4033-4035 [11, 12, 13]. DNSSEC was designed to enhance Vanilla DNS by adding authenticated denial of existence, message authentication, and integrity properties. DNSSEC adds security properties to the resource records of DNS zones by using four primary resource records, namely Resource Record Signature (RRSIG), DNSKEY, Delegation Signer (DS), and NextSECure (NSEC) records.

As illustrated in Figure 2.10, DNSSEC adds two public-private key pairs to each zone, which are referred to as Key Signing Key (KSK) and Zone Signing Key (ZSK), published as DNSKEY records. ZSK is used to sign all the resource records sets (RRSets) in a zone (black arrows in Figure 2.10), except for DNSKEY RRSet. On the other hand, KSK is used exclusively for signing the DNSKEY RRSet (red arrows). RRsets are a collection of DNS records with the same name, class, and types, but potentially different data values [13]. DNSSEC signs each RRSet to ensure the authenticity and integrity of all records within the set, preventing attackers from modifying or removing individual records. DNSSEC uses DS records to establish a secure trust relationship between a parent zone and a child zone in the DNS hierarchy. Each DNS zone, except for the root, generates a cryptographic hash of its public KSK along with additional metadata, which is sent to the parent zone to be published as a DS record. Similar to other RRSets in a zone, the DS record in a parent zone is signed with its ZSK.

Since no zones are above the root, the root zone's public KSK is defined as a trust anchor and embedded in DNSSEC-validating resolvers. Using the root zone's public

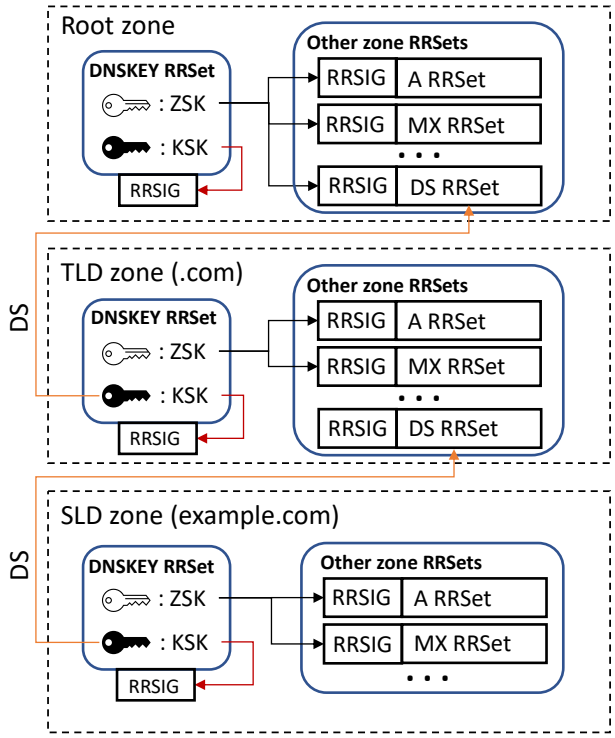


Figure 2.10: An overview of DNSSEC records and chain of trust (H())

KSK, a validating resolver can validate the root zone's ZSK. The resolver can then validate the DS records of the TLDs in the root using the public ZSK of the root zone. The DS records contain the cryptographic hash of the public KSK associated with the child zones (*e.g.,* .com). Subsequently, when the validating resolver obtains the DNSKEY of the child zone, the resolver verifies its consistency with the DS record that it obtained from the parent zone. The child zone's KSK is then used to validate the public ZSK of the child zone (*i.e.,* validating DNSKEY). This iterative process continues until the resolver reaches the name server authoritative for the requested DNS record, and validates the requested resource record using the ZSK associated with the final zone. Finally, the response is transmitted to the client entity that initially sent a query to the recursive resolver.

DNSSEC is typically implemented and used in Stage 2 of the name resolution process, and the stub-to-recursive communication in Stage 1 remains unsecured. Stub resolvers in DNSSEC are often configured as *non-validating* and *security-aware* stub resolvers. The term "*security-aware*" denotes that the stub resolver understands DNSSEC-related bits, extensions, and records [11]. "*Non-validating*" indicates that

these stub resolvers do not completely validate DNSSEC record signatures. Instead, they rely on one or more recursive resolvers to perform the resource record validation [11]. Thus, the communication channel between stub and recursive resolvers must be secured separately, and the recursive resolver must be relied on to provide correct responses.

Regarding caching DNSKeys, similar to other original DNS records, DNSKEYs also have a Time To Live (TTL) field, which is a 32-bit value that determines the duration for which these keys should be cached on the resolvers. While caching the DNSKEYs of DNSSEC-protected zones for short durations provides more flexibility and responsiveness to key compromise situations, short-time caching periods impose additional computational and bandwidth load on ANSes and resolvers. In addition, short-time caching for the zone keys increases the name resolution times, as the keys expire from DNS caches more rapidly, and resolvers need to traverse the DNS hierarchy to obtain the keys of the intended zone for authenticating DNS responses. On the other hand, long caching durations for the zone keys result in a lack of flexibility in the key compromise situations. However, larger TTL values for DNSKEYs improve the name resolution performance, as the keys are queried less frequently as their presence in the resolver caches is more likely. Taking both sides into account, the caching time should neither be excessively long to mitigate the damage of key compromise situations, nor very short to minimize the name resolution delay.

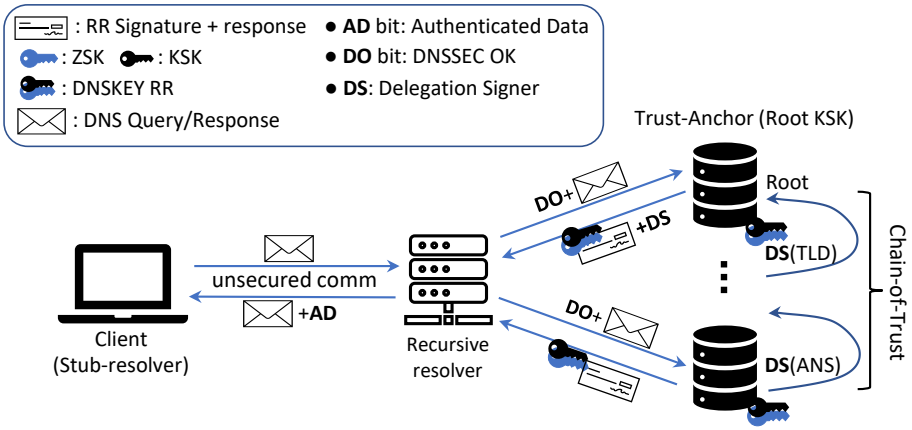


Figure 2.11: **DNSSEC:** Providing message authentication using digital signatures in Stage 2.

Figure 2.11 illustrates the process of name resolution in DNSSEC using a *non-validating security-aware* stub resolver. The client sends a DNS query to a *validat-*

ing security-aware recursive resolver. Subsequently, the recursive resolver traverses the DNS hierarchy, fetching DNS records along with DNSKEYs, DS records, and RRSIGs, which are used for validating the DNSSEC response, per Figure 2.10. Finally, the recursive resolver retrieves the requested resource record from the ANS of its zone and verifies the response. Upon successful validation, the recursive resolver caches the response and sends it to the client with Authenticated Data (AD) bit set.

Traffic Amplification in DNSSEC: DNSSEC uses UDP as transport layer protocol with EDNS (see Section 2.1.1) [41], which enables transmitting DNS responses larger than the original DNS maximum response size (512 bytes) over UDP. These design choices alongside the added signatures and keys in DNSSEC responses, enable reflection amplification attacks with significant amplification factors up to 100× [25, 162]. Thus, attackers can exploit DNSSEC to amplify the traffic in Distributed Denial of Service (DDoS) attacks by sending queries that produce large responses directed at targeted destinations [37, 162].

Unsigned Records: In DNS, non-authoritative delegating records within a zone are those that delegate authority to another DNS zone (*e.g.,* NS or glue records), without providing authoritative responses for the delegated zone. In DNSSEC, non-authoritative delegating records within DNSSEC-protected zones are not signed. Specifically, glue and NS resource records of child zones are not part of the authoritative DNS data secured in the parent zones. Thus, these non-authoritative records in the parent zones are transferred unsigned [11, 75], and these unsigned records are not protected by DNSSEC. The injection and caching of these unsigned records in a validating recursive resolver can result in DNSSEC validation failure, potentially causing disruptions in the resolution of DNS queries when attempting to access legitimate nameservers. Additionally, in instances where the recursive resolver falls back to Vanilla DNS or accepts unathenticated responses, these unsigned records can result in false responses and *downgrade attacks* [75].

Authenticated Denial of Existence in DNSSEC: Aside from message authentication, DNSSEC adds another property, known as *authenticated denial of existence*. When a resource record does not exist within a zone in Vanilla DNS, the authoritative name server returns an NXDOMAIN (Non-Existent Domain) response. However, a single authenticated NXDOMAIN resource record in a DNSSEC-signed zone could be injected as a response to all queries to a zone and thereby disrupt the availability of legitimate zone records. To address this problem, DNSSEC introduced

NSEC resource records. When a resolver queries for a non-existent record in a zone, NSEC records return the name before and the name after the non-existent queried domain name as a signed response. Consider a zone such as `example.com`, which contains `alice.example.com` and `charlie.example.com` subdomains. When a resolver queries `bob.example.com` from a name server of this zone, the authoritative name server returns the following record:

```
alice.example.com. NSEC charlie.example.com. A RRSIG NSEC
```

This signed NSEC record indicates that there is no `bob.example.com` subdomain, and the name before it is `alice`, and the name after it is `charlie`.

Due to design of NSEC records, they generate an alphabetically-ordered chain of existing records within a zone. This enables a *zone walking* or *zone enumeration* attack, whereby an attacker can iteratively query all of the existing NSEC records in a zone to discover all of the domain names included within that zone. Although DNS zone contents are not inherently secret, zone contents can reveal valuable information about the targeted domain (*e.g.,* existing servers or applications) as a part of reconnaissance phase in an attack.

NSEC3 [14] was proposed to address this zone enumeration vulnerability in the original NSEC design. The original NSEC uses plaintext resource record names in NSEC records, NSEC3 aims to mitigate zone enumeration attacks by using cryptographic hashes to obscure the resource record names used in NSEC3 records. Furthermore, NSEC3 employs random salt values and iterative hashing, to mitigate pre-computed and offline guessing attacks that aim to discover NSEC3 records. When a client queries for a non-existent domain in a zone, NSEC3 responds by providing two cryptographic hashes of two resource record names between which the hash of the queried name is placed. NSEC3 significantly increases the difficulty of zone enumeration, but remains vulnerable to offline-guessing attacks.

NSEC5 [67] was proposed to prevent offline guessing attacks on hashed resource record names in NSEC3. Instead of using cryptographic hash functions as above, NSEC5 employs a Verifiable Random Function (VRF). In NSEC5, a VRF is considered as a public-key variant of the hash function, which accepts a resource record name and a private key (VRF key) as inputs and generates a corresponding NSEC5 proof [166]. A client with access to the corresponding VRF public key of the zone can validate the obtained NSEC5 records; however, without the VRF private key, an entity is unable to generate NSEC5 records and perform offline guessing attacks.

Stale Records: Another limitation in DNSSEC is the presence of signed and unexpired but stale resource records. DNSSEC RRSIGs have an expiry window, determined by their `Inception` and `Expiration` fields. Stale records in DNSSEC come to existence when a signed resource record exists, and before its expiration (the time in the `Expiration` field has not yet been reached), a new resource record with identical `name` but different `data` fields gets signed. Although the resource record has been updated and a new, valid resource record is now available, the stale resource record is signed and has a valid and unexpired signature. Stale resource records in DNSSEC are susceptible to replay, enabling stale/false response injection. Replaying resource records can also be exploited to misdirect clients to non-optimal CDN nodes [71]. The absence of real-time (fresh) signatures in DNSSEC enables replay of stale records in DNSSEC.

Expired Zone: DNSSEC RRSIGs have a fixed expiry window and DNSSEC-protected zones must renew these signatures before the `Expiration` time. Failing to renew DNSSEC signatures results in zone records becoming invalid, rendering responses unacceptable to DNSSEC-validating resolvers. Zone records would thus become unreachable to clients that use validating resolvers. Expired zone records that result in DNSSEC validation failures are not only prevalent in smaller zones but have also been frequently observed within TLDs.[8]

2.2.10 (Stage 2) DNSCurve

DNSCurve [25, 46] was proposed in 2009 as a high-speed ECC-based approach to enhance security and privacy of Vanilla DNS in Stage 2. DNSCurve was developed as an alternative to DNSSEC, aiming to resolve the security, privacy, and amplification problems associated with DNSSEC. This scheme employs authenticated encryption, using the keys established through Curve25519 ECC functions, to secure transmission of DNS messages within Stage 2. The public keys of ANSes are encoded (using Base32) and prepended (as a subdomain) to the domain names of ANSes (*e.g.,* "uz5jm...235c1.dnscurve.org"). These public keys are 54 bytes long, including a hard-coded string 'uz5', added at the beginning, indicating support of DNSCurve by an ANS.

As Figure 2.12 illustrates, Stage 1 communication in DNSCurve remains unse-

[8] `https://ianix.com/pub/dnssec-outages.html`

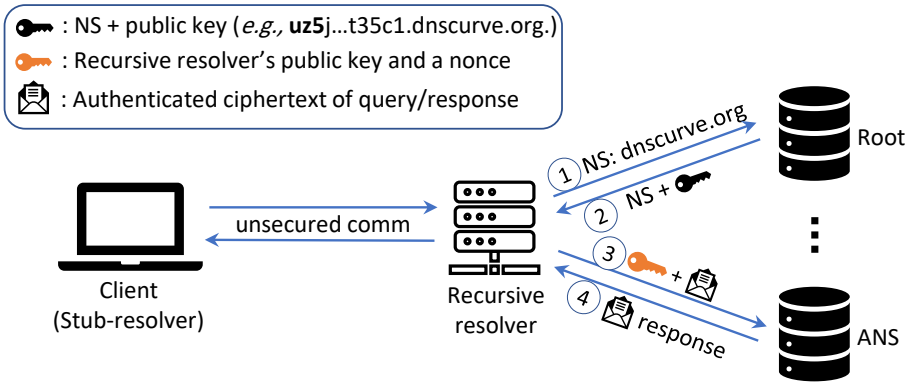


Figure 2.12: **DNSCurve**: Using authenticated encryption in Stage 2.

cured. Unlike DNSSEC, which uses the AD bit to indicate signature verification to clients in Stage 1, DNSCurve does not provide any explicit signaling to clients in Stage 1—clients must rely on the recursive resolver and ANSes to correctly implement and use DNSCurve [6]. However, in Stage 2, when the recursive resolver queries the name servers (*i.e.,* NS records) of 'dnscurve.org' from its parent zone (org.) name server (Step 1), the TLD (org.) name server returns the NS records of 'dnscurve.org' (Step 2). The NS records of 'dnscurve.org' start with the encoded public key indicated by the 'uz5' hard-coded string (*e.g.,* 'uz5j...t35c1' in Figure 2.12). The recursive resolver uses the name server's public key and its own private key to generate a shared secret used with a unique nonce to encrypt a DNS query and generate a Message Authentication Code (MAC) [24, 46]. Afterwards, the resolver sends the encrypted query with its public key and the nonce to the nameserver of 'dnscurve.org' (Step 3). The name server generates the same key stream using its private key, resolver's public key, and the nonce. Subsequently, the nameserver decrypts and verifies the query, generates an authenticated and encrypted response, and sends it back to the recursive resolver (Step 4). In the final step, the recursive resolver decrypts the received response and verifies the MAC of the received response, stores the response in its cache, and sends it back to the client.

As demonstrated, DNSCurve establishes a secure link between a recursive resolver and an ANS. Instead of associating long-term secrets with each zone as DNS records, in DNSCurve, the long-term keys are associated with name servers. DNSCurve uses Curve25519 to generate public key pairs and establish shared secrets [24]. Then, a per-packet nonce is used along with the established secret to generate a key stream. The key stream is then used in the Salsa20 stream cipher to

encrypt DNS messages. Additionally, the same key stream is used with the Poly1305 MAC algorithm to provide message authentication. Due to the absence of multiple signatures and key records in DNSCurve, responses are smaller than DNSSEC. Moreover, responses in DNSCurve are often limited to 512 bytes prior to encryption [46]. If a response exceeds this limit, it is truncated, and the client must retry over TCP [46]. Consequently, adversaries cannot use DNSCurve infrastructure for reflection amplification attacks.

Unlike DNSSEC, there is no need for additional resource records in DNSCurve. All messages in DNSCurve, including the NXDOMAIN responses, are uniquely encrypted and authenticated in real-time using the generated key stream with a per-packet nonce. Thus, DNSCurve does not employ NSEC records to provide authenticated denial of existence and is not vulnerable to NSEC-based zone enumeration attacks. Besides, as different recursive resolvers and ANSes generate different shared secrets, responses to one resolver cannot be replayed to another. Also, due to the used per-packet nonce in generating stream key, an old message cannot be replayed to the same recursive resolver.

Similar to DNSSEC, in DNSCurve, Stage 1 must be secured separately. Also, in DNSCurve, resolvers do not signal clients regarding the use of DNSCurve in Stage 2 [6]. Therefore, even if Stage 1 is secured, clients do not have means to know that the name resolution was protected by DNSCurve in Stage 2. In order to employ real-time authenticated encryption, DNSCurve requires private keys to be present on the nameservers of a zone. Therefore, when anycast is implemented by a zone owner for load balancing and enhancing performance, the private key must be present on all nameserver instances to facilitate online cryptographic operations [145]. The anycast instances are distributed across distinct geographical locations and administered in different regions, which the zone owner may not completely trust (*e.g.,* the root zone [168]). Consequently, vulnerabilities of anycast server instances will impact the duplicated private key on said servers.

Aside from the replication of long-term secrets on nameserver instances and the absence of appropriate key distribution mechanisms to distribute keys among the nameserver instances in DNSCurve, the public keys of DNSCurve are prepended (as noted earlier) as a subdomain to the nameserver names. Consequently, recursive resolvers typically obtain the nameserver keys from the nameserver of the parent zone. The nameserver records will be obtained with confidentiality and authenticity

only if the parent zones up to the root zone have also implemented DNSCurve and DNSCurve public keys have been retrieved authentically. Therefore, it is necessary to incorporate a DNSSEC-like trust anchor in the root zone and resolver software for DNSCurve to ensure secure communication with the root nameservers, securely obtain NS records and public keys of subordinate nameserver in the DNS hierarchy, and thereby, protect transmitted DNS messages. The absence of a clearly defined trust model and chain of trust, by which the resolvers can trust the keys of nameservers in DNSCurve, is another problem of this scheme [145].

In order to have a DNSSEC-like trust anchor in DNSCurve, the root nameservers are required to include their public keys in the recursive resolvers and replicate their private keys on all of the nameserver instances for live cryptographic operations. ICANN, the entity responsible for maintaining the root zone, opted against replicating DNSCurve private keys across all nameserver instances to avoid the risks associated with replicating private keys on nameserver instances [168]. On the other hand, the root zone has adopted DNSSEC, leveraging pre-signed resource records, which eliminates the need to duplicate private keys across nameserver instances within the root zone.

2.2.11 DNS Enhancements Requiring Modest Changes

Here we review techniques and simple schemes that have been proposed to address security or privacy issues of DNS resolution through the use of relatively minor infrastructure modifications.

QName Minimization

Although it is not necessary, recursive resolvers typically send the full QName (*i.e.,* the queried domain name such as *www.example.com*) to ANSes as they traverse the DNS hierarchy. Therefore, the ANSes can gather information about the queried domain names from resolvers, and sometimes ECS is also included in those queries, which can leak client-related information. QName minimization [27] aims to reduce the amount of information sent to upstream nameservers. Therefore, as a recursive resolver traverses the hierarchy of the DNS to resolve a domain name, it includes only the necessary part of the QName in a query. For example, when a recursive resolver is resolving '*www.example.com*', it queries the root name servers

with only '*.com*' as the QName instead of the complete domain name.

Increasing Entropy Against Off-path Attackers

Since Kaminsky [98] demonstrated the vulnerability of recursive resolvers to off-path cache poisoning, various enhancements have been proposed to increase the entropy in DNS responses, thereby reducing the probability of adversaries successfully injecting false responses. The increased entropy hinders an off-path adversary from injecting a false response that is accepted as if generated by a legitimate name server. However, in-line adversaries with access to DNS queries and responses can still observe the randomized fields and inject false responses.

As a part of DNS header, Transaction Identifier (TXID) is a 16-bit value chosen by clients and typically generated randomly to match DNS queries and responses. Since the amount of entropy that was added by TXID to DNS responses was not sufficient to prevent the Kaminsky attack, Daniel J. Bernstein [88] initially introduced and implemented source port randomization in *djbdns*.[9] Source port randomization in DNS queries introduces another 16-bit entropy to DNS responses that an off-path attacker must guess correctly and use in a false response to be accepted. Nevertheless, source port randomization is not effective in scenarios where a recursive resolver that issues a query is behind a firewall, or in cases of Network Address Translation or Port Address Translation (NAT/PAT), which modifies the source port number into a predictable value that can be guessed by off-path adversaries [133]. Additionally, researchers have demonstrated that network side-channels can be used to infer the value of randomized source port values in DNS queries, which renders the entropy introduced by source port randomization ineffective [114].

In addition to TXID and source port randomization, Dagon *et al.* [42] proposed enhancing DNS security by incorporating case randomization in queried domain names, *i.e.,* mixing uppercase and lowercase letters. This increases the entropy of DNS responses, making cache poisoning attacks more challenging for off-path adversaries. However, since this technique only modifies the queried domain name, the entropy introduced is limited by the length of the domain name. To increase the entropy of DNS responses by altering queried domain names independent of the length of domain names, Predisci *et al.* [133] proposed Wildcard SECure DNS (WSEC DNS). WSEC DNS does not require software changes on the server side but

[9] `http://cr.yp.to/djbdns.html`

requires adding specific `TXT` and `CNAME` wildcard records to the zone files to indicate support of WSEC and to operate. When a recursive resolver generates a query to a WSEC-enabled name server, it appends a random string to the queried domain name and formats the query in a specific way so that the name server can process it correctly without the added random string.

Since on-path adversaries have access to the included randomness in DNS queries and responses, these randomness-based solutions are effective only against off-path adversaries. Moreover, researchers have demonstrated attacks by off-path adversaries that infer or bypass the random values included in DNS messages. For example, Herzberg *et al.* [73] introduced a technique for predicting the source ports of queries of resolvers behind a Network Address Translation (NAT). In other researchwork, Herzberg *et al.* [74] demonstrated a method for bypassing source port randomization of responses, when the responses from ANSes are fragmented. Man *et al.* [114] used network side-channels for inferring the DNS query source ports and cache poisoning.

Defenses Against Centralization

The growing trend of adopting remote public recursive resolvers (*e.g.,* Cloudflare's resolver at 1.1.1.1) in client-side software, such as web browsers and pre-configured devices, including IoT devices [122, 139] has raised privacy concerns about centralized data being gathered by resolvers [83]. Hounsel *et al.* [84] proposed and evaluated various approaches for distributing DNS queries to prevent centralization of DNS queries in a single recursive resolver. Similarly, Hoang *et al.* [76] proposed the K-resolver mechanism to distribute queries among multiple resolvers. This splits the queries sent to each resolver to prevent any single resolver from gaining complete knowledge of all queries over time. In order to address query-centralization-related privacy issues, other solutions such as transferring Vanilla DNS [144] and DoH queries [123, 144] over Tor network [144], and ODNS [147] have been proposed to hide the IP address of the client who issues a DNS query from recursive resolvers.

Aside from approaches for mitigating query centralization in Stage 1, QName minimization (Section 2.2.11) has been proposed to enhance the privacy of client queries in Stage 2 of name resolution. QName minimization significantly reduces the query information exposed to and potentially collected by higher-level zones in the DNS hierarchy (*e.g.,* root or TLDs) in Stage 2.

Additionally, Asoni *et al.* [17] proposed paged DNS, as a scheme to enhance the privacy of resolvers querying the root and TLDs as centralized points in Stage 2. In paged DNS, DNS records on ANSes of the root and TLDs are grouped into pages, and resolvers query pages instead of single specific queries. Therefore, PageDNS requires modifications to ANSes and recursive resolvers to enable querying, responding, and updating the pages of records. However, Stage 1 and subordinate zones under the TLDs continue to use Vanilla DNS.

Another technique proposed to improve DNS resolution privacy is known as Range Queries [174], which includes a set of cover queries alongside each actual query, thus obscuring the specific query being made. Range Queries are susceptible to a *Semantic Intersection* attack by which the resolver can learn the hostnames belonging to a site and using this to separate dummy queries from real ones [62]. Additionally, timing attacks are another threat to Range Queries, as adversaries can analyze the behavior of the entity issuing queries and infer the real queried record based on the timing of queries [62]. If the querying entity initiates a subsequent set of range queries before all queries from the initial set have been processed, the resolver may infer that the remaining queries in the first set are dummy queries [62]. Both PageDNS and Range Queries are compatible with DNSSEC to add message authentication; however, their implementation would result in significant overhead when used with DNSSEC. This is because the additional responses retrieved for the records within pages and dummy queries would also include DNSSEC RRSIGs, thereby increasing the overhead and processing requirements.

2.2.12 DNS Enhancements Requiring Substantial Changes

Here, we briefly discuss DNS schemes that require substantial modifications to the name resolution procedure. These are categorized into two groups: P2P-based schemes and Blockchain-based.

P2P-based Schemes: As one of the first attempts to use peer-to-peer networks for name resolution, Cox *et al.* [38] proposed and evaluated DDNS as an alternative to DNS. DDNS employs Distributed Hash Tables (DHTs) based on the Chord [156] peer-to-peer protocol to store and query DNS records. Although DDNS benefits from the load-balancing and resilience of peer-to-peer networks in the face of attacks against availability, it add time overhead versus Vanilla DNS. Overlook is another proposed scheme that uses DHTs on top of the Pastry [143] overlay network. As

a part of the GNUnet project, the GNS [167] is proposed as a decentralized and censorship-resistant alternative to DNS. Its objective is to enhance name resolution privacy by incorporating DHTs for distributing and resolving key-value mappings.

Blockchain-based schemes: Namecoin was the first censorship-resistant blockchain-based name system, Namecoin was forked from Bitcoin and its first block was mined in 2011 [126]. Kalodneret *et al.* [97] conducted an empirical analysis of Namecoin ecosystem. Since its development many other blockchain-based naming systems, such as Blockchain Name System (BNS) [3] and Ethereum Name Service (ENS) have been proposed and implemented [96]. Since all these schemes rely on decentralized blockchains, they are censorship-resistant and no centralized entity acts as a single point of failure or has control over their registered domains. However, due to the need for radical modifications to the existing DNS infrastructure and DNS resolution process, none of these schemes have gained widespread adoption as replacements for Vanilla DNS.

2.3 Secure Delegation of Authorization

In Internet protocols that employ long-term secrets, one method to avoid replication of these secrets is to implement secure delegation of authority to other entities in the protocol without directly sharing the long-term key. In this section, we provide background on methods for delegation of authorization to explore techniques that avoid sharing (replicating) long-term private keys.

Secure delegation of authorization and its revocation have been fundamental challenges in different PKIs [33]. For example, in the intersection of CDNs and web X.509 PKI, there is a need for delegation of authorization for TLS termination. A CDN is made up of numerous geographically distributed servers, known as edge servers. These are used to efficiently and often securely transfer content to end-users primarily based on their geographical location. In some instances, as a simple delegation method, domain owners employing CDNs for TLS-based services share their long-term private keys with the CDN to enable TLS sessions to end-users. By transferring private keys to CDN instances, the private key owner effectively delegates authority to the CDN to provide TLS-based services on their behalf. However, the long-term key is then replicated on all the CDN edge servers and exposed to attacks targeting these servers.

Delegation methods can likewise be employed within the DNS context to grant authority to the DNS server instances (*e.g.,* when anycast is used) within a zone, facilitating secure communication with the DNS server instances and eliminating the need to duplicate long-term keys across all server instances.

Long-term private key sharing, as described in the context of delegation for TLS-based services, is highly discouraged. Private key sharing does not provide fine-grained authorization for delegatees. Furthermore, the compromise of the long-term secret on any of the delegatees jeopardizes security across all other delegatees. In order to address the challenges of long-term key sharing, various methods have been proposed to enable partial or complete delegation without requiring key sharing, while also supporting the revocation of such privileges [33].

One class of delegation schemes involves securely storing the long-term secret on a designated server. Each communication with delegatees, then requires interaction with the server that stores the long-term secret. For example, in the TLS context, Keyless SSL [157] relies on TLS handshake proxying [155] and completes each handshake to the edge servers of a CDN by communicating to a designated server that stores the long-term TLS private key.

SSL splitting [106] is another example of this class of delegation. In this approach, after SSL connection establishment, the designated server that holds the long-term key sends the MAC and a unique identifier for each TLS record to the delegatee. The delegatee then verifies its cache for the corresponding payload associated with the MAC and sends it to the requester. In Keyless SSL [157], the delegatee interacts with the designated server only during the TLS handshake process. In contrast, SSL splitting requires per-record communication with the designated origin server, limiting the advantages of delegation in the context of CDNs. Chuat *et al.* [33] conducted an extensive analysis of delegation and revocation schemes, and developed a framework for their systematic analysis.

Proxy certificates [169] and delegated credentials [22] are another type of secure delegation protocols that do not require per-connection or per-record communications with the designated server that stores the long-term key. In these schemes, the designated server that holds the long-term secret generates data structures (*e.g.,* certificates), and signs them using the long-term secret. These signed structures are then used by the delegatees to securely offer services without the need for the long-term secret or frequent communications with the designated server.

2.3.1 Delegated Credentials

Here, we provide a detailed description of delegated credentials, which, when combined with an appropriate revocation system, offer a promising solution for the delegation and replication of long-term secrets, without introducing per-connection delays [33]. In the web X.509 PKI domain owners have a limited control over the fields of the TLS certificates (*e.g.,* expiry window or supported algorithms) that a CA issues. Additionally, the use of short-term certificates increases the load on CAs, and a brief outage of the CAs may result in the unavailability of TLS-based services. To address these limitations through delegation of authorization, a method known as delegated credentials has been proposed and standardized [22]. In the web PKI, the CA-issued certificates for end-users cannot be used to sign other certificates; however, if their *digital signature* bit is enabled in their *key usage* extension, such certificate can be used to sign and validate other objects beyond certificates and Certificate Revocation Lists (CRLs). Delegated credentials are small data structures, digitally signed by endpoint X.509 certificates, thereby facilitating the delegation of authorization from endpoint TLS certificates. These delegated credentials are then distributed among the edge servers within the CDN context and their corresponding private keys are used to effectively complete TLS handshakes without requiring long-term private key replication.

Listing 2.1: Delegated credendtials structure (based on [21])

```
struct {
    struct {
        uint32 valid_time; // DC lifetime
        SignatureScheme signature_algorithm;
        Pkey DC_public_key; // public key of DC
    } Credential;
    SignatureScheme algorithm; // DC signature algorithm
    Signature signature; // delegation signature
} DelegatedCredential; // short-form name: DC
```

As shown in Listing 2.1, a delegated credential is composed of a structure that contains an other structure for the "Credential." In the Credential structure, there is a *valid_time* field, which specifies the duration in seconds relative to the `NotBefore` field of the certificate responsible for issuing (*i.e.,* signing) the delegated credential. This indicates the period during which the delegated credential is considered valid.

The *DC_public_key* and *signature_algorithm* specify the public key and signature algorithm of the Credential key pair. The *DelegatedCredential.algorithm* and *DelegatedCredential.signature* specify the respective signature algorithm and the signature that establishes that binding between the end entity certificate and the delegated credential.

In Chapter 4, we use a structure analogous to delegated credentials within the DNS context to mitigate the challenges associated with sharing long-term private keys among nameserver instances within DNS zones.

2.4 Formal Analysis of Security Protocols

Verifying security and privacy properties of protocols often involves significant challenges, primarily due to the difficulty of manually analyzing or proving these properties, including manually exploring all the possible protocol behaviors. Consequently, researchers have investigated mathematical and automated methods to facilitate the verification of security properties in protocols. Formal verification methods can be broadly classified into two categories: symbolic and computational [29].

In Chapter 6 we employ a symbolic analysis approach to increase assurance in the provided properties of our new scheme. In the symbolic model of security protocols, cryptographic primitives are represented as symbols, while messages are expressed as *terms* used by these primitives. These primitives are employed in *rules* that define the protocol's specification, with the rules deterministically modeling the protocol's behavior. This approach simplifies the analysis by ignoring the mathematical complexities of cryptography and focusing on the logical interactions between entities within the protocol.

For example, `senc(m, k)` is a symbolic function for symmetric encryption of a message m with a key k. If a rule has access to k or the defined adversary obtains knowledge of k, they can use the decryption function `sdec(c, k)` to decrypt the plaintext message m. Notably, cryptographic primitives in the symbolic model are assumed to be perfect. Thus, in this example, only with access to the key k the message can be decrypted.

2.4.1 The Tamarin Prover

Tamarin is a tool used for symbolic modeling and formal verification of security protocols [117]. It offers pre-defined symbolic cryptographic primitives, such as Diffie-Hellman (DH) key agreement, symmetric encryption, and digital signatures. Tamarin has been employed for modeling and analysis of security protocols such as TLS 1.3 [39], 5G authentication [23], RHINE [55], and other security protocols [146]. Thus, Tamarin is as an appropriate tool for a formal security analysis of the secure DNS scheme that we propose in this thesis (DNSSEC+).

The default attacker model in Tamarin is the Dolev-Yao [54] adversary model, which assumes complete control over the network: the adversary can read, modify, drop, or fabricate and inject network messages. A Tamarin user defines a symbolic model of a protocol in Tamarin syntax and specifies the properties to be proven. Tamarin is capable of automatically generating proofs to establish the validity of the specified properties, without requiring manual interaction. Tamarin also provides an interactive mode with a graphical interface that offers a step-by-step manual proof development process, with visualization of the constraint solver and of any discovered attacks.

Protocol Specification: In Tamarin, the allowed protocol interactions and steps are specified using multiset rewriting rules. These rules form a labeled transition system, with a global state composed of *facts* that represent the state information. The transition system begins with an empty multiset of facts and this multiset evolves as rules are executed.

The following example demonstrates a Tamarin rule, named `Example_DH`. Each rule has a left-hand-side (LHS/premise) and a right-hand-side (RHS/conclusion) that specify the multisets before and after the rule execution. If all the LHS facts exist in the current state, the rule can be executed, resulting in the removal of the LHS facts from the current global state and the addition the RHS facts to the state.

Listing 2.2: Example Tamarin rule for DH key generation

```
1 rule Example_DH:
2     [Fr(a)] // premise (LHS)
3     --[Secret(a)]→ // action/event facts
4     [!Ltk($A, a), Out(g^a)] // conclusion (RHS)
```

In Tamarin, facts are typically represented as `F(t1,...,tn)`, where `F` represents

the name of the fact, and `t1` to `tn` are the terms that represent variables, messages, functions, and other symbolized elements. On the LHS (line 2), the fact `Fr()` is a built-in Tamarin fact, used for modeling fresh random values such as keys, and a is a freshly generated term, which will be used as a DH private key in this example. On the RHS, `Out()` is another built-in fact in Tamarin that models an entity sending out a message to the adversary-controlled network. Here $(g \hat{} a)$ models the public component of a DH key in the built-in DH theory of Tamarin. To use the bulitin DH model within Tamarin we define the DH base (generator) as a public term. The fact `!Ltk()` stores identity A with its fresh private key a. The facts within the LHS and RHS are state facts that, upon execution of the rule, are respectively removed and added to the global state. The '!' symbol indicates that a fact is persistent: it will not be removed from the system's state when it is consumed by rules and can be used as often as needed. There is another type of fact, such as `Secret()` in `Example_DH`, which is not present in the LHS and RHS and is categorized as an event/action fact. Action facts are not added to or consumed from the system's state. Instead, they serve to label transitions by recording the execution of rules, and they are added to the protocol execution trace. The different possible executions of the rules generate a set of potential execution traces made up from action facts. These traces serve as the basis for defining and verifying properties.

Defining Properties: In Tamarin, properties are formulated as lemmas, which are guarded first-order logic formulas defined over action facts and time points. For a security property to be verified, its lemma must hold across all possible execution traces, with no counterexamples found. Tamarin checks and either provides a proof when the property holds in all protocol behaviors, or returns a counterexample that demonstrates an attack. In some cases, the automatic proof construction process in Tamarin may fail to produce a proof or a counterexample with the allocated time and resources, and the verification process fails to terminate. In this case, the property is neither proved nor disproved. In such cases, the interactive mode of Tamarin can be used to identify the underlying problem and guide the tool by writing special intermediate lemmas and refining the protocol model to help the automatic proof generation process to reach a conclusive termination.

The example in Listing 2.3 presents a simple secrecy lemma, named `Example_Secrecy`. Here, `K()` (line 4) is a built-in action fact that represents the adversary's knowledge of a specific term, such as a 'key' in this example. The `#i` and `#j` are

temporal variables, representing time points. This lemma asserts that for all possible protocol behaviors and values of the variable 'key' (line 2), if the 'key' is captured in the execution trace by the action fact `secret()` at time point `i` (line 3), there exists no time point `#j` such that (denoted by '.') the adversary knows the 'key' (line 4). If Tamarin identifies a counterexample at which the value 'key' is known to the adversary, it stops the proof process and presents the attack in the output.

Besides lemmas employed to demonstrate the security and privacy properties of a modeled protocol, there are also executability lemmas whose objective is to ensure that the modelled protocol can properly execute to completion. Unlike the regular property lemmas, which must hold for all protocol traces, executability lemmas need only be valid for a single trace and are defined using the term `exists-trace` in Tamarin. This indicates that there is at least one trace in the set of protocol execution traces in which the defined protocol executes completely.

2.5 Summary

In this chapter, we primarily reviewd and analyzed the provided properties and weaknesses of schemes that have been proposed to augment the security or privacy of Stages 1 and 2 of the DNS resolution process. Chapter 3 will develop a threat model for the DNS resolution process and establish an evaluation framework for assessing its security, privacy, and availability properties. Using this framework, we summarize and systematically evaluate the reviewed schemes. Additionally, we provided an overview of PKI and delegation mechanisms, which will be used in Chapter 4 for the design of DNSSEC+. Finally, we provided background on formal analysis of security protocols, which will be applied in Chapter 6 to formally verify a set of security and privacy properties of DNSSEC+.

Listing 2.3: Example Tamarin lemma for proving secrecy

```
1       lemma Example_Secrecy:
2           "All key #i .
3           Secret(key) @i
4           ==> not (Ex #j .  K(key) @j)"
```

Chapter 3

Systematic Analysis of Secure DNS Schemes

As the literature review in Chapter 2 shows, Vanilla DNS was originally designed without security or privacy goals, rendering it susceptible to various attacks. To mitigate the vulnerabilities of Vanilla DNS, numerous schemes have been proposed to improve the security and privacy of the DNS resolution process. However, the absence of a comprehensive, DNS resolution-specific threat model makes it difficult to systematically compare existing threats within the DNS resolution process and to evaluate the effectiveness of proposed secure DNS schemes. To address this, in this chapter, we develop a comprehensive threat model of the DNS resolution process. We begin by defining the scope of the threats being analyzed and identifying attacks specifically associated with the DNS resolution process. Based on this, we provide a taxonomy of attack techniques that essentially enumerate threats to DNS resolution. Then, using this threat model as a foundation, we define security, privacy and availability properties to mitigate the identified attacks and vulnerabilities. Finally, we develop an objective evaluation framework based on the defined threats and use it to assess 11 previously proposed secure DNS schemes.

3.1 Threat Model and Attack Taxonomy

Initially, we define a DNS resolution threat model for use as a basis for defining security, privacy, and availability properties that per our model, suffice to secure the DNS resolution process. DNS resolution refers to the process by which a client generates a DNS query and, through a configured stub resolver, sends it to a recursive resolver. The recursive resolver interacts with various entities in the DNS hierarchy until it obtains the resource record associated with the queried record or gets an error response. The recursive resolver then returns the obtained response or error message to the client. Our focus herein is on the network-based attacks on the name resolution path, which involve exploiting network protocols and specifically interfering with or affecting the name resolution process. Host-based attacks are excluded from our scope (see Section 1.1.1).

Herein, attacks such as domain name squatting (*e.g.,* typosquatting [2]), domain

name hijacking [130], and parked domains [164] are excluded from our scope, as they are not network-based threats. Furthermore, attacks that involve registrars or search engines [130] are excluded from our scope, as these entities are not directly involved in the DNS resolution procedure. Botnet-related DNS threats are excluded as these attacks do not interfere with the name resolution procedure (*e.g.,* DNS tunneling or data exfiltration [125] or using DNS for inter-botnet communication [50]), or use legitimate functionality of DNS (*e.g.,* DNS-based fast-flux [113] or Domain Generation Algorithms (DGAs) [9, 134]). DNS reflection and amplification attacks [7], whose primary target is not the availability of DNS servers, are also beyond scope.

In Figure 3.1, DNS threats are classified based on their technical goals and the techniques employed to achieve these goals. The main technical goals are categorized into five groups, namely *domain name impersonation*, *censorship*, *degradation of service*, *information gathering*, and *evading detection*.

Monetization, while typically viewed as the direct goal of domain parking, can be considered a general objective for most DNS attacks and falls outside our scope. The *domain name impersonation* technical goal means fraudulently representing a service point (IP address). While web services are a primary target of *domain name impersonation*, this technical goal can also apply to non-web (non-HTTP) services. The service might be, *e.g.,* a web server, a server running File Transfer Protocol (FTP) [43], or an NTP server [95] (one technique to achieve this is *pharming*). *Censorship* refers to blocking specific DNS schemes or altering DNS responses of specific domains. We assume that nationwide adversaries such as governments (*e.g.,* China [77] or Iran [16]) are the main DNS-based censorship agents. *Degradation of service* is the objective of adversaries that disrupt the availability of the DNS infrastructure to legitimate clients. Adversaries with the *information-gathering* goal collect client and query-related information on the name resolution path, often violating the privacy of clients.

Subsections 3.1.1 to 3.1.5 discuss the five categories of technical goals used by adversaries to actively or passively exploit vulnerabilities in the DNS resolution process.

3.1.1 Domain Name Impersonation Techniques

This subsection describes the technical goal of *domain name impersonation*, which represents the first technical goal in Figure 3.1. One of the main methods for per-

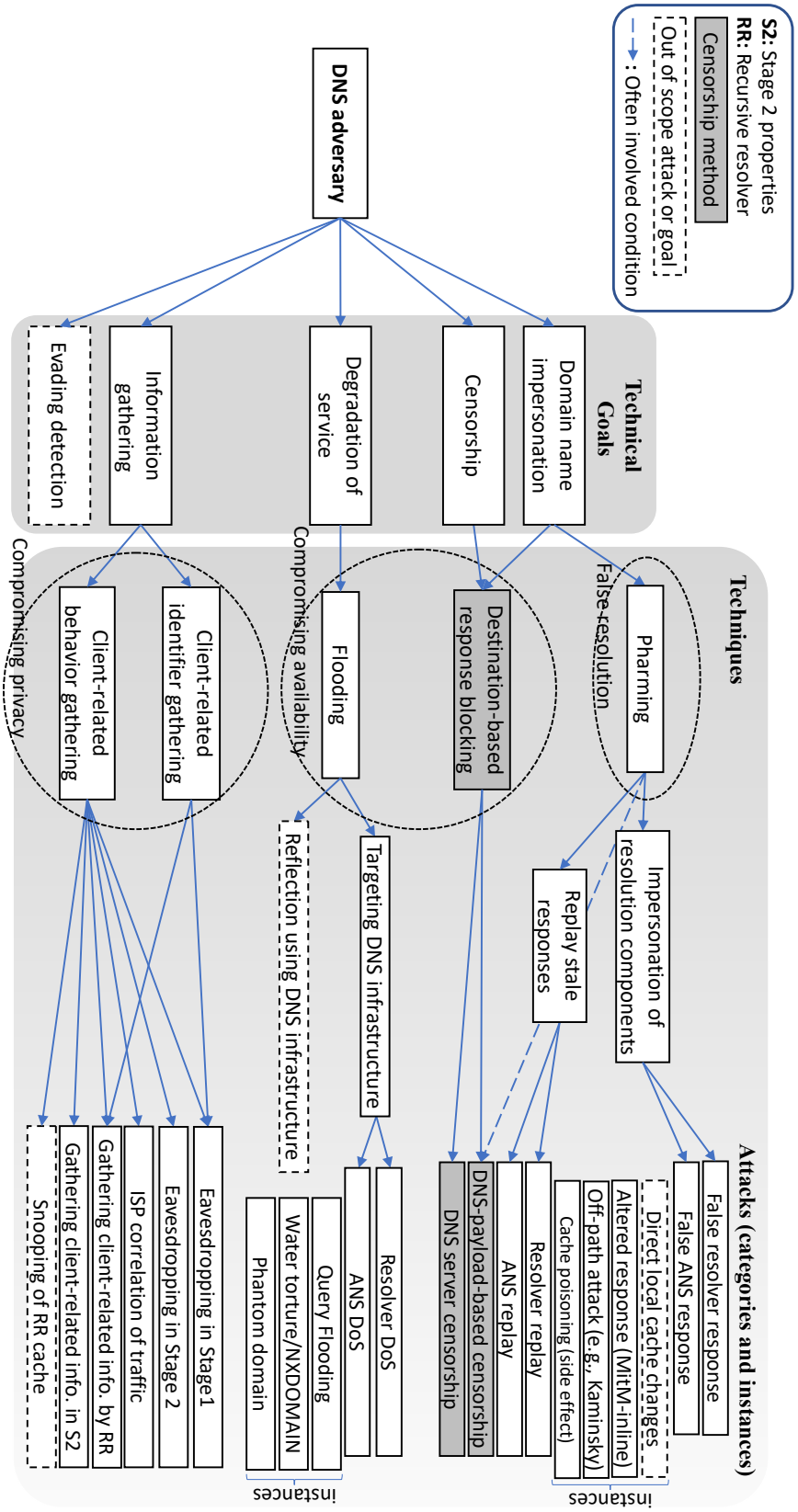


Figure 3.1: **Threat Model and Attack Taxonomy**: Network-based attacks in DNS resolution process.

forming *domain name impersonation* is through *pharming*. From the viewpoint of a client that issues a DNS query for resolution, we define *pharming* as any such method in which the response record (*e.g.,* IP address in '`A`' record) that the client receives is different from the one that the ANS has currently associated with the queried record. One technique of *pharming* is through changing the DNS record association (*e.g.,* domain-name/IP-address) in the cache of recursive resolvers or clients, known as DNS cache poisoning. Cache poisoning can be done by *direct local cache changes* (*e.g.,* by installing malware on a recursive resolver or client and manipulating their cached DNS records). However, *direct local cache changes* are host-based attacks, beyond our scope.

Inline adversaries in the name resolution path can transmit altered/false responses and impersonate authorized entities to carry out *cache poisoning* [107]. Off-path adversaries over the Internet can perform *cache poisoning* using techniques such as *Kaminsky* [98], *fragmentation* [74], network side-channels [114], or *birthday attack* [152].[1] *Cache poisoning* often involves *injection of false responses* by impersonating legitimate entities. In network-based *cache poisoning* attacks, adversaries impersonate legitimate recursive resolvers or ANSes to deceive the victim into believing that the responses received are from the legitimate entities that were queried.

Due to the lack of anti-replay means in DNS/DNSSEC, an adversary can capture and *replay false/stale responses* [15]. For example, DNSSEC responses are typically signed once and are valid for a specific period (see Section 2.2.9). In instances where an ANS has reassigned a domain name to a different IP address, replaying previously signed DNSSEC records can be used to carry out *pharming* and *cache poisoning* to misdirect clients.

3.1.2 Censorship Techniques

This subsection discusses the censorship-related technical goal, focusing on techniques that leverage DNS payloads for censorship, as well as methods used to censor access to DNS-based schemes. In Figure 3.1, techniques that can be used for DNS-based censorship are depicted as gray boxes. *Destination-based response blocking* refers to blocking DNS queries/responses that satisfy specified criteria by a censoring

[1]The vulnerability associated with birthday attacks has been mitigated in modern DNS software implementations by not sending multiple queries for the same domain name at the same time.

agent. For example, a query could include a specified domain name, a destination IP address associated with a recursive DNS server, or a destination port number associated with a particular DNS scheme. Direct IP-address-based censorship of Internet services other than DNS (based on packet headers) falls outside the scope of our model, as it is distinct from the DNS infrastructure and is not related to DNS resolution. In *DNS-payload-based censorship*, typically a Stage 1 intermediate adversary (*i.e.,* inline or on-path, *e.g.,* an ISP or recursive resolver) blocks DNS queries or responses by targeting specific payload fields. This may involve blocking or falsifying DNS resolution based on the payload's QName within a query or the IP address within a response [16]. As the legitimate association between a domain name and IP address changes, *DNS-payload-based censorship* often involves a type of *pharming*. While *DNS-payload-based censorship* provides adversaries with a fine-grained technique of censorship, *DNS server censorship* can block the entire traffic directed at the DNS servers of a specific DNS scheme based on their IP addresses, domain names [78], or their designated port number. For example, a censoring government can block DoT by blocking all traffic associated with TCP port 853 or disrupt DoTor by restricting access to Tor guard relays.

3.1.3 Degradation of Service Techniques

Degradation of service is another goal by which adversaries degrade or disrupt the availability of DNS resolution. We limit our primary scope to attacks where the targeted entities are DNS resolution components.

Flooding is a technique of degrading the availability of DNS. This can be broken down into attacks that *target DNS infrastructure* and attacks that perform *reflection using DNS infrastructure*. In *targeting DNS infrastructure*, adversaries flood recursive resolvers (*resolver-DoS*) or ANSes (*ANS-DoS*) using techniques such as *query flooding*, *water torture/NXDOMAIN* [111], or an attack known as *phantom domain* [145].

In *query flooding* adversaries send a large number of DNS queries to a UDP-based recursive resolver or ANS to exhaust their resources (primarily the processing power of a DNS servers in UDP-based DNS). In the variation of this technique known as the *water torture/NXDOMAIN* attack, adversaries generate random strings and append them as prefixes (*i.e.,* subdomains) to the targeted domain in the DNS queries generated for flooding attacks [111]. The targeted ANS looks up for these

non-existent domains, which exhausts the ANS's processing resources and fills the cache of the recursive resolver with non-existent responses.

The objective of *phantom domain* attack is to exhaust the resources of recursive resolvers. Here, an adversary establishes slow or unresponsive ANSes, known as phantom domains. Next, the adversary directs queries for these phantom domains to a victim recursive resolver, and these slow or unresponsive ANSes consume the resources of the recursive resolver.

In *Reflection using DNS infrastructure* attacks, due to the stateless nature of UDP-based DNS, DoS attackers can reflect their flooding traffic using DNS infrastructure to amplify their attack traffic and obscure the source of their attack. Furthermore, the inclusion of additional cryptographic records in DNSSEC increases its amplification factor, thereby increasing its susceptibility to reflection amplification attacks (see 2.2.9). Attacks that perform reflection using DNS infrastructure are excluded from our scope, as their primary target is not the availability of DNS itself.

3.1.4 Information Gathering Techniques

This subsection discusses the techniques used by adversaries and entities within the DNS resolution process to collect information about clients and their queried DNS records. As the entities involved in the name resolution process typically have access to the data and metadata of transmitted DNS messages (*e.g.,* query payload or metadata), they can gather various information about a client, thus violating the client's privacy. *Information Gathering* is the third goal by which adversaries (typically intermediate entities in the name resolution process) gather information on client-related behavior and identity, using DNS queries or responses and their associated metadata.

Collecting a client's browsing or DNS query history, and combining these queries with additional information, has been shown to allow identification or re-identification of the client across time periods [26, 127]. Domain names within DNS queries may disclose various information with different privacy sensitivity levels. For example, DNS queries can leak personal information about a client (*e.g.,* the applications that a client uses or the type of IoT devices that a client owns [10]). In some cases, DNS queries may allow information to be associated with a real-world entity, at which point the information itself becomes Personally Identifiable Information (PII). For

example, if a client issues a DNS query with the QName '*admin.example.com*,' there is a high probability that the client is the administrator of '*example.com*.' Thus, this domain name can be associated with a real-world identity and is considered PII.

In the DNS context, some part of the payload can directly disclose client-related behavior and some part of the metadata (transmission-related information) can be used for client identification. For example, the Questions, Answers, Authority, and Additional sections of a DNS message payload can directly reveal client-related behavioral information. On the other hand, timestamps, DNS header, and other network layer headers can be collected to identify a client or form client-related behavioral information over time. In Figure 3.1, *client-related behavior gathering* refers to collecting a client's actions through DNS query information that can directly reveal the client's behavior, and *client-related identifier gathering* refers to collecting metadata or payload that could be considered as direct (*e.g.,* IP addresses) or indirect (*e.g.,* timestamps) identifiers of clients. Indirect identifiers can be used in association with other information to increase the accuracy of identification or lead to the formation of PII [127].

Eavesdropping can occur in both stages. In Stage 1, an adversary can collect both client-related behavioral patterns and identifying information (*e.g.,* IP address) from a DNS query by *eavesdropping*. In Stage 2, adversaries can collect client-related behavioral pattern by *eavesdropping*, and the metadata (*i.e.,* identifying information) belong to a recursive resolver. Although metadata in Stage 2 does not belong to clients, the DNS query payload may contain client-related identifier information such as ECS (Section 2.1.2). In Figure 3.1, we assume that there is no ECS included in Stage 2; thus, *eavesdropping in Stage 2* only reveals client-related behavior.

In Stage 1, the ISP of a client serves as the convergence point for the client's interaction with the Internet. As such, it is capable of correlating an encrypted DNS query with the subsequent traffic originating from the client, thus enabling the inference of the encrypted queried DNS record. For example, if a client sends an encrypted DNS query and the subsequent traffic is HTTPS, the ISP can do a reverse lookup or check the plaintext Server Name Indication (SNI) field in the TLS `client hello` message to discover the QName of the earlier encrypted DNS query. As all Internet Engineering Task Force (IETF) standardized protocols are recommended to mitigate pervasive monitoring attacks on the Internet [61], Encrypted SNI (ESNI)

was initially proposed as a mechanism to prevent privacy leaks resulting from the exposure of the SNI during TLS handshakes. Subsequently, Encrypted Client Hello (ECH) was introduced to provide enhanced protection by mitigating information leakage from the entire Client Hello message within the TLS handshakes [141]. In Figure 3.1, this attack against the confidentiality of encrypted DNS messages is called *ISP correlation of traffic*, which reveals client-related behavior.

Aside from ISPs, recursive resolvers can gather various information from the DNS queries they receive for different purposes.[2] In *gathering client-related info. by (RR)*, a recursive resolver gathers both a client's behavior-related (*e.g.,* QName) and identifying information (*e.g.,* IP address). Recursive resolvers can also geolocate clients based on their IP addresses. In *gathering client-related information (S2)*, only the DNS message payload is privacy-sensitive to the clients, which might also contain ECS, and the query metadata (*i.e.,* identifying information) belongs to the recursive resolver. Therefore, different entities involved in Stage 2 (*i.e.,* intermediate ASes and ANSes) can gather behavior-related information about clients that use a recursive resolver but cannot often associate the gathered information with the clients (unless ECS is included in the queries).

Lastly, DNS *cache snooping* is an active technique by which attackers collect information about resolved domain names by a recursive resolver by checking cached domain names on the recursive resolver. *Cache Snooping* is typically performed by sending DNS queries with the Recursion Desired (RD) flag unset (RD=0), analyzing the remaining TTL in the responses from a recursive resolver, or measuring the response time of the DNS cache [68]. However, *cache Snooping* of the multi-layered centralized resolvers with distributed caches is more sophisticated than single-cache resolvers [139]. As in cache snooping, the adversaries either employ legitimate DNS functionality (*e.g.,* non-recursive requests) or side channels (*e.g.,* time analysis); cache snooping is beyond the scope of this model.

3.1.5 Evading Detection Techniques

In this subsection, we provide a brief discussion of attacks that abuse the legitimate functionality of DNS as a benign protocol to hide malicious activities. As almost all Internet-connected networks require name resolution, DNS is typically not blocked

[2] *e.g.,* logged information from Google's public DNS servers: `https://developers.google.com/speed/public-dns/privacy`

by firewalls. In *evading detection* based on DNS, adversaries try to leverage DNS as a benign protocol to hide their malicious traffic or infrastructure. For example, botnets use DGAs and DNS-based fast-flux [113] to hide their Command and Control (C&C) server [4, 134]. Furthermore, an attacker can exfiltrate arbitrary data from a compromised target through DNS traffic (also known as DNS tunneling) [125, 154, 173]. In DNS-based *evading detection*, adversaries do not directly interfere with or disrupt the name resolution process. Instead, they exploit the legitimate functionality of DNS for malicious goals. Therefore, the technical goal of *evading detection* is beyond the scope of our taxonomy.

3.2 Resolving The Threats of DNS Resolution

In the threat model and attack taxonomy (Section 3.1), we systematically analyzed and categorized existing threats within the DNS resolution process, providing a comprehensive overview of the threats and attack techniques. Building on this, we now define 14 security, privacy, and availability properties for the DNS resolution process. These properties are designed to address and mitigate the previously identified threats.

3.2.1 Security Properties

The security properties that we define here are specific to the DNS resolution process, and are designed to mitigate attacks that compromise the authenticity and integrity of DNS messages or expose long-term secrets (*e.g.,* private keys) to adversaries.

S1 ***Resilient-to-False-Resolver-Response*** **(Stage 1)**: This property is achieved if a DNS scheme prevents adversaries from injecting false responses to clients by impersonating legitimate recursive resolvers.

In Stage 1, if an adversary has access to the issued DNS queries by a client or can trigger a client to issue DNS queries (*e.g.,* through embedded images on a webpage), the adversary can inject false responses to the client by impersonating the legitimate recursive resolver in the queries. Impersonating a recursive resolver means generating a fabricated response using the parameters in the legitimate client's query (*e.g.,* the resolver's IP address, TXID, and other sections) to make it appear as if the response is generated by the legitimate recursive resolver.

An adversary in Stage 1 can be an inline or intermediate entity (*e.g.,* a router or an ISP) that became rogue or an outsider that redirects traffic through network traffic hijacking (*e.g.,* Address Resolution Protocol (ARP) spoofing or Border Gateway Protocol (BGP) hijacking). An inline adversary has access to the DNS queries issued by a client and the query parameters (*e.g.,* random TXID and source port). However, an off-path adversary first needs to trigger a client to issue DNS queries, so that the adversary knows some of the query sections. Subsequently, the adversary must accurately guess the correct values within various sections of the issued queries or somehow match them to successfully inject false responses on behalf of the legitimate resolver.

In addition, the inline adversary can block the client's query or the legitimate response from the recursive resolver. However, the off-path adversary cannot directly block a legitimate query or response without mounting a DoS attack on the destination of the query or traffic hijacking. Thus, the off-path adversary must inject the false response before the legitimate response arrives at the client's machine.

Message authentication in Stage 1 is one way to mitigate active inline or off-path *false response injection* attacks. Message authentication also guarantees the integrity of DNS messages, mitigating benign bit errors [118]. Due to the lack of message authentication or other means to mitigate false response injection between clients and recursive resolvers in Vanilla DNS, *false response injections* and impersonations of recursive resolvers go undetected by clients. Consequently, if a client caches the response, *false response injection* can result in client-side *cache poisoning* (*e.g.,* stub resolver or the DNS cache of the client's web browser).

S2 ***Resilient-to-Resolver-Replay-Attack*** **(Stage 1)**: A DNS scheme provides this property if recursive resolver responses to a client cannot be successfully replayed to other clients, and previous responses sent to a client cannot be replayed later to the same client.

As an example of S2, if a DNS scheme employs authenticated encryption with different session keys per client-resolver interaction in Stage 1, this stops successful replay of the transferred messages to different clients. Thus, having separate protocol runs with fresh keys for each client prevents replay of DNS responses from one client to others. Also, not reusing session keys prevents the replay

of messages from previous sessions in subsequent sessions of the same client. Moreover, adding Time-Variant Parameters (TVPs) [161] to the exchanged DNS messages adds uniqueness to messages and prevents replay of messages in the same session. Ensuring that interactions adhere to a one-to-one relationship guarantees the uniqueness of messages, and effectively prevents replay attacks. In the absence of anti-replay means in a DNS scheme, replay of stale responses can be used to perform *pharming* or to degrade specific services that rely upon DNS (*e.g.,* CDN services [71]).

S3 ***Resilient-to-False-ANS-Response*** **(Stage 2)**: In Stage 2, if a DNS scheme prevents adversaries from successfully injecting false responses to recursive resolvers by impersonating legitimate ANSes, the scheme satisfies S3.

In general, an adversary may be able to impersonate legitimate ANSes used in the queries of a victim recursive resolver and inject false responses to the recursive resolver. As open resolvers are exposed to the Internet and respond to recursive queries that they receive from any IP address, inline and off-path adversaries can trigger them to issue DNS queries. Adversaries responsible for triggering these queries then inject false responses, enabling cache poisoning (*e.g.,* using the Kaminsky [98, 114] technique) and thereby resulting in *pharming*.

For a DNS scheme, one way to satisfy *Resilient-to-False-ANS-Response* is by providing message authentication in Stage 2 for the responses received by recursive resolvers from ANSes. For example, in Vanilla DNS, due to the lack of message authentication and other means to mitigate false message injection in Stage 2, recursive resolvers would not detect false message injections by impersonation of ANSes.

False message injection in Stage 2 affects clients that use the targeted recursive resolver, while *false message injection* in Stage 1 often affects a single client. Therefore, attacks that result in *false ANS response injection* (*i.e.,* cache poisoning of a recursive resolver) typically affect more clients than similar attacks in Stage 1 (*i.e.,* poisoning cache of a single client).

S4 ***Resilient-to-ANS-Replay-Attack*** **(Stage 2)**: A DNS scheme provides this property if ANS responses sent to a recursive resolver cannot be replayed to other recursive resolvers. Furthermore, old responses sent to a recursive resolver cannot be replayed later to the same recursive resolver.

In the absence of anti-replay means in Stage 2, an adversary can capture and replay old responses, which may be no longer correct. For example, in DNSSEC, although responses are signed, they can be replayed by entities other than ANSes before their signatures expire [172]. The stale records in DNSSEC (see Section 2.2.9) can be used to inject false responses to the cache of resolvers [15] and thereby misdirect clients to now-incorrect IP addresses. A solution to ensure anti-replayability in DNS schemes is for ANSes to use different keys for message authentication when interacting with different recursive resolvers in Stage 2. This approach mitigates replay attacks by ensuring that each interaction is uniquely authenticated. Moreover, incorporating TVPs into the exchanged messages helps prevent the replay of previously authenticated messages to the same client, in cases where a key is reused for authenticating multiple messages with the same resolver [25].

S5 ***Avoid-Duplicating-Longterm-Secret*** **(Stage 2)**: A DNS scheme that provides means for DNS zones to avoid duplication of long-term private keys on all server instances and minimize the exposure of such keys to attacks targeting these servers satisfies this property.

ANSes use long-term private keys in cryptographic operations (*e.g.,* encryption or message authentication) to provide security properties in Stage 2. The method by which ANSes employ long-term private keys can expose the keys to attacks targeted at ANS server instances. For example, if IP anycast is used to improve the performance of content delivery and the long-term private key is present on all CDN servers, the private key is exposed to attacks targeting these servers. In DNSSEC (with offline-signing), where signatures are pre-computed, long-term private key can be stored more securely, and there is no need for their presence on all ANS servers; thus, DNSSEC satisfies S5.

3.2.2 Availability Properties

We now define two properties designed to mitigate attacks that disrupt the DNS resolution process by blocking DNS messages or blocking access to DNS servers. Additionally, we define two properties to address attacks targeting the DNS infrastructure, which could degrade or disrupt the name resolution process.

A1 ***Resilient-to-DNS-payload-based-Censorship*** **(Stage 1)**: A DNS scheme must prevent a Stage 1 adversary from gaining access to plaintext DNS query and response feilds (message sections) to provide this property. Consequently, an attacker cannot intercept DNS messages based on specified DNS payload sections to block the name resolution process.

In Stage 1, countries with a history or policy of censorship, such as China [77] or Iran [16], have been observed to leverage DNS to block access to specific domain names. Various sections in DNS queries and responses can be used by censoring agents; QName inside the Question section is a particularly useful field. In this work, if a DNS scheme prevents censoring agents from accessing all DNS Question, Answer, Authority, and Additional in a DNS payload, it satisfies this property.[3] As the DNS message header section contains non-privacy-sensitive information (*e.g.,* control flags) about a DNS message, if in a DNS scheme this section is accessible to censoring entities, but the other sections are concealed, the scheme receives full credit for this property.

Inline censoring entities in Stage 1 (*e.g.,* ISP) can target specific DNS queries/responses based on one or more DNS payload sections. The targeted DNS queries or responses may then be blocked or responded with an NXDOMAIN answer [151], a routable public IP address (*e.g.,* Facebook address [77]), or a false IP address (*e.g.,* a placeholder "black hole" address [16]).

Resilient-to-DNS-payload-based-Censorship is typically achieved by encrypting queries and responses in Stage 1. Strongly obfuscating DNS messages is another technique to achieve this property. Our rationale for defining this property only in Stage 1 is that an informed user or privacy-aware Internet software (e.g., web browsers) can choose recursive resolvers outside a censoring region. Thus, if a DNS scheme provides *Resilient-to-DNS-payload-based-Censorship* in Stage 1, the recursive resolver could be selected in a non-censoring region and the *Resilient-to-DNS-payload-based-Censorship* is not required in Stage 2 to mitigate censorship. For example, a client residing in a censoring region (*e.g.,* Iran [16]), who uses Cloudflare's recursive resolver, might know that it is not controlled by a censoring entity and is located in a non-censoring region; thus, if queries from the client

[3]Some public recursive resolvers remove Authority or Additional sections from DNS responses. We assume that these sections are not stripped from DNS responses since this behavior is not consistent across all resolvers.

to this resolver (Stage 1) are *Resilient-to-DNS-payload-based-Censorship*, this property is not required in Stage 2.

A2 ***Resilient-to-DNS-Server-Censorship*** **(Stage 1)**: A DNS scheme provides this property if clients' interactions with recursive resolvers do not have distinguishable characteristics that can be used to block access to the recursive resolvers.

As a more coarse-grained technique than A1, in DNS server censorship, a censoring entity completely blocks access to the recursive resolvers of a DNS scheme. Inline middleboxes under the control of a censoring government can be used to block DNS requests/responses that satisfy specific characteristics. For instance, access to the servers of a DNS scheme might be blocked based on their IP addresses, domain names, or port number. If a DNS scheme is limited to a set of discoverable IP addresses, domain names, or uses a distinguishable port number, these characteristics can be used to censor this scheme by blocking access to its resolvers.

Additionally, if a DNS scheme uses a known port number that indicates the use of that specific DNS scheme (*e.g.,* TCP/853 for DoT), the IP addresses of its servers can be effectively discovered over the Internet (*e.g.,* using ZMap [56]). Thus, a DNS scheme that uses a distinguishable port number can be censored directly by its port number or by discovering the IP address of its servers and blocking access to those IP addresses.[4]

Also, if a DNS scheme is built upon another protocol that can get blocked by a censoring government, the underlying protocol can get blocked to censor access to the DNS servers of this scheme. For example, a censoring government can block access to Tor relays or directory authorities to censor access to DNS-over-Tor resolvers.

One method to achieve *Resilient-to-DNS-Server-Censorship* is that a proposed DNS scheme does not rely upon another censorable protocol, does not have distinguishable characteristics (*e.g.,* specific port number), and is not limited to a set of discoverable IP addresses or domain names that can be used to censor the scheme.

[4]If the IP addresses of the DNS servers are frequently changed, persistent scanning and censoring can still effectively block access to these servers.

DNS schemes that disguise DNS traffic using ports of other protocols tend to be resilient against port-based censorship. In that case, censoring traffic requires fingerprinting and traffic analysis, which is more costly. For instance, DoH traffic uses port 443—its traffic gets merged with HTTPS; hence, censoring DoH is more costly (albeit not impossible). Schemes whose traffic can be effectively identified and isolated with additional analysis (*e.g.,* Machine-Learning-based (ML-based) classification or statistical analysis) receive half credit for A2.

A3 ***Resilient-to-Resolver-DoS*** **(Stage 1)**: A DNS scheme provides this property if the used transport layer protocol in Stage 1 provides resistance mechanisms against DoS attacks or the scheme employs DoS mitigation techniques at the application layer to protect recursive resolvers.

In order to ensure the availability of the DNS resolution process, recursive resolvers must be accessible to legitimate clients and resist DoS attacks. Although perfect availability over the Internet is idealistic, some DNS schemes are more resilient to DoS attacks than others. The primary type of attack against availability is DDoS, which overwhelms the resources of the target from multiple locations. Bandwidth depletion is a general type of DoS that can target any Internet-connected service, including recursive resolvers. However, DNS-related DoS attacks often target DNS at the application layer (*e.g.,* flooding in UDP-based DNS) or target the protocols upon which a DNS scheme relies (*e.g.,* TCP SYN flooding in TCP-based schemes).

As a UDP-based scheme, Vanilla DNS does not verify the source address of queries at the transport layer, and queries are answered at the application layer. As the number of queries in a UDP-based DNS scheme increases in the face of a DDoS attack, server CPU usage, response latency, and the number of unanswered queries also increase [176]. Also, if, *e.g.,* source address spoofing was used in the UDP-based DNS packets, query rate-limiting based on IP address is rendered futile [176]. On the other hand, DDoS attacks on TCP-based DNS overwhelm the pre-allocated resources to TCP connections through SYN flooding, preventing legitimate users from accessing name resolution services [161]. However, before completing the TCP handshake, DNS queries are not served at the application layer, and after the handshake, IP-based rate-limiting is a viable defense. Additionally, in TCP-based DNS, SYN cookies and SYN cache [105]

can be employed to mitigate SYN flooding attacks [176].

In summary, if a DNS scheme works over TCP, as it has means to mitigate DDoS attacks (*e.g.,* SYN cache and SYN cookies [105]) and the query does not get served at the application layer before the completion of the TCP handshake, we consider it resilient against DoS. However, due to the lack of means to detect spoofed messages in UDP, spoofed queries are responded in the application layer, and UDP-based DNS schemes are often susceptible to DoS attacks. In order to fortify UDP-based schemes against DoS attacks, anti-DoS mechanisms can be implemented at the application layer (*e.g.,* in QUIC).

A4 ***Resilient-to-ANS-DoS*** **(Stage 2)**: Similar to A3, a DNS scheme that provides ANSes with DoS resistance mechanisms at the transport layer or employs DoS mitigation techniques at the application layer satisfies this property.

In addition to recursive resolvers, ANSes are necessary components for a successful DNS resolution process. Similarly, ANSes are susceptible to DoS attacks. We apply the same rule as A3 for rating the resilience of ANSes to DoS. In Stage 2, TCP-based schemes and UDP-based protocols that employ DoS mitigation techniques at the application layer to protect ANSes provide *Resilient-to-ANS-DoS*.

3.2.3 Privacy and Anonymity Properties

In this part, we define five privacy-related properties for the DNS resolution process, aimed at protecting against potential attacks that expose name resolution information to unauthorized entities.

P1 ***Resilient-to-Eavesdropping-in-s1*** **(Stage 1)**: A scheme provides this property if the plaintext content of (application layer) DNS messages sent in Stage 1 is accessible only to authorized entities (*i.e.,* clients and recursive resolvers).

Due to the lack of confidentiality protection in Vanilla DNS, DNS queries and responses in Stage 1 are susceptible to eavesdropping. Despite being a security property, we categorize confidentiality of DNS queries and responses as a privacy property. Our rationale is that while DNS information, which maps domain names to IP addresses, is publicly accessible, the specific destinations a client intends to connect to may be considered private. Therefore, transmitting DNS query and response data in plaintext can harm clients' privacy. Encrypting

the DNS messages transmitted from stub to recursive is one approach to achieve *Resilient-to-Eavesdropping-in-s1*. This way, inline entities, such as a client's local network, ISP or other Stage 1 ASes, cannot have access to the content of DNS messages. If the used encryption algorithm cannot resist traffic analysis attacks that reveal domains visited by users, the scheme does not receive complete credit for this property. In the evaluation of this property, it is assumed that the traffic of a DNS scheme can be isolated, and given this, its resistance to eavesdropping (of plaintext message) is assessed. Therefore, this property is evaluated independently from the rating of P2.

P2 ***Conceal-DNS-Msg-Nature-in-s1*** **(Stage 1)**: To provide this property, detecting and isolating a DNS scheme's traffic by its explicit headers or behavioral characteristics must be prevented in Stage 1.

If DNS traffic is distinguishable in Stage 1, intermediate entities can block, redirect or censor DNS traffic, and in case of backward compatibility, force the communication to fall back [87] to less secure schemes (*e.g.,* Vanilla DNS). Furthermore, if the traffic of an encrypted DNS scheme is detectable, it can be filtered for traffic analysis (*e.g.,* in DoT/DoH [150]) to identify the domain names visited by users with high accuracy.

In Stage 1, if the traffic of a DNS scheme is not easily distinguishable by explicit headers in the network or transport layer (*e.g.,* destination port number), this scheme partially provides *Conceal-DNS-Msg-Nature-in-s1*. For instance, DNS schemes such as DoT use a specific destination port number (853), which renders their traffic easily distinguishable; these do not satisfy this property. Beyond explicit headers, the traffic of a DNS scheme may not be distinguishable by behavioral characteristics, such as time or size analysis, for the scheme to receive full credit for this property.

To resist behavioral analysis, a DNS scheme may use techniques such as padding before encryption or traffic-flow security [158] (*e.g.,* sending cover traffic) to render its traffic less distinguishable. For example, DoH merges its traffic with web traffic using port 443, transferring DNS queries as HTTPS traffic. However, although DoH traffic is not distinguishable by its explicit headers, it is still susceptible to behavioral analysis (*e.g.,* packet size [129], connection duration [163]) to detect and isolate DoH traffic. Thus, DoH partially conceals DNS message

nature in Stage 1.

P3 ***Hides-Client-IPaddr-From-Resolver*** **(Stage 1)**: A DNS scheme satisfies this property if it provides any means that successfully conceals a client's IP address from the queried recursive resolvers in Stage 1.

Although DNS message confidentiality in Stage 1 (see P1) prevents unauthorized access by inline entities to DNS queries, recursive resolvers usually have access to both client-related behavior (*e.g.,* QName) and identifying information (*e.g.,* IP address). In Stage 1, recursive resolvers can collect query-related information and then use it to examine the behavior of DNS clients. For example, Bird *et al.* [26] showed that clients can be accurately and uniquely identified in different periods based on their web browsing histories. The same identification and re-identification could be achieved by collecting DNS-related data from clients. Moreover, in the context of IoT devices, the DNS query payload and metadata can disclose information about the types of IoT devices owned by a client to entities involved in Stage 1 of the resolution process (*e.g.,* recursive resolver) [10].

Some DNS schemes use techniques such as proxies, relay servers, or the Tor network to enhance the privacy of clients against resolvers, separating a client's IP address from its DNS queries. Clearly, if a client (stub resolver) includes its IP address or subnet as ECS in DNS queries, the anonymizing DNS scheme adds overhead but still fails to hide the client's IP address. Thus, schemes that hide clients' IP addresses from recursive resolvers must explicitly remove or truncate ECS from their queries to gain full credit for this property.

P4 ***Resilient-to-Eavesdropping-in-s2*** **(Stage 2)**: Similar to P1, this property is met if the plaintext content of (application layer) DNS messages in Stage 2 is only accessible to authorized entities (*i.e.,* recursive resolvers and ANSes).

If the traffic between recursive resolvers and ANSes (*i.e.,* Stage 2) remains confidential, we consider a DNS scheme *Resilient-to-Eavesdropping-in-s2*; this implies that inline entities cannot access DNS message contents.

Encryption is typically used to preserve the confidentiality of DNS messages in Stage 2. An inline adversary in Stage 2 may be able to infer the QName of a recursive resolver's encrypted query through inter-domain dependencies, the set of name servers queried by the recursive resolver [137, 148], and other side

channels. However, confidentiality in Stage 2 still prevents access of inline entities to client-related information for queries that include ECS. If the used encryption algorithm is vulnerable to traffic analysis attacks that can reveal visited domains, the scheme is not fully credited for this property. We rule out use of end-to-end encryption between clients and ANSes as that noticeably increases the query overhead of ANSes [148], and bypasses the benefits that come with using caching recursive resolvers.

P5 ***Hides-Client-IPaddr-in-s2*** **(Stage 2)**: A DNS scheme that removes, includes only a subset of, or conceals ECS in DNS queries to preserve clients' anonymity in Stage 2 satisfies *Hides-Client-IPaddr-in-s2*.

Two types of entities can compromise privacy of clients in Stage 2: ANSes queried by a recursive resolver and intermediate entities between a recursive resolver and ANSes (when they can observe the plaintext of queries). If a client includes ECS in DNS queries, recursive resolvers that support ECS will also include ECS in their iterative queries as they traverse the DNS hierarchy. Thus, the intermediate entities in Stage 2 and the ANSes (from root to the name server authoritative for the zone of the queried record) will have access to the ECS added by the client.

If the exact IP address of a client is included (*i.e.,* /32 in IPv4 or 128 bits in IPv6), it can be used for user tracking, gathering information, selective cache poisoning [101], or ECS-based censoring. To hide a part of ECS, if we assume the default truncation of 24 bits for IPv4 and 56 bits for IPv6 based on the ECS RFC's default recommendation, these truncated addresses are likely to reveal client-related information (*e.g.,* country, city, or organization). Thus, as a scheme that truncates ECS might still disclose client-related information of use to attackers (*e.g.,* for selective cache poisoning [101] at country or organization level); such a scheme receives only partial credit for this property.

If a DNS scheme eliminates ECS in DNS queries to prevent Stage 2 intermediate entities and ANSes from accessing clients' IP addresses or subnet, it receives full credit. Completely removing ECS improves the privacy and security of clients, and may reduce page load time by excluding the overhead caused by ECS record cache misses [82].

DNS schemes that encrypt queries in Stage 2, precluding access of Stage 2 intermediate entities to ECS, receive partial credit for this property. In such schemes,

the access of intermediate entities to ECS is precluded, but ANSes still have access to the included ECS. If clients do not include ECS in DNS queries, no client-related identifier is disclosed in Stage 2.[5] In our evaluation, we assume that ECS is included in DNS queries. A DNS scheme must explicitly remove, encrypt, or truncate ECS at stub or recursive resolvers to satisfy this property.

3.3 Evaluation of Secure DNS Schemes

The DNS resolution threat model and attack taxonomy outlined in Section 3.1, was used as the foundation for Section 3.2, where 14 security, privacy, and availability properties were defined to mitigate the identified threats effectively. Now, using the properties defined in Section 3.2, we present an evaluation framework for assessing the security, privacy, and availability properties of secure DNS schemes. This framework provides an objective approach for analyzing existing secure DNS schemes, identifying their security, privacy, and availability weaknesses. Moreover, this framework provides valuable insights into existing threats and security properties that protect the DNS resolution process. It is our hope that this contributes to the advancement and informed development of future secure DNS schemes.

In Table 3.1, we present a comparative evaluation of 11 previously proposed DNS schemes whose theories of operation were discussed in our literature review (Chapter 2). Subsequently, we provide insights from our DNS evaluation framework regarding association between properties, high-level comparison of schemes, and relevant discussions on the existing security, privacy, and availability challenges of the name resolution process.

3.3.1 Relationships Between Properties

Table 3.1 presents a comparative evaluation of the DNS schemes reviewed in Chapter 2, which were proposed to augment the security of DNS resolution in Stages 1 and 2. In Table 3.1, a full dot at row i and column j indicates that the scheme in column j fully satisfies the property in row i. A half dot means that the scheme partially satisfies the corresponding property, while an empty dot denotes that the property is not satisfied by the scheme.

[5] Assuming that the queried record does not contain a client-identifier information.

Before discussing the bigger picture in Table 3.1, we explore the relationships and clarify distinctions between properties. Although similarities can suggest redundancy between properties, they can also be an artifact of the specific schemes rated. When possible, where a similarity exists, we explain a hypothetical DNS scheme that would achieve one property but not the other.

A2 and P2: Based on the definitions, there is a relationship between *Resilient-to-DNS-Server-Censorship* and *Conceal-DNS-Msg-Nature-in-s1*. The absence of *Conceal-DNS-Msg-Nature-in-s1* is sufficient to prevent the realization of *Resilient-to-DNS-Server-Censorship*. However, the presence of *Conceal-DNS-Msg-Nature-in-s1* does not guarantee *Resilient-to-DNS-Server-Censorship*. In other words, if a DNS scheme's traffic is distinguishable, then the distinguishable characteristics can be used to isolate communications with DNS resolvers and block access to these resolvers. On the other hand, if the traffic of a DNS scheme is not distinguishable, if the protocol upon which it relies is susceptible to censorship, access to its servers can be blocked by blocking the underlying protocol. For example, DoTor conceals DNS messages by converting them into fixed-sized Tor cells, making DNS traffic indistinguishable from other Tor traffic. However, censoring agents can limit access to the entire Tor network, censoring DoTor alongside other Tor traffic. Thus, DoTor satisfies *Conceal-DNS-Msg-Nature-in-s1* but fails to satisfy *Resilient-to-DNS-Server-Censorship*.

S1 and S2: The evaluations of *Resilient-to-False-Resolver-Response* and *Resilient-to-Resolver-Replay-Attack* may at first reading appear to be equivalent. *Resilient-to-False-Resolver-Response* is provided through implementation of countermeasures against malicious false message injection, such as including MACs or digital signatures in a scheme. *Resilient-to-Resolver-Replay-Attack* can be provided by integrating anti-replay mechanisms with a scheme; *e.g.,* by using distinct session keys per session to provide session freshness and per-record nonces to ensure freshness of messages in a session. Therefore, it is possible to have a DNS scheme that satisfies *Resilient-to-False-Resolver-Response* (*e.g.,* utilizing MACs) but not *Resilient-to-Resolver-Replay-Attack* due to the absence of anti-replay mechanisms. In other words, anti-replayability requires a more robust form of authentication, wherein the interactions between entities have a one-to-one relationship, which inherently implies the simpler form of message authentication without anti-replay mechanisms [109].

A1, P1, and P2: If a secure DNS scheme prevents unauthorized access to its

Table 3.1: Comparative evaluation framework for assessing properties of secure DNS schemes (s1: Stage 1, s2: Stage 2)

			Stage 1								**Stage 2**					
DNS Scheme	*Reference*	*Proposed Working Stage*	*S1. Resilient-to-False-Resolver-Response(s1)*	*S2. Resilient-to-Resolver-Replay-Attack(s1)*	*A1. Resilient-to-DNS-payload-based-Censorship*	*A2. Resilient-to-DNS-Server-Censorship*	*A3. Resilient-to-Resolver-DoS*	*P1. Resilient-to-Eavesdropping-in-s1*	*P2. Conceal-DNS-Msg-Nature-in-s1*	*P3. Hides-Client-IPaddr-From-Resolver*	*S3. Resilient-to-False-ANS-Response(s2)*	*S4. Resilient-to-ANS-Replay-Attack(s2)*	*S5. Avoid-Duplicating-Longterm-Secret*	*A4. Resilient-to-ANS-DoS*	*P4. Resilient-to-Eavesdropping-in-s2*	*P5. Hides-Client-IPaddr-in-s2*
Vanilla DNS	[119, 120]	1&2	○	○	○	○	○	○	○	○	○	○		○	○	○
DNSSEC	[11, 12, 13]		○	○	○	○	○	○	○	○	●	○	●	○	○	○
Live DNSSEC	[53]	2	○	○	○	○	○	○	○	○	●	○	○	○	○	○
DNSCurve	[25, 46]		○	○	○	○	○	○	○	○	●	●	○	○	●	◐
DNSCrypt*V3	[48]		●	●	◐	○	○	◐	○	○	○	○	○	○	○	○
Strict DNS-over-TLS	[86]		●	●	◐	○	●	◐	○	○	○	○	○	○	○	○
DNS-over-HTTPS	[80]		●	●	◐	◐	●	◐	◐	○	○	○	○	○	○	○
Strict DNS-over-DTLS	[140]	1	●	●	◐	○	●	◐	○	○	○	○	○	○	○	○
Strict DNS-over-QUIC(S1)	[89]		●	●	◐	○	●	◐	○	○	○	○	○	○	○	○
DNS-over-Tor	[144]		●	●	●	○	●	●	●	●	○	○	○	○	○	●
Oblivious DNS	[147]		○	○	◐	○	○	◐	○	●	○	○	○	○	○	●

messages (*i.e.,* fulfills *Resilient-to-Eavesdropping-in-s1* (P1)) or renders its traffic indistinguishable (*i.e.,* satisfies *Conceal-DNS-Msg-Nature-in-s1* (P2)), the lack of access to DNS payload prevents censoring entities from DNS payload-based censorship. Therefore, to the extent that a given scheme satisfies either P1 or P2, it fulfills A1 (*Conceal-DNS-Msg-Nature-in-s1*), as P1 or P2 render plaintext of DNS message inaccessible to adversaries.

P5 for Stage 1 schemes: Property *Hides-Client-IPaddr-in-s2* (P5) implies preventing exposure of client IP addresses or subnets in Stage 2. Client IP addresses in Stage 2 are typically included in DNS payload in form of ECS. Removing ECS from the client's query at stub resolvers or recursive resolvers ensures that the client's IP address remains undisclosed in Stage 2. In Stage 1, a scheme that *Hides-Client-IPaddr-From-Resolver* (P3) must additionally remove ECS from its DNS queries, as failure to do so would render the added overhead futile. Stage 1 schemes (*i.e.,* DoTor or ODNS) that remove ECS from DNS queries at stub or recursive resolvers achieve

full credit for *Hides-Client-IPaddr-in-s2*.

3.3.2 Comparison of Schemes

Several observations can be made from the analysis of Table 3.1. First, none of the secure DNS schemes provide security in the whole path of DNS resolution (*i.e.,* both stages), and none satisfy all properties in one stage. The majority of secure-DNS schemes in our evaluation are designed to operate in Stage 1. To ensure a secure DNS resolution process, these Stage 1 schemes have to assume that other Stage 2 schemes (*e.g.,* DNSSEC) are employed to secure the communication between recursive resolvers and ANSes. However, Stage 2 schemes such as DNSSEC and DNSCurve are as yet not widely deployed and do not provide all of the defined properties in Stage 2. Thus, clients using Stage 1 schemes might falsely assume that their DNS resolution is fully protected, while the security and privacy of their queries can be compromised in Stage 2.

DNSSEC (A disproportionate amount of effort for a minimal return): Table 3.1 illustrates that DNSSEC only provides two properties: *Resilient-to-False-ANS-Response* (S3) and *Avoid-Duplicating-Longterm-Secret* (S5). As stub resolvers are typically configured as non-validating and depend on recursive resolvers for DNSSEC usage and validation, DNSSEC only protects Stage 2. Recursive resolvers in Stage 1 use designated bits to signal the clients that a response is DNSSEC-validated. However, the communication between stub resolvers and recursive resolvers remains unprotected. The lack of security in Stage 1 allows an attacker to inject false responses to clients, while possibly misleading them into believing that DNSSEC has provided response validation. DNSSEC is not resilient against DoS attacks, and due to the increased response size is subject to reflection amplification attacks. Furthermore, due to long-lived signatures, DNSSEC-signed resource records are susceptible to replay attacks, enabling the replay of stale records or sub-optimal responses in CDN services [71]. Additionally, DNSSEC was not designed to provide privacy-related properties; thus, it does not satisfy any of our privacy properties.

DNSSEC, while providing minimal properties, introduces a significant key management and signature generation burden adding computational and bandwidth overhead to DNS resolution in Stage 2.

The live DNSSEC implementation in Table 3.1 denotes a version of DNSSEC that generates signatures in real-time, requiring the presence of signature generation

keys on the servers. Since the generated signatures are not unique for each query, this real-time approach remains vulnerable to replay attacks. Furthermore, the requirement to store signature keys on server instances increases the risk of key exposure to potential threats. Consequently, unlike offline-signed DNSSEC records, this implementation does not satisfy P5.

Beyond encryption: All of the evaluated schemes that were proposed to work in Stage 1 use encryption to prevent unauthorized access to plaintext DNS messages. However, encryption alone has been found to be insufficient to protect against traffic analysis attacks. These attacks leverage metadata and encrypted data patterns, such as time, size, direction, or order of packets, to infer or classify the queried domain name or visited websites through statistical or ML-based techniques. Simple padding schemes have been demonstrated to be ineffective in mitigating traffic analysis attacks that reveal the domain names visited by users [150].

One technique for obscuring size-related metadata and patterns is employing large padding schemes prior to encryption. However, this results in a significant increase in bandwidth overhead. An alternative approach is to adopt schemes that repackage traffic into uniform-sized packets, such as Tor cells in DoTor. However, using Tor also results in significant time, computational, and bandwidth overhead. A viable approach for mitigating traffic analysis, which requires further research, can be using various techniques such as padding, merging traffic with other protocols, and implementing traffic-flow security measures (*e.g.,* sending cover traffic) in the DNS context to effectively mitigate such attacks with a reasonable overhead.

TLS-based DNS schemes: Among the TLS-based schemes, DoH provides more privacy and availability properties than the others. In addition to the properties of other schemes, DoH partially provides *Conceal-DNS-Msg-Nature-in-s1* (P2), and *Resilient-to-DNS-Server-Censorship* (A2) by merging its traffic with the web. TLS-based schemes that use UDP as the transport layer protocol, such as DoDTLS and DoQ, are designed to enhance the performance of DoT while maintaining similar security and privacy properties.

Near-identical schemes: Except for *Conceal-DNS-Msg-Nature-in-s1* (P2) and *Resilient-to-DNS-Server-Censorship* (A2), DNSCrypt*V3 and DoH provide similar properties. Due to the distinguishability of DNSCrypt's traffic, previous research demonstrated that isolating DNSCrypt traffic is relatively easy without sophisticated techniques [131]. However, isolating DoH traffic from web traffic requires

more complex ML-based and statistical techniques. Consequently, isolation of DoH traffic is more complex than isolation of DNSCrypt traffic; thus, DoH partially satisfies *Conceal-DNS-Msg-Nature-in-s1* and *Resilient-to-DNS-Server-Censorship*, while DNSCrypt does not. Moreover, DoH does not provide any mechanism to conceal the IP addresses of clients from recursive resolvers; therefore, DoH does not satisfy *Hides-Client-IPaddr-From-Resolver* (P3). Although DNSCrypt employs relays to decouple DNS queries from client IP addresses, its lack of explicit removal or truncation of ECS in DNSCrypt rules out credit for *Hides-Client-IPaddr-From-Resolver*.

Confidentiality in Stage 2: All of the secure DNS schemes that work in Stage 1 provide confidentiality in that stage. Hence, based on their use of encryption, they all provide some degree of *Resilient-to-Eavesdropping-in-s1* (P1). However, in Stage 2, only DNSCurve provides confidentiality, which has not been adopted by the root and TLDs. That is, the majority of DNS traffic in Stage 2 consists of DNSSEC and Vanilla DNS, with their plaintext data being accessible to unauthorized entities. If Stage 2 traffic contains client-related information, such as ECS or complete QName, it will disclose client-related information in Stage 2. This highlights the importance of eliminating or truncating ECS from DNS messages and employing QName minimization to reduce client-related information leaks in Stage 2.

Censorship resilience: Schemes that use a distinguishable port number are vulnerable to port-based server censorship. Among the evaluated schemes, only DNSCrypt, DoH, and DoTor employ port numbers that are shared other widely used protocols. Furthermore, it is evident that schemes susceptible to port-based censorship are also at risk of censorship based on IP address, as their name servers can be identified by Internet scanning techniques. Given that censoring organizations (agents) employ various strategies to restrict access to the Tor network [1], they can similarly block or degrade DoTor traffic along with other Tor-related communications. Additionally, DNSCrypt traffic has distinguishable characteristics [131], making it easily recognizable and, therefore, susceptible to censorship. Overall, among the evaluated schemes, DoH appears to have the highest resilience against traffic isolation and server censorship.

Availability: Regarding the availability of recursive resolvers, only Stage 1 schemes that work on top of TCP or implement some means of source IP address verification before resolving a DNS query are *Resilient-to-Resolver-DoS*. Additionally, such schemes also resist DNS-based reflection amplification attacks by verifying

the source IP address. However, none of the evaluated Stage 2 schemes protects ANSes against DoS attacks. One possible explanation for the lack of DoS resistance schemes in Stage 2 is the limited number of in Stage 2, and ANSes may be reluctant to deploy stateful protocols that require maintaining the communication states and consequently result in significant per-query overhead compared to UDP-based schemes.

Traffic distinguishability: The distinguishability of DNS resolution traffic from other network traffic can harm the availability and privacy of DNS resolution. Two secure DNS alternatives satisfy some degree of *Conceal-DNS-Msg-Nature-in-s1* (P2), namely DoH and DoTor. However, it was shown that DoH traffic can be distinguished from web traffic [163]; thus, DoH received partial credit for *Conceal-DNS-Msg-Nature-in-s1*. Therefore, only DoTor fully renders DNS traffic indistinguishable, and thereby mitigates attacks on isolated traffic of a DNS scheme, such as downgrade attacks or exclusive censorship of DNS traffic.

Summary: According to Table 3.1, Stage 1 protocols primarily provide confidentiality (P1), message authentication (S1), anti-replay mechanism (S2), resilience to payload-based censorship (A1), and resilience to resolver DoS (A3) in that stage. However, Stage 1 schemes often do not conceal the nature of DNS messages (P2) and most are susceptible to DNS server censorship (A2).

Regarding Stage 2 schemes, we observe that the number of Stage 2 schemes is limited, and the evaluated schemes provide a different set of properties compared to one another. Moreover, none of the schemes provides resilience to ANS DoS (A4); however, both of the evaluated schemes mitigate cache poisoning attacks by preventing false response injection (S3) and neither of the two hides clients IP addresses completely in Stage 2 (P5).

3.4 Discussion

In this section, based on our literature review from Section 2.2 and the comparative evaluation in Chapter 3, we discuss several high-level insights regarding the name resolution process. One general observation is that the schemes designed to work in Stage 2 are limited in that they provide only a handful of the defined security benefits. On the other hand, Stage 1 schemes are diverse, and provide various combinations of properties but none provides all.

Slow migration away from Vanilla DNS. Although numerous secure DNS schemes have been proposed over time, Vanilla DNS remains dominant for name resolution in practice, particularly in Stage 2 [34, 168]. A subset of the secure DNS schemes have security weaknesses and deployability obstacles that impede their large-scale adoption. Therefore, designing new secure DNS schemes that are more secure and deployable appears to be necessary for widespread adoption. To enhance the adoption rate of secure DNS schemes, it appears crucial to incentivize and educate clients and administrators of recursive resolvers and ANSes on their benefits and implementation strategies. Furthermore, setting secure DNS schemes as the default configuration in client-side software (*e.g.,* web browsers) may be another effective approach to progress widespread adoption. However, for both current and future schemes, to prevent the centralized collection of client data in Stage 1, client software must, by default, use multiple resolvers from different organizations and distribute queries accordingly.

The privacy benefits of TLS-based schemes are questionable. TLS-based schemes (*i.e.,* DoT, DoH, DoQ, DoDTLS) rely on the evolved TLS and web PKI [91], but mainly provide security; in the DNS context, they often fail to satisfy privacy properties. For P1, encryption in TLS-based schemes must be enhanced with defense mechanisms (*e.g.,* strong padding) against traffic analysis attacks that reveal or allow identification of domains visited by users. Additionally, most TLS-based schemes require means to conceal DNS message nature in Stage 1 (P2) and hide the IP addresses of clients from the recursive resolvers (P3). Aside from lack of privacy properties, the TLS-based schemes are primarily proposed to secure Stage 1, and Stage 2 remains unsecured. Taking into account these points, introducing the TLS-based schemes as secure and privacy-preserving schemes to clients risks misleading clients about name resolution security and privacy. For example, web browsers, such as Google Chrome or Microsoft Edge, introduce DoH as *secure DNS*, but this only secures Stage 1 and does not provide any of our properties in Stage 2.

Stronger privacy in other alternatives. Regarding Stage 1 schemes, schemes that transmit DNS messages over the Tor network (*e.g.,* DoTor) provides more privacy benefits than others. DoTor is the only scheme that offers robust privacy against intermediate entities in Stage 1, as it provides strong confidentiality and conceals the nature of DNS traffic; thus, it prevents intermediate entities (including ISP) from detecting name resolution. Additionally, DoTor [144] hides the IP address

of clients from recursive resolvers. To take advantage of the benefits of Tor and preserve security and privacy of DNS messages in Tor exit relays, other secure DNS schemes in Stage 1 (*e.g.,* DoT or DoH) could be combined with Tor when a regular (non-onion) recursive resolver is queried (*e.g.,* DoH-over-Tor [144]). However, using Tor results in significant latency, bandwidth increase, and computational overhead, which must be considered before integrating Tor with a secure DNS scheme. Aside from integrating DNS schemes with Tor, combining Stage 1 schemes with ODNS-like relays (*e.g.,* Oblivious DoH (ODoH) [100]) enhances their privacy by hiding client IP addresses from recursive resolvers. Among the Stage 2 schemes, DNSCurve illustrates stronger privacy properties mainly by utilizing authenticated encryption.

ISP correlation. ISPs can correlate encrypted DNS traffic and subsequent traffic originating from a client, and thus infer queried domain names from encrypted DNS messages. ISP correlation-based information gathering is enabled by two factors: the distinguishability of encrypted name resolution and domain name leaks from other protocols. This suggests that, a secure DNS scheme that renders its traffic indistinguishable (*i.e.,* provides P2) from other traffic can effectively mitigate these correlations. Mitigating domain name leaks in other protocols (*e.g.,* using ESNI or ECH [141]) can also alleviate such correlation-based domain name leaks.

Complete availability. Availability is required for both recursive resolvers and ANSes to ensure that name resolution takes place effectively. A subset of secure DNS schemes aim to only mitigate reflection amplification by enforcing TCP when DNS responses are larger than queries (*e.g.,* DNSCrypt) or larger than a specific threshold (*e.g.,* DNSCurve). While such schemes address reflection amplification attacks with minimum overhead, attackers can still perform DoS attacks by sending UDP-based queries with spoofed source IP addresses that do not result in large responses to enforce TCP, and expend the computation resources of such servers. Therefore, using TCP as the default transport-layer protocol of a DNS scheme can mitigate DoS with spoofed source IP addresses attacks targeting DNS servers.

Alternatively, source IP address verification can be implemented at the application layer of a UDP-based scheme to mitigate DoS attacks targeting DNS servers. However, validating the source IP address typically requires either an additional round-trip or the maintenance of state from previous interactions, introducing computational and network overhead. As a result, designers of secure DNS schemes may choose to focus on mitigating reflection amplification attacks without enforcing

source address validation to reduce performance costs.

Deployability issues. Due to the two-staged structure of DNS resolution and the difference in required properties of each stage, proposing a comprehensive and efficient DNS scheme that provides security, privacy, and availability properties in both stages appears to be challenging. In addition, complex DNS alternatives that require significant changes or introduce considerable overhead are unlikely to achieve widespread adoption. Moreover, ANSes in Stage 2 have been reluctant to adopt schemes that are complex or add considerable time or computation overhead. Therefore, proposing a secure DNS scheme that meets the required properties of Stage 2 while overcoming deployability barriers to achieve large-scale adoption remains an open challenge. One of the objectives in Chapter 4 is to propose a new secure DNS scheme for Stage 2 that enhances security and privacy while mitigating the major deployment challenges associated with previously proposed schemes.

Combining schemes. Since designing a single secure DNS scheme for both stages potentially results in complexity and significant changes to the name resolution process, combining schemes is a potential solution. There are two main approaches for combining secure DNS schemes: combining schemes within the same stage and combining schemes in different stages. Integrating schemes in the same stage augments the provided properties by the combined schemes in that stage, while combining schemes in different stages secures both stages (the entire DNS resolution path) based on the selected schemes. For example, by combining DoT with DoTor [144], or DoH with ODNS-like relays [100] in Stage 1, the resultant scheme increases the provided properties only in Stage 1. On the other hand, by integrating the TLS-based schemes in Stage 1 with DNSSEC in Stage 2, each scheme provides benefits in one stage, and by complementing each other the entire DNS resolution path is enhanced.

When combining schemes, the resulting overhead must not outweigh the augmented properties. For instance, while integrating Stage 1 schemes with Tor may enhance privacy, Tor introduces significant performance and computational overhead, making it unsuitable for latency-sensitive or resource-constrained scenarios.

Centralization. It is widely accepted that relying on a single resolver is harmful to clients' privacy in Stage 1 [81, 147]. Accumulated name resolution history of clients can be used by resolution services for client identification and re-identification, and other goals, such as financial gains.

The various types of stakeholders in a DNS resolution process, such as clients, network administrators, and organizations maintaining resolution services or ANSes may prioritize their objectives differently. However, prioritizing client privacy and security, a name resolution must use mechanisms that avoid centralization. For example, DoTor [144] employs Tor relays, and ODoH [100] or DNSCrypt [48] use specific relays to conceal the IP address of clients from recursive resolvers. Beyond relay-based solutions, Hoang *et al.* [76] proposed the K-resolver to distribute DNS queries among different DoH resolvers, which mitigates the privacy risks of a single centralized resolver.

Bypassing censorship. Internet censoring entities often leverage DNS messages or resolvers to block access to specific services [16, 132]. Therefore, using non-censoring recursive resolvers, residing out of the authority of censoring agents, can help circumvent DNS-based censorship. The majority of schemes proposed for Stage 1 offer some degree of resilience against payload-based censorship. However, all these schemes remain vulnerable to server-based censorship, wherein censoring authorities can effectively block the entire scheme by blocking access to the associated servers.

3.5 Summary

In this chapter, we developed a comprehensive threat model and attack taxonomy for the DNS resolution process (Section 3.1). Based on the identified threats, we defined 14 security, privacy, and availability properties (Section 3.2), which were used to develop a framework for the objective evaluation of DNS schemes (Section 3.3). Using this framework, we conducted a comparative evaluation of a number of previously proposed secure DNS schemes, and provided high-level insights into the security, privacy, and availability aspects of the DNS resolution process (Section 3.4). The systematic analysis in this chapter highlights the absence of, and the need for, a secure DNS scheme that augments the security of DNS resolution in Stage 2 while minimizing deployment barriers as much as possible. In Chapter 4, we present the design goals and theory of operation of DNSSEC+, a secure DNS scheme proposed to augment the security and privacy of Stage 2 while minimizing deployability barriers and mitigating significant performance overhead.

Chapter 4

DNSSEC+: A Secure Scheme in Stage 2

As discussed in Chapter 3, the absence of security and privacy measures between DNS recursive resolvers and authoritative nameservers has been exploited by both on-path and off-path attackers. Moreover, while many security proposals have been introduced in practice and in previous literature, they typically focus on enhancing security in Stage 1. In contrast, Stage 2 schemes are limited, and often face deployability barriers or fail to provide security and privacy properties that mitigate the threats within this stage. The absence of a broadly adopted security solution between resolvers and ANSes motivates a new scheme that mitigates these issues in previous proposals.

In this chapter, by drawing insights from shortcomings and benefits of previous secure DNS schemes in Stage 2, specifically DNSSEC, we present DNSSEC+.[1] Primarily influenced by DNSSEC, it operates in Stage 2 and aims to preserve the beneficial goals and properties of DNSSEC, while addressing its security and privacy deficiencies. DNSSEC+ is designed to also incorporate beneficial Stage 2 properties of other secure DNS proposals while avoiding their security and privacy vulnerabilities, and deployability obstacles. In a nutshell, DNSSEC+ provides real-time message authentication while avoiding the duplication of long-term secrets on zone server replicas. It also provides query and response confidentiality, including forward secrecy for responses, without requiring extra network round-trips.

To achieve these benefits, we introduce a novel short-term delegation mechanism, where a DNS zone delegates record-signing to its nameserver replicas (which may not be fully trusted by the zone owner) using the nameservers' own keys. Short-term delegation approaches (see Section 2.3), such as the use of delegated credentials [22] in CDNs for TLS-based communications, have been found to be effective in protecting the long-term keys of security protocols [33]. This short-lived delegation enables zone nameserver instances to serve zone records using real-time cryptographic operations, while limiting the exposure of long-term keys within the zone and minimizing risks of short-term key compromise. Because the performance and adoption require-

[1] We are in the process of pursuing publication under the revised name ss2DNS to avoid the misinterpretation that DNSSEC+ (our proposal) is a direct extension of DNSSEC.

ments of a secure DNS scheme is expected to have a crucial impact on its adoption, we aim for DNSSEC+ to have performance comparable to practical schemes. Besides requiring the usual (single) round-trip between the resolver and each ANS (thus no added network overhead or increased network resolution delays), DNSSEC+ uses the existing DNSSEC trust model, thus requiring no new infrastructure.

4.1 Desired Properties of DNSSEC+

In this section, building on the strengths while addressing the shortcomings of DNSSEC, we define nine design properties to satisfy in the development of DNSSEC+. These are thus positioned as required properties; however, the technical details of DNSSEC are not incorporated into DNSSEC+.

As explained in Chapter 3, an adversary can mount off-path or on-path active and passive attacks in Stage 2. Active attacks enable security and availability threats, whereas passive attacks are often sufficient to compromise privacy. Since DNSSEC+ aims to secure Stage 2 of the DNS resolution process, and the threats of Stage 1 differ (see Sec. 3.1), these must be mitigated using a secure DNS scheme in Stage 1. Furthermore, the required properties and trust model for a Stage 1 scheme differ from those of Stage 2, as each stage involves different entities with varying interests in the DNS resolution process.

4.1.1 Desirable Properties to Retain from DNSSEC

In this section, we identify the desirable properties from DNSSEC that we retain in DNSSEC+.

Message Authentication: False response injection can be performed by on-path and off-path attackers in Stage 2 (Section 3.1). Similar to DNSSEC, DNSSEC+ provides message authentication and integrity to prevent unauthorized manipulation and injection of responses.

Avoid duplicating long-term secret: In DNSSEC, ANSes within a zone contain and serve the pre-signed DNS records for which these nameservers are authoritative. Therefore, there is no need to duplicate the long-term private keys (*i.e.,* KSK or ZSK; see Section 2.2.9) on each nameserver instance within DNSSEC-enabled zones. The root and TLD zones typically store the KSK or even the ZSK

on a secure system, which is separate from the nameservers. DNSSEC's signature lifetime facilitates the secure storage of long-term private keys on a server. These signatures have a defined expiry window, during which they can be served without requiring access to the private key. In secure DNS schemes that employ real-time cryptographic operations (*e.g.,* encryption or signing), the private key is required to be present on the ANSes. The duplication of private keys on the nameservers of a zone exposes these keys to attacks targeted at ANS instances. As another secure DNS scheme in Stage 2, DNSCurve requires private keys to be present on all ANS instances to securely transmit DNS messages [145]. To minimize exposure of long-term secrets within each zone, DNSSEC+ aims to avoid duplication of long-term keys by using a secure delegation mechanism.

Single round-trip: DNSSEC employs UDP with single round-trip DNS resolution to transfer DNS responses alongside their corresponding signature. Since the communications over the Internet are typically preceded by a DNS query, any viable designed DNS scheme must minimize latency. Therefore, one of the main deployability and usability goals of DNSSEC+ is to have a single round-trip for the transmission of a query and the reception of its corresponding response, thereby minimizing the overall delay associated with name resolution in Stage 2.

Established trust model: Stage 2 DNS security schemes need to provide mechanisms for recursive resolvers to trust the keys used by the nameservers—that is, a *trust model*. The web trust model is prevalent over the Internet, with billions of issued certificates [35]. The web PKI has been used by Stage 1 schemes, such as DoH and DoT [91]. However, in Stage 2, the web PKI has been rarely used. We believe the reason for this is that TLS-based schemes (*e.g.,* DoT, DoH) are relatively expensive for Stage 2, and the root zone as a core authority within the Internet infrastructure is reluctant to rely on external entities (*e.g.,* CAs) in the web PKI as its trust anchors. For DNSSEC+, we use a DNSSEC-like trust model, which has been accepted and adopted by the root and TLDs within the DNS hierarchy.

4.1.2 DNSSEC Shortcomings to Be Addressed

This section outlines the weaknesses and vulnerabilities of DNSSEC that are addressed in DNSSEC+.

Significant amplification: As explained in Section 2.2.9, DNSSEC employs UDP for single round-trip query resolutions. While efficient, this results in suscep-

tibility to reflection attacks with considerable amplification factors. In DNSSEC+, we aim to keep the single round-trip resolution, while minimizing the amplification factor.

Replay attacks: In DNSSEC, signed resource records can be replayed within their expiry window, resulting in vulnerability to stale-response injection. To prevent response replay attacks from previous interactions, in DNSSEC+ we use TVPs [161] for freshness.

Failing open: DNSSEC was designed with algorithm agility, enabling the use of new cryptographic algorithms, and the removal of deprecated ones. However, the lack of specified mechanisms for handling failures in validation or support of new algorithms has resulted in vulnerable DNSSEC implementations on DNSSEC-validating resolvers [72]. Adversaries can exploit these vulnerable resolvers by injecting false responses with unsupported cryptographic fields (*e.g.,* signatures or keys) [72]. Vulnerable recursive resolvers accept these false responses as they do not support and validate their cryptographic fields, thereby rendering them susceptible to cache poisoning attacks, even when the zones are DNSSEC-protected [72]. In a secure DNS scheme, if message authentication fails, DNS messages cannot be trusted and should be considered invalid. Adhering to the *safe-defaults* [161] principle, if at any point in the name resolution process of DNSSEC+ any verification fails, name resolution should be terminated and results discarded. Thus, by failing closed, potential downgrading attacks that could bypass the security validations within a secure DNS scheme can be mitigated.

Lack of confidentiality and forward secrecy: DNSSEC does not provide confidentiality. In the schemes that provide confidentiality by encrypting queries in Stage 2, adversaries may still infer the queried domain name by analyzing query patterns to ANSes [148]. However, certain queries in Stage 2 contain client-specific information, such as ECS [36] or domain names that can be directly associated with individual clients. Therefore, encrypting DNS messages in Stage 2 can protect client-related information and increase the difficulty of identifying the queried record. DNSSEC+ has the primary aim of response authenticity and integrity, which is achieved by transmitting authenticated and encrypted responses thus providing response confidentiality as well. In DNSSEC+, responses also provide forward secrecy; thus, the compromise of long-term keys does not compromise the confidentiality of previously transmitted responses. DNSSEC+ also provides an optional

query-encryption mode for privacy protection, as detailed in the following section.

4.2 DNSSEC+ Technical Details

DNSSEC+ is primarily motivated by the lack of real-time record signing in DNSSEC (as explained in Section 2.2.9). This design choice in DNSSEC provides the benefit of avoiding the duplication (copying) of *precious* zone signing keys across hundreds of potentially untrusted *nameserver instances* (the physically distributed server replicas deployed globally). In DNSSEC+, records are signed in real-time, without duplicating private keys. The main approach here is to allow each (untrusted) nameserver instance to sign data using its own unique key, and have that key being *authorized* by a central key server constituting the main ANS of the zone. The central key server authorizes the key of a nameserver instance by signing it, and revokes the key by refraining from renewing the signature. Such key signing can be implemented in an automated fashion, thus allowing very short key lifetimes (*e.g.,* few hours).

This design fundamentally shifts the perception of the replicated DNS zone server instances, from the standard *"logically centralized but physically distributed"* notion, to a *"delegated (decentralized) servers"* notion.[2]

In what follows, we detail how DNSSEC+ operates, how a recursive resolver follows the chain of trust to verify the authorization of a server instance, and how query-response privacy can be added to DNSSEC+ without introducing new network round-trips between the resolver and any nameserver instance.

Nameserver delegation. Figure 4.1 shows a zone in DNSSEC+. Each zone has a *central key server* (*"key server"* for short), which is trusted by, and under direct control of, the zone administrator. Its purpose is to store the long-term signing key of the zone, and delegate a limited authority to nameserver instances within the zone by signing their keys. This delegation authorizes nameserver instances to respond to queries using DNSSEC+.

Trust model. Similar to DNSSEC, a reveres-tree chain of trust is used in DNSSEC+, where the public component of the long-term signing key of each zone (*i.e.,* the verification key) is transferred and published in the parent zone. The public component of the long-term key of the root zone is installed in DNS resolver

[2]Not to be confused with DNS *zone delegation*, where an entire DNS zone is delegated to other ANSes. The new delegation we are referring to in DNSSEC+ happens *within* a zone.

software as a trust anchor.

Real-time integrity protection of DNS responses. Having access to the long-term verification key of the zone (either from the parent zone, or hard-coded as a trust anchor for the root), a recursive resolver querying a nameserver instance will first verify the signature on the structure holding the nameserver's short-term key. This short-term key will then be used to establish a symmetric key between the nameserver instance and the recursive resolver, and transfer an encrypted DNS response back to the resolver.

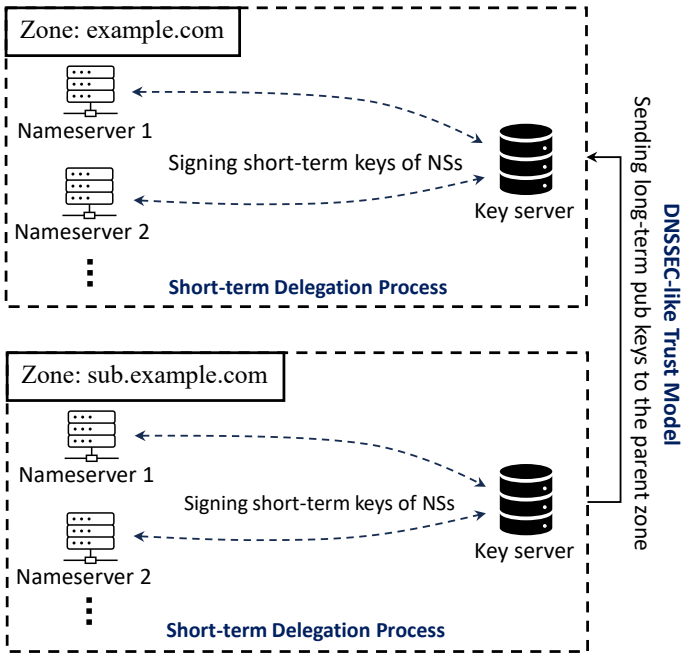


Figure 4.1: Zones in DNSSEC+ consist of a DNSSEC-like trust model (public key of each zone is authenticated by its parent zone) and short-term delegation.

Additional feature: query confidentiality. In addition to response integrity, DNSSEC+ also provides query confidentiality (for privacy) as an optional feature. It thus has two modes of operation: *no-privacy* and *privacy-enforcing*. The former provides confidentiality and integrity for DNS responses only, the latter for requests and responses. A notable challenge in the privacy-enforcing mode is that a recursive resolver must obtain the short-term nameserver key first from the nameserver itself, as it is not present in the parent zone (which costs a round-trip with the nameserver instance), then use it to encrypt the query and send it to the nameserver instance (a second round-trip). Doubling the round-trip would be a major hindrance to the

adoption of DNSSEC+ in practice.

To avoid requiring an additional round-trip, we use two different symmetric keys: one to encrypt the query, the other the response. To establish the query key with a nameserver instance, the resolver obtains all needed information from the parent nameserver. When it transmits the encrypted query, it sends with it its own DH agreement key (the g^a) in the same transmission. Upon receiving this, the nameserver instance generates the response key, encrypts the response with it, and sends it along with its freshly-generated DH agreement key (the g^b), which it signs with its own short-term key (itself was signed by the long-term zone key).[3] Note that while, the query-encryption key is now accessible to all nameserver instances (unlike the response-encryption key), an adversary compromising that key does not impact the integrity and authenticity of the responses (as they use a different key), which, similar to DNSSEC, is the primary goal of DNSSEC+ (hence the name).

Forward Secrecy. DNSSEC+ implements a half-static DH key agreement approach for queries to enable DNS resolution within a single round-trip. Thus, forward secrecy is not provided for queries. However, DNSSEC+ provides forward secrecy for responses, as the nameserver uses the resolver's ephemeral key (included in the query) and generates a new ephemeral key for each response transmission.

4.3 Zones in DNSSEC+

In each zone, there is a key server trusted by the zone owner (Fig. 4.1), and there are other nameserver instances that may not be completely trusted by the zone owner. The key server securely stores the long-term private signing key of the zone. The nameserver instances can be nameservers under the control of the zone administrator, or globally distributed nameserver instances managed by a CDN service provider, which are not directly controlled by the nameserver administrator, and do not have access to the long-term private key of the zone.

Table 4.1 lists the symbols used for specifying keys, zones and nameservers. A zone with level l in the DNS hierarchy has a long-term signing key ($\underline{\mathrm{w}}_l$), stored on the key server of the zone. By a secure but unspecified means,[4] the nameservers and the key server within a zone must be able to mutually authenticate each other

[3] We use modular exponentiation (modp) for exposition, but expect EC to be used in practice.

[4] Different CDN providers may use different methods to ensure secure interactions with their edge servers, and we do not impose any specific constraints on these options.

and confidentially exchange messages. A nameserver with ID i in a zone with level l in the DNS hierarchy generates a fresh short-term signing key structure (ω_l^i), which is a customized type of certificate. Subsequently, the nameserver sends its short-term public key structure (ω_l^i) through the described secure channel to the key server of the zone. The key server verifies the short-term key structure and its origin nameserver, and upon successful validation, signs the short-term public key structure of nameservers (ω_l^i) using the zone's long-term signing key ($\underline{w}_l$). Finally, the key server returns the signed short-term key structure to the nameserver.

Listing 4.1: Short-term nameserver signing key structure

```
1   struct {
2       struct {
3           int inception;
4           int expiration;
5           Pubkey STK_public_key;
6           int nameserver_ID;
7           int zone_level;
8       } short_term_key_structure;
9       Signature signature;
10  } Signed_short-term_key_structure;
```

As Listing 4.1 shows, the short-term signing key of a nameserver consists of a key value plus 4 attributes. `inception` and `expiration` values indicate the lifetime of the short-term key structure. `STK_public_key` is the short-term signing public key of a nameserver (ω_l^i), and `nameserver_ID` indicates the unique ID of a nameserver within a zone. Finally, `zone_level` field indicates the level of the zone in the DNS hierarchy within which this short-term key is signed. These five fields constitute the `short_term_key_structure`, which will be signed by the long-term key of a zone. The signed structure with the included `Signature` field then forms the `Signed_short-term_key_structure`.

For instance, Fig. 4.2 illustrates the process of signing short-term key structures in the root zone. As the top arrow shows, *Nameserver 1* generates a short-term key structure (ω_0^1), where "1" represents the nameserver ID and "0" indicates the zone level. This structure is then transmitted to the root zone's key server using an authenticated and encrypted channel. Upon securely receiving the short-term public key structure of *Nameserver 1* (ω_0^1), and validating the key structure and

Symbol	Meaning
$\mathcal{A}$, $\underline{\mathcal{A}}$	Long-term public, private agreement key
Λ, $\underline{\Lambda}$	Short-term public, private agreement key
w, $\underline{\mathrm{w}}$	Long-term verifying, signing key
ω, $\underline{\omega}$	Short-term verifying, signing key
r	Unique random number
l	Zone level in the DNS hierarchy (subscript) ($0 \leq l$)
i	Nameserver ID number (superscript) ($0 \leq i$)
R	Recursive resolver (superscript)

Table 4.1: Symbols used in the abstract description of DNSSEC+ operation: The top four are asymmetric keys, and the bottom three are ownership annotation. The asymmetric key symbols (top four) will represent the public component of the key (agreement or signature verification), and for their private component (agreement or signing), the symbol is underlined.

authenticating the nameserver, the zone's key server signs the short-term public key structure of the nameserver ($Ss.1 = S_{\underline{w}_0}(\omega_0^1)$) using the long-term signing key of the zone ($\underline{\mathrm{w}}_0$). Subsequently, the key server securely transfers the signed short-term key of the nameserver to *Nameserver 1*. The signed structure of short-term keys of nameservers have a validity period that specifies their lifetime. The signed short-term key structures have a relatively brief lifetime (*e.g.,* hours to days), defined by the `inception` and `expiration` fields. Thus, short-term key structures minimize the threat and exposure of compromised keys and ensure implicit revocation of nameserver keys in short time intervals.

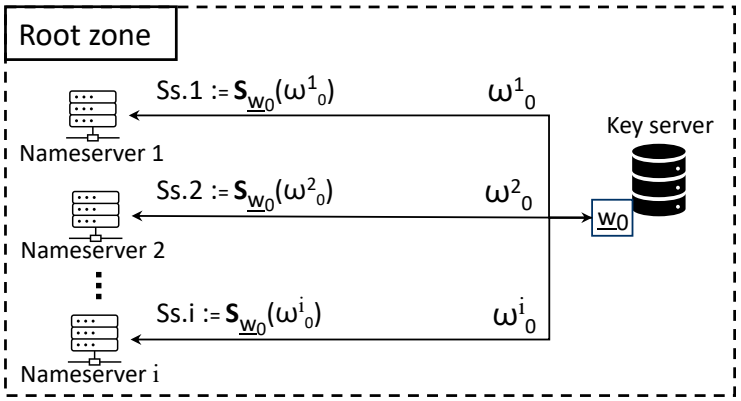


Figure 4.2: The delegation process in DNSSEC+: signing the nameserver short-term key structures by the long-term signing key of the zone.

Function	Used to	Notation
Symmetric A-encryption	Encrypt message m with key a	$E_a(m)$
Symmetric A-decryption	Decrypt message m with key a	$D_a(m)$
Signature generation	Sign message m with key a	$S_a(m)$
Signature verification	Verify signature on message m with key b	$V_b(m, S_a(m))$
Key establishment	Produce DH key using private key $\underline{A}$ and public B	$DH(\underline{A}, B)$
Generate ephemeral key pair	Generate ephemeral agreement keys	$(\underline{A}, A) := GenDH()$

Table 4.2: List of functions used in DNSSEC+ (A-encryption/decryption: Authenticated encryption/decryption)

Label	Key type	Used to
	Zone Keys	
$\underline{\mathcal{A}_l}$	Long-term zone **private agreement key**	Establish shared secret for query A-decryption
$\mathcal{A}_l$	Long-term zone **public agreement key**	Establish shared secret for query A-encryption
$\underline{\mathrm{w}_l}$	Long-term zone **signing key** (private)	Sign short-term nameserver keys
w_l	Long-term zone **verifying key** (public)	Verify short-term nameserver keys
	Nameserver Keys	
$\underline{\omega_l^i}$	Short-term nameserver **signing key** (private)	Sign ephemeral session agreement keys
ω_l^i	Short-term nameserver **verifying key** (public)	Verify ephemeral session agreement keys
$\underline{\Lambda_l^i}$	Ephemeral nameserver **private agreement key**	Establish shared secret for response A-encryption
Λ_l^i	Ephemeral nameserver **public agreement key**	Establish shared secret for response A-decryption
	Resolver Keys	
$\underline{\Lambda^R}$	Ephemeral resolver **private agreement key**	Establish shared secret for query and response
Λ^R	Ephemeral resolver **public agreement key**	Establish shared secret for query and response

Table 4.3: List of keys used in DNSSEC+

Before the expiration of the current signed key structure, the nameserver instances generate a new short-term signing key structure. Subsequently, this newly generated key structure is transmitted to the zone's key server via an authenticated and encrypted channel to be signed. If the nameservers within a zone do not renew their short-term signing key structures prior to the expiration of the current key, the resolvers cannot validate the responses after expiration of the current key and the DNSSEC+ resolution fails. The long-term keys in DNSSEC+ are stored securely on the key server of each zone. Thus, the attack surface of the long-term keys is significantly smaller compared to the short-term key structures, which are stored on the nameserver instances.

Aside from the long-term signing key of each zone (w_l), which is stored on a key server within each zone, there is another long-term agreement key associated with each zone ($\mathcal{A}_l$). See Table 4.3 for a complete list of DNSSEC+ keys. To provide confidentiality of DNS queries, resolvers need to have access to a trusted (authentic) public agreement key from the nameservers. Retrieval of this key from the nameserver would require an additional round-trip, violating our desired sin-

gle round-trip policy (Section 4.1). In order to satisfy the single round-trip and confidentiality properties at the same time, each zone with level l contains another long-term initial agreement key ($\mathcal{A}_l$). Unlike the zone's private long-term signing key ($\underline{\mathrm{w}}_l$), which is stored only on the key server within each zone, the private long-term initial agreement key ($\underline{\mathcal{A}}_l$) is transferred to all the nameserver instances within each zone. In DNSSEC+, when a zone generates $\mathcal{A}_l$, it is required to transmit it to the parent zone along with the zone's long-term signing key (w_l). Then, $\mathcal{A}_l$ is used to provide confidentiality of DNS queries, as we explain next in Section 4.4. Based on the decision of resolvers on the privacy level of queries, they can use the long-term agreement key of zones for query encryption.

4.4 Name Resolution in DNSSEC+

In DNSSEC+, ANS i within a zone with level l has two keys (see Section 4.3): one short-term for signing (ω_l^i) and one long-term for key agreement ($\mathcal{A}_l$). The key ω_l^i is signed by the long-term signing key of the zone ($\underline{\mathrm{w}}_l$), which is stored on the zone's key server. A DNSSEC+ resolver has access to the long-term public keys of the root (w_0, $\mathcal{A}_0$) as trust anchors.

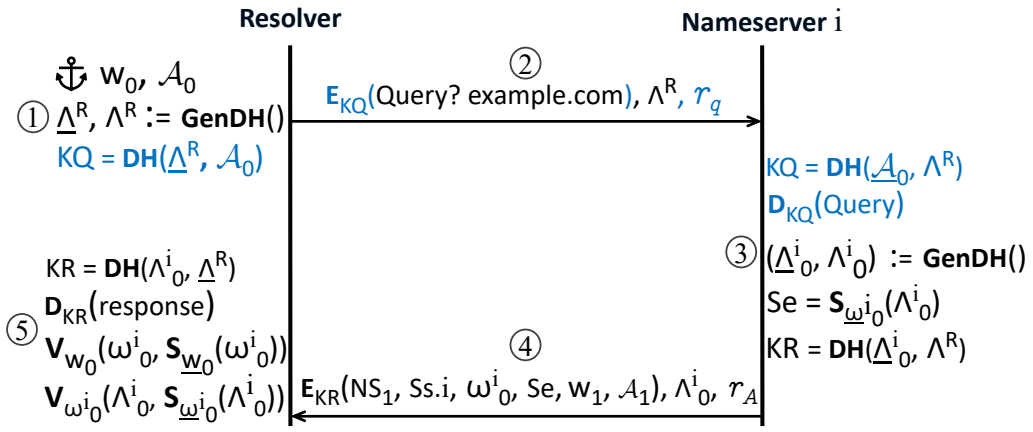


Figure 4.3: DNSSEC+ query resolution from a resolver to nameserver i of the root zone. The steps in black occur in both no-privacy and privacy-enforcing modes; steps in blue only occur in privacy-enforcing mode (query encryption).

In DNSSEC+, resolvers can operate in two modes: Privacy-enforcing and no-privacy. Based on the privacy-sensitivity of queries (*e.g.,* when ECS [36] is included) or per client (stub resolver) request, they have the option to encrypt the transmitted

queries in the privacy-enforcing mode. We use the notation in Table 4.2 to represent cryptographic functions. Also, Table 4.3 classifies the keys within DNSSEC+ based on their owner entities. The private part of an asymmetric key pair is expressed using underlined letters (*e.g.,* $\underline{\mathcal{A}}$ is a private key and $\mathcal{A}$ is its corresponding public key).

4.4.1 No-privacy Mode

Figure 4.3 illustrates the process of name resolution in DNSSEC+, when resolving a domain name from nameserver i of the root zone. The steps written in black occur when a resolver is in the default no-privacy mode. We first describe the name resolution process in the no-privacy mode, followed by an explanation of the privacy-enforcing mode. In Step 1, to initiate the query transmission, the resolver generates an ephemeral agreement key pair ($\underline{\Lambda}^R$, Λ^R). Subsequently, in Step 2, the resolver transmits the plaintext query (*Query? example.com*) alongside the resolver's ephemeral public agreement key (Λ^R) to nameserver i. Upon receiving the query and looking up the response in Step 3, nameserver i generates an ephemeral agreement key pair ($\underline{\Lambda}_0^i$, Λ_0^i). Then, the ephemeral public key of the nameserver (Λ_0^i) is signed ($Se = S_{\underline{\omega}_0^i}(\Lambda_0^i)$) using the short-term signing key of the nameserver ($\underline{\omega}_0^i$). At this point, the nameserver generates a master key (KR) using DH key agreement with the ephemeral private key of the nameserver ($\underline{\Lambda}_l^i$) and the ephemeral public key of the resolver (Λ^R). The generated master key and the fresh random number (r_A) are used as inputs of a Key Derivation Function (KDF) to derive the encryption key of the response. In addition to the standard DNS response, additional cryptographic parameters are appended to the response prior to encryption.

As Figure 4.3 shows, in this example name resolution, the resolver queries the root zone nameserver for a record associated with '*example.com*', and the root zone nameservers are not authoritative for providing the final response for this query. Therefore, nameserver i within the root zone returns a nameserver 'NS' record for the TLDs at level 1 in the DNS hierarchy. As demonstrated in Step 4, the nameserver uses the master key (KR) derived in Step 3 with a fresh random number (r_A) and a KDF to encrypt the 'NS_1' record of the TLD with level 1. Additionally, the nameserver appends the short-term key structure (ω_0^i) of the nameserver with its corresponding signature ($Ss.i$). The signature is generated by the long-term key of the zone on the key server within the zone ($Ss.i = S_{\mathrm{w}_l}(\omega_l^i)$), as described in Sec. 4.3.

Moreover, the signature of the ephemeral key of the nameserver ($Se = S_{\underline{\omega}_l^i}(\Lambda_l^i)$) is appended to the message before encryption. In this example, the NS_1 in the response belongs to a delegated TLD zone, so the long-term signing (w_1) and initial agreement key ($\mathcal{A}_1$) of the TLD are also appended to the response message. These long-term keys of the TLD will be used when the resolver initiates queries directed at the TLD nameservers. After encrypting the DNS response with additional cryptographic signatures and keys, the nameserver appends the ephemeral public key (Λ_l^i) along with the random number (r_A) used for encrypting the response. Subsequently, the nameserver transmits the response to the resolver in Step 4.

Upon receiving the response, in Step 5, the resolver generates the master key (KR) using the ephemeral public key of the nameserver (Λ_0^i) and the resolver's ephemeral private key ($\underline{\Lambda}^R$). It then decrypts the message within the response, and verifies the signature of the short-term public key structure of the nameserver ($S_{\underline{\mathrm{w}}_0}(\omega_l^i)$) using the long-term signing key of the root zone (w_0). Next, the resolver verifies the signature of the nameserver ephemeral public key ($S_{\underline{\omega}_0^i}(\Lambda_0^i)$), which was used to encrypt the response. If the decryption or signature verifications fail, the response is considered invalid and discarded. Otherwise, if all checks in Step 5 complete successfully, the resolver caches and uses the DNS response.

After securely resolving the NS record of TLD from the root zone, the resolver has access to the long-term public keys of the '*.com*' zone (*i.e.,* w_1, $\mathcal{A}_1$). The resolver is now able to repeat the same steps for resolving Second-Level Domain (SLD) NS records. When the resolver reaches the nameserver authoritative for the queried record, it repeats the same steps. However, the response does not contain the long-term keys of the child zone (*i.e.,* w_{l+1}, $\mathcal{A}_{l+1}$), as at that point the resolver has reached the authoritative nameservers for the queried record, the resolver does not need to traverse other subordinate zones.

4.4.2 Privacy-enforcing Mode

To resolve names in one round-trip while encrypting queries, we separated the long-term zone key used for providing security and privacy properties of queries from the long-term key used for responses. The blue steps in Figure 4.3 are used in addition to the black steps in the privacy-enforcing mode. After generating the ephemeral key pair, the resolver generates a master key using DH key agreement ($GenDH()$) with the root zone's initial agreement public key ($\mathcal{A}_0$) and the resolver's ephemeral

agreement private key ($\underline{\Lambda}^R$). The generated master key (KQ) is then used for query confidentiality (and integrity). In Step 2, the resolver uses authenticated encryption with a key derived from (KQ) to protect the integrity and confidentiality of the query. The authenticated encryption herein uses random numbers (192 bits) used once as TVP to ensure freshness of encryption keys [20]. The resolver then transmits the encrypted query, along with the resolver's ephemeral public key (Λ^R) and the random number (r_q) used in derivation of the encryption key, to nameserver i of the root zone.

The nameserver i receives the encrypted query with the resolver's ephemeral public agreement key and the random number from Step 2. The nameserver generates the same master key (KQ), using DH key agreement with the resolver's ephemeral agreement public key (Λ^R) and the root zone's long-term agreement private key ($\underline{\mathcal{A}}_0$). The nameserver uses the generated master key and the received nonce from the resolver to decrypt the query. The next steps after decrypting the query is the same as the steps explained in the no-privacy mode.

4.5 Caching

The caching mechanism for standard DNS records remains the same in DNSSEC+. The resource records are transmitted as authenticated and encrypted messages. After decryption and verification, they will be treated as Vanilla DNS messages. Caching the long-term keys of the zones in DNSSEC+ is essential to achieve a comparable performance to Vanilla DNS. Otherwise, each time a new record needs to be resolved by a resolver, the resolver needs to traverse the DNS hierarchy to obtain the long-term keys of the intended zone to securely resolve the query. Regarding the period for which long-term keys are cached in DNSSEC+ by resolvers, caching for long- and short-term durations have similar advantages and drawbacks as DNSSEC keys (Section 2.2.9). Caching for shorter durations enhances security, while longer caching periods improve performance. Administrators have to determine an optimal balance to effectively address both security and performance considerations.

The long-term signing key in DNSSEC+ is stored on a trusted key server within each zone and not used directly in the interaction of nameservers and resolvers. With that in mind, caching long-term public keys associated with zones for periods longer than DNS record TTL values is less likely to raise security concerns, while

providing performance benefits. For example, the public keys of a zone can be cached for 24 hours, and whenever the resolver intends to resolve a query from the nameservers within the caching period, the cached keys can be used without requiring communication with the parent zones to obtain the long-term public keys of the zone. A practical approach for setting the caching time of the zone keys is to set the caching time of the long-term keys of the top zones in the DNS hierarchy (*e.g.,* root or TLDs) relatively longer compared to their subordinate zones. In this manner, when a resolver wants to resolve a domain name within a given zone, if the long-term keys of the intended zone are not cached, the resolver does not need to traverse the entire DNS hierarchy to obtain the long-term keys of the intended zone. This is because there is a greater likelihood of the long-term keys for higher-level zones having been previously cached.

4.6 Updating Records and Keys

Standard DNS records: As DNS messages are now in DNSSEC+ sent securely using the original DNS zone files, the process of updating records of zone files in DNSSEC+ remains the same as in Vanilla DNS.

Updating Short-term keys (ω_l^i): As such keys have short lifetime, nameservers need to obtain a new signed short-term key structure before the expiration of the current one (see Section 4.3).

Updating long-term zone agreement keys ($\mathcal{A}_l$): Since long-term zone keys are used in the DNSSEC+ trust model, the process of updating long-term keys require considerations to avoid name resolution failures. For updating the long-term agreement key ($\mathcal{A}_l$) of a zone with level l, the zone administrator initially generates a new agreement key $\mathcal{A}_l$ on its key server and sends it its zone nameservers, so that they can decrypt incoming queries encrypted using the new key. In the next step, the zone owner removes the old key from the parent zone and adds the new $\mathcal{A}_l$ to the parent zone using the OOB authenticated channel between the zones.[5] After waiting for enough time, so that the old $\mathcal{A}_l$ is removed from the caches of resolvers, the zone owner removes the old agreement key from its nameservers.

Updating long-term zone signing keys ($\mathbf{w}_l$): Updating long-term signing

[5]In DNSSEC domain registrars typically provide web interfaces for adding/removing keys in parent zones.

keys is similar to updating KSKs in DNSSEC, where three update methods exist [121]. However, for updating the zone signing keys (w_l) in DNSSEC+, we use a customized approach, which is similar to the double-DS method in DNSSEC [121]. This method is the most efficient regarding the number of additional bytes added to the responses during the long-term zone signing key updates.

For updating the long-term zone signing key, denoted as w_l (old), to the new key, denoted as w_l (new), the zone owner first adds w_l (new) to the parent zone. At this point, the parent zone publishes both the old and new keys in DNS responses, and the zone owner waits for enough time to ensure the expiration of w_l (old) from the resolver caches, and the w_l (new) is cached alongside the old key in the caches of resolvers. Next, the zone owner removes the w_l (old) from its zone and starts using w_l (new) for signing the short-term key structures. Following this step, the zone owner waits for enough time, ensuring the expiration of short-term key structures signed by w_l (old) in its zone. Finally, the zone owner removes the w_l (old) from the parent zone and the process is complete.

4.7 Discussion

This section provides further discussion for the points presented above and introduces additional ideas to be explored in future.

Targeting Stage 2. DNSSEC+ operates in Stage 2, requiring a secure scheme (*e.g.,* DoT [86]) in Stage 1 if the overall goal is to secure the entire DNS resolution path. The focus on Stage 2, rather than developing a new protocol for the full resolution path, is justified by the deployment challenges faced by schemes that require fundamental changes to the original two-stage name resolution in the DNS design (see Section 2.2.12). Furthermore, various secure DNS schemes for Stage 1 have already been introduced and are increasingly adopted on both the client side (e.g., web browsers) and recursive resolver side [102, 110]. By combining DNSSEC+ in Stage 2 with an existing secure Stage 1 protocol, the security of the entire DNS resolution path can be enhanced.

Availability of Key Servers. Availability of key servers within each zone is critical. If a key server becomes unavailable before the short-term signing keys of the nameservers are updated, and those keys have expired, name resolution will fail. Since key servers play such a critical role, aside from their security, their availability

also needs to be ensured through means such as server redundancy. In practice, external trustworthy key servers can be introduced in the trust infrastructure, which can be used by zone owners to reliably outsource the functionality of acting as their key server.

Reduce Response Size. In our current model and implementation in Chapter 5, to authenticate the ephemeral keys of the DNSSEC+ nameservers, these keys are signed by the short-term signing key of the nameservers. This signature is appended as part of response and used by the recursive resolvers for verifying the authenticity of the ephemeral key. An alternative is to use implicitly authenticated key agreement protocols, such as MQV [104]. In this method, the key agreement function establishes a shared master key based on the short-term key of the nameserver (ω_l^i) and the ephemeral key of the nameserver (Λ_l^i), which is implicitly authenticated. Therefore, by employing an implicit, unilateral authenticated key agreement function (*i.e.,* where only the server-side is authenticated), inclusion of the ephemeral keys' signature in the responses becomes unnecessary. As a result, the additional parameters in DNSSEC+ responses can be reduced by ∼70 bytes, which further alleviates the amplification factor.

Notifying Clients. In the current design (Section 4.4) and implementation (Chapter 5) of DNSSEC+, no means have been defined to inform clients regarding successful use of DNSSEC+ in Stage 2. Similar to the AD flag (see Section 2.2.9) in DNSSEC, a DNS header bit can be defined for DNSSEC+ by which resolvers inform clients regarding effective implementation and use of DNSSEC+ in Stage 2. Thereby, if a recursive resolver is trusted by a client and Stage 1 is secured, a securely-communicated confirmation to the use of DNSSEC+ provides the client assurance that the name resolution process was DNSSEC+-protected in Stage 2.

Mitigating Query Flooding. Since DNSSEC+ is a UDP-based scheme without source IP address validation, nameservers are susceptible to query flooding, exhausting computational resources. Such attacks can be mitigated by rate-limiting techniques [45], forcing TCP use, or application-layer source IP address validation. Additionally, zone owners can use CDN instances for their nameservers [168], enabling more resilience against flooding attacks by distributing queries among nameservers.

Delegation in the Internet. In DNSSEC+, the short-term delegation of ANSes within a zone is analogous to delegated credentials [22]. These short-term

delegations mechanisms are useful in situations where a long-term secret owner does not fully trust all the servers hosting its service, and reduces the risk of attacks on the long-term secret. Short-term delegations also minimize the threat of key compromise, as they are implicitly revoked in short intervals, rendering them useless to the attackers after their expiry [33].

Downgrade Attacks. Similar to downgrade attacks on HTTPS, where the attacker forces a fallback to HTTP, and in which mitigations are implemented outside of TLS protocol (*e.g.,* HSTS [79]), attacks that block name resolution in DNSSEC+ and thereby downgrade DNSSEC+ to Vanilla DNS require mitigations outside of the DNSSEC+ protocol itself. We do not discuss this here as it is out of our current scope. Fallback to a less secure/private scheme is a deployment choice, which can be configured to favor security/privacy or incremental deployment with incremental benefits. Moreover, when fallback mechanisms are properly configured, passive adversaries who do not alter or block name resolution are unable to execute downgrade attacks. As a result, the security and privacy benefits of DNSSEC+ are preserved. We note, however, that DNSSEC+ is designed to fail closed (Section 3.2), thus mitigating *within-protocol* downgrade attacks [72].

Traffic Interception. Vanilla DNS traffic is transmitted in plaintext, which enables a wide range of network-based traffic analysis and debugging techniques that rely on the accessibility of plaintext DNS messages [153]. While the move to encrypted DNS enhances privacy, it also introduces certain trade-offs. Encrypted DNS can be abused by adversaries and malware to facilitate malicious activities such as data exfiltration [173] and botnet communication [131]. The shift to encrypted DNS is particularly concerning for stakeholders who depend on access to plaintext DNS traffic through software and hardware middleboxes for purposes such as network management and monitoring, troubleshooting and debugging, intrusion detection, and the identification of malware and data exfiltration.

The encryption of DNS messages in DNSSEC+ and the prevention of traffic processing within the DNS context are conceptually analogous to interception mechanisms observed in the TLS ecosystem [44]. Various techniques for traffic interception have been proposed for TLS-protected communications, which may offer valuable insights for addressing similar challenges in DNSSEC+. Readers are encouraged to read this work [44], as it provides a comprehensive analysis of traffic interception methods in the context of TLS communications.

Enhancing Deployability. To facilitate deployability, the trust model of DNSSEC+ could be extended to leverage DNSSEC's existing chain of trust for the authenticated retrieval of zone keys. This approach enables the distribution of DNSSEC+ keys as signed DNSSEC records, ensuring their authenticity within their respective zones. This approach enables a DNSSEC+ zone to remain compatible with a DNSSEC-enabled parent zone, thereby supporting incremental deployability without necessitating the immediate adoption of DNSSEC+ by all zones. Additionally, automation techniques such as those outlined in RFC 7344 [103] (Automating DNSSEC Delegation Trust Maintenance), facilitate key management and the continuous establishment and maintenance of the DNSSEC trust model. Given that DNSSEC+ employs a PKI and trust model similar to DNSSEC, one may reasonably believe that these automation mechanisms can be directly applied to support the deployment of DNSSEC+.

As reliance on third-party DNS infrastructure providers continues to grow, with approximately 20 million websites utilizing Cloudflare DNS for authoritative hosting and query resolution,[6] the adoption and deployment of DNSSEC+ by these providers could significantly facilitate and accelerate its widespread implementation on the ANS-side. In addition to ANSes, resolvers are also required to implement and use DNSSEC+. Given the increasing adoption of public resolvers across the Internet [139], encouraging their implementation and use of DNSSEC+ represents an effective strategy for promoting the widespread deployment and use of this scheme on a large scale. Since DNS infrastructure providers already possess the technical capacity and the ability to influence Internet users to adopt DNSSEC+ in Stage 2, they are well positioned to accelerate its initial deployment. In the absence of such involvement, the responsibility for implementing and configuring DNSSEC+ falls to individual zone administrators. As a result, the cost of adoption, whether computational, configurational, or financial, must be handled either by infrastructure providers or by zone owners and resolvers themselves.

With respect to scalability, while rigorous scalability testing is beyond our scope, DNSSEC+ introduces no new significant overhead. Modern computing systems are sufficiently powerful to handle its requirements, and cryptographic primitives are expected to become more efficient over time.

Furthermore, to improve deployability, the key server does not need to be hosted

[6] `https://trends.builtwith.com/ns/Cloudflare-DNS`

on a dedicated machine; instead, a zone's primary nameserver can serve as the key server, thereby reducing the overhead associated with deployment. Moreover, DNSSEC+ will be installed as a software and operates in the user-space of the resolvers and ANSes without requiring kernel or firmware updates. The objective is not to replace DNSSEC in Stage 2, but rather to position DNSSEC+ as a secure Stage 2 DNS scheme that can benefit from the current deployment of DNSSEC. Finally, we acknowledge that certain elements of the DNSSEC design (*e.g.,* key management and the establishment of a trust model), which are also used in DNSSEC+ with modifications, may result in similar resistance to the adoption of DNSSEC+.

4.8 Summary

After identifying the lack of a widely adopted secure DNS scheme in Stage 2 in Chapter 3, this chapter introduced the properties and design goals of DNSSEC+, based on the advantages and limitations of DNSSEC. Subsequently, we outlined the new processes and entities introduced on the zone side, as well as the required modifications to resolvers for DNSSEC+ implementation. Additionally, we detailed the name resolution process in both non-privacy and privacy-enforcing modes. In Chapter 5, we present the implementation of a DNSSEC+ prototype and conduct an evaluation of its security and privacy properties alongside computational and time performance in comparison with previously proposed secure DNS schemes.

Chapter 5

Implementation and Evaluation of DNSSEC+

In this chapter, we begin with an analysis of the amplification factor in DNSSEC+, followed by comparing the properties of DNSSEC+ with DNSSEC and DNSCurve. Subsequently, a prototype implementation of DNSSEC+ is presented, and then used for evaluating its performance across key metrics: server-side processing latency, resolution time, and CPU usage. We conducted a comparative analysis of DNSSEC+ with other schemes, including TCP-based DNS [119, 120], live-DNSSEC [53], and DoT [86]. The results of our evaluation indicate that, in terms of total name resolution time, DNSSEC+ achieves performance comparable to previously proposed, and also less secure DNS schemes, with an acceptable overhead on the server side (an unavoidable consequence of the additional cryptographic operations). Furthermore, DNSSEC+ demonstrates a significantly better performance compared to DoT in total name resolution time and server-side CPU utilization.

5.1 Comparative Evaluation of DNSSEC+

In this section, we begin by examining the susceptibility of DNSSEC+ to amplification attacks by conducting a comparative analysis of its amplification factor compared to DNSSEC. Subsequently, we evaluate the security and privacy properties of DNSSEC+ compared to DNSSEC and DNSCurve, the two schemes that have been proposed and implemented in Stage 2. Using the evaluation framework developed in Chapter 3, we demonstrate the enhanced properties provided by DNSSEC+ in comparison with DNSSEC and DNSCurve.

5.1.1 Amplification Factor

As explained in Section 4.1, it is crucial for DNSSEC+ to resolve queries in a single round-trip. There are trade-offs associated with a single round-trip, and amplification is one of the important aspects to consider. One of the schemes with a notable bad reputation regarding traffic amplification in Stage 2 is DNSSEC. Although the amplification factor in DNSSEC can theoretically exceed 100×, the empirically ob-

served average amplification factor for queries of type `ANY` for TLDs in DNSSEC 2014 was approximately 47× [162]. The queries of type `ANY` often result in a greater amplification factor. When an attacker abuses `ANY` queries to target a DNSSEC-enabled nameserver, the nameserver returns any type of resource records available on the nameserver for the given domain name in response. In a DNSSEC-protected zone, in addition to the resource records, the nameserver also returns the RRSIGs associated with each resource record. Therefore, relative to the number of resource records included in the response, a DNSSEC-enabled server returns RRSIGs, which results in a greater amplification of traffic.

In DNSSEC+, regardless of the DNS record type and the number of records in the response, the number of bytes added to the response for encryption and authentication are constant (Figure 5.1). The reason is that, unlike DNSSEC, for each DNS record a separate signature is not required. Consequently, the amplification factor in DNSSEC+ is restricted and cannot be abused for considerable amplifications in DDoS attacks. With Elliptic Curve Digital Signatures (ECDSA) and NaCl cryptography [24], the number of additional bytes for a non-delegating response is ∼245 bytes and for a delegating response ∼310 (see Sec. 5.2.1). The number of added bytes by DNSSEC+ can be further decreased (Section 4.7). Compared to DNSSEC, which can possibly add thousands of bytes to the response of a query of type `ANY`, with DNSSEC+ only a limited number of bytes are added to each response for authentication and encryption.

5.1.2 Comparative Analysis: DNSSEC, DNSCurve

Compared to DNSSEC, which only provides message authentication to DNS responses, DNSSEC+ provides real-time authenticated encryption for encrypting DNS queries and responses, thereby providing both confidentiality and message authentication. Therefore, DNSSEC+ does not require NSEC-like records [13, 14, 67] for negative responses, and regular NXDOMAIN responses can be transmitted securely. As explained in Section 5.1.1, compared to DNSSEC, which is susceptible to significant traffic amplification rates, responses in DNSSEC+ only contain a fixed number of additional bytes. Besides, in DNSSEC the responses are susceptible to be captured and replayed by an adversary, within the lifetime of their signature. However, due to the use of ephemeral agreement keys, the DNS messages in DNSSEC+ cannot be replayed between different sessions. Moreover, the added TVP introduces fresh-

ness to the messages within a session. Consequently, if more than one query is sent with the same ephemeral key, the queries or responses cannot be replayed within the same session. Another difference between DNSSEC and DNSSEC+ is that DNSSEC requires separate queries to obtain the DNSKEY records from a zone's nameservers. Although both queries can be transmitted simultaneously and the delay would remain the same, in DNSSEC+ the keys are appended as part of the response and one less query is required. Finally, DNSSEC requires modifications to the zone files, while in DNSSEC+ the zone files remain unchanged, which saves the administrative time of updating zone signatures.

Now consider DNSCurve [25, 46] in the context of key management. DNSCurve lacks a specified mechanism for distributing nameserver keys among anycast instances, in cases of anycasting. Furthermore, it requires long-term keys to be stored on nameserver instances, making them exposed to attacks. To address these issues, DNSSEC+ introduces a delegation approach where a key server within the zone signs short-term key structures for nameserver instances. DNSSEC+ thus avoids duplicating long-term secrets, and provides means for distributing the keys of nameserver instances within a zone.

DNSCurve [46] does not provide forward secrecy to queries nor to responses, because the ANS's public key is not ephemeral. DNSSEC+ implements a half-static DH approach for queries to enable DNS resolution within a single round-trip, thus, also not providing forward secrecy for queries. However, DNSSEC+ provides forward secrecy for responses.

Finally, DNSCurve does not provide a chain of trust in the DNS hierarchy. Thus, resolvers cannot validate the authenticity of an NS record that contains a public key, rendering DNSCurve susceptible to false nameserver injections [145, 168]. In DNSSEC+, the long-term keys of ANSes establish a DNSSEC-like chain of trust up to the root, making it resilient to the aforementioned attack.

5.1.3 Overall Evaluation

In addition to the amplification analysis (Section 5.1.1) and the examination of the properties and design choices between DNSSEC+ and DNSSEC and DNSCurve (Section 5.1.2), Table 5.1 provides a comparative overview of the security, privacy, and availability properties of DNSSEC+ in comparison with DNSSEC (live and offline) and DNSCurve, using our evaluation framework from Chapter 3. This eval-

Table 5.1: Comparative evaluation of DNSSEC+ with secure DNS schemes in Stage 2

DNS Scheme	*Reference*	*S1. Resilient-to-False-ANS-Response*	*S2. Resilient-to-ANS-Replay-Attack*	*S3. Avoid-Duplicating-Longterm-Secret*	*A1. Resilient-to-ANS-DoS*	*P1. Resilient-to-Eavesdropping-in-s2*	*P2. Hides-Client-IPaddr-in-s2*
Live DNSSEC	[53]	●	○	○	○	○	○
DNSSEC	[11, 12, 13]	●	○	●	○	○	○
DNSCurve	[25, 46]	●	●	○	○	●	◐
DNSSEC+	[92]	●	●	●	○	●	◐

uation highlights that DNSSEC+ offers enhanced security and privacy properties compared to DNSSEC and delivers stronger security properties than DNSCurve. Although deployability properties were excluded from the scope of our evaluation framework in Chapter 3, the *established trust-model* was another deployability property that was considered and satisfied in both DNSSEC and DNSSEC+, while not satisfied by DNSCurve.

As demonstrated in Table 5.1, regarding availability, none of the Stage 2 schemes, including DNSSEC+, satisfy the *Resilient-to-ANS-DoS* property because they rely on UDP-based communication without application-layer mechanisms for source address validation. Implementing source address validation to enhance resilience against ANS DoS attacks requires either an additional round-trip, which contradicts the single round-trip design objective of DNSSEC+, or the maintenance of additional state information, leading to increased resource consumption and added protocol complexity.

For ANSes with UDP-based schemes in Stage 2, using the distributed infrastructure and defense mechanisms provided by CDNs can serve as a compensatory measure against ANS DoS attacks. Additionally, implementing rate limiting mecha-

nisms helps mitigate reflection attacks targeting specific entities. Similar to Vanilla DNS and DNSSEC, DNSSEC+ requires the enforcement of TCP once response sizes exceed a specified threshold, thereby establishing an upper bound for the amplification factor and ensuring reliable response transmission.

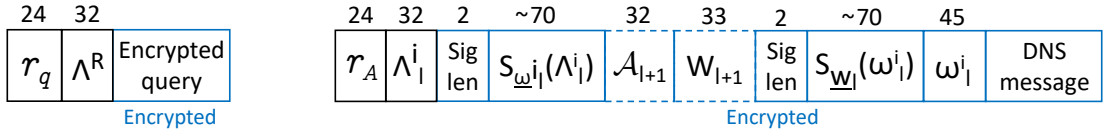
5.2 Implementation and Performance Evaluation

We built a proof-of-concept prototype of DNSSEC+ for testing its performance in practice, and for comparison with other DNS schemes. The prototype consists of two parts: ANS-side and resolver-side.

5.2.1 Prototype Implementation

ANS-side. So as to implement the encryption and decryption functions, we modified the DNS library used in `CoreDNS` [65]. As demonstrated in Figure 5.1 (b), the nameserver adds its short-term public key structure (ω_l^i) with its signature generated by the zone's key server ($S_{\underline{\mathrm{w}}_l}(\omega_l^i)$). Additionally, the signature of the ephemeral public agreement key ($S_{\underline{\omega}_l^i}(\Lambda_l^i)$) is added to the response prior to encryption. Finally, the public ephemeral key of the nameserver (Λ_l^i), and the random number (r_A) used to encrypt the response are added to the response. The dashed boxes (Figure 5.1 (b)) represent the long-term keys associated with the child zone ($\mathcal{A}_{l+1}$, w_{l+1}), and are added when the response is referring to a delegated zone. In responses to the queries for which a nameserver is authoritative, the dashed boxes are omitted. We used ECDSA with curve P-256 and SHA256 [66] for signing and verifying the signatures, and NaCl [24] library for authenticated encryption.

Resolver-side. The resolver encrypts the DNS queries (Figure 5.1 (a)) using NaCl-based [24] authenticated encryption and sends the encrypted queries alongside the freshly generated ephemeral key of the resolver (Λ^R) and the random number (r_q), which were used to encrypt the query. Upon receiving an encrypted DNSSEC+ response, the resolver extracts the random number and public key from the message and decrypts the encrypted part. Subsequently, the resolver parses and extracts the included keys, signatures, and the DNS message from the response. The resolver initially verifies the included digital signatures of the short-term key structure of the nameserver and the ephemeral key of the nameserver. If the verification process



(a) Query (b) Response

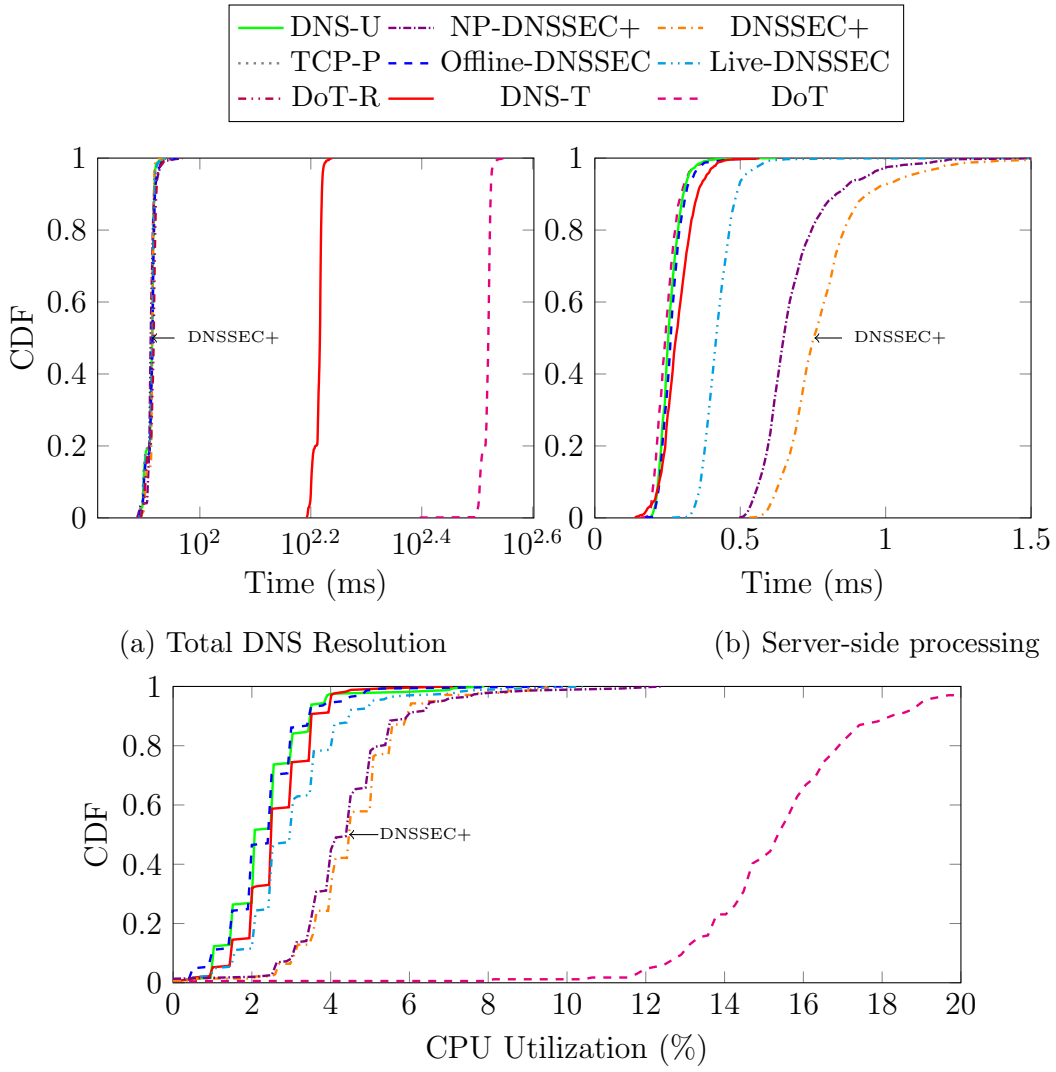
Figure 5.1: Query and response format in DNSSEC+ (the dashed boxes are only included in delegating responses). The numbers on top represent bytes.

succeeds, the resolver proceeds to process the DNS response; otherwise, the response is discarded. To implement the resolver-side in DNSSEC+, we modified `q` [64], which is a similar DNS resolution tool to `dig`, implemented in Go.

5.2.2 Performance Evaluation

In this section, we test our prototype implementation against several other DNS schemes, focusing on three performance parameters: server-side processing time (and how it is affected with various Link-Layer Maximum Transmission Unit (MTU) values), total resolution time, and CPU utilization. We setup the study using two VMs deployed on cloud servers, and acknowledge that such studies typically serve as a preliminary evaluation of expected performance in practice. They do not replace performance indicators obtained from a real-world, Internet-scale deployment of the scheme in question. An Internet-scale evaluation is beyond the scope of this thesis. Nonetheless, our preliminary experimental setup provides initial confidence in the relative performance of DNSSEC+ compared to existing schemes. While absolute performance measurements are not pursued, our CPU utilization analysis focuses on demonstrating the relative computational overhead of DNSSEC+. Evaluating scalability bottlenecks or the impact of DoS attacks on servers falls outside the current scope and is left for future work.

Server-side Processing: We used two cloud servers in the United States, each equipped with a 2-core (2.4 GHz) CPU and 2GB of RAM, running Ubuntu 20.04. Figure 5.2b illustrates the CDFs of server-side processing times for five DNS schemes compared to DNSSEC+ in both privacy modes. The processing time is measured as the interval between the arrival of the last DNS query datagram fragment at the ANS and the departure of the first response fragment, captured at the Link layer. While server-side processing latencies include some noise from process switching, this was considered negligible as it mainly affects the upper end of the CDFs in all



(a) Total DNS Resolution

(b) Server-side processing

(c) CPU utilization (120 QPS)

Figure 5.2: Total DNS resolution time, server-side processing delay, and CPU utilization of different schemes. UDP-based DNS (DNS-U), TCP-based DNS (DNS-T), TCP-P (Persistent), DoT-R (Resumption), NP-DNSSEC+ (No privacy). Figures (b) and (c) do not include TCP-P and DoT-R (see inline for explanation).

the schemes that were analyzed.

The results of 1,000 queries show that DNSSEC with live signing requires over 65% more server-side processing time than UDP-based DNS at the 80th percentile. Despite the cryptographic overhead, DNSSEC+ processes over 90% of queries in under one millisecond, adding approximately half a millisecond to the server-side processing time compared to UDP-based DNS. In DNSSEC+ without privacy, the

absence of query decryption reduces server-side processing time by around 0.1ms compared to the privacy-enforcing mode. Performance of DNSSEC+ can be further improved by reusing ephemeral keys for multiple queries within a short time, with minimal impact on forward secrecy. Aside from Live DNSSEC and DNSSEC+, all other schemes exhibit similar server-side processing latency. Moreover, server-side processing times for TCP with persistent connections (TCP-P) and TLS with session resumption (DoT-R) are expected to be equivalent to standard TCP (DNS-T) and DoT, as they measure query processing time after session establishments. Thus, they are excluded from Figure 5.2b.

Total DNS Resolution Latency: Total DNS resolution latency was measured between both VMs, with an average round-trip time (RTT) of 81ms, based on 35,000 RTT measurements, and a standard deviation of 0.131ms. As shown in Fig. 5.2a, UDP-based DNS, DNSSEC, and DNSSEC with live signing have similar total latencies of approximately 82ms. Since most DNSSEC+ queries are processed in under 1ms on the server-side, the total resolution latency for DNSSEC+ is similar to UDP-based DNS and DNSSEC, with a negligible increase. Based on the DNSSEC+ measurements obtained within a single zone, it can be asserted that the overhead associated with resolving records through the DNS hierarchy is also minimal. Thus, the impact of DNSSEC+ on the end-users' experience of DNS resolution is negligible, as the introduced additional latency is minimal. Due to the TCP handshake, TCP-based DNS results in double the latency of single round-trip schemes, while DoT is four times slower due to the extra overhead of the TLS handshake. The TLS termination messages (FIN, ACK, RST) are excluded from this calculation. If termination packets were included, DoT would be five times slower than UDP-based DNS. Although in-code timestamping for TCP-P and DoT-R was not included to measure server-side processing (Fig. 5.2b) and utilization (Fig. 5.2c), we measured the resolution time for these schemes (Fig. 5.2a) by using the timestamps of query packet departures and response packet arrivals at the resolver.

CPU Utilization: Figure 5.2c presents the CPU utilization of different DNS schemes when a multi-threaded program sends 20,000 queries at a rate of ∼120 queries per second. At the 90th percentile, UDP-based DNS and offline-DNSSEC exhibit identical CPU utilization, while TCP-based DNS demonstrates approximately 0.5% higher utilization. Live DNSSEC shows an increase of 1% in CPU usage compared to UDP-based DNS at this percentile. DNSSEC+ in both modes consumes

approximately 2.5% more CPU. Notably, DoT without session resumption requires around five times more CPU than UDP-based DNS and three times more than DNSSEC+ in both privacy modes at the 90th percentile.

Effect of Network-Layer Fragmentation: Because DNSSEC+ messages are typically longer than those of Vanilla DNS and DNSSEC, we examined the effect of network-layer fragmentation, particularly for messages exceeding the MTU, on the server-side processing time in all three schemes. We used four MTU values: 1450, 1000, 500, and 200.

For larger MTU values (Figures 5.3a, 5.3b), where fragmentation is less likely, the CDFs of Vanilla DNS and DNSSEC largely overlap, and DNSSEC+ similarly experiences no fragmentation. When the MTU is set to 200B (Figure 5.3d), both DNSSEC and Vanilla DNS are prone to fragmentation, showing similar patterns, and DNSSEC+ experiences ∼0.03 ms increase at 90th percentile. However, at 500B (Fig. 5.3c), Vanilla DNS experiences less fragmentation, while DNSSEC remains mostly fragmented, resulting in slightly slower performance for DNSSEC. Overall, the impact of fragmentation on server-side processing time is negligible.

5.2.3 Server-side Time Breakdown

To have a better understanding of the server-side processing latency in DNSSEC+, Figure 5.4 presents a detailed breakdown of the processing times for different server-side operations in the privacy-enforcing mode. The results are the average values upon resolving 1,000 queries. In the privacy-enforcing mode of DNSSEC+, upon receiving a query, the ANS must first decrypt the query. The total decryption time is 0.1 ms, which is made up of two components: symmetric authenticated decryption (0.08 ms) and query preparation (0.02 ms). The query preparation process involves parsing the query and preparing the decryption key.

Upon decrypting the query and looking up for the corresponding response, the ANS in DNSSEC+ must encrypt the response. The total encryption process requires 0.3 ms. This time is composed of generating an ephemeral agreement key and a nonce (0.08 ms), signing the ephemeral key using the nameserver's short-term signing key (0.11 ms), symmetric encryption of the response (0.08 ms), and preparing the response (0.01 ms). The response preparation involves looking up for the response and reading the cryptographic keys from memory, which are used in generating the response. An additional 0.02 ms is spent on the time measurement

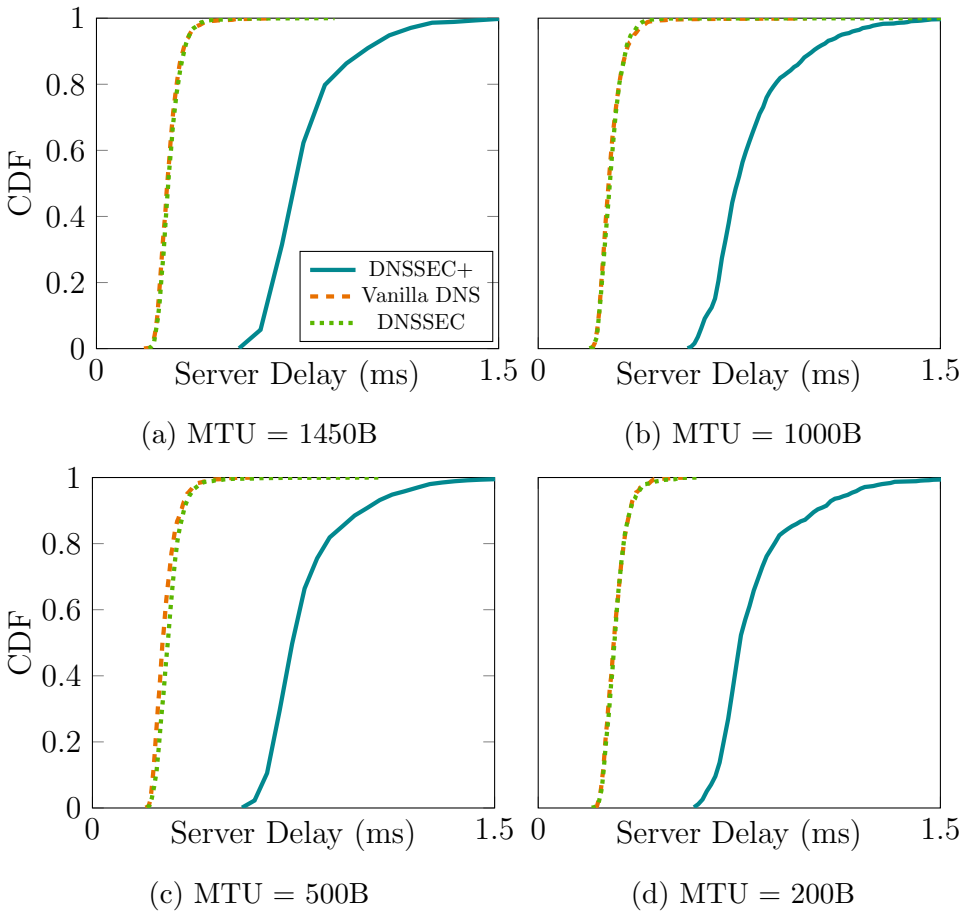


Figure 5.3: Server-side processing delay for DNSSEC+, Vanilla DNS, and DNSSEC in various MTUs.

functions and the execution of server-side functions within the code.

5.2.4 Takeaways

We evaluated the server-side processing latency and the total DNS resolution time using a single-zone prototype. Based on these measurements, we expect that the performance across other zones will have similar results when traversing the DNS hierarchy. Although the server-side processing latency of DNSSEC+ is relatively higher compared to other evaluated secure DNS schemes, it remains < 1 millisecond for the majority of queries. The total resolution time of DNSSEC+ is comparable to that of less secure schemes, as the server-side processing latency is overshadowed by network delays in the total DNS resolution time [40, 115]. Moreover, in cases of

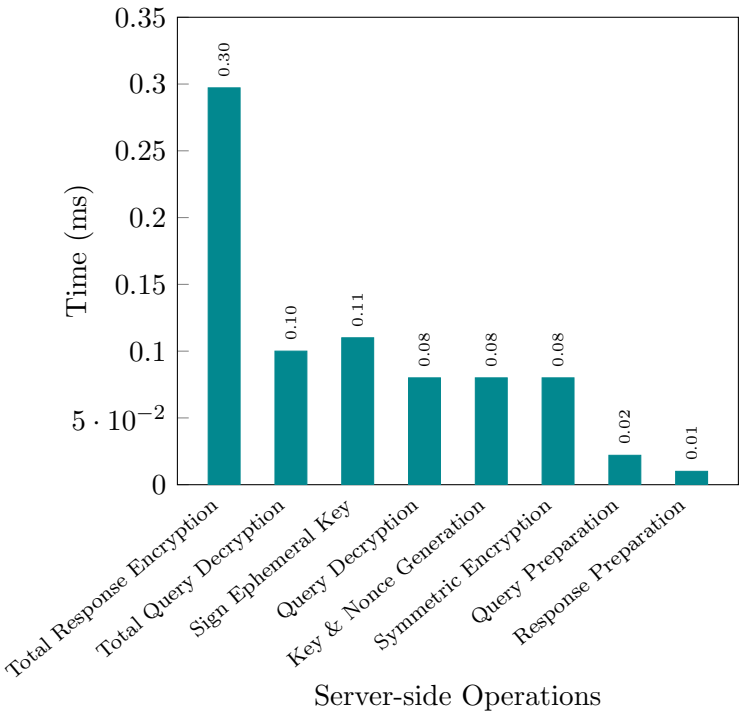


Figure 5.4: Breakdown of server-side processing latency for cryptographic operations

fragmentation of large responses in DNSSEC+, we showed that fragmentation has a negligible effect on server-side processing time. Finally, DNSSEC+ demonstrates CPU utilization comparable to that of less secure schemes, and significantly lower than DoT.

5.3 Summary

Following the design and proposal of DNSSEC+ in Chapter 4, this chapter examines traffic amplification in DNSSEC+ compared to DNSSEC. Additionally, we discussed the comparative advantages of DNSSEC+ over DNSSEC and DNSCurve. Performance evaluation results indicate that DNSSEC+ has a negligible impact on the total DNS resolution time. Moreover, server-side processing time and CPU utilization for DNSSEC+ are not significantly higher than those of other secure schemes, demonstrating its feasibility for deployment. Finally, it is shown that the effect of fragmentation on server-side latency is minimal, with almost no impact on total name resolution latency. In Chapter 6, we will develop a symbolic model

of DNSSEC+ and formally verify the defined security and privacy properties to establish confidence in the correctness of the properties.

Chapter 6

Formal Analysis of DNSSEC+

As described in Chapter 4, DNSSEC+ is designed to augment real-time security and privacy properties in Stage 2 [92]. Use of DNSSEC+ in Stage 2 effectively mitigates vulnerabilities within this stage, including cache poisoning [43, 74, 98], replay attacks using stale responses [15, 25, 71], eavesdropping [18], and large-scale surveillance [70]. DNSSEC+ utilizes a delegation mechanism to mitigate the risk of exposure of a zone's long-term key to attacks by avoiding the need to replicate this key throughout the ANSes of the zone. Additionally, it secures transmission of DNS messages by employing symmetric authenticated encryption, with encryption keys generated using DH key agreement.

The security and privacy properties of DNSSEC+ require formal analysis to validate their fulfillment. To do so, we develop a symbolic model of the protocol using the Tamarin Prover [117] (Section 2.4.1). Security and privacy properties of DNSSEC+ are then formally proven using this model. Finally, the model is used to assess the impact of key and entity compromises, demonstrating the consequences of such compromises on these properties. Our analysis shows that all of the originally defined properties of DNSSEC+ hold as expected when none of the keys are compromised. Moreover, compromise of any single long-term private key within the protocol does not, in isolation, affect the validity of the security and privacy properties of responses.

6.1 Stated Properties and Assumptions

This section begins by listing the design properties of DNSSEC+ and specifying the subset related to security and privacy, which are formally verified by Tamarin in this chapter. Next, the underlying assumptions and threat model used in the development of the symbolic model for DNSSEC+ and the proof of its properties are defined.

6.1.1 Stated Properties of DNSSEC+

In Chapter 4, nine properties were defined to be fulfilled by DNSSEC+. Table 6.1 summarizes these, and highlights the subset of security and privacy properties, formally proven herein using Tamarin. DNSSEC+ encrypts both queries and responses to achieve *message confidentiality*. Additionally, the response encryption keys are freshly generated, with the objective of providing *forward secrecy*. The next security property that DNSSEC+ provides is *entity authentication* for the nameservers, and guarantees that the transmitted responses are both authenticated and also *resilient to replay attacks*. The defined entity authentication in DNSSEC+ is unilateral, as only the nameservers authenticate themselves to the resolvers; the resolvers do not authenticate themselves to the nameservers.

For the remaining properties, Chapter 4 demonstrated how the DNSSEC+ design satisfies the objectives of *avoiding duplication of long-term secrets* through the secure delegation process, achieving *single round-trip* DNS resolution, and employing an *established trust model*. The property of *minimum amplification* is satisfied by the design of messages which maintain a constant amplification factor compared to DNSSEC (see Section 5.1.1). Furthermore, in Chapter 4 we presented the *fail-closed* objective as a procedural design choice aimed at mitigating within-protocol downgrade attacks in DNSSEC+.

DNSSEC+ Property	Target	notes
Message confidentiality	DNS queries/responses	Tamarin proof
Forward secrecy	DNS response keys	Tamarin proof
Entity authentication	Nameservers	Tamarin proof
Mitigate replay	DNS responses	Tamarin proof
Avoid duplicating LTK	DNS zones	by design
Single round-trip	DNS resolution	by design
Established trust model	DNS hierarchy	by design
Minimum amplification	DNS responses	by design
Fail-closed	DNS resolution process	by design

Table 6.1: Stated properties of DNSSEC+: the first four are supported by Tamarin proofs herein, while the remaining five were established by design (see Chapter 4).

6.1.2 Assumptions

As outlined in Chapter 4, there are three main components in DNSSEC+: the delegation of authorization to the nameservers within each zone, a trust model, and the DNS resolution process. The establishment of the trust model, in which the child zones below the root send their long-term public keys to their parent zone, takes place through a secure OOB channel. Thus, it is assumed that this process is securely completed prior to any name resolution. The delegation process, by signing the short-term keys within a zone and sending them to the nameservers, is also conducted through a confidential and authenticated OOB channel established between the zone's key server and its nameservers. The delegation process must be completed prior to the initiation of any name resolution. The security and privacy properties defined in Table 6.1 (*i.e.,* first four properties) are for the DNS resolution process in Stage 2, and the preceding processes of trust model establishment and delegation to nameservers are assumed to be completed via their secure OOB channels.

We model the communication between the nameservers and key servers for the delegation process within zones as secure OOB channels whose messages the adversary (in Tamarin) cannot access. Additionally, we assume that there is a trust model established between the zones within the DNS hierarchy through the authenticated OOB channel between the zones. An example of such channel in DNSSEC is the use of HTTPS-based communications via domain registrars for establishing the chain of trust. Using these two assumptions, we formally verify the properties of the DNS resolution process in DNSSEC+ using Tamarin.

In our model, we limit the name resolution to the root zone and do not continue the resolution down the DNS hierarchy. By proving the security and privacy properties for DNS resolution with a root's nameserver, it is assumed that, in DNSSEC+, the resolver obtains the child (*i.e.,* TLD) zone's authentic long-term public keys from the root. The long-term public keys of the child zone are assumed to be transmitted authentically to the root during the trust model establishment phase. Thus, in turn, the formal verification of the name resolution for the TLD zone would be similar to the root, but this time with the TLD's long-term keys. In this manner, proving the properties of DNS resolution with the root zone extends the same properties for subsequent resolutions down the DNS hierarchy, assuming authentic trust model establishment.

As outlined in Section 4.4, DNSSEC+ operates in two modes: privacy-enforcing

and no-privacy. In the privacy-enforcing mode, DNS queries are encrypted to ensure confidentiality. In the no-privacy mode, queries remain unencrypted. Responses are both authenticated and encrypted in both modes to maintain data authenticity and confidentiality. We model and prove the security and privacy properties of DNSSEC+ in the privacy-enforcing mode for which we can also prove the secrecy of queries. Moreover, in Section 6.4, where we show the implications of key compromises, we demonstrate that the response properties remain verifiable if query secrecy is compromised (*e.g.,* when `LTAK` is compromised), which is an analogous scenario to the no-privacy mode (where queries are in plaintext).

In addition to the above protocol-related assumptions, in symbolic analysis we exclude computational attacks and primarily focus on the interactions and abstract operation of the protocol. Consequently, symbolic analysis inherently incorporates certain assumptions as part of its abstraction. The first assumption is that the cryptographic primitives within the model are perfect. For example, an adversary can generate a valid signature if and only if it knows the signing private key. Freshly generated random terms, such as nonces and cryptographic keys, are assumed to be unpredictable and unique. Since the information in the symbolic model is abstracted to terms, an entity, including the adversary, either has complete knowledge of a term or no knowledge at all, with no possibility of having partial knowledge of the term.

Common threats in Stage 2 of the DNS resolution include inline adversaries—between a recursive resolver and ANSes. These can be located within distinct Autonomous Systems (ASes), each maintained by different administrative policies. Such entities have access to DNS messages and can read, modify, delete, and inject fabricated messages. Additionally, off-path adversaries can send queries to a recursive resolver and inject false responses, thereby poisoning the cache of the resolver [98][74]. In Section 6.3, we focus on proving the properties of DNSSEC+ against both in-line and off-path network-based attacks, and we use the default Dolev-Yao [54] attacker in the Tamarin prover. In Section 6.4, in order to analyze the impact of host and key compromises on the properties, we model host-based attacks by adding the private keys of DNSSEC+ to the adversary's knowledge and analyzing how it affects the verified properties.

6.2 Modelling DNSSEC+ in Tamarin

In this section, we develop a symbolic model of the delegation process and DNS resolution in DNSSEC+, using Tamarin rules.

6.2.1 Secure Delegation

We invite the readers to review the entire process of secure delegation within DNSSEC+ zones in Figure 4.2 of Chapter 4. To model the DNSSEC+ delegation process in Tamarin, we use four Tamarin rules:

D1. The initialization of a key server within the root zone, responsible for generating and securely storing the long-term keys of the zone (*i.e.,* LTSK and LTAK).

D2. The initialization of the zone's ANS instances and their respective short-term signing keys (STSKs), followed by the transmission of these STSKs to the key server via an OOB channel modeled in Tamarin, which assumes that the messages are transferred over a channel not accessible to the adversary.

D3. Receiving the nameserver's STSKs by the initialized key server in Step 1, which then uses the zone's LTSK to sign the STSKs. The signed STSKs are subsequently returned to the nameservers through a secure OOB channel.

D4. Finally, the nameservers receive their signed STSK, enabling their use in the DNS resolution process for responding to queries.

Listing 6.1 shows Step 1 of the secure delegation process. In this step, a key server is initialized with two fresh long-term keys (Lines 2-3): a long-term signing key (`ltkKS`) and a long-term agreement key (`ltkKA`) in the premise. In the conclusion, these keys are added to the state as permanent facts (Lines 6-9) to be used as premises of the next rules. Additionally, the public components of the long-term keys are sent to the network and added to the adversary's knowledge using the `Out()` fact in Line 10. The action fact `OnlyOnce()` is used to restrict the protocol traces, so that at most one key server being initialized. The `KeyServerInit()` action fact is used in executability lemmas for ensuring the proper execution of the protocol and correctness of proofs.

Listing 6.1: Key Server Initialization

```
1   rule Key_Server_Init:
2          [ Fr(~ltkKS) // long-term signing key
3          , Fr(~ltkKA) ] // long-term agreement key
4          --[OnlyOnce('Key_Server_Init')
5          , KeyServerInit($K, ~ltkKS, pk(~ltkKS))]->
6          [ !ZKS_Sk($K, ~ltkKS)
7          , !ZKS_Pk($K, pk(~ltkKS))
8          , !ZKA_Sk($K, ~ltkKA)
9          , !ZKA_Pk($K, 'g' ^ ~ltkKA)
10         , Out(<pk(~ltkKS), 'g' ^ ~ltkKA>)]
```

In the next rule, demonstrated in Listing 6.2, a nameserver is initialized with a fresh ID (`nid`) and fresh short-term signing key (`stsk`) in the premise.

Listing 6.2: Nameserver Requests Signing

```
1   rule NS_Sign_Request:
2       [ Fr(~stsk) // short-term signing key
3       , Fr(~nid) ] // nameserver ID
4       --[ NS_Sign_Req($N, ~nid, ~stsk) ]->
5       [ NS_Wait_Sig(< $N, ~nid, ~stsk, pk(~stsk)>)
6       , NS_Sends_to_Sign(<$N, ~nid, pk(~stsk)>)] // OOB channel
```

The action fact `NS_Sign_Req()` is added to verify correct execution of the protocol. In the conclusion (lines 5-6, Listing 6.2), the `NS_Wait_Sig()` fact is used to store the current state and terms and wait for the key server to sign the nameserver's short-term signing key. Finally, the `NS_Sends_to_Sign()` fact is used to model sending the public key of the nameserver through a secure OOB channel to the key server to be signed. Unlike the rule for key server initialization (Listing 6.1), this rule can be executed more than once, as the number of the nameservers within a zone is not limited.

The next rule models the key server signing the short-term key of a nameserver. As Listing 6.3 shows, the key server has the long-term private signing key of the zone (`!ZKS_SK()`) and receives the sign request from the nameserver through an OOB channel (`NS_Sends_to_Sign()`) as the premise. In the conclusion (line 5), the signed short-term signing key of the nameserver is added to the state by `Signed_NS_key()` fact. This fact is assumed to be received by the nameserver through a secure OOB

channel. The KS_Signs() is used in the executability lemmas.

Listing 6.3: Key Server Signs

```
1  rule Key_Server_Signs_NS:
2         [ !ZKS_Sk($K, ltkKS)
3         , NS_Sends_to_Sign(< $N, nid, stpk >)]
4         --[ KS_Signs($N, nid, stpk, sign(<$N, nid, stpk>, ltkKS)) ]->
5         [ Signed_NS_key(sign(<$N, nid, stpk>, ltkKS), <$N, nid, stpk> )]
```

In the final step of the delegation process, the nameserver receives the signed short-term key structure from the key server and adds it to the state. As demonstrated in the premises of Listing 6.4, the nameserver state fact NS_Wait_Sig is used here as a premise to ensure that this rule is preceded by the NS_Sign_Request rule from Listing 6.2. The nameserver receives the signed key structure from the key server through an OOB channel using the Signed_NS_key() fact from the previous rule (Listing 6.3). The !ZKS_PK() contains the public component of the zone's long-term signing key and added as a premise for verifying the signature generated by the key server. In the action facts, the equality restriction (Eq()) is used for verifying the signature of the short-term key structure, and the NS_Rcv_Signed() is used in executability lemmas. In line 5, the rule verifies that the signature received from the OOB channel is generated by the zone's long-term signing key.

Listing 6.4: Nameserver Receives Signed

```
1  rule NS_Receives_Signed:
2      [ NS_Wait_Sig(< $N, nid, stsk, stpk >)
3      , Signed_NS_key(sig_stpk, <$N, nid, stpk> )
4      , !ZKS_Pk($K, pkKS)]
5      --[ Eq(verify(sig_stpk, <$N, nid, stpk> , pkKS), true)
6      , NS_Rcv_Signed(sig_stpk, <$N, nid, stpk>)
7      ]->
8      [
9        !NS_sstk_Signed_PK(sig_stpk, <$N, nid, stpk>)
10     , !NS_sstk_Signed_SK(sig_stpk, <$N, nid, stpk>, stsk)
11     , Out(< stpk, sig_stpk >)]
```

In the conclusion, the signature of the nameserver with the short-term public and private signing key are added to the state as permanent facts (Lines 8-10), which

will be used in the name resolution process for signing the ephemeral keys by the nameserver. Furthermore, the nameserver's short-term public key and the signature of the short-term key structure are sent to the network and added to the adversary's knowledge.

6.2.2 DNS Resolution

After completion of the delegation process, the zone's long-term signing and agreement keys are initialized, and the nameserver's signing keys are signed by the key server. At this point, resolvers can use the zone's long-term agreement key to securely transmit queries to the nameserver instances of the zone. The process of DNS resolution occurs between a resolver and a zone's nameserver instance. Assuming that the queried zone is the root, the public long-term keys of the root zone are securely included into the resolver software, as trust anchors, through secure means. The DNS resolution process is modeled using three Tamarin rules demonstrated in Listings 6.5 to 6.7.

The rule `Res_1`, Listing 6.5, models a resolver sending an encrypted query. In the premises, the fact `!ZKA_Pk()` holds the root zones long-term agreement public key (`ZKAP`). Incorporating this fact in the premise models the insertion of the root zone's long-term public agreement key as trust anchor in the resolver's software. The next three premises generate three fresh values: `ephR`, representing the resolver's ephemeral private agreement key; `query`, denoting the query data including domain name and *qid*; and `Rnc`, indicating the resolver's nonce. In this rule, the query is modeled as a fresh value which represents the entire query, which is unknown to the adversary. In the action facts, `QKeySecret()` is here to be used for proving the secrecy of session keys used for the authenticated encryption of queries. Additionally, `QDataSecret()` action fact is included here for proving the secrecy of query data. Both these action facts will be used when we are defining the query key and data secrecy properties. The `ResolverSendQuery()` is another action fact used in the executability lemmas for ensuring proper execution of the protocol model. `Role('R')` action fact is used for distinguishing the role of the resolver from the nameserver in the proofs.

In the conclusion, the nonce (`Rnc`), the ephemeral public agreement key of the resolver (`'g'^ephR`) and the symmetrically encrypted query are sent out to the network using the `Out()` fact. The query encryption key is generated using a KDF

Listing 6.5: Resolver Sends Query

```
rule Res_1:
    let
      queryKey = kdf(< ~Rnc, ZKAP ^ ~ephR >)
      query_data = < ~query, 'Query' >
    in
    [ !ZKA_Pk($K, ZKAP) // public long-term key of the root zone
    , Fr(~ephR) // resolver's ephemeral key
    , Fr(~query) // query
    , Fr(~Rnc) ] // resolver's nonce
    --[
      ResolverSentQuery($R, ~qid, ~ephR)
    , Role('R')
    , QKeySecret($R, ~ephR, queryKey)
    , QDataSecret($R, query_data)
    , OUT_R_1(<~Rnc, 'g' ^ ~ephR, senc(query_data, queryKey)>)
    ]->
    [ Out( <~Rnc, 'g' ^ ~ephR, senc(query_data, queryKey)> )
    , Res_State_1($R, ~ephR, ~qid, ~Rnc)
    , !EskR($R, ~ephR) ]
```

that takes the nonce and a DH established key. The `Res_State_1()` fact stores the query state and maintains the order of executed rules. This fact will be used in the premise of the rule wherein the resolver receives the response.

As demonstrated in the next rule (Listing 6.6), we model a nameserver that receives an encrypted query and sends an encrypted response back to the resolver. In the premises of this rule, the fact `!ZKA_Sk()` models the possession of the nameserver to the zone's long-term agreement private key. The resolver used the public component of this key to generate a session key for the query encryption. By using this key and the resolver's ephemeral public agreement key and nonce, the nameserver is able to decrypt the received query. The fact `In()` is for receiving an encrypted query sent by a resolver from the network (from conclusion of Listing 6.5). The fact `!NS_sstk_Signed_Sk()` contains the short-term signing key of the nameserver, as well as the signature for this short-term key, which was generated by the zone's key server in the delegation phase. The nameserver also generates two fresh values: one is `ephN` that represents the nameserver's ephemeral agreement key, and the other is `Nnc`, which is the nonce used for generating the response session key. The `Running()` action fact is added for proving the authentication of the name-

server in the properties. The `Role('N')` action fact is added for distinguishing the nameserver from the resolver roles in the definition of properties. Additionally, the `NS_Sends_Response()` is an action fact which is used in the executability lemmas for ensuring that the name resolution executes properly.

Listing 6.6: Nameserver Responds a Query

```
1   rule NS_1:
2          let
3            queryKey = kdf(<Rnc, epkR ^ ZKA>)
4            ResponseMaster = kdf(< ~Nnc, Rnc, epkR ^ ~ephN >)
5            query_data = < query, 'Query' >
6            ephSig = sign('g' ^ ~ephN, stsk)
7            response_data = < 'Response', ~response, query, $N, nid, stpk
                 , ephSig, stk_sig >
8          in
9          [ !ZKA_Sk($K, ZKA) // long-term agreement private key
10         , In( <Rnc, epkR, senc(query_data, queryKey) > ) // query
11         , !NS_sstk_Signed_SK(stk_sig, <$N, nid, stpk>, stsk)
12         , Fr(~ephN) // nameserver's ephemeral key
13         , Fr(~Nnc) // nameserver's nonce
14         , Fr(~response) ] // response
15         --[
16             Role('N')
17           , NS_Sends_Response($R, $N, qid, 'Response', nid, stsk,
                 stk_sig)
18           , Running($N, $R, 'Nameserver', < qid, ResponseMaster >)
19           , IN_N_1(qid, <Rnc, epkR, senc(query_data, queryKey)>)
20         ]->
21         [
22           Out(< ~Nnc, 'g' ^ ~ephN, senc( response_data, ResponseMaster
                 ) >) // send encrypted response to network
23         ]
```

In the conclusion of this rule, the nameserver sends out to the network the symmetrically encrypted response alongside the nameserver's freshly generated ephemeral agreement public key (`'g'^ephN`) and nonce (`Nnc`).

As demonstrated in Listing 6.7, the resolver receives the response from the nameserver rule (Listing 6.6) and verifies the signatures included within it. In the premises of this rule, the resolver receives the response from the network using the `In()` fact. Furthermore, the `Res_State_1()` fact is derived from the conclusion of the rule in

Listing 6.5, where the resolver sent the query and stored the query state information, including the resolver's ephemeral agreement key and the fresh nonce used for query encryption. Additionally, the `ZKS_Pk()` fact, initialized by the key server rule in Listing 6.1, contains the zone's long-term public signing key (herein included as the trust anchor). The private component of this key was used in the delegation process to sign the short-term signing key of the nameserver instances.

Listing 6.7: Resolver Receives Response

```
1   rule Res_2:
2          let
3          ResponseKey = kdf(< Nnc, Rnc, epkN ^ ephR >)
4          response_data = < 'Response', $N, qid, nid, stpk, sig_epkN,
               stk_sig >
5          in
6          [ In(< Nnc, epkN, senc(response_data, ResponseKey)>)
7          , Res_State_1($R, ephR, qid, Rnc)
8          , !ZKS_Pk($K, ltkPK)
9          ]
10         --[
11           Eq(verify(sig_epkN, epkN, stpk), true)
12         , Eq(verify(stk_sig, <$N, nid, stpk>, ltkPK), true)
13         , Role('R')
14         , SecretR(response_data)
15         , RKeySecret(ResponseKey)
16         , Commit($R, $N, 'Resolver', < qid, ResponseKey >)
17         , ResolverReceivesResponse( 'Response', qid)
18         ]->
19         []
```

The first two action facts within this rule (Lines 11–12) are used to verify the signatures included in the received responses. Line 11 verifies that the ephemeral agreement key of the nameserver is signed by its short-term signing key, and Line 12 validates that the short-term signing key of the nameserver was signed using the zone's long-term signing key during the delegation phase. The `Role('R')` action fact will be used for distinguishing the role of resolver from the nameserver. `SecretR()` is the action fact, which will be used in the proof of response data secrecy. In Line 15, the fact `RKeySecret()` will later be used in the proof lemmas for proving the secrecy of the response keys received by the resolvers. The `Commit()` action fact is added to be used in the entity authentication proof. Finally, the `ResolverReceivesResponse()`

is a fact which is used in the executability lemmas for ensuring that the protocol successfully completes, and the properties are not proved in without the proper execution of the protocol. As there is no further steps in the protocol run, the conclusion of this rule is empty.

6.3 Modelling the Security Properties

Following defining the symbolic mode of DNSSEC+ using Tamarin rules, in this section, we model the specified security properties of DNSSEC+ as lemmas using Tamarin's syntax, and subsequently verify that these properties hold.

6.3.1 Secrecy of DNS Messages

Message secrecy in formal verification is generally defined as follows:

$$msg\_secrecy \triangleq \forall\ msg\ \ i.\ \ Secret(msg)\ @i\ \rightarrow not\ \exists\ j.\ K(msg)@j$$

This means: the definition of *msg_secrecy* is for all instances of the term *msg*, where *msg* is captured by the *Secret()* action fact at time point i, there does not exist a time point j at which the adversary knows the *msg*.

Secrecy of Query Data

We formally define the secrecy property of queries (*i.e.,* in the privacy-enforcing mode of DNSSEC+) generated and sent by a legitimate resolver by the lemma specified in Listing 6.8. This lemma asserts that whenever an encrypted query is sent by a legitimate resolver at time point i (captured by the instance of `QDataSecret(R, QData)` action), and none of the private agreement or signing keys of DNSSEC+ are compromised (lines 4-8), then there is no time point j at which the adversary knows the plaintext of the query message. In the verification of this property in Tamarin, it is assumed that all keys within the protocol remain unknown to the adversary.

We used the `query_secrecy` lemma and verified the secrecy of the query data in Tamarin. Additionally, we independently verified the secrecy of the query encryption key using another lemma in Tamarin. In the context of DNSSEC+, we assert that the `query_secrecy` property also implicitly proves the confidentiality of the query session key (*i.e.,* the derived key used for query encryption). If the query encryption

```
1 lemma query_secrecy:
2  "All R QData #i .
3   QDataSecret(R, QData) @i & Role('R') @i &
4   not (Ex Z #t1 .  RevLTAK(Z) @t1) &
5   not (Ex R #t2 .  RevEphR(R) @t2) &
6   not (Ex N #t3 .  RevEphN(N) @t3) &
7   not (Ex Z #t4 .  RevLTSK(Z) @t4) &
8   not (Ex N #t5 .  RevSTSK(N) @t5)
9   ==> not (Ex #j.  K(QData) @j)"
```

Listing 6.8: Lemma for query secrecy (verified by Tamarin)

key is compromised, this property will be violated, as the adversary can decrypt and gain knowledge of the query data in plaintext.

Secrecy of Response Data

The DNSSEC+ response encryption key is distinct from the encryption key used for transmitting its corresponding query. In addition to the secrecy of the query data, we also verify the secrecy of the response data received by a legitimate resolver as an answer to a query sent by the same resolver. We define the lemma for response data secrecy, as demonstrated in Listing 6.9, and use Tamarin to prove this lemma. This lemma asserts that: for all protocol behaviors, if an encrypted response to a query of a legitimate resolver is received by the resolver at time point i (captured by the instance of `RDataSecret(RData)` action in line 3), and none of the private agreement or signing keys are compromised, there is no time point j at which the adversary knows the plaintext response. In this lemma, similar to query secrecy, we assume that none of the protocol private keys are known to the adversary.

```
1 lemma response_secrecy:
2   All RData #i .
3   RDataSecret(RData) @i & Role('R') @i &
4   not (Ex Z #t1 .  RevLTAK(Z) @t1) &
5   not (Ex R #t2 .  RevEphR(R) @t2) &
6   not (Ex N #t3 .  RevEphN(N) @t3) &
7   not (Ex Z #t4 .  RevLTSK(Z) @t4) &
8   not (Ex N #t5 .  RevSTSK(N) @t5)
9   ==> not (Ex #j.  K(RData) @j)"
```

Listing 6.9: Lemma for response secrecy (verified by Tamarin)

The proof of the `response_secrecy` lemma in Tamarin verifies that DNS re-

sponses of queries received by a legitimate resolver remain unknown to the adversary. Additionally, we independently verified the secrecy of the response encryption key using a separate lemma in Tamarin. Furthermore, in the context of DNSSEC+, the `response_secrecy` lemma implicitly ensures the secrecy of the response encryption keys. If the response encryption key becomes known to the adversary, this property would not hold, as the adversary would then be able to decrypt and learn the response. Thus, *response encryption key secrecy* is a necessary condition for *response message secrecy* in DNSSEC+ (because the response is encrypted once using a single key).

A note on the secrecy of DNS queries and responses. In DNSSEC+, if an adversary gains access to the plaintext of the response data, the secrecy of the query is also compromised, as the query data (*e.g.,* question and query ID fields) is included in the response, as required by the DNS standard [119]. On the other hand, the compromise of the plaintext query does not directly lead to the compromise of the response in DNSSEC+. This is because the keys used to encrypt queries and responses are distinct, and the response section of data is not included in the query messages. However, given that DNS data is generally public, an adversary with knowledge of the query's question might independently resolve the same query to determine the corresponding response. Thus, while the adversary cannot directly extract the response from DNSSEC+ queries, access to plaintext query enables inference of the response through independent resolution. These two inherent characteristics of DNS highlight the importance of the confidentiality of both queries and responses, as the lack of secrecy in one may render the secrecy of the other futile.

6.3.2 Forward Secrecy

In DNSSEC+, the response session keys are derived from ephemeral keys of both resolvers and nameservers. As a result, these session keys satisfy forward secrecy, meaning that even if the long-term zone and nameserver signing keys are compromised (*i.e.,* LTSK and STSK), the session keys from prior DNS resolutions remain unknown to the adversary. As illustrated in Listing 6.10, forward secrecy lemma is formally defined within our Tamarin model and subsequently proved using Tamarin. For all protocol behaviors, when a resolver receives a response encrypted with a session key at time point i (modeled by `RKSecret(Rkey) action`) and none of the

private agreement keys are compromised at any time, and also none of the long-term signing keys are compromised before the response is received (lines 4 and 5), then there is no time point j at which the adversary knows the session key.

As stated in lines 4 and 5 (Listing 6.10), the long-term keys `LTSK` and `STSK` must not be compromised before the response is received and captured by the `RKSecret()` fact at time point i. Therefore, by adding (`t1 < i` and `t2 < i`) to the conditions, we are allowing the possibility that the adversary can know `LTSK` and `STSK` after the response is received at i. If the signing keys `LTSK` and `STSK` are compromised before the response is received, and the adversary has access to a query, the adversary could use these signing keys to impersonate the zone or nameserver and inject a false response, and the encryption key would then be known to the adversary, which violates both secrecy and forward secrecy of the response encryption keys. Thus, the compromise of query secrecy and one of these long-term keys undermines the forward secrecy of DNS response keys and secrecy of DNSSEC+ responses.

```
1  lemma response_sesskey_FwdSecrecy:
2   "All Rkey #i .
3    RKSecret(Rkey) @i & Role('R') @i &
4    not (Ex Z #t1 .  RevLTSK(Z) @t1 & t1 < i) &
5    not (Ex N #t2 .  RevSTSK(N) @t2 & t2 < i) &
6    not (Ex Z #t3 .  RevLTAK(Z) @t3) &
7    not (Ex R #t4 .  RevEphR(R) @t4) &
8    not (Ex N #t5 .  RevEphN(N) @t5)
9    ==> not (Ex #k.  K(RKey) @k)"
10
```

Listing 6.10: Lemma for response key forward secrecy (verified by Tamarin)

6.3.3 Unilateral Authentication

Nameserver unilateral authentication is formally defined as a unilateral injective agreement [109, 170], as illustrated in the lemma in Listing 6.11. The unilateral authentication lemma is defined as an agreement on a specified set of values, including keys and identities, in the matching runs between resolvers and nameservers [51, 109], and then proved using Tamarin. It asserts that for all protocol behaviors, each `Commit` action by the resolver upon receiving a response with the specified data implies that, if none of the private agreement and signing keys within the protocol are known to the adversary, there exists a unique `Running` action

(lines 10–12) performed by the nameserver with the same data. As there is a unique `Running` action preceding each `Commit` action, DNSSEC+ effectively prevents replay attacks by utilizing a fresh nonce for each response, which is included in the agreed upon `data` in this lemma.

The proof of the authentication property in Listing 6.11 guarantees that the resolver can verify the authenticity of responses received from a nameserver, providing only the authentication of the nameserver. However, this lemma cannot validate that `RES2` is equal to `RES`, as unilateral authentication implies that only the resolver authenticates the nameserver (*i.e.,* `NS` in lines 3 and 10 are equal). Therefore, no mutual authenticity guarantee, by including the equality between `RES2` in the `Running` action and `RES` in the `Commit` action, can be inferred from the unilateral authentication property defined in this lemma. This is because in DNSSEC+, only resolvers are responsible for authenticating nameservers, whereas the nameservers do not authenticate resolvers.

```
1  lemma unilateral_injective_agreement:
2   "All RES NS data #i.
3    Commit(RES, NS, 'Resolver', data) @i &
4    not (Ex Z #t1 .  RevLTAK(Z) @t1) &
5    not (Ex R #t2 .  RevEphR(R) @t2) &
6    not (Ex N #t3 .  RevEphN(N) @t3) &
7    not (Ex Z #t4 .  RevLTSK(Z) @t4) &
8    not (Ex N #t5 .  RevSTSK(N) @t5)
9   ==>
10  (Ex RES2 #j.  Running(NS, RES2, 'Nameserver', data) @j & j < i &
11  not (Ex RES3 NS2 #i2.  Commit(RES3, NS2, 'Resolver', data) @i2
12   & not (i2 = i)))"
```

Listing 6.11: Lemma for unilateral nameserver authentication (verified by Tamarin)

Summary. In this section, we formally modeled the security and privacy properties of DNSSEC+ by defining their lemmas. Subsequently, we used Tamarin to automatically construct proofs for each property in the presence of the Dolev-Yao adversary. The analysis did not yield any counterexamples indicating attacks, and the proof for each lemma was generated in Tamarin.

6.4 Implications of Key Compromises

In Section 6.3, we defined and proved the properties of DNSSEC+ under the assumption that if none of the agreement or signing keys within the protocol are compromised, then the defined properties for the protocol are valid and verifiable in Tamarin. In this section, we investigate the implications of key compromises in DNSSEC+ using Tamarin. To this end, we allow each agreement key or signing key to be compromised (become known to the adversary), and then determine whether the properties from Section 6.3 still hold. Subsequently, we extend our analysis for multiple protocol keys being compromised, modeling the compromise of different entities within DNSSEC+. As an example, we find that if resolvers are compromised, only the ephemeral private agreement key of the resolvers (`ephR`) will be compromised. Table 6.2 summarizes the results, which are explained in details in the following two subsections: 6.4.1 and 6.4.2.

6.4.1 Single Key Compromises

Long-term Agreement Key (`LTAK`): The long-term agreement key of a zone (`LTAK`) is used by resolvers to derive query encryption keys. We allow this key to become known to the adversary and retry to generate the proofs for the lemmas that were proved in Section 6.3. As summarized in Table 6.2, if the `LTAK` is compromised, only the *query secrecy* property can no longer be proved, and Tamarin identifies a corresponding attack. Using the `LTAK`, with the resolver's nonce and public agreement key, an adversary can derive the query encryption key, and thereby decrypt the query. However, since the remaining properties are related to responses and the response encryption keys are different from those for queries, the other properties remain valid.

Long-term Signing Key (`LTSK`): The long-term signing key (`LTSK`) of a zone is used by the key server to sign the public signing keys of the nameservers as part of the zone-side delegation process, thereby enabling the nameservers to respond to client queries with verifiable authenticity. We allow this key to become known to the adversary and attempt to recreate the proofs from Section 6.3. The compromise of `LTSK` alone does not undermine the *query secrecy* property, as it does not impact the confidentiality of queries. Tamarin successfully proves this property in this scenario. Similarly, *response secrecy* remains intact and is proved by Tamarin, since

an adversary possessing the LTSK cannot decrypt responses, nor can they generate or inject false responses without access to the corresponding query, which is then embedded within the response.

Furthermore, the *response key forward secrecy* property is proved by Tamarin, as the compromise of LTSK does not impact the private agreement keys required to derive the response encryption key. For an adversary to impersonate a nameserver or inject a known false response, access to the query is necessary for generating an acceptable false response.

Finally, when the LTSK is compromised, the resolver can still authenticate the nameserver, preserving the validity of the *entity authentication* property in Tamarin. Since the adversary cannot access the query information and response encryption keys, the adversary cannot compromise the authenticity of the nameserver responses, received by legitimate resolvers (as proved by Tamarin). Overall, by compromising LTSK alone, an adversary can delegate authority to malicious nameservers by signing their short-term key structures. However, these malicious nameservers cannot respond to legitimate queries without access to query decryption keys.

Short-term Signing Key (STSK): Similar to the long-term signing key (LTSK), as proved by Tamarin, the compromise of private (STSK) does not impact the *query secrecy* property, as this key is not utilized for query encryption. Furthermore, the compromise of a nameserver's STSK alone does not undermine the proof of *response secrecy* in Tamarin. This is because the STSK is not directly used for encrypting responses, and without access to the query data embedded in responses, an adversary cannot generate and inject false responses, which are acceptable (by the legitimate resolver) and known to the adversary.

Regarding the *response forward secrecy* property, possession of the STSK alone does not allow the adversary to compromise the forward secrecy of response keys. Additionally, the adversary cannot impersonate the nameserver, as the query data is still required to generate responses, and this data cannot be accessed by the attacker solely by having access to STSK. Regarding *entity authentication*, it is proved by Tamarin as the adversary does not have access to the query to generate false responses, and also does not have access to response encryption keys to compromise the authenticity of the legitimate nameserver responses directly.

Ephemeral Nameserver Agreement Key (ephN): ephN is the ephemeral private agreement key used on the nameserver-side to derive the response encryption

key. Compromise of this key directly undermines the *response secrecy* property in Tamarin, enabling the adversary to derive the response encryption key. Additionally, as the query data is embedded within responses, this compromise would also undermine the *query secrecy* as the adversary obtains query information from its corresponding response.

Regarding the *response key forward secrecy* property, the compromise of the ephemeral agreement keys render the response encryption key accessible to the adversary, thereby violating forward secrecy regardless of the secrecy of the long-term keys. Furthermore, possession of the response encryption key allows the adversary to directly violate the *nameserver authentication* property (*e.g.,* by modifying responses).

Ephemeral Resolver Agreement key (`ephR`): `ephR` is the ephemeral private agreement key used for deriving both query and response encryption keys. Thus, if the `ephR` of resolvers are known to the adversary, then both the *query secrecy* and *response secrecy* would be directly compromised. Additionally, as the response encryption key becomes known to the adversary, the *response key forward secrecy* property does not hold regardless of the secrecy of the long-term keys. Finally, the *nameserver authentication* would also be compromised, if `ephR` is compromised, because an adversary knowing this key can directly modify the nameserver-generated responses.

6.4.2 Entity Compromises

Three main types of entities are involved in the the DNS resolution of DNSSEC+: resolvers, nameservers, and key servers. Here we explain the implications of compromise of each of these entities on the properties of DNSSEC+.

Resolver: In DNSSEC+, the only private key that resolvers have access to is the ephemeral private agreement keys (`ephR`) they generate. Consequently, if resolvers are compromised, the adversary would gain access to `ephR`. The impact of a resolver compromise on the defined security properties is therefore equivalent to the compromise of `ephR` itself. Thus, all of the defined query and response properties would be violated by the attacker if resolvers are compromised, as shown in Table 6.2.

Nameserver: In DNSSEC+, nameservers have access to the zone's long-term private agreement key (`LTAK`), their short-term signing key (`STSK`), and ephemeral

Compromised key/entity	Query Secrecy	Response Secrecy	Response Key Fwd Secrecy	Nameserver Authentication
LTAK	✗	✓	✓	✓
LTSK	✓	✓	✓	✓
STSK	✓	✓	✓	✓
ephN	✗	✗	✗	✗
ephR	✗	✗	✗	✗
Nameserver	✗	✗	✗	✗
Key server	✗	✗	✗	✗
Resolver	✗	✗	✗	✗

Table 6.2: Impact of key and entity compromises on the properties of DNSSEC+. Columns represent the defined protocol properties, and rows show the keys and entities within DNSSEC+. Each cell in the table indicates whether a property still holds or not if the corresponding key or entity is compromised. (Keys compromised due to entity compromises in DNSSEC+: *Nameserver*: ALTK, STSK, and EphN. *Key server*: ALTK and SLTK. *Resolver*: EphR)

agreement keys (`ephN`). Consequently, if nameservers are compromised, all these keys become known to the adversary. Possession of the ephemeral agreement key (`ephN`) alone enables the adversary to compromise all defined security properties (as shown in Table 6.2), regardless of the secrecy of the other keys. Therefore, the compromise of a nameserver renders all the properties ineffective.

Key Server: In DNSSEC+, key servers are responsible for generating and managing the long-term agreement key of a zone (`LTAK`), which is used for deriving query encryption keys. Furthermore, key servers also manage the long-term signing key of the zone (`LTSK`), which facilitates the delegation of authorization to the nameservers within a zone. Compromise of the `LTAK` enables an adversary to decrypt queries, thereby undermining the *query secrecy* property. Access to plaintext queries, combined with the compromised `LTSK`, allows the adversary to impersonate the zone or associated nameservers and inject false responses. The compromise of these two long-term keys also enables an adversary to impersonate a zone and to inject false responses using arbitrary response encryption keys, violating both *response secrecy* and *response key forward secrecy*. Additionally, the ability to impersonate nameservers and to inject false responses violates the *nameserver authentication* property.

Overall, as illustrated in the bottom three rows of Table 6.2, the compromise of any entity involved in the DNS resolution process of DNSSEC+ undermines all four

properties formally established in Section 6.3.

6.5 Discussion

Query Compromise. The compromise of a query in DNSSEC+ alone does not result in the adversary gaining knowledge of its corresponding response. However, since DNS records are not typically secret, an adversary who knows the query can resolve the query independently to obtain its corresponding response. This is not a flaw in DNSSEC+, but an inherent property of DNS itself. While the lack of query confidentiality in DNS allows adversaries to independently resolve the same queries, DNS responses to a single query can vary in specific scenarios. For example, responses may differ in cases where Content Delivery Networks (CDNs) return responses based on the geographic/network location of the query's source IP address, or other scenarios where different load-balancing mechanisms are employed.

Additionally, some nameservers are configured to provide different responses based on query metadata and querent profile, including the transport layer protocol, source address, and other metadata parameters. For example, a nameserver might white-list "`ANY`" queries to specific network locations or addresses. Such configurations may result in different DNS responses for identical queries. Consequently, even when queries are not encrypted, an adversary resolving the same query may not necessarily receive a response identical to the one provided to the original query.

Secrecy in Tamarin. In Tamarin, secrecy is defined as the adversary's lack of knowledge regarding a specific term, such as a message or a cryptographic key. However, there are two main cases within our model through which an adversary may gain knowledge of a specific term. The case involves accessing the plaintext of encrypted messages from a legitimate protocol execution. The second requires the adversary impersonating one of the protocol's entities and injecting a false response, thereby having access to the plaintext of the injected response. The latter does not constitute a direct secrecy attack but rather an impersonation attack; consequently, the encrypted legitimate protocol interactions are not necessarily compromised. Nevertheless, the adversary's ability to inject an arbitrary message that it is aware of represents a violation of the message secrecy. In this paper, we have proven message secrecy in this Tamarin conventional manner and have considered both scenarios as violations of secrecy.

Model Details. Tamarin has a pre-computation phase during which it determines the sources of protocol and intruder facts to reuse them in later analysis. However, for certain protocols, Tamarin may fail to correctly identify the sources of specific facts, resulting in an incomplete pre-computation phase. To address this issue, users must manually define a special type of lemma, referred to as a "source lemma", to explicitly clarify the origins of values for which Tamarin lacks sufficient information. In our work, we defined and used a source lemma to enable Tamarin to successfully complete the pre-computation phase. This source lemma and other details defined in the model are included in the publicly available source code of our theory.[1]

6.6 Summary

In this chapter, we developed a symbolic model of DNSSEC+ and conducted a formal verification of its security and privacy properties. Furthermore, we analyzed the impact of key and entity compromises on these verified properties. In Chapter 7, we revisit the research questions of this thesis, review how they have been addressed, and discuss potential future research directions.

[1] `https://github.com/Ali-Jahromi/FormalAnalysisDNSSECPlus`

Chapter 7

Discussion and Concluding Remarks

DNS is a critical component of the Internet, enabling the resolution of hundreds of billions of queries each day.[1] Consequently, the security, privacy, and availability of these entities relying upon DNS are inevitably tied to the security and privacy of the name resolution schemes used.

Based on the research questions outlined in Chapter 1, the initial objective of this thesis was to identify the threats of the name resolution process and to define a set of properties sufficient to mitigate these threats. The second objective was to develop a systematic evaluation framework for assessing schemes designed to enhance the security of the DNS resolution process. The third was to propose an improved scheme to augment the security and privacy of the DNS resolution process in Stage 2.

7.1 Addressing the Research Questions

In this section, we revisit the research questions and explain how they have been addressed in this thesis.

RQ1. What are the primary security and privacy threats facing the DNS resolution process; and, what security, privacy, and availability properties are sufficient to effectively mitigate these threats?

In Chapter 3, we developed a comprehensive threat model of the DNS resolution process with the primary focus on network-based attacks. The identified DNS resolution threats were categorized into five main groups based on their technical goals, with the techniques employed by attackers to achieve these objectives summarized in Figure 3.1. To mitigate the DNS resolution threats outlined in the threat model and attack taxonomy, we subsequently defined 14 security, privacy, and availability properties for the DNS resolution process. These properties were established based on insights from the threat model and the literature review of DNS security and shortcomings of previously proposed secure DNS schemes (Chapter 2).

[1] `https://vercara.com/resources/2023-dns-traffic-and-trends-analysis`

RQ2. What type of evaluation framework could be developed to objectively evaluate the security and privacy of both existing and future DNS schemes designed to enhance the security or privacy of the DNS resolution process?

To address this question, Chapter 2 presents a comprehensive literature review of secure DNS schemes in Stages 1 and 2. This chapter also includes an analysis of security and privacy enhancements to the DNS resolution process, distinguishing between approaches that require minimal modifications to the existing DNS and those that require significant changes to the Vanilla DNS infrastructure and processes.

Building on this analysis, a systematic evaluation framework was developed to assess DNS security schemes based on 14 defined security, privacy, and availability properties that we asserted to be sufficient to address the identified threats to the DNS resolution process. Utilizing this framework, Table 3.1 provides an evaluation of 11 secure DNS schemes identified in the literature review (Chapter 2). The findings from this evaluation were further used for a discussion on the existing challenges in the DNS resolution process.

This framework offers insights into the limitations of the analyzed schemes and we hope that it may be of use as a foundation for the design and analysis of future DNS security solutions by providing an objective approach for their evaluation.

RQ3. How can an enhanced Stage 2 DNS resolution scheme be designed and developed to augment the security and privacy in Stage 2 of the DNS resolution process?

Chapter 4 introduced DNSSEC+, a novel secure DNS scheme in large part motivated by the benefits and shortcomings of DNSSEC. We defined 9 security, privacy, deployability, and performance-related design goals for DNSSEC+. To achieve these objectives, we described the required entities and processes involved in secure delegation, trust model establishment, and name resolution within DNSSEC+.

The secure delegation mechanism here is a key feature of DNSSEC+. It mitigates the risks associated with replicating long-term secrets across ANS instances within zones, thereby reducing the risk of the exposure of long-term secrets via attacks on these ANS instances. Additionally, DNSSEC+ was designed to complete the name resolution process in a single round-trip, and as the network latency is the dominant latency in DNS resolution, the total DNS resolution latency of DNSSEC+ is comparable to that of Vanilla DNS.

One of the primary barriers to the adoption and deployability of secure DNS

schemes is the absence of an accepted trust model by root and TLD authorities in the DNS hierarchy. To address this challenge, DNSSEC+ uses a DNSSEC-like trust model, which has been accepted and adopted by root and TLD authorities. This model is suitable for the DNS context as it maintains root zone authority without reliance on external entities. Using this trust model enables DNSSEC+ to include nameserver keys within DNS responses while simultaneously achieving the single round-trip resolution goal.

To assess effectiveness, Chapter 5 presents a comparative evaluation of DNSSEC+ with other Stage 2 schemes with respect to their security, privacy, and availability properties. Additionally, we implemented a prototype of DNSSEC+ and conducted performance measurements for comparison with previously proposed secure DNS schemes. The results demonstrate that DNSSEC+ achieves performance comparable to existing schemes while offering enhanced security and privacy properties.

To gain further confidence in the security and privacy properties of DNSSEC+, Chapter 6 presents a formal verification of the protocol using Tamarin, based on a symbolic model of DNSSEC+. Furthermore, we analyzed the impact of key and entity compromises on the security and privacy properties of DNSSEC+, demonstrating the protocol's resilience against potential threats.

7.2 Future Research Directions

In this section, we outline potential research directions for future work. These extend investigations conducted in this thesis or address new problems identified herein.

Expanding Threat Modeling to Additional Aspects. In Section 3, a comprehensive threat model and attack taxonomy was developed, focusing specifically on network-based attacks within the DNS resolution process. However, other critical processes and threats within the DNS ecosystem, such as domain registration, zone transfers, and the exploitation of DNS for malicious activities—such as botnet communication or data exfiltration—were beyond the scope of this thesis. Expanding threat model to include these would be worthwhile and serve as a stepping stone for the development of effective mitigation mechanisms.

Systematic Evaluation of Usability and Deployability. In Section 3.3, an objective framework was developed to evaluate the security, privacy, and availability properties of secure DNS schemes. However, to achieve a broader systematic

understanding, additional factors may include those related to usability and deployability. Extending our objective evaluation framework for these aspects would be worthwhile, as usability and deployability barriers can prevent adoption of secure DNS schemes.

Adapting DNSSEC+ for Stage 1. In Chapter 4, DNSSEC+ was proposed and designed to augment security and privacy in Stage 2. The current design employs a reverse-tree DNSSEC-like PKI, specifically tailored for a Stage 2 scheme. However, we believe that DNSSEC+ can be modified to work with alternative PKIs, enabling its application in Stage 1 to secure communication between clients and recursive resolvers while maintaining a single round-trip resolution with minimal latency and overhead. For instance, DNSSEC+ could be implemented in Stage 1 using the web PKI, with CA public keys configured as trust anchors in the client (stub resolver) software. A potential avenue for future research is to explore how DNSSEC+ can be extended to Stage 1 by modifications that incorporate PKIs with trust models different from the current reverse-tree trust model used in DNSSEC+, maintaining its security and privacy benefits while minimizing overhead.

Utilizing DNSSEC+ to Enhance the Security of Internet Protocols. As discussed in the literature review (Chapter 2) and the threat model (Chapter 3), vulnerabilities in the DNS infrastructure have been exploited to compromise the security and privacy of various Internet protocols and ecosystems. Future research could explore the practical applications of DNSSEC+ in securing critical Internet processes, such as domain validation in the web PKI [30], enhancing security and privacy in VPN connections [138], and strengthening other network protocols that rely on DNS [43].

Securing the Entire DNS Resolution. To enhance the security of the entire DNS resolution path, the integration or combination of DNSSEC+ with other Stage 1 schemes could be explored, aiming to deliver new combinations of security, privacy, and performance properties, that meet the requirements of different use cases. As a potential future research direction these combinations might be analyzed to demonstrate their properties and suitability across different contexts.

Practical Adoption of DNSSEC+. For practical adoption, we expect that the proper path would be to introduce discussion of DNSSEC+ to the IETF community through the appropriate IETF Working Group (WG), which would be the one that is responsible for DNSSEC (and other similar proposals). The steps would be

(after an initial academic-style publication as is already in progress), to prepare an IETF working draft that would receive technical feedback from the IETF community (drawing upon its practical and deployability expertise in particular). Based on this feedback, the document could evolve into a formal RFC, endorsed by the working group, most likely incorporating technical revisions to the scheme as specified in the thesis. Whether or not that would actually lead to adoption of DNSSEC+ is impossible to predict, as even DNSSEC itself has seen a very slow uptake over time despite substantial expert support behind it.

Bibliography

[1] Sadia Afroz and David Fifield. Timeline of Tor censorship. `https://web.archive.org/web/20240229081919/http://www.icsi.berkeley.edu/~sadia/tor_timeline.pdf`, 2021.

[2] Pieter Agten, Wouter Joosen, Frank Piessens, and Nick Nikiforakis. Seven months' worth of mistakes: a longitudinal study of typosquatting abuse. In *Network and Distributed System Security Symposium (NDSS)*, 2015.

[3] Muneeb Ali, Ryan Shea, Jude Nelson, and Michael J Freedman. Blockstack technical whitepaper. *Blockstack PBC, October*, 12, 2017.

[4] Kamal Alieyan, Ammar ALmomani, Ahmad Manasrah, and Mohammed M Kadhum. A survey of botnet detection based on DNS. *Neural Computing and Applications*, 28(7):1541–1558, 2017.

[5] Mario Almeida, Alessandro Finamore, Diego Perino, Narseo Vallina-Rodriguez, and Matteo Varvello. Dissecting DNS stakeholders in mobile networks. In *ACM International Conference on Emerging Networking Experiments And Technologies (CoNEXT)*, pages 28–34, 2017.

[6] Marios Anagnostopoulos, Georgios Kambourakis, Elisavet Konstantinou, and Stefanos Gritzalis. DNSSEC vs. DNSCurve: a side-by-side comparison. In *Situational Awareness in Computer Network Defense: Principles, Methods and Applications*, pages 201–220. IGI Global, 2012.

[7] Marios Anagnostopoulos, Georgios Kambourakis, Panagiotis Kopanos, Georgios Louloudakis, and Stefanos Gritzalis. DNS amplification attack revisited. *Computers & Security*, 39, 2013.

[8] Manos Antonakakis, Roberto Perdisci, David Dagon, Wenke Lee, and Nick Feamster. Building a dynamic reputation system for DNS. In *USENIX Security Symposium*, 2010.

[9] Manos Antonakakis, Roberto Perdisci, Yacin Nadji, Nikolaos Vasiloglou, Saeed Abu-Nimeh, Wenke Lee, and David Dagon. From throw-away traffic to bots: detecting the rise of DGA-based malware. In *USENIX Security Symposium*, 2012.

[10] Noah Apthorpe, Dillon Reisman, and Nick Feamster. Closing the blinds: four strategies for protecting smart home privacy from network observers. In *IEEE Workshop on Technology and Consumer Protection (ConPro)*, 2017.

[11] Roy Arends, Scott Rose, Matt Larson, Dan Massey, and Rob Austein. DNS Security Introduction and Requirements. RFC 4033, 2005.

[12] Roy Arends, Scott Rose, Matt Larson, Dan Massey, and Rob Austein. Protocol Modifications for the DNS Security Extensions. RFC 4035, 2005.

[13] Roy Arends, Scott Rose, Matt Larson, Dan Massey, and Rob Austein. Resource Records for the DNS Security Extensions. RFC 4034, 2005.

[14] Roy Arends, Geoffrey Sisson, David Blacka, and Ben Laurie. DNS Security (DNSSEC) Hashed Authenticated Denial of Existence. RFC 5155, 2008.

[15] Suranjith Ariyapperuma and Chris J Mitchell. Security vulnerabilities in DNS and DNSSEC. In *ACM International Conference on Availability, Reliability and Security (ARES)*, 2007.

[16] Simurgh Aryan, Homa Aryan, and J Alex Halderman. Internet censorship in Iran: a first look. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI)*, 2013.

[17] Daniele E Asoni, Samuel Hitz, and Adrian Perrig. A paged domain name system for query privacy. In *International Conference on Cryptology and Network Security*. Springer, 2018.

[18] Derek Atkins and Rob Austein. Threat Analysis of the Domain Name System (DNS). RFC 3833, 2004.

[19] Tuomas Aura. Cryptographically Generated Addresses (CGA). RFC 3972, 2005.

[20] Elaine Barker, Lily Chen, Allen Roginsky, Apostol Vassilev, and Richard Davis. Recommendation for pair-wise key-establishment schemes using discrete logarithm cryptography. Technical Report NIST Special Publication 800-56A, National Institute of Standards and Technology, 2018.

[21] Richard Barnes, Subodh Iyengar, Nick Sullivan, and Eric Rescorla. Delegated Credentials for (D)TLS, 2022. Internet-Draft (IETF).

[22] Richard Barnes, Subodh Iyengar, Nick Sullivan, and Eric Rescorla. Delegated Credentials for TLS and DTLS. RFC 9345, 2023.

[23] David Basin, Jannik Dreier, Lucca Hirschi, Saša Radomirovic, Ralf Sasse, and Vincent Stettler. A formal analysis of 5G authentication. In *ACM SIGSAC conference on computer and communications security*, 2018.

[24] Daniel J Bernstein. Cryptography in NaCl. *Networking and Cryptography library*, 3(385), 2009.

[25] Daniel J Bernstein. DNSCurve: Usable security for DNS. `https://dnscurve.org`, 2009.

[26] Sarah Bird, Ilana Segall, and Martin Lopatka. Replication: Why we still can't browse in peace: on the uniqueness and reidentifiability of web browsing histories. In *USENIX Symposium on Usable Privacy and Security (SOUPS)*, 2020.

[27] Stéphane Bortzmeyer. DNS Query Name Minimisation to Improve Privacy. RFC 7816, 2016.

[28] Timm Böttger, Felix Cuadrado, Gianni Antichi, Eder Leão Fernandes, Gareth Tyson, Ignacio Castro, and Steve Uhlig. An empirical study of the cost of DNS-over-HTTPS. In *ACM Internet Measurement Conference (IMC)*, 2019.

[29] Colin Boyd, Anish Mathuria, and Douglas Stebila. *Protocols for Authentication and Key Establishment*. Springer, 2020.

[30] Markus Brandt, Tianxiang Dai, Amit Klein, Haya Shulman, and Michael Waidner. Domain validation++ for MitM-resilient PKI. In *ACM Conference on Computer and Communications Security (CCS)*, 2018.

[31] Marc Buijsman, Matthijs Mekking, and Jeroen van der Ham. Securing the last mile of DNS with CGA-TSIG. *Research Project*, 2, 2014.

[32] Rishabh Chhabra, Paul Murley, Deepak Kumar, Michael Bailey, and Gang Wang. Measuring DNS-over-HTTPS performance around the world. In *ACM Internet Measurement Conference (IMC)*, 20201.

[33] L. Chuat, A. Abdou, R. Sasse, C. Sprenger, D. Basin, and A. Perrig. SoK: delegation and revocation, the missing links in the Web's chain of trust. In *IEEE European Symposium on Security & Privacy*, 2020.

[34] Taejoong Chung, Roland van Rijswijk-Deij, Balakrishnan Chandrasekaran, David Choffnes, Dave Levin, Bruce M. Maggs, Alan Mislove, and Christo Wilson. A longitudinal, end-to-end view of the DNSSEC ecosystem. In *USENIX Security Symposium*, 2017.

[35] Cloudflare. Merkle Town. `https://ct.cloudflare.com/`, 2018. Accessed: 2024.

[36] Carlo Contavalli, Wilmer van der Gaast, David C. Lawrence, and Warren "Ace" Kumari. Client Subnet in DNS Queries. RFC 7871, 2016.

[37] Alex Cowperthwaite and Anil Somayaji. The futility of DNSSEC. In *Annual Symposium Information Assurance (ASIA)*. Citeseer, 2010.

[38] Russ Cox, Athicha Muthitacharoen, and Robert T. Morris. Serving DNS using a peer-to-peer lookup service. In *Peer-to-Peer Systems*. Springer, 2002.

[39] Cas Cremers, Marko Horvat, Sam Scott, and Thyla van der Merwe. Automated analysis and verification of TLS 1.3: 0-RTT, resumption and delayed authentication. In *IEEE Symposium on Security and Privacy (S&P)*, 2016.

[40] Mark Crovella and Balachander Krishnamurthy. *Internet Measurement: Infrastructure, Traffic and Applications*. John Wiley & Sons, Inc., 2006.

[41] Joao da Silva Damas, Michael Graff, and Paul A. Vixie. Extension Mechanisms for DNS (EDNS(0)). RFC 6891, 2013.

[42] David Dagon, Manos Antonakakis, Paul Vixie, Tatuya Jinmei, and Wenke Lee. Increased DNS forgery resistance through 0x20-bit encoding: security via leet queries. In *ACM Conference on Computer and Communications Security (CCS)*, 2008.

[43] Tianxiang Dai, Philipp Jeitner, Haya Shulman, and Michael Waidner. From IP to transport and beyond: cross-layer attacks against applications. In *ACM SIGCOMM Conference*, 2021.

[44] Xavier de Carné de Carnavalet and Paul C. van Oorschot. A survey and analysis of TLS interception mechanisms and motivations: exploring how end-to-end TLS is made "end-to-me" for web traffic. *ACM Comput. Surv.*, 55(13s), July 2023.

[45] Casey Deccio, Derek Argueta, and Jonathan Demke. A quantitative study of the deployment of DNS rate limiting. In *International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2019.

[46] Matthew Dempsky. DNSCurve: link-level security for the domain name system, 2010. Internet-Draft (IETF).

[47] Frank Denis. DNSCrypt version 2 protocol specification. `https://github.com/DNSCrypt/dnscrypt-protocol/blob/master/DNSCRYPT-V2-PROTOCOL.txt`, 2017.

[48] Frank Denis. Anonymized DNSCrypt specification. `https://github.com/DNSCrypt/dnscrypt-protocol/blob/master/ANONYMIZED-DNSCRYPT.txt`, 2019.

[49] Sara Dickinson, Daniel Kahn Gillmor, and Tirumaleswar Reddy.K. Usage Profiles for DNS Over TLS and DNS Over DTLS. RFC 8310, 2018.

[50] Christian J Dietrich, Christian Rossow, Felix C Freiling, Herbert Bos, Maarten Van Steen, and Norbert Pohlmann. On botnets that use DNS for command and control. In *European Conference on Computer Network Defense*. IEEE, 2011.

[51] Whitfield Diffie, Paul C Van Oorschot, and Michael J Wiener. Authentication and authenticated key exchanges. *Designs, Codes and cryptography*, 2(2):107–125, 1992.

[52] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: the second-generation onion router. Technical report, Naval Research Lab Washington DC, 2004.

[53] DNSSEC - Enables on-the-fly DNSSEC signing of served data. `https://coredns.io/plugins/dnssec/`. Accessed: 2024-10-10.

[54] Danny Dolev and Andrew Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[55] Huayi Duan, Rubén Fischer, Jie Lou, Si Liu, David Basin, and Adrian Perrig. RHINE: robust and high-performance Internet naming with E2E authenticity. In *USENIX NSDI*, 2023.

[56] Zakir Durumeric, Eric Wustrow, and J Alex Halderman. ZMap: fast Internet-wide scanning and its security applications. In *USENIX Security Symposium*, 2013.

[57] Donald E. Eastlake 3rd and Mark P. Andrews. Domain Name System (DNS) Cookies. RFC 7873, 2016.

[58] Donald E. Eastlake 3rd and Charles W. Kaufman. Domain Name System Security Extensions. RFC 2065, 1997.

[59] Donald E. Eastlake 3rd, Ólafur Guðmundsson, Paul A. Vixie, and Brian Wellington. Secret Key Transaction Authentication for DNS (TSIG). RFC 2845, 2000.

[60] Wesley Eddy. TCP SYN Flooding Attacks and Common Mitigations. RFC 4987, 2007.

[61] Stephen Farrell and Hannes Tschofenig. Pervasive Monitoring is an Attack. RFC 7258, 2014.

[62] Hannes Federrath, Karl-Peter Fuchs, Dominik Herrmann, and Christopher Piosecny. Privacy-preserving DNS: analysis of broadcast, range queries and mix-based protection methods. In *European Symposium on Research in Computer Security (ESORICS)*. Springer, 2011.

[63] Electronic Frontier Foundation. NSA spying on Americans. `https://www.eff.org/nsa-spying`, 2024. Accessed: 2024-10-21.

[64] Miek Gieben. exDNS. `https://github.com/miekg/exdns`, 2013. Accessed: 2024.

[65] Miek Gieben. CoreDNS. `https://github.com/coredns/coredns`, 2016. Accessed: 2024.

[66] Go. ECDSA. `https://pkg.go.dev/crypto/ecdsa`, 2014. Accessed: 2024.

[67] Sharon Goldberg, Moni Naor, Dimitrios Papadopoulos, Leonid Reyzin, Sachin Vasant, and Asaf Ziv. NSEC5: provably preventing DNSSEC zone enumeration. *Cryptology ePrint Archive*, 2014.

[68] Luis Grangeia. DNS cache snooping. Technical report, Security Team—Beyond Security, 2004.

[69] Benjamin Greschbach, Tobias Pulls, Laura Roberts, Philipp Winter, and Nick Feamster. The effect of DNS on Tor's anonymity. In *Network and Distributed System Security Symposium (NDSS)*, 2017.

[70] C. Grothoff, M. Wachs, and M. Ermert. NSA's MORECOWBELL: knell for DNS. `https://git.gnunet.org/bibliography.git/plain/docs/mcb-en.pdf`, 2017.

[71] Shuai Hao, Yubao Zhang, Haining Wang, and Angelos Stavrou. End-users get maneuvered: empirical analysis of redirection hijacking in content delivery networks. In *USENIX Security Symposium*, 2018.

[72] Elias Heftrig, Haya Shulman, and Michael Waidner. Downgrading DNSSEC: how to exploit crypto agility for hijacking signed zones. In *USENIX Security Symposium*, 2023.

[73] Amir Herzberg and Haya Shulman. Security of patched DNS. In *Computer Security–European Symposium on Research in Computer Security (ESORICS)*. Springer, 2012.

[74] Amir Herzberg and Haya Shulman. Fragmentation considered poisonous, or: one-domain-to-rule-them-all.org. In *IEEE Conference on Communications and Network Security (CNS)*, 2013.

[75] Amir Herzberg and Haya Shulman. Towards adoption of DNSSEC: availability and security challenges. *Cryptology ePrint Archive*, 2013.

[76] Nguyen Phong Hoang, Ivan Lin, Seyedhamed Ghavamnia, and Michalis Polychronakis. K-resolver: towards decentralizing encrypted DNS resolution. In *NDSS Workshop on Measurements, Attacks, and Defenses for the Web (MADWeb)*, 2020.

[77] Nguyen Phong Hoang, Arian Akhavan Niaki, Jakub Dalek, Jeffrey Knockel, Pellaeon Lin, Bill Marczak, Masashi Crete-Nishihata, Phillipa Gill, and Michalis Polychronakis. How great is the great firewall? measuring China's DNS censorship. In *USENIX Security Symposium*, 2021.

[78] Nguyen Phong Hoang, Michalis Polychronakis, and Phillipa Gill. Measuring the accessibility of domain name encryption and its impact on Internet filtering. In *International Conference on Passive and Active Network Measurement (PAM)*. Springer, 2022.

[79] Jeff Hodges, Collin Jackson, and Adam Barth. HTTP Strict Transport Security (HSTS). RFC 6797, November 2012.

[80] Paul E. Hoffman and Patrick McManus. DNS Queries Over HTTPS (DoH). RFC 8484, 2018.

[81] Austin Hounsel, Kevin Borgolte, Paul Schmitt, and Nick Feamster. D-DNS: towards re-decentralizing the DNS. `https://www.arxiv.org/abs/2002.09055v3`, 2020.

[82] Austin Hounsel, Kevin Borgolte, Paul Schmitt, Jordan Holland, and Nick Feamster. Comparing the effects of DNS, DoT, and DoH on web performance. In *Proceedings of The Web Conference 2020*, 2020.

[83] Austin Hounsel, Paul Schmitt, Kevin Borgolte, and Nick Feamster. Designing for tussle in encrypted DNS. In *ACM Workshop on Hot Topics in Networks (HotNets)*, 2021.

[84] Austin Hounsel, Paul Schmitt, Kevin Borgolte, and Nick Feamster. Encryption without centralization: distributing DNS queries across recursive resolvers. In *ACM Applied Networking Research Workshop (ANRW)*, 2021.

[85] Rebekah Houser, Zhou Li, Chase Cotton, and Haining Wang. An investigation on information leakage of DNS over TLS. In *ACM International Conference on Emerging Networking Experiments And Technologies (CoNEXT)*, 2019.

[86] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. Hoffman. Specification for DNS Over Transport Layer Security (TLS). RFC 7858, 2016.

[87] Qing Huang, Deliang Chang, and Zhou Li. A comprehensive study of DNS-over-HTTPS downgrade attack. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI)*, 2020.

[88] Bert Hubert and Remco Mook. Measures for Making DNS More Resilient Against Forged Answers. RFC 5452, 2009.

[89] Christian Huitema, Sara Dickinson, and Allison Mankin. DNS over Dedicated QUIC Connections. RFC 9250, 2022.

[90] Jana Iyengar and Martin Thomson. QUIC: a UDP-Based Multiplexed and Secure Transport. RFC 9000, 2021.

[91] Ali Sadeghi Jahromi and AbdelRahman Abdou. Comparative analysis of DoT and HTTPS certificate ecosystems. In *NDSS Workshop on Measurements, Attacks, and Defenses for the Web (MADWeb)*, 2021.

[92] Ali Sadeghi Jahromi, AbdelRahman Abdou, and Paul C. van Oorschot. DNSSEC+: an enhanced DNS scheme motivated by benefits and pitfalls of DNSSEC. `https://arxiv.org/abs/2408.00968`, 2024.

[93] Ali Sadeghi Jahromi, AbdelRahman Abdou, and Paul C. van Oorschot. Formal security analysis of DNSSEC+, 2025. Submitted for publication.

[94] Ali Sadeghi Jahromi, AbdelRahman Abdou, and Paul C. van Oorschot. SoK: an evaluation framework for secure DNS schemes, 2025. In preparation for publication.

[95] Philipp Jeitner, Haya Shulman, and Michael Waidner. The impact of DNS insecurity on time. In *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2020.

[96] Nick Johnson. ERC-137: Ethereum Domain Name Service - Specification. `https://eips.ethereum.org/EIPS/eip-137`. Accessed: 2025.

[97] Harry A Kalodner, Miles Carlsten, Paul M Ellenbogen, Joseph Bonneau, and Arvind Narayanan. An empirical study of Namecoin and lessons for decentralized namespace design. In *WEIS*, volume 1, 2015.

[98] Dan Kaminsky. Black Ops 2008: It's the end of the cache as we know it., 2008.

[99] Aminollah Khormali, Jeman Park, Hisham Alasmary, Afsah Anwar, Muhammad Saad, and David Mohaisen. Domain name system security and privacy: a contemporary survey. *Computer Networks*, 185, 2021.

[100] Eric Kinnear, Patrick McManus, Tommy Pauly, Tanya Verma, and Christopher A. Wood. Oblivious DNS over HTTPS. RFC 9230, 2022.

[101] Panagiotis Kintis, Yacin Nadji, David Dagon, Michael Farrell, and Manos Antonakakis. Understanding the privacy implications of ECS. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2016.

[102] Mike Kosek, Trinh Viet Doan, Malte Granderath, and Vaibhav Bajpai. One to rule them all? a first look at DNS over QUIC. In *International Conference on Passive and Active Network Measurement (PAM)*. Springer, 2022.

[103] Warren "Ace" Kumari, Ólafur Guðmundsson, and George Barwood. Automating DNSSEC Delegation Trust Maintenance. RFC 7344, September 2014.

[104] Laurie Law, Alfred Menezes, Minghua Qu, Jerry Solinas, and Scott Vanstone. An efficient protocol for authenticated key agreement. *Designs, Codes and Cryptography*, 28, 2003.

[105] Jonathan Lemon et al. Resisting SYN flood DoS attacks with a SYN cache. In *BSDCon*, volume 2002, pages 89–97, 2002.

[106] Chris Lesniewski-Laas and M. Frans Kaashoek. SSL splitting: securely serving data from untrusted caches. *Computer Networks*, 48(5):763–779, August 2005.

[107] Baojun Liu, Chaoyi Lu, Haixin Duan, Ying Liu, Zhou Li, Shuang Hao, and Min Yang. Who is answering my queries: understanding and characterizing interception of the DNS resolution path. In *USENIX Security Symposium*, 2017.

[108] Gavin Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *International Workshop on Tools and Algorithms for the Construction and Analysis of Systems*, pages 147–166. Springer, 1996.

[109] Gavin Lowe. A hierarchy of authentication specifications. In *Computer Security Foundations Workshop*. IEEE, 1997.

[110] Chaoyi Lu, Baojun Liu, Zhou Li, Shuang Hao, Haixin Duan, Mingming Zhang, Chunying Leng, Ying Liu, Zaifeng Zhang, and Jianping Wu. An end-to-end, large-scale measurement of DNS-over-Encryption: how far have we come? In *ACM Internet Measurement Conference (IMC)*, 2019.

[111] Xi Luo, Liming Wang, Zhen Xu, Kai Chen, Jing Yang, and Tian Tian. A large scale analysis of DNS water torture attack. In *International Conference on Computer Science and Artificial Intelligence*, 2018.

[112] Minzhao Lyu, Hassan Habibi Gharakheili, and Vijay Sivaraman. A survey on DNS encryption: current development, malware misuse, and inference techniques. *ACM Comput. Surv.*, 55(8), 2022.

[113] Muhammad Mahmoud, Manjinder Nir, and Ashraf Matrawy. A survey on botnet architectures, detection and defences. *IJ Network Security*, 17(3):264–281, 2015.

[114] Keyu Man, Zhiyun Qian, Zhongjie Wang, Xiaofeng Zheng, Youjun Huang, and Haixin Duan. DNS cache poisoning attack reloaded: revolutions with side channels. In *ACM Conference on Computer and Communications Security (CCS)*, 2020.

[115] Sam Marsh, Achiel van der Mandele, and Shih-Chiang Chien. Are you measuring what matters? a fresh look at Time To First Byte. `https://blog.cloudflare.com/ttfb-is-not-what-it-used-to-be`, 2023. Accessed: 2024.

[116] Alexander Mayrhofer. The EDNS(0) Padding Option. RFC 7830, May 2016.

[117] Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. The Tamarin prover for the symbolic analysis of security protocols. In *Computer Aided Verification (CAV)*, pages 696–701. Springer, 2013.

[118] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of Applied Cryptography*. CRC press, 1996.

[119] P. Mockapetris. Domain Names - Implementation and Specification. RFC 1035, 1987.

[120] Paul Mockapetris. Domain Names - Concepts and Facilities. RFC 1034, 1987.

[121] Stephen Morris, Johan Stenstam, John Dickinson, and Matthijs Mekking. DNSSEC Key Rollover Timing Considerations. RFC 7583, 2015.

[122] Giovane CM Moura, Sebastian Castro, Wes Hardaker, Maarten Wullink, and Cristian Hesselman. Clouding up the Internet: how centralized is DNS traffic becoming? In *ACM Internet Measurement Conference (IMC)*, 2020.

[123] Alec Muffett. No port 53, who dis? a year of DNS over HTTPS over Tor. In *NDSS DNS Privacy Workshop (DNSPriv)*, 2021.

[124] Steven J Murdoch and Piotr Zieliński. Sampled traffic analysis by Internet-exchange-level adversaries. In *Symposium on Privacy Enhancing Technologies (PETS)*. Springer, 2007.

[125] Asaf Nadler, Avi Aminov, and Asaf Shabtai. Detection of malicious and low throughput data exfiltration over the DNS protocol. *Computers & Security*, 80:36–53, 2019.

[126] Namecoin. Decentralized secure names. `https://www.namecoin.org/resources/whitepaper/`. Accessed: 2025.

[127] Arvind Narayanan and Vitaly Shmatikov. Myths and fallacies of "personally identifiable information". *Communications of the ACM*, 53(6):24–26, 2010.

[128] Roger M Needham and Michael D Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12), 1978.

[129] F.J. Nijeboer. Detection of HTTPS encrypted DNS traffic. `http://essay.utwente.nl/82085/`, 2020.

[130] Gunter Ollmann. The pharming guide: understanding & preventing DNS-related attacks by phishers. Technical report, 2005.

[131] Constantinos Patsakis, Fran Casino, and Vasilios Katos. Encrypted and covert DNS queries for botnets: challenges and countermeasures. *Computers & Security*, 2020.

[132] Paul Pearce, Ben Jones, Frank Li, Roya Ensafi, Nick Feamster, Nick Weaver, and Vern Paxson. Global measurement of DNS manipulation. In *USENIX Security Symposium*, 2017.

[133] Roberto Perdisci, Manos Antonakakis, Xiapu Luo, and Wenke Lee. WSEC DNS: protecting recursive DNS resolvers from poisoning attacks. In *IEEE/IFIP International Conference on Dependable Systems & Networks*, 2009.

[134] Daniel Plohmann, Khaled Yakdan, Michael Klatt, Johannes Bader, and Elmar Gerhards-Padilla. A comprehensive measurement study of domain generating malware. In *USENIX Security Symposium*, 2016.

[135] The Tor Project. Reporting bad relays. `https://trac.torproject.org/projects/tor/wiki/doc/ReportingBadRelays`, 2018.

[136] J. Purushothaman, E. Thompson, and A. Abdou. Certificate root stores—an area of unity or disparity? In *USENIX Workshop on Cyber Security Experimentation and Test (CSET)*, 2022.

[137] Venugopalan Ramasubramanian and Emin Gün Sirer. Perils of transitive trust in the domain name system. In *ACM Internet Measurement Conference (IMC)*, 2005.

[138] Reethika Ramesh, Leonid Evdokimov, Diwen Xue, and Roya Ensafi. VPNalyzer: systematic investigation of the VPN ecosystem. In *Network and Distributed System Security Symposium (NDSS)*, 2022.

[139] Audrey Randall, Enze Liu, Gautam Akiwate, Ramakrishna Padmanabhan, Geoffrey M. Voelker, Stefan Savage, and Aaron Schulman. Trufflehunter:

cache snooping rare domains at large public DNS resolvers. In *ACM Internet Measurement Conference (IMC)*, 2020.

[140] Tirumaleswar Reddy, Dan Wing, and Prashanth Patil. DNS over Datagram Transport Layer Security (DTLS). RFC 8094, 2017.

[141] Eric Rescorla, Kazuho Oku, Nick Sullivan, and Christopher A. Wood. TLS encrypted Client Hello, September 2024. Internet-Draft (IETF).

[142] Eric Rescorla, Hannes Tschofenig, and Nagendra Modadugu. The Datagram Transport Layer Security (DTLS) Protocol Version 1.3. RFC 9147, 2022.

[143] Antony Rowstron and Peter Druschel. Pastry: scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms*. Springer, 2001.

[144] Mahrud Sayrafi. Introducing DNS resolver for Tor. `https://blog.cloudflare.com/welcome-hidden-resolver/`, 2018.

[145] Giovanni Schmid. Thirty years of DNS insecurity: current issues and perspectives. *IEEE Communications Surveys & Tutorials*, 23(4), 2021.

[146] Benedikt Schmidt, Simon Meier, Cas Cremers, and David Basin. Automated analysis of Diffie-Hellman protocols and advanced security properties. In *Computer Security Foundations Symposium*. IEEE, 2012.

[147] Paul Schmitt, Anne Edmundson, and Nick Feamster. Oblivious DNS: practical privacy for DNS queries. In *Symposium on Privacy Enhancing Technologies (PETS)*, 2019.

[148] Haya Shulman. Pretty bad privacy: pitfalls of DNS encryption. In *Workshop on Privacy in the Electronic Society (WPES)*, 2014.

[149] Sandra Siby, Ludovic Barman, Christopher Wood, Marwan Fayed, Nick Sullivan, and Carmela Troncoso. You get padding, everybody gets padding! You get privacy? evaluating practical QUIC website fingerprinting protections for the masses. `https://arxiv.org/abs/2203.07806`, 2022.

[150] Sandra Siby, Marc Juarez, Claudia Diaz, Narseo Vallina-Rodriguez, and Carmela Troncoso. Encrypted DNS -> privacy? a traffic analysis perspective. In *Network and Distributed System Security Symposium (NDSS)*, 2020.

[151] Kushagra Singh, Gurshabad Grover, and Varun Bansal. How India censors the web. In *ACM Conference on Web Science (WebSci)*, 2020.

[152] Sooel Son and Vitaly Shmatikov. The hitchhiker's guide to DNS cache poisoning. In *International Conference on Security and Privacy in Communication Systems*. Springer, 2010.

[153] Etienne Stalmans and Barry Irwin. A framework for DNS based detection and mitigation of malware infections on a network. In *2011 Information Security for South Africa*, pages 1–8, 2011.

[154] Jacob Steadman and Sandra Scott-Hayward. DNSxD: detecting data exfiltration over DNS. In *Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE, 2018.

[155] Douglas Stebila and Nick Sullivan. An analysis of TLS handshake proxying. In *2015 IEEE Trustcom/BigDataSE/ISPA*, volume 1, 2015.

[156] Ion Stoica, Robert Morris, David Karger, M Frans Kaashoek, and Hari Balakrishnan. Chord: a scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM Computer Communication Review*, 31(4), 2001.

[157] Nick Sullivan. Keyless SSL: the nitty gritty technical details. `https://blog.cloudflare.com/keyless-ssl-the-nitty-gritty-technical-details/`, 2014.

[158] Kutub Thakur, Md Liakat Ali, Sandra Kopecky, Abu Kamruzzaman, and Lixin Tao. Connectivity, traffic flow and applied statistics in cyber security. In *IEEE International Conference on Smart Cloud (SmartCloud)*, 2016.

[159] Martin Thomson and Sean Turner. Using TLS to Secure QUIC. RFC 9001, 2021.

[160] Sadegh Torabi, Amine Boukhtouta, Chadi Assi, and Mourad Debbabi. Detecting Internet abuse by analyzing passive DNS traffic: a survey of implemented

systems. *IEEE Communications Surveys & Tutorials (COMST)*, 20:3389–3415, 2018.

[161] Paul C. van Oorschot. *Computer Security and the Internet: Tools and Jewels From Malware to Bitcoin (2nd edition)*. Springer International, 2021.

[162] Roland van Rijswijk-Deij, Anna Sperotto, and Aiko Pras. DNSSEC and its potential for DDoS attacks: a comprehensive measurement study. In *ACM Internet Measurement Conference (IMC)*, 2014.

[163] Dmitrii Vekshin, Karel Hynek, and Tomas Cejka. DoH insight: detecting DNS over HTTPS by machine learning. In *ACM International Conference on Availability, Reliability and Security (ARES)*, 2020.

[164] Thomas Vissers, Wouter Joosen, and Nick Nikiforakis. Parking sensors: analyzing and detecting parked domains. In *Network and Distributed System Security Symposium (NDSS)*, 2015.

[165] Paul A. Vixie. Extension Mechanisms for DNS (EDNS0). RFC 2671, 1999.

[166] Jan Včelák, Sharon Goldberg, Dimitrios Papadopoulos, Shumon Huque, and David C Lawrence. NSEC5, DNSSEC authenticated denial of existence, 2018. Internet-Draft (IETF).

[167] Matthias Wachs, Martin Schanzenbach, and Christian Grothoff. A censorship-resistant, privacy-enhancing and fully decentralized name system. In *Cryptology and Network Security*. Springer, 2014.

[168] Matthaus Wander. An overview of secure name resolution. `https://media.ccc.de/v/29c3-5146-en-an_overview_of_secure_name_resolution_h264`, 2012.

[169] Von Welch, Ian Foster, Carl Kesselman, Olle Mulmo, Laura Pearlman, Steven Tuecke, Jarek Gawor, Sam Meder, and Frank Siebenlist. X.509 proxy certificates for dynamic delegation. In *PKI R&D workshop*, volume 14, 2004.

[170] Johannes Wilson, Mikael Asplund, and Niklas Johansson. Extending the authentication hierarchy with one-way agreement. In *Computer Security Foundations Symposium (CSF)*. IEEE, 2023.

[171] Philipp Winter, Richard Köwer, Martin Mulazzani, Markus Huber, Sebastian Schrittwieser, Stefan Lindskog, and Edgar Weippl. Spoiled onions: exposing malicious Tor exit relays. In *Symposium on Privacy Enhancing Technologies (PETS)*, 2014.

[172] He Yan, Eric Osterweil, Jon Hajdu, Jonas Acres, and Dan Massey. Limiting replay vulnerabilities in DNSSEC. In *IEEE Workshop on Secure Network Protocols*, 2008.

[173] Mengqi Zhan, Yang Li, Guangxi Yu, Bo Li, and Weiping Wang. Detecting DNS over HTTPS based data exfiltration. *Computer Networks*, 209:108919, 2022.

[174] Fangming Zhao, Yoshiaki Hori, and Kouichi Sakurai. Analysis of privacy disclosure in DNS query. In *2007 International Conference on Multimedia and Ubiquitous Engineering (MUE)*, pages 952–957. IEEE, 2007.

[175] Yury Zhauniarovich, Issa Khalil, Ting Yu, and Marc Dacier. A survey on malicious domains detection through DNS data analysis. *ACM Comput. Surv.*, 51(4), 2018.

[176] Liang Zhu, Zi Hu, John Heidemann, Duane Wessels, Allison Mankin, and Nikita Somaiya. Connection-oriented DNS to improve privacy and security. In *IEEE Symposium on Security and Privacy (S&P)*, 2015.

[177] Futai Zou, Siyu Zhang, Bei Pei, Li Pan, Linsen Li, and Jianhua Li. Survey on domain name system security. In *IEEE First International Conference on Data Science in Cyberspace (DSC)*, 2016.