Privacy-preserving communications for IoT based on DNS and its security extensions

Ibrahim Ayoub

université PARIS-SACLAY

Privacy-preserving communications for IoT based on DNS and its security extensions

Communications préservant la confidentialité pour l'IoT basées sur le DNS et ses extensions de sécurité

Thèse de doctorat de l'université Paris-Saclay

École doctorale n° 580, Sciences et Technologies de l'information et de la communication (STIC)
Spécialité de doctorat: Sciences des réseaux, de l'information et de la communication
Graduate School : informatique et sciences du numérique
Référent : Université de Versailles-Saint-Quentin-en-Yvelines

Thèse préparée dans l'unité de recherche **DAVID (Université Paris-Saclay, UVSQ)**, sous la direction de **Kinda KHAWAM**, Maître de Conférences/HDR et le co-encadrement de **Sandoche BALAKRICHENAN**, Ingénieur Recherche et Développement chez Afnic

Thèse soutenue à Versailles, le 28 novembre 2024, par

Ibrahim AYOUB

Composition du jury

Membres du jury avec voix délibérative

Véronique VEQUE Professeur, Université Paris-Saclay	Présidente
Abdallah MAKHOUL Professeur, Université de Franche-Comté	Rapporteur & Examinateur
Laurent TOUTAIN Professeur, IMT Atlantique	Rapporteur & Examinateur
Gregory BLANC Enseignant-Chercheur (ENAC, ISAE), Télécom SudParis	Examinateur

THESE DE DOCTORAT

NNT : 2024UPASG074

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre: Communications préservant la confidentialité pour l'IoT basées sur le DNS et ses extensions de sécurité

Mots clés: IdO, DNS, Confidentialité, Sécurité, DANE, DNSSEC

Résumé: Les technologies de l'Internet des Objets (IoT) ont transformé notre manière d'interagir avec le monde et les machines, devenant une partie intégrante de notre quotidien. Cette thèse vise à relever certains des défis rencontrés dans les environnements IoT en utilisant le Domain Name System (DNS) ainsi que ses extensions et protocoles de sécurité. Bien que le DNS soit principalement un système de recherche distribué permettant de mapper les noms de domaine aux adresses IP, il a considérablement évolué grâce à diverses de ses extensions. Cette évolution lui a permis de jouer un rôle plus large, notamment en atténuant certains des défis liés aux environnements IoT.

Notre première contribution identifie quatre grandes catégories de défis de l'IoT : la nature contrainte des dispositifs IoT, l'identification des objets de l'IoT, leur sécurité et interopérabilité. Nous effectuons également une revue de la littérature pour examiner comment le DNS est utilisé dans la recherche et l'industrie pour répondre à ces défis. La deuxième contribution propose l'utilisation du DNS-based Authentication of Named Entities (DANE), un protocole DNS conçu pour renforcer l'Infrastructure à Clé Publique (PKI), afin de mettre en place un mécanisme d'authentification mutuelle entre deux serveurs backend LoRaWAN, sécurisant la connexion sans dépendre des autorités de certification commerciales (CAs).

La troisième contribution présente LoRaDANCE, un mécanisme de sécurité qui permet à un *End-Device (ED)* LoRaWAN de rejoindre un réseau sans avoir à pré-partager des clés secrètes avec les serveurs backend, comme requis dans le LoRaWAN standard. L'authentification mutuelle avec le Join Server (JS) est assurée par le protocole DANE, tandis que la cryptographie asymétrique permet au dispositif et au serveur de générer la clé secrète nécessaire, éliminant ainsi le besoin de clés secrètes pré-partagées.

Pour notre quatrième contribution, nous avons mené une étude approfondie des noms de domaine IoT et évalué les différences entre ces derniers et les noms de domaine non-IoT. Dans ce contexte, les noms de domaine IoT font référence à ceux des serveurs backend IoT résolus via DNS, tandis que les noms de domaine non-IoT correspondent aux serveurs accessibles par des dispositifs génériques. L'étude a été réalisée en trois phases : une analyse statistique, une analyse DNS, et une classification des deux classes de noms de domaine à l'aide de l'apprentissage automatique.

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Title: Privacy-preserving communications for IoT based on DNS and its security extensions
Keywords: IoT, DNS, privacy, security, DANE, DNSSEC

Abstract: The Internet of Things (IoT) technologies have transformed how we interact with the world and machines, becoming an integral part of our daily lives. This thesis aims to address some of the challenges faced by IoT environments using the Domain Name System (DNS) and its security extensions and protocols. While DNS is primarily a distributed lookup system that maps domain names to IP addresses, it has evolved significantly through various extensions and DNS-based protocols. This evolution has enabled DNS to play a broader role, particularly in mitigating some of the challenges in IoT environments. Our first contribution identifies four major categories of IoT challenges: the constrained nature of IoT devices, identification in IoT, IoT security, and interoperability. We also conduct a literature review to examine how DNS is used in both research and industry to address these challenges. The second contribution proposes using DNS-based Authentication of Named Entities (DANE), a DNS protocol designed to strengthen Public Key Infrastructure (PKI), to establish a mutual authentication mechanism between two LoRaWAN backend servers, securing the connection without relying on commercial Certificate Authorities (CAs). The third contribution, introduces LoRaDANCE, a security mechanism that allows a LoRaWAN *ED* to join a network without pre-sharing any secret keys with the backend servers, as required in standard LoRaWAN. Mutual authentication with the Join Server (JS) is ensured through DANE, while asymmetric cryptography enables the device and server to generate the necessary secret key, eliminating the need for pre-shared secret keys. For our fourth contribution, we conducted an in-depth study of IoT domain names and evaluated the differences between them and non-IoT domain names. In this context, IoT domain names refer to those of IoT backend servers resolved via DNS, whereas non-IoT domain names correspond to servers accessed by generic devices and humans. The study was carried out in three phases: a statistical analysis, a DNS analysis, and a machine learning-based classification of the two domain name classes.

To my family ..

Acknowledgments

Completing this PhD has been one of the most rewarding yet challenging experiences of my life. It is a journey that I could not have undertaken alone, and I am deeply grateful to those who supported and guided me throughout these years.

First and foremost, I would like to express my heartfelt gratitude to my supervisors, Kinda Khawam and Sandoche Balakrichenan. The journey was marked by its fair share of ups and downs, but your guidance and perseverance ensured that we made it to the finish line together. Thank you for your unwavering commitment and belief in my work.

I would also like to extend my sincere thanks to Afnic as a whole and, in particular, the Labs Department for welcoming me and providing a supportive environment throughout my three years of research. A special thanks to Benoît for your steadfast support since day one—it has meant so much to me.

To my colleagues, Hélène, Caroline, Gaël, Alexandre, Thomas, Michaël, Marc, Rémy, and Lotfi, thank you for your encouragement, collaboration, and friendship. It has been a pleasure to work alongside such talented individuals. My heartfelt thanks also go out to Samia, who left us far too soon—your memory remains with us.

Finally, I am deeply grateful to my friends and family for their constant love, patience, and encouragement.

Résumé

Les technologies de l'Internet des Objets (IoT) ont évolué de manière significative au cours des dernières années, devenant une partie intégrante de nombreux aspects de notre quotidien. Les objets autrefois considérés comme isolés et sans besoin de connectivité réseau sont désormais couramment connectés, et il est difficile d'imaginer leur fonctionnement autrement. Ces objets connectés, ou "things", vont de petits capteurs et gadgets domestiques, qui sont le sujet de ce travail, à de grands véhicules, machines industrielles, et bien plus encore.

Le succès de l'IoT a attiré de nombreux investisseurs et innovateurs, chacun contribuant à l'évolution rapide de l'écosystème. Cela a abouti à la création d'un large éventail de technologies et de standards IoT, conduisant à un paysage riche et diversifié, mais également hautement fragmenté. L'adoption généralisée de l'IoT et la fragmentation de son écosystème ont fait émerger plusieurs défis. Ce travail vise à adresser certains des défis auxquels sont confrontés les environnements IoT en utilisant le Domain Name System (DNS) et ses extensions, ainsi que certains protocoles de sécurité. Bien que le DNS soit principalement un système de recherche distribué conçu pour mapper les noms de domaine aux adresses IP, il a évolué de manière significative grâce à diverses extensions. Cette évolution lui a permis de jouer un rôle plus large, notamment en tant que contributeur clé à la mitigation de certains des défis auxquels sont confrontés les environnements IoT.

Les objectifs globaux de cette thèse sont les suivants : premièrement, nous identifions et catégorisons les défis dans les environnements IoT et examinons comment le DNS est actuellement utilisé pour les adresser. Ensuite, nous utilisons le DNS pour implémenter nos propres solutions afin de résoudre certains de ces défis.

State of the Art: DNS to Address IoT Challenges

Pour notre première contribution, nous identifions quatre catégories principales de défis auxquels l'IoT est confronté et étudions comment le DNS est utilisé dans la recherche et l'industrie pour s'attaquer à un nombre significatif de ces défis. Le premier défi concerne la nature contrainte des dispositifs IoT. La majorité des dispositifs IoT sont limités en termes de puissance de traitement, de mémoire et de ressources énergétiques. Ces contraintes réduisent considérablement les choix que les fabricants peuvent faire concernant les systèmes d'exploitation qu'ils utilisent et les mécanismes de sécurité qu'ils déploient, faisant des dispositifs IoT le maillon faible en termes de performance et de sécurité. Le deuxième défi concerne l'identification dans l'IoT, car la diversité des technologies et des standards IoT aboutit à de nombreux schémas de nommage et d'identification souvent incompatibles. Le troisième défi concerne la sécurité dans l'IoT. Les contraintes des dispositifs IoT les empêchent de mettre en œuvre des mécanismes de sécurité à la pointe, les exposant à une gamme d'attaques et compromettant potentiellement les réseaux auxquels ils sont connectés. Le quatrième défi est le manque d'interopérabilité entre les technologies IoT. Ce problème provient de l'absence de standardisation face à la grande variété de technologies IoT, dont la plupart utilisent des protocoles, schémas de nommage et formats de données propriétaires, rendant les tech-

nologies IoT incompatibles et divisant fortement l'écosystème IoT.

Le DNS, tel qu'observé dans les derniers articles de recherche et standards industriels, apparaît comme un outil approprié pour relever plusieurs des défis auxquels l'IoT est confronté. Notre première contribution est complétée par une revue de la littérature qui montre comment le DNS est utilisé pour répondre partiellement ou totalement à chaque défi dans les environnements IoT. Par exemple, les protocoles DNS adaptés aux dispositifs contraints servent ces dispositifs IoT en leur fournissant la capacité de résoudre efficacement les noms de domaine dans les limites de leurs ressources. Pour l'identification IoT, la fonction originelle du DNS, qui est de résoudre les noms en identifiants, peut être adaptée aux dispositifs IoT nécessitant un identifiant. En fait, de nombreux schémas d'identification IoT, tels que l'Object Name Service (ONS) pour les Electronic Product Codes (EPC) et l'OID Resolution System (ORS) pour les Object Identifiers (OID), utilisent déjà cette approche. Pour la sécurité IoT, les DNS Security Extensions (DNSSEC) et les protocoles basés sur le DNS, tels que DANE, améliorent la sécurité des communications en fournissant l'authentification et l'intégrité. Le DNS contribue également à l'atténuation des défis d'interopérabilité grâce à des protocoles standardisés pour la découverte et la communication des dispositifs. Cependant, l'exposition du DNS à l'IoT doit être abordée avec prudence en raison des répercussions potentielles des dispositifs IoT compromis, qui pourraient être exploités pour mener des attaques basées sur le DNS ou perturber le processus de résolution, compromettant ainsi la sécurité et la fiabilité globales du réseau.

Mutual Authentication of IoT Backend Servers

La deuxième contribution consistait à utiliser DANE pour implémenter un mécanisme d'authentification mutuelle entre deux serveurs backend IoT.

DANE est un protocole DNS qui permet à un propriétaire de domaine de publier un enregistrement TLSA dans ses zones DNS, liant son nom de domaine à sa clé publique ou à son certificat numérique. Cela permet de valider la clé publique ou le certificat lors des handshakes TLS, similaire au rôle des CA traditionnelles. DANE a été proposé comme un complément ou un remplaçant potentiel des CA traditionnelles, selon son utilisation. Cependant, à l'origine, DANE ne validait que les clés publiques brutes ou les certificats des serveurs, pas ceux des clients.

Pour réaliser l'authentification mutuelle entre le client et le serveur TLS, nous avons implémenté deux drafts Internet émis par le groupe de travail IETF DANCE (DANE Authentication for Network Clients Everywhere). Ces drafts montrent comment DANE peut être utilisé pour valider les clés publiques brutes ou les certificats numériques des clients TLS, permettant ainsi une authentification mutuelle. La technologie IoT à laquelle nous avons appliqué cela est LoRaWAN. Nous avons implémenté notre solution sur un véritable *End-Device (ED)* LoRaWAN, en utilisant DANE pour permettre l'authentification mutuelle entre le NS et le JS pendant le processus de join. Nous avons évalué les performances de notre solution en analysant la durée du processus join tout en variant la localisation du résolveur DNS, et en étudiant l'impact du caching DNS sur les performances globales.

LoRaDANCE: Using The DNS for Mutual Authentication Without Pre-shared Keys

La troisième contribution a proposé LoRaDANCE, un mécanisme permettant à un *ED* LoRaWAN de joindre un réseau LoRaWAN sans avoir besoin de pré-partager l'*AppKey*, qui est la clé secrète utilisée dans les réseaux LoRaWAN pour générer des clés de session pendant le processus join. LoRaDANCE facilite également l'authentification mutuelle entre le *ED* et le JS en utilisant DANE. Comme son nom l'indique, LoRaDANCE s'inspire de DANCE, en utilisant DANE pour l'authentification mutuelle, bien que ce ne soit pas

dans le contexte conventionnel d'un client et d'un serveur TLS.

Nous avons implémenté notre solution sur un véritable *ED* LoRaWAN, en utilisant DANE pour permettre l'authentification mutuelle entre le *ED* et le JS pendant le processus de join. La clé secrète est dérivée en utilisant Elliptic Curve Diffie-Hellman. Enfin, nous avons évalué les performances de LoRaDANCE en étudiant la durée du processus join et sa consommation d'énergie.

IoT Backend Servers: Studying IoT Domain Names

La quatrième contribution est une étude sur les serveurs backend IoT, en se concentrant sur les noms de domaine de ces serveurs. Les dispositifs IoT sont rarement autonomes. Ils doivent généralement contacter des serveurs backend pour transmettre des informations, recevoir des commandes ou obtenir des mises à jour de firmware. Ces dispositifs sont généralement équipés des noms de domaine de ces serveurs, qu'ils résolvent via le DNS. Dans ce contexte, un nom de domaine IoT fait référence au nom de domaine d'un serveur contacté par des dispositifs IoT, spécifiquement le nom de domaine d'un serveur backend IoT. Par exemple, un serveur backend IoT pourrait être un serveur qui reçoit et stocke des séquences vidéo provenant de caméras IP.

Nous avons compilé une liste de noms de domaine IoT réels à l'aide de jeux de données provenant d'études antérieures qui utilisaient des bancs d'essai avec de véritables dispositifs IoT. De plus, nous avons utilisé deux listes publiques des sites web les plus visités comme noms de domaine non-IoT. Notre étude s'est déroulée en trois phases.

La première phase est une comparaison statistique entre les noms de domaine IoT et non-IoT, basée sur leur longueur maximale, minimale et moyenne (en caractères) et le nombre d'étiquettes. Cette phase comprenait également une analyse des étiquettes les plus fréquentes dans chaque jeu de données.

La deuxième phase est une analyse du DNS. Pour chaque jeu de données, nous avons résolu plusieurs enregistrements de ressources (RR). Pour chaque jeu de données et chaque RR, nous avons calculé le pourcentage de noms de domaine ayant ce RR, la durée moyenne des requêtes, le Time-to-Live (TTL) moyen, et la taille moyenne des réponses.

La troisième et dernière phase consiste à entraîner plusieurs modèles d'apprentissage automatique pour classifier les noms de domaine IoT et non-IoT. L'objectif de cette phase est d'identifier des différences intrinsèques entre les deux classes de noms de domaine, qui ne peuvent être détectées visuellement ou par une analyse statistique.

Abstract

The Internet of Things (IoT) technologies have transformed how we interact with the world and machines, becoming an integral part of our daily lives. This thesis aims to address some of the challenges faced by IoT environments using the Domain Name System (DNS) and its security extensions and protocols. While DNS is primarily a distributed lookup system that maps domain names to IP addresses, it has evolved significantly through various extensions and DNS-based protocols. This evolution has enabled DNS to play a broader role, particularly in mitigating some of the challenges in IoT environments. Our first contribution identifies four major categories of IoT challenges: the constrained nature of IoT devices, identification in IoT, IoT security, and interoperability. We also conduct a literature review to examine how DNS is used in both research and industry to address these challenges. The second contribution proposes using DNS-based Authentication of Named Entities (DANE), a DNS protocol designed to strengthen Public Key Infrastructure (PKI), to establish a mutual authentication mechanism between two LoRaWAN backend servers, securing the connection without relying on commercial Certificate Authorities (CAs). The third contribution, introduces LoRaDANCE, a security mechanism that allows a LoRaWAN *ED* to join a network without pre-sharing any secret keys with the backend servers, as required in standard LoRaWAN. Mutual authentication with the Join Server (JS) is ensured through DANE, while asymmetric cryptography enables the device and server to generate the necessary secret key, eliminating the need for pre-shared secret keys. For our fourth contribution, we conducted an in-depth study of IoT domain names and evaluated the differences between them and non-IoT domain names. In this context, IoT domain names refer to those of IoT backend servers resolved via DNS, whereas non-IoT domain names correspond to servers accessed by generic devices and humans. The study was carried out in three phases: a statistical analysis, a DNS analysis, and a machine learning-based classification of the two domain name classes.

List of Abbreviations

DNS	Domain Name System
RFC	Request for Comments
TLD	Top-Level Domain
SLD	Second-Level Domain
gTLD	Generic Top-Level Domain
ccTLD	Country Code Top-Level Domain
RR	Resource Record
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
QNAME	Query Name
DNSSEC	Domain Name System Security Extensions
DS	Delegation Signer
ZSK	Zone Signing Key
KSK	Key Signing Key
DANE	DNS-based Authentication of Named Entities
PKI	Public Key Infrastructure
CA	Certificate Authority
TLS	Transport Layer Security
DoS	Denial of Service
DDoS	Distributed Denial of Service
NS	Nameserver
DoT	DNS-over-TLS
IETF	Internet Engineering Task Force
QTYPE	Query Type
ODNS	Oblivious DNS
IoT	Internet of Things
LoRaWAN	LoRa Wide Area Network
ITU	International Telecommunication Union
ITU-T	International Telecommunication Union Standardization Sector
RFID	Radio Frequency Identification

SOA	Service Oriented Architecture
IoMT	Internet of Medical Things
IIoT	Industrial Internet of Things
CoAP	Constrained Application Protocol
AIoTI	Alliance for Internet of Things Innovation
IP	Internet Protocol
DOI	Digital Object Identifier
EPC	Electronic Product Code
ONS	Object Name Service
OID	Object Identifier
ORS	OID Resolution System
API	Application Programming Interface
WoT	Web of Things
W3C	World Wide Web Consortium
DoC	DNS over CoAP
DTLS	DNS over Datagram Transport Layer Security
ENUM	Electronic NUmber Mapping
DNSNA	DNS Name Autoconfiguration
ND	Neighbor Discovery
DHCP	Dynamic Host Configuration Protocol
SDNSNA	Secure Domain Name System Name Autoconfiguration
C&C	Command & Control
DNS-SD	DNS-Based Service Discovery
mDNS	Multicast DNS
SIoT	Social Internet of Things
DSN	Distributed Sensor Network
PKIX	Public Key Infrastructure using X.509 digital certificates
DANE	DANE Authentication for Network Clients Everywhere
WG	Working Group
TTL	Time To Live
MTU	Maximum Transfer Unit
ED	End Device
JS	Join Server
AS	Application Server
ECDH	Elliptic Curve Diffie–Hellman
OTAA	Over-The-Air Activation
ABP	Activation by Personalization
PHYPayload	Physical Payload

MIC	Message Integrity Code
SoA	Start of Authority
CBOR	Concise Binary Object Representation
CNAME	Canonical Name
CAA	Certification Authority Authorization
M2M	Machine-to-Machine
DGA	Domain Generation Algorithm
NLP	Natural Language Processing
TF-IDF	Term Frequency-Inverse Document Frequency
NB	Naïve Bayes
LR	Logistic Regression
KNN	K-Nearest Neighbors
SVM	Support Vector Machine
DT	Decision Tree
RF	Random Forest
CBOW	Continuous Bag-of-Words
PDF	Probability Density Function
ROC	Receiver Operation Characteristic
AUC	Area Under the Curve

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

Chapter Overview

This chapter introduces the core concepts of this thesis. This work revolves around using the Domain Name System (DNS) in Internet of Things (IoT) environments to reinforce their security properties. To set the foundation for the rest of this thesis, we begin by defining IoT, providing a generalized definition that encompasses the wide range of technologies, protocols, and standards that make up the IoT ecosystem, as well as the various applications of IoT technologies. Understanding the diversity and complexity of IoT is essential for exploring how DNS can play a critical role in securing these interconnected devices and networks, as will be examined in the following chapters.

The second part of this chapter introduces LoRaWAN (Long Range Wide Area Network), the technology of choice for our implementations in this thesis. We introduce the basic architecture of the LoRaWAN network and provide an in-depth view into the join process, *i.e.*, the process by which a LoRaWAN *End-Device (ED)* joins a LoRaWAN network.

The third part introduces the Domain Name System (DNS), covering both its architecture and the DNS resolution process. Additionally, it explores several DNS security and privacy extensions, along with DNS-based protocols.

Finally, the thesis objectives and contributions are presented.

1.1 The Internet of Things (IoT)

Even though the Internet of Things (IoT) has been around for a while—the term *Internet of Things* was coined in 1999 by Kevin Ashton [48])—it is only in recent years that it has gained significant attention in both industry and research. The past few years have seen rapid IoT integration across various industries and research domains. Unlike generic Internet devices which are predominantly IP-based and use the TCP/IP protocol suite—such as servers, personal computers, and smart devices—IoT is not a single technology that is governed by a single organization. Consequently, it cannot be encapsulated by a single definition or technology framework. IoT is, in fact, a general term that encompasses a wide range of technologies with significant diversity in communication protocols, data representation, and transmission methods.

In the following, we provide the definition of IoT that is adopted by vendor-neutral regulatory bodies, and give an overview of IoT architectures and applications.

1.1.1 Definition of IoT

A standard definition of IoT has yet to be agreed upon, but a definition can be inferred from its applications. Based on interactions with the various, diverse objects collectively referred to as IoT, it can be defined as the network formed by connecting physical or virtual objects to the Internet or other communication networks. A large portion of the devices that now constitute IoT were not traditionally considered to require Internet connectivity, or any form of network connectivity. Previously viewed as standalone and isolated, everyday objects and devices are increasingly being equipped with network capabilities, making communication with such devices more commonplace. Nevertheless, defining IoT in this manner does not hint at any specific technology, standards, data formats, hardware, or software. This is due to the rich and diverse ecosystem that is IoT. It encompasses various technologies across the electromagnetic spectrum with a wide range of often non-interoperable standards and protocols. Technologies such as Narrowband IoT (NB_IoT) [22] [127], Bluetooth Low Energy (BLE) [2], Sigfox [21], Zigbee [24], and LoRaWAN [15] can all be labeled as IoT.

Hence, it is useful to turn to regulatory bodies for a more formal definition, which, while not necessarily providing specific technical specifications, captures the essence of what IoT is and how its technologies should be conceived and implemented. For example, the ITU Telecommunication Standardization Sector (ITU-T) defines IoT in Y4000: Overview of the Internet of things [32] as *a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies (ICT)*.

The devices in an IoT network, again, according to Y4000[32], could be:

- **Data-carriers:** These are usually static, such as barcodes, and attached to physical things.
- **Data-carrying devices:** These devices might have the information stored in them altered by a data-capturing device, *e.g.*, Radio-frequency identification (RFID).
- **Data-capturing devices:** These devices can read from or write to a physical thing. The physical thing could be a data-carrier or a data-carrying device.

- **Sensing and actuating devices:** This category includes sensors that can interact with their environment, gather data and measurements, and send them over the network for processing. This category also includes actuators that can perform operations in their environment based on information received over the network or their measurement of the environment around them.
- **General devices:** These include industrial machinery, home appliances, personal computers, and mobile phones that have embedded processing and wired or wireless communication capabilities.

1.1.2 IoT Architecture

The lack of specificity in IoT definitions also extends to its architecture. Unlike TCP/IP communication, which follows a single model—or two if we consider the OSI reference model—IoT has multiple architectures, none of which is standardized. The most notable are the three-layer [36][124], four-layer [98] also referred to as Service Oriented Architecture (SOA), and five-layer [124] architectures. See Figure 1.1.

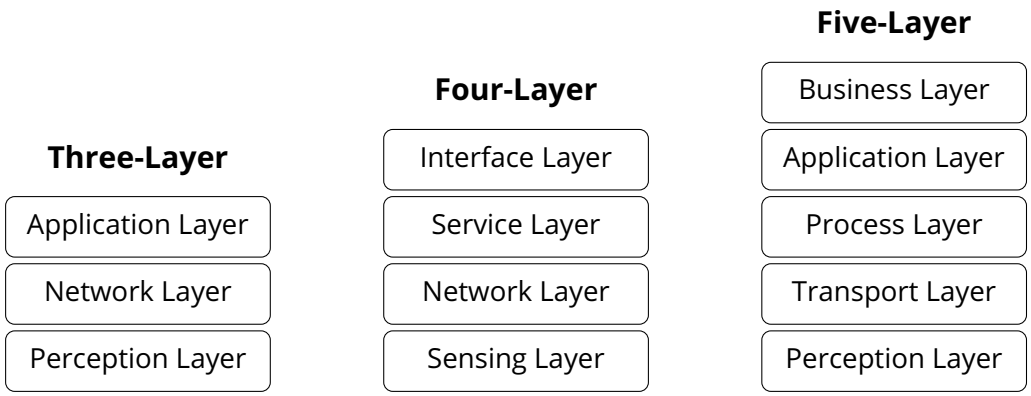


Figure 1.1: IoT architectures.

Three-Layer Architecture

- **Perception Layer:** This layer contains devices like sensors, actuators, barcodes, and RFID tags. It is mainly the layer that interacts with the environment to collect data to be sent to the upper layers.
- **Network Layer:** This layer receives data from the Perception layer. It includes the network infrastructure that transfers data between the Perception and Application layers.
- **Application Layer:** This layer processes and analyzes data passed through the Network layer.

Four-Layer (SOA) Architecture

- **Sensing Layer:** This layer is similar to the Perception layer in the three-layer model. It contains sensing tools to perform the measurements in the environments of these devices.
- **Network Layer:** This layer is the network infrastructure (wired or wireless) that ensures connectivity between things among themselves and between things and their backend.
- **Service Layer:** This layer stores and processes information. Services required by users are created and managed here.

- **Interface Layer:** This layer defines interaction rules between users and devices. It also attempts to solve compatibility issues between devices from different vendors that follow different standards.

Five-Layer Architecture

- **Perception Layer:** This layer is similar to the Perception Layer in the three-layer architecture.
- **Transport Layer:** Also referred to as network layer. This layer receives data from the Perception layer and passes them to the Processing layer.
- **Processing Layer:** This layer processes and analyzes data passed through the Network layer.
- **Application Layer:** Data from the Processing layer are used here for IoT applications.
- **Business Layer:** This layer contains the management of the IoT system.

1.1.3 Applications of IoT

Applications of IoT are numerous and diverse. Smart cities use IoT devices to monitor traffic conditions [90], predict pollution levels [82], and provide smart parking [68]. Furthermore, cities deploy IoT networks for security purposes such as asset monitoring [81] and identification [103]. The agriculture industry benefits from IoT for produce distribution [104], supply chain management [60], and overall smart agriculture [52].

IoT plays a significant role in healthcare and is commonly known as the Internet of Medical Things (IoMT). IoMT helps improve the quality of life while decreasing the pressure on the medical system by allowing patients to self-diagnose when possible or to get a healthcare professional's recommendation from a distance [191][39]. Care for the elderly [191], monitoring the state of mind [159], physical activity [116], and even eating habits [171] are a few examples of what IoMT has to offer in terms of ameliorating the healthcare system.

The digital transformation also found its way to different industries leading to what is referred to as the fourth industrial revolution (Industry 4.0) with the Industrial Internet of Things (IIoT) [164]. In addition, IoT proved to be helpful in emergencies where communication with the individuals at risk is of utmost importance [144] [130] [201] [61]. Finally, in light of the COVID-19 pandemic, IoT was used to help trace, detect and consequently mitigate the spread of the virus [183] [157].

1.2 LoRaWAN

In this thesis, we focus on LoRaWAN as the primary IoT technology. The following sections introduce the LoRaWAN architecture and the join process through which a LoRaWAN *ED* connects to a network.

1.2.1 The LoRaWAN Architecture

LoRaWAN (Long Range Wide Area Network) is a wireless communication protocol and is considered one of the most popular Low-Power Wide-Area Network (LPWAN) technologies. LoRaWAN excels in providing long-range, low-power communications which enables communications for constrained, battery-fed devices over wide geographical areas.

The basic architecture of a LoRaWAN network is presented in Figure 1.2. LoRaWAN *EDs* use the LoRa modulation technologies over unlicensed radio frequency bands to communicate with radio *gateways*

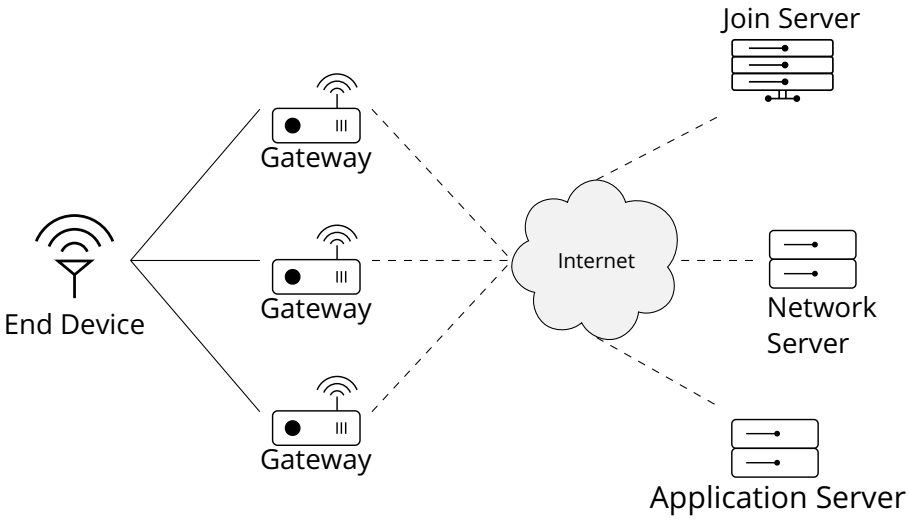


Figure 1.2: The LoRaWAN architecture.

which act as a relay between the radio and IP space where the LoRaWAN backend servers reside. The LoRaWAN backend servers are IP-enabled servers connected to the Internet and are:

- **The Network Server (NS):** The NS facilitates communication between LoRaWAN *EDs* and the network through radio *gateways*. It manages device identification by generating a *DevAddr* (Device Address) for each *ED*, allowing it to be recognized within the network. The NS also ensures device authenticity and integrity, preventing unauthorized devices from joining the network during the activation process. Additionally, the NS regulates data rates by sending Adaptive Data Rate (ADR) commands, adjusting them based on the specific conditions of each device.
- **The Join Server (JS):** The JS is responsible for the activation of *EDs* and the generation of session keys. It manages key distribution, ensuring secure communication between devices and the network.
- **The Application Server (AS):** The AS receives and processes data sent by the *EDs*. It has the application logic which end-users can interact with.

An *ED* that wishes to join a LoRaWAN network does so using Activation by Personalization (ABP) or Over-The-Air Activation (OTAA). OTAA is the more secure method [142] and is the method we consider in this work.

1.2.2 Over-The-Air-Activation

Before the join procedure begins, several keys and identifiers should be provisioned on the *ED* and the backend servers. These keys and identifiers are:

- **DevEUI**: a 64-bit globally unique Extended Unique Identifier in the IEEE EUI64 address space for each *ED* that identifies it in the network.
- **JoinEUI**: a 64-bit globally unique Extended Unique Identifier in the IEEE EUI64 address space that identifies a JS.
- **AppKey**: an AES-128 root key specific to each *ED* and is used by the *ED* to generate the session keys after having joined the network and ensure data confidentiality.

The provisioning is done as follows:

- **Provisioned on the ED**: *DevEUI*, *JoinEUI*, and *AppKey*.
- **Provisioned on the NS**: *DevEUI*.
- **Provisoned on the JS**: *DevEUI* and *AppKey*.

The OTAA procedure in LoRaWAN 1.0.X is depicted in Figure 1.3. The OTAA procedure is as follows:

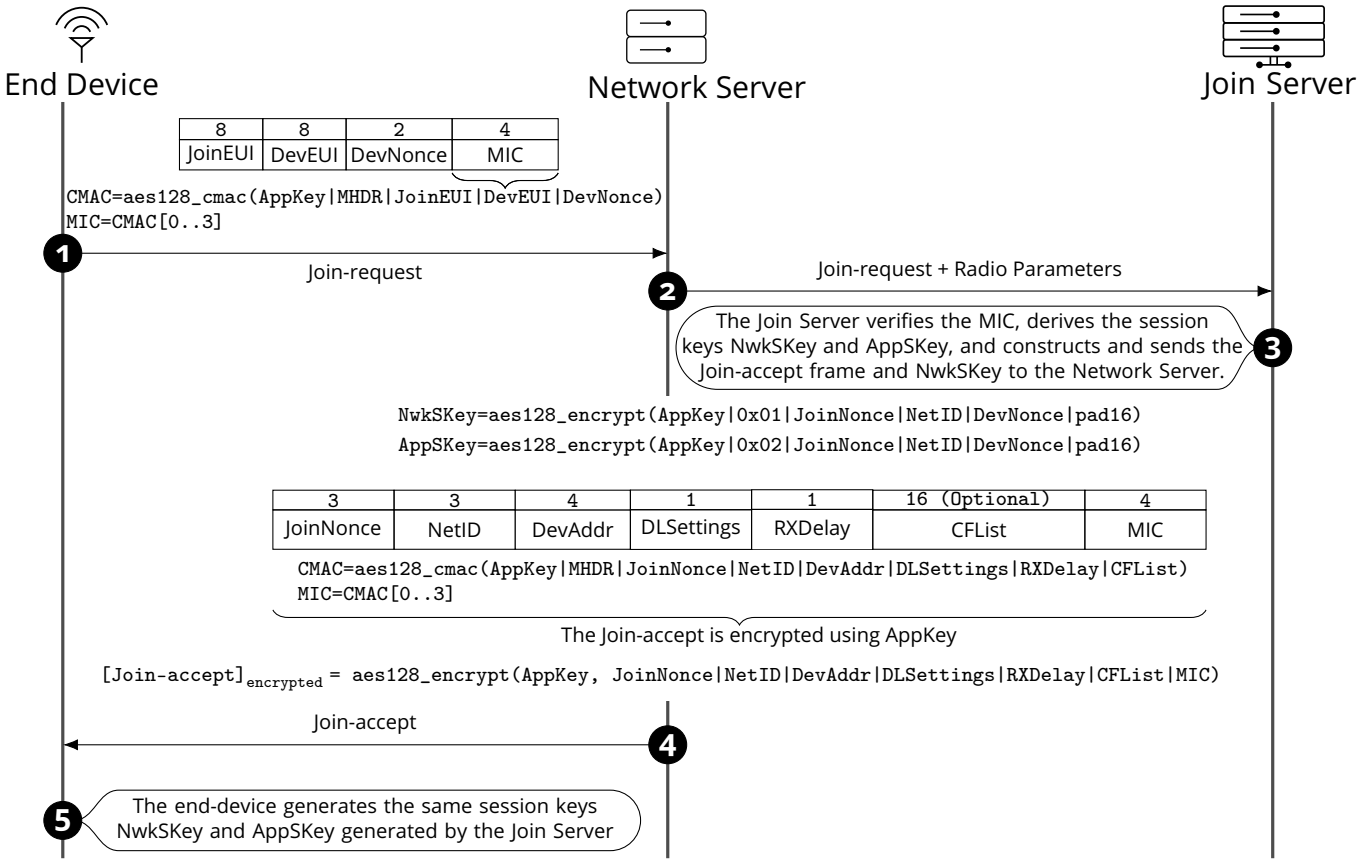


Figure 1.3: The LoRaWAN 1.0.X OTAA.

1. The *ED* sends a *Join-request* with *PHYPayload* containing the *JoinEUI*, *DevEUI*, and a 16-bit *DevNonce* which is a counter that starts at 0 after powering up the *ED*, and is incremented with every *Join-request* sent to prevent replay attacks. The *Join-request* is sent in plaintext with a 4-byte message integrity code (MIC) generated using *AppKey* to ensure its integrity.

2. The *Network Server* (NS) receives the *Join-request* and verifies if it is the dedicated NS for the *ED* by inspecting the *DevEUI*. If not, the NS ignores the *Join-request*. Otherwise, the NS uses DNS to look up the IP address of the *Join Server* (JS) using the *JoinEUI*. After that, the NS generates *DevAddr*, a 32-bit address that identifies the *ED* in the current network and sends *JoinReq* to the JS. The *JoinReq* contains the *PHYPayload* of the *Join-request* and radio parameters.

3. The JS verifies the MIC and generates the session keys *NwkSKey* and *AppSKey*. The JS then constructs the *Join-accept* frame that includes the *JoinNonce* which is used to generate the session keys. The *Join-accept* is encrypted using the *AppKey*. The JS then sends a Result=Success and the *JoinAns* back to the NS with the *PHYPayload* containing the encrypted *Join-accept* and the *NwkSKey*.

4. If the Result=Success, the NS forwards the *PHYPayload* with the *Join-accept* to the *ED*.

5. After receiving the *Join-accept*, the *ED* generates the same session keys *NwkSKey* and *AppSKey* generated by the JS.

1.3 DNS and DNS security

Creating a network where devices can exchange data requires developing a method to uniquely identify and address each device individually. On the Internet, where a large number of devices—including servers, routers, and hosts—form a large interconnected network, this was achieved using IP addresses. Early on, it was discovered that while IP addresses are machine-friendly, they are tedious for humans to memorize. This led to the creation of hostnames and domain names, which are textual identifiers used to identify devices and resources on the network in a human-friendly manner. Mapping between domain names and IP addresses was eventually entrusted to the Domain Name System (DNS), but this was not always the case.

In the early stages of the Internet, a simple `'HOSTS.txt'` file located on a single machine was responsible for the domain-name-to-IP-address translation, with all hosts on the network retrieving this file via FTP. The bandwidth required to download this file grew proportionally to N^2 for a network of N hosts [27]. This solution worked properly when N was small, but the growth of the Internet mandated that a more scalable solution is found. The solution was DNS. DNS is a distributed lookup service that is used to translate domain names, such as `'www.afnic.fr'`, to IPv4 addresses, such as `'192.134.5.37'`, or IPv6 addresses, such as `'2001:67c:2218:302::51:231'`. The DNS acts as the Internet's phonebook and ensures that communication on the Internet, such as email or simple web browsing, are easily and efficiently done.

Many standardization documents describe the functions of DNS. The Internet Engineering Task Force (IETF) regularly publishes Requests for Comments (RFCs) that define or update DNS functions. Since the first steps to naming hosts were laid out in 1982 by RFC 819 [25], which was followed by RFC 920 [26] that introduced the idea of domain names, DNS has undergone many updates, one of the major ones being RFC 1034 [27], and RFC 1035 [28], which were the blueprint of the modern DNS.

The fundamental elements of DNS architecture are the nameservers, each responsible for resolving a group of domain names into their corresponding data, such as IP addresses, mail server information, or other types of Resource Records (RRs) like text or service data. When DNS clients send resolution requests, these requests are first received by resolvers, which then query the appropriate nameservers to retrieve the RRs.

Originally, DNS did not account for the security and privacy of DNS queries and responses. This means that DNS traffic, including the requested domain names, response data, and other related information, were transmitted in plaintext, giving malicious actors the chance to intercept, tamper, or spoof the DNS queries and responses. To address these shortcomings, several extensions and DNS-based protocols were proposed to add a layer of security and/or privacy to DNS, such as DNS security extensions (DNSSEC) which guarantees the integrity—but not privacy—of DNS responses, and several other protocols that ensured the privacy of DNS traffic.

1.3.1 DNS Architecture

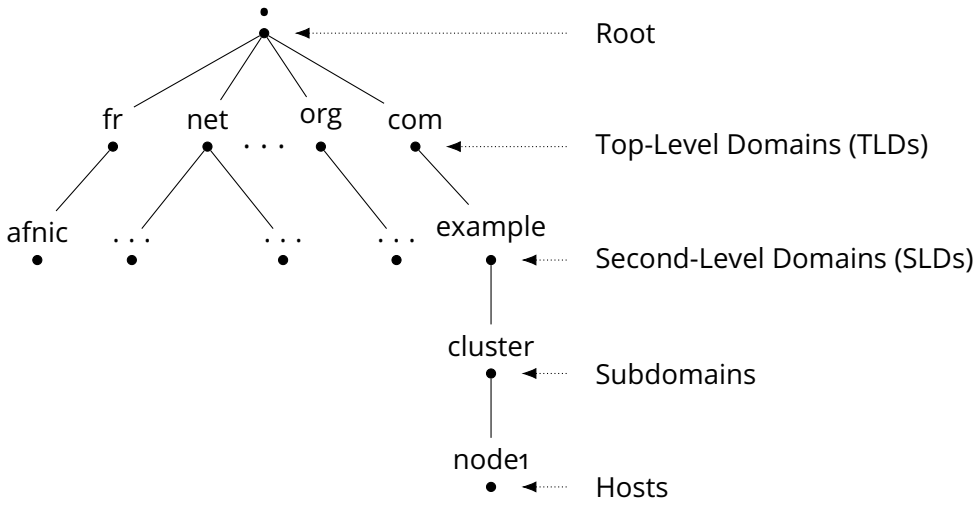


Figure 1.4: DNS namespace structure.

The set of all domain names registered in the DNS is collectively referred to as the DNS namespace, which is organized hierarchically in an inverted tree-like structure. The DNS servers at each level of this hierarchy are called nameservers. See Figure 1.4. The DNS is a distributed lookup service, meaning its namespace is distributed among different nameservers across the DNS tree. A segment of the namespace is called a DNS zone and a nameserver that can definitively answer queries about a zone—providing a definitive answer to the original DNS query—is said to be authoritative over that zone.

At the root of the DNS tree are the root nameservers represented by a `'.'` (dot). Currently, there are 13 root nameservers, identified as `'a.root-servers.net'` through `'m.root-servers.net'` [118], which serve as the starting point of the DNS resolution process. When a root nameserver receives a query that it is not authoritative to answer definitively, it refers the client to the next level in the DNS namespace, which contains the Top-Level Domains (TLDs).

TLDs are the last non-empty part of any domain name, such as `'com'` in `'example.com'` or `'fr'` in `'afnic.fr'`[1]. TLD zones are stored at the root nameservers, meaning the root nameservers are authoritative over these zones and can definitively answer queries related to them. An extensive list of all TLDs can be found here [117], but they can be categorized as:

- **Generic Top-Level Domains (gTLD):** for generic use, *e.g.*, `'.com'` and `'.net'`.
- **Country Code Top-Level Domains (ccTLD):** reserved for countries or independent sovereign territories, *e.g.*, `'.fr'` for France and `'.it'` for Italy.
- **Reserved Top-Level Domains:** reserved for testing and documentation, *e.g.*, `'test'` and `'example'`.
- **Infrastructure Top-Level Domains:** include `'.arpa'` which is reserved for the management of network infrastructure and for reverse DNS resolution.

Further down the tree are the Second-Level Domains (SLDs). SLDs are typically registered by individuals or organizations to create a unique web address such as `'example'` in `'example.com'` or `'afnic'`

[1] Afnic is the French Internet Registry

in `'afnic.fr'`. Registration is usually done through accredited registrars. The SLD authoritative nameservers, which are the nameservers hosting the SLD zones are either hosted by the domain owner on private infrastructure, by the registrar, or by a third-party host.

Subdomains are created by adding a label to the left of an SLD or an existing subdomain. The original domain owner delegates authority to subdomains below their SLD or existing subdomain. This creates a hierarchy within a specific DNS zone which provides flexibility for different services, departments, or functions within an organization.

Hosts, which are at the bottom of the DNS tree, refer to individual devices or systems on the network that are reachable via their domain names regardless if their IP addresses change.

1.3.2 DNS Resource Records

DNS RRs are the fundamental building blocks of a DNS zone. Depending on their type, RRs contain various types of information about the domain name. The DNS resolution process involves a DNS client querying the DNS infrastructure for one or more of these RRs. What follows is a brief overview of the most commonly used RRs.

- **A Record:** holds a 32-bit IPv4 address and is used for mapping between IPv4 addresses and domain names.
- **AAAA Record:** holds a 128-bit IPv6 address and is used for mapping between IPv6 addresses and domain names.
- **MX Record (Mail Exchange):** specifies a mail server that receives emails on behalf of a domain name.
- **CNAME Record (Canonical Name):** creates an alias for one domain name that points to another domain name.
- **NS Record (Nameserver):** specifies the nameserver which is authoritative over the domain name.
- **PTR Record (Pointer):** maps an IP address to a domain name and is used for reverse DNS lookups.
- **SRV Record (Service):** specifies a host and a port number for a specific service such as Voice over IP (VoIP).
- **TXT Record (Text):** used to store text information about the domain name.
- **DNSKEY Record:** contains a public key that is used to verify Domain Name System Security Extensions (DNSSEC) signatures for that zone.
- **DS Record (Delegation Signer):** holds information linking a child zone to its parent zone in DNSSEC. It is used to establish a chain of trust.
- **TLSA Record:** associates a TLS certificate with the domain name, ensuring secure connections through DANE (DNS-Based Authentication of Named Entities).
- **CAA Record (Certification Authority Authorization):** specifies which certificate authorities (CAs) are allowed to issue certificates for the domain name.

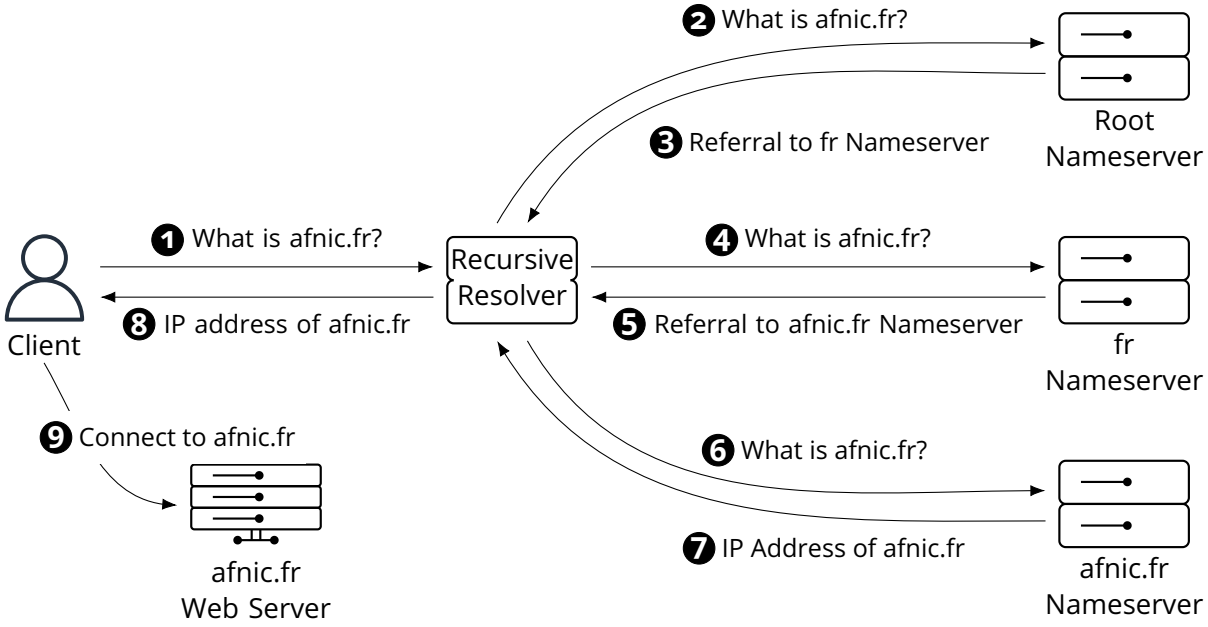


Figure 1.5: The DNS resolution process.

1.3.3 The DNS Resolution Process

The primary function of DNS is to map domain names to IP addresses. DNS has also evolved into a versatile system that also supports email routing, service discovery, security, and other functionalities. The domain names are usually given to resources on the Internet that users wish to access. The mapping process is referred to as domain name resolution. The DNS resolution process is depicted in Figure 1.5.

For example, a user that wishes to visit the website of Afnic will type in their browser `'www.afnic.fr'`. The first entity that receives the request, also known as the DNS query, is the stub resolver which is usually part of the operating system. The stub resolver then transfers the query to the DNS recursive resolver. Recursive resolvers are the main interface between users and the DNS infrastructure. Upon receiving a query, the recursive resolver initiates a hierarchical query/response process with the DNS nameservers, following a recursive pattern to retrieve the required information.

The recursive resolver first sends the query to one of the root nameservers. The root nameserver answers with a referral to the `'.fr'` nameserver. The recursive resolver then sends the request to the `'.fr'` nameserver, which answer with a referral to afnic.fr nameserver, which is authoritative over `'afnic.fr'`. Lastly, the recursive resolver sends the query to afnic.fr nameserver, which answers with the IP address of `'afnic.fr'`. The recursive resolver then sends back the answer to the user which then connects to afnic.fr web server.

1.3.4 DNS Security and Privacy Standards

When DNS was first deployed, privacy and security were not considered [194]. RFC 9076 [194] titled *DNS Privacy Considerations* studies the security—mainly privacy—drawbacks of DNS that end users should be aware of. The drawbacks include: sending queries in plain text, which jeopardizes privacy; using the User Datagram Protocol (UDP), whereas most security mechanisms are designed for the Transmission Control Protocol (TCP); sending the full QNAME (Query Name, *i.e.*, the requested domain name) at every stage of the resolution process, even when it is not necessary; generating unnecessary DNS requests due to

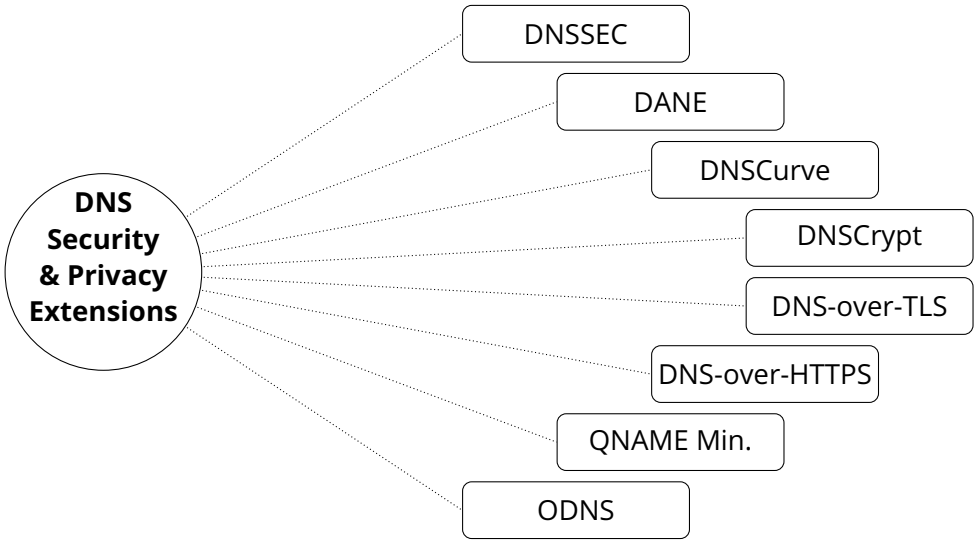


Figure 1.6: DNS security and privacy standards.

resource prefetching and auto-completion; cache snooping; and the passive collection of DNS data by powerful companies running public DNS servers, such as Google [11] and Cloudflare [3]. Passive DNS data collected by major companies running public DNS servers may be abused or sold to third-party companies for commercial purposes, or even misused for surveillance and spying.

Finally, the Server Name Indication (SNI), the last plain-text part of a secured web connection, is used in the case of multi-hosting to identify the intended resource requested on the webserver. SNI is still sent without encryption, which exposes the browsing habits of users [113].

The following section discusses the most prominent DNS security and privacy extensions and protocols. See Figure 1.6.

DNSSEC DNSSEC [174, 176, 175] stands for DNS security extensions. DNSSEC guarantees the integrity—not privacy—of DNS responses that DNS resolvers receive using an authentication chain of trust. The chain starts at the root nameservers, which are the trust anchors and are trusted by default. Going down the chain, every level in the hierarchy vouches for the level below it. Cryptographic signatures are used to assure the DNS resolver that the response it receives comes from the legitimate DNS nameserver and has not been tampered with in transit. The DNSSEC mechanism is presented in Figure 1.7.

RFC 4033 [174] provides an introduction to DNS security and requirements, RFC 4034 [176] defines the RRs for the DNS Security Extensions, and RFC 4035 [175] is about the modifications required in the initial DNS protocols following the introduction of DNSSEC. RFC 4398 [122] discusses storing certificates in the DNS, RFC 5155 [47] presents DNSSEC hashed authenticated denial of existence, and RFC 6014 [109] explains how cryptographic algorithm identifiers, required to implement DNSSEC, are allocated within the Internet Assigned Numbers Authority (IANA) registries.

In the context of DNSSEC, a Resource Record Set (RRset) refers to a group of RRs that share the same name, class, and type. A DNSSEC-enabled zone contains two public/private key pairs:

- **Key Signing Key (KSK):** a cryptographic key pair used to sign the key records of a zone.
- **Zone Signing Key (ZSK):** a cryptographic key pair used to sign the non-key records of the zone.

Some DNSSEC-related RRs are:

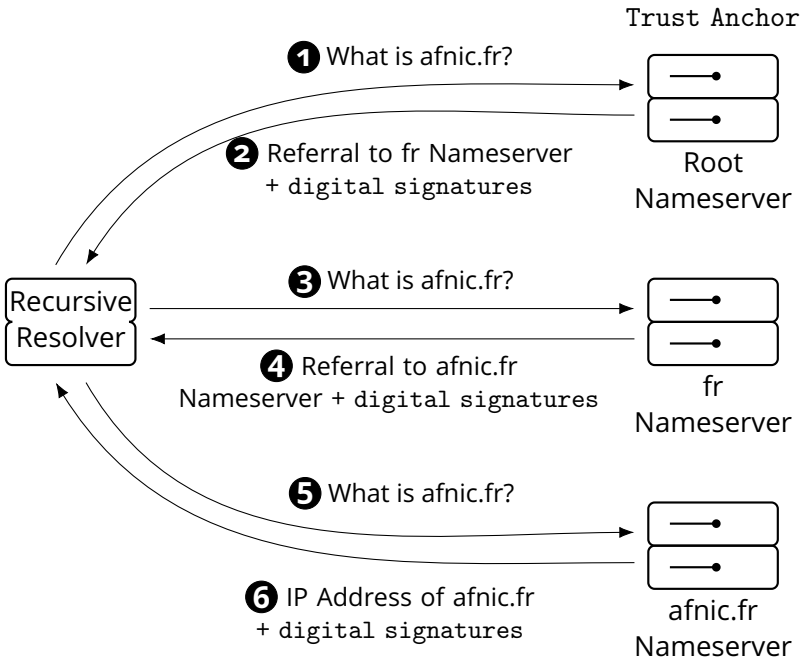


Figure 1.7: DNSSEC Verification.

- **DNSKEY:** stores public keys (KSK and ZSK) used for validating DNSSEC signatures.
- **DS (Delegation Signer):** links the current zone to its child zone via a hash of the child zone's KSK. The DS record is used to enable a parent zone to vouch for a child zone, establishing the chain of trust.
- **RRSIG (Resource Record Signature):** contains the cryptographic signature that validates the authenticity of DNS RRs. It is signed using the private key corresponding to the DNSKEY.
- **NSEC/NSEC3 (Next Secure):** used to prove the non-existence of DNS data.

A detailed demonstration of how DNSSEC functions is presented in Figure 1.8. The recursive resolver holds $\texttt{PubKSK}^{\texttt{Root}}$, the public KSK of the root nameserver, which serves as the trust anchor for the chain of trust in DNSSEC. This key allows the resolver to verify the signatures of the root zone and, in turn, the entire chain of trust down to the queried domain.

1. The recursive resolver sends a query to the root nameserver to resolve `'afnic.fr'`.

2. The root nameserver responds with a referral to the `'fr'` nameserver responsible for the `'fr'` zone.

- The response message contains the following:
  1. Referral to `'fr'` nameserver
  2. DNSKEY RRset of the root zone containing the $\texttt{PubKSK}^{\texttt{Root}}$ and $\texttt{PubZSK}^{\texttt{Root}}$
  3. RRSIG for DNSKEY RRset of the root zone: digital signature over the DNSKEY RRset signed using the $\texttt{PvtKSK}^{\texttt{Root}}$
  4. DS RR of `'fr'` zone: contains a hash of the $\texttt{PubKSK}^{\texttt{fr}}$

5. RRSIG for DS RR of `'fr'` zone: signed using the $\texttt{PvtZSK}^{\texttt{Root}}$

- The resolver verifies the RRSIG for the DNSKEY RRset of the root zone using $\texttt{PubKSK}^{\texttt{Root}}$
- The resolver then uses $\texttt{PubZSK}^{\texttt{Root}}$ (from the DNSKEY RRset of the root zone) to verify the RRSIG for the DS RR of `'fr'` zone.

3. The recursive resolver sends a query to the `'fr'` nameserver to resolve `'afnic.fr'`.

4. The `'fr'` nameserver responds with a referral to the `'afnic.fr'` nameserver responsible for the `'afnic.fr'` zone.

- The response message contains the following:

1. Referral to `'afnic.fr'` nameserver
2. DNSKEY RRset of the `'fr'` zone containing the $\texttt{PubKSK}^{\texttt{fr}}$ and $\texttt{PubZSK}^{\texttt{fr}}$
3. RRSIG for DNSKEY RRset of the `'fr'` zone: digital signature over the DNSKEY RRset signed using the $\texttt{PvtKSK}^{\texttt{fr}}$
4. DS RR of `'afnic.fr'` zone: contains a hash of the $\texttt{PubKSK}^{\texttt{afnic.fr}}$
5. RRSIG for DS RR of `'afnic.fr'` zone: signed using the $\texttt{PvtZSK}^{\texttt{fr}}$

- The resolver verifies the RRSIG for the DNSKEY RRset of the `'fr'` zone using $\texttt{PubKSK}^{\texttt{fr}}$
- The resolver then uses $\texttt{PubZSK}^{\texttt{fr}}$ (from the DNSKEY RRset of the `'fr'` zone) to verify the RRSIG for DS RR of the `'afnic.fr'` zone.

5. The recursive resolver sends a query to the `'afnic.fr'` nameserver to resolve `'afnic.fr'`.

6. The `'afnic.fr'` nameserver responds with the *A RR* which contains the IPv4 address of `'afnic.fr'`.

- The response message contains the following:

1. *A RR* of `'afnic.fr'`
2. RRSIG for *A RR* of the `'afnic.fr'` zone: digital signature over the *A RR* signed using the $\texttt{PvtZSK}^{\texttt{afnic.fr}}$
3. DNSKEY RRset of the `'afnic.fr'` zone containing the $\texttt{PubKSK}^{\texttt{afnic.fr}}$ and $\texttt{PubZSK}^{\texttt{afnic.fr}}$
4. RRSIG for DNSKEY RRset of the `'afnic.fr'` zone: digital signature over the DNSKEY RRset signed using the $\texttt{PvtKSK}^{\texttt{afnic.fr}}$

- The resolver verifies the RRSIG for the DNSKEY RRset of the `'afnic.fr'` zone using $\texttt{PubKSK}^{\texttt{afnic.fr}}$
- The resolver then uses $\texttt{PubZSK}^{\texttt{afnic.fr}}$ (from the DNSKEY RRset of the `'afnic.fr'` zone) to verify the *A RR* of the `'afnic.fr'` zone.

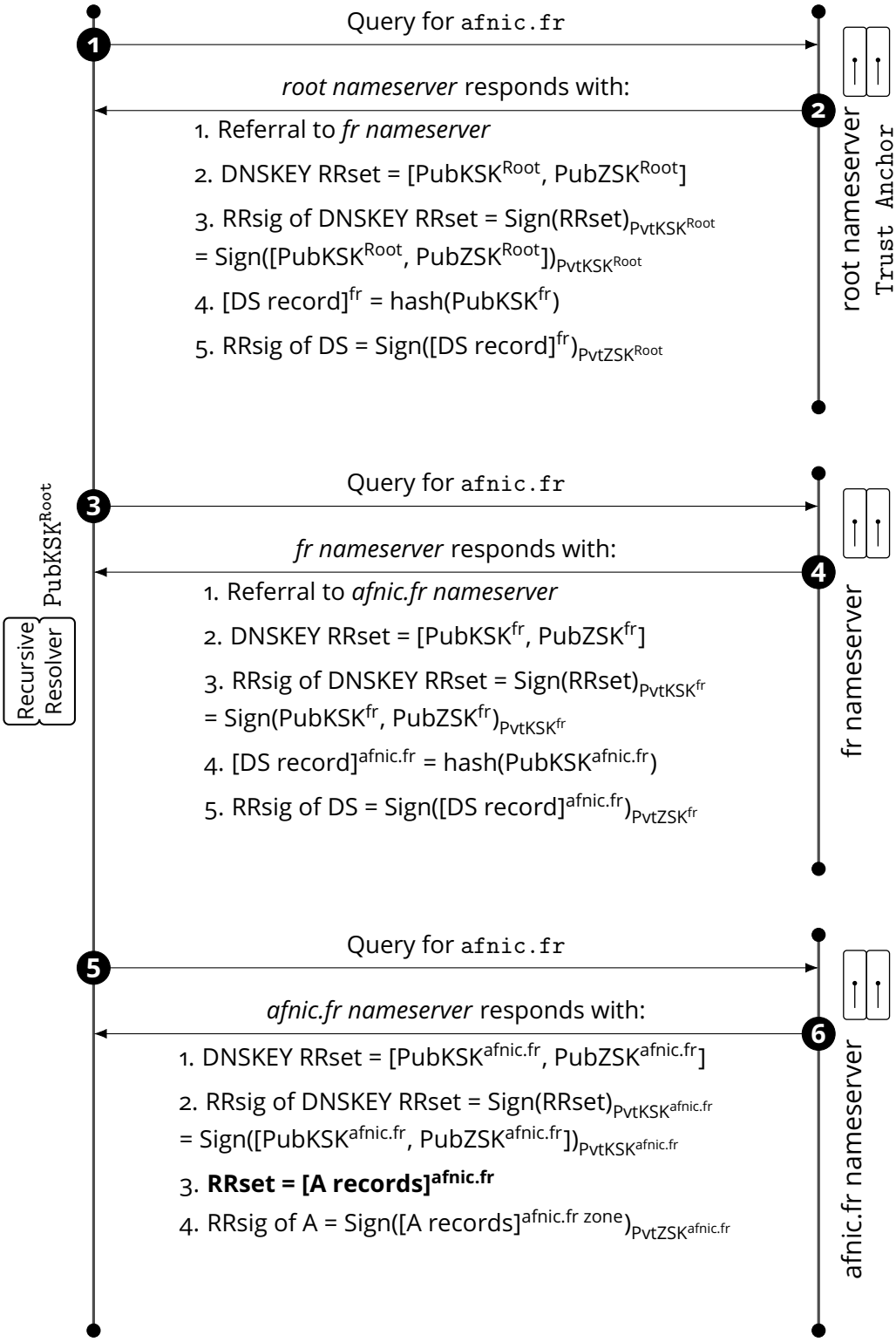


Figure 1.8: Illustration of the DNSSEC verification process.

DNS-Based Authentication of Named Entities (DANE) The DNS-based Authentication of Named Entities (DANE) [111] is a DNS protocol whose goal is to give power back to domain name owners and allow them to decide who validates their certificates and how these certificates are validated.

Communications on the Internet today rely on the Public Key Infrastructure (PKI), with the X.509 digital certificate as its main element. Entities that want their public key validated present their certificate to the entities attempting to establish a secure connection with them. The validation of a certificate depends on its issuer. Self-signed certificates are rarely accepted on the Internet. Most digital certificates are issued by trusted Certificate Authorities (CAs). Trusted root CAs issue and sign certificates, and only those issued by them are inherently trusted. Root CAs often create intermediate CAs, which are authorized to issue and sign certificates for end entities, known as leaf certificates. This establishes a hierarchical chain of trust, beginning with the root CAs, passing through the intermediate CAs, and ending with the leaf certificates.

The PKI, which uses X.509 digital certificates, is widely adopted and supports the majority of secure connections on the Internet. However, it is not without its limitations. End users that receive and validate digital certificates must have a root or *Trust Store* containing the self-signed certificates of trusted CAs. If a certificate is trusted by any of the trusted CAs, then it is validated. This means that one compromised CA could pose a significant security threat by, for example, issuing false certificates especially to popular websites like the CA Diginotar did in 2011 [99].

DANE is a DNS protocol that has the potential to support—or, in certain use cases, entirely replace—the current PKI that relies on X.509 digital certificates. It aims to enhance the authentication and integrity of Internet communications by binding public keys to DNS names and securing this binding with DNSSEC.

Figure 1.9 gives an overview of how certificates are verified during the TLS handshake if DANE is used. In this scenario the TLS server is supposed to have published a TLSA RR in its DNS zone. The client would initiate a TLS handshake before connecting to the TLS server. The server then shares its certificate with the TLS client. The client, which would normally verify the certificate using a CA chain of trust, queries DNS to retrieve the TLSA RR of the TLS server. The integrity of the DNS response is guaranteed as DNSSEC is used. The client then proceeds to verify the certificate based on the parameters set in the TLSA RR.

The primary feature of DANE is the TLSA RR. See Figure 1.10.

The TLSA RR is composed of:

- **Port**: The port number the Transport Layer Security (TLS) server listens on.
- **Protocol**: The protocol used, *e.g.*, TCP.
- **Domain name**: The domain name of the TLS server which is the domain name associated with the TLSA RR.
- **Certificate Usage field**: 1 byte that represents the association that will be used to match the certificate provided in the TLS handshake. It could have the following values:
  - **PKIX-TA(0)**: Also referred to as *CA constraint*. This Certificate Usage value indicates that the Certificate Association Data contains the certificate or the public key of a trust anchor that can issue certificates for the server and that the client must find in its trust store to finally validate the certificate presented by the server during the TLS handshake.
  - **PKIX-EE(1)**: Also referred to as *service certificate constraint*. This Certificate Usage value indicates that the Certificate Association Data contains the certificate or the public key of the end entity that must be matched with the certificate provided by the server during the TLS handshake. The certificate must also be validated using a CA chain of trust.
  - **DANE-TA(2)**: Also referred to as *Trust Anchor Assertion*. This Certificate Usage value indicates that the Certificate Association Data contains the certificate or the public key of a Trust Anchor that must be used to validate the end entity certificate provided by the server during the TLS

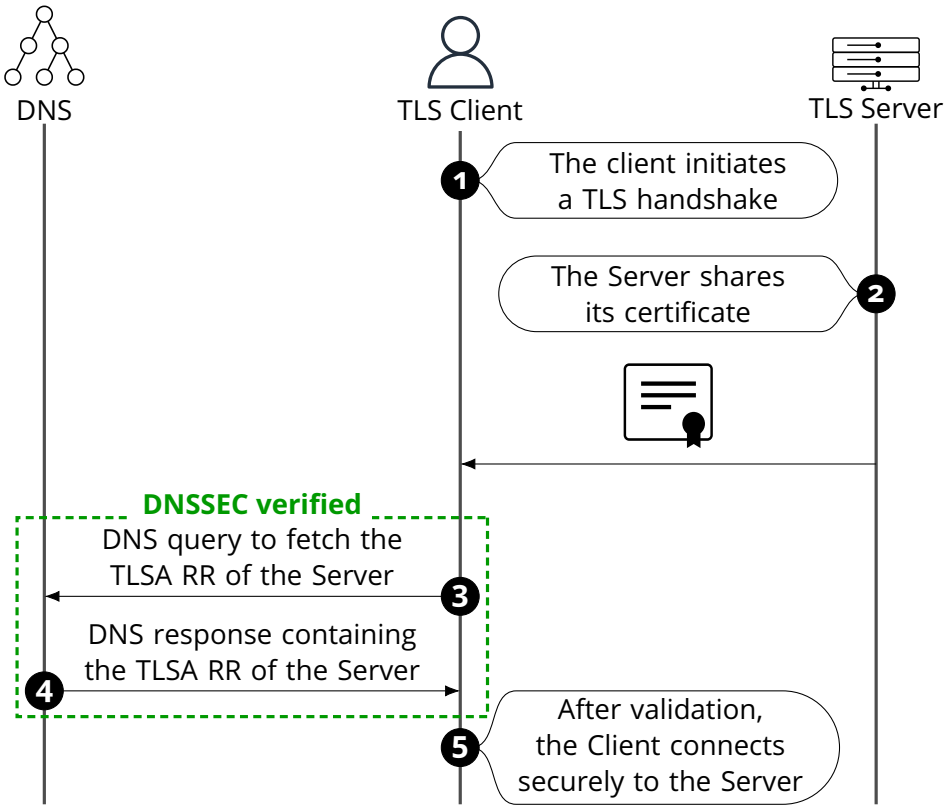


Figure 1.9: Certificate verification using DANE.

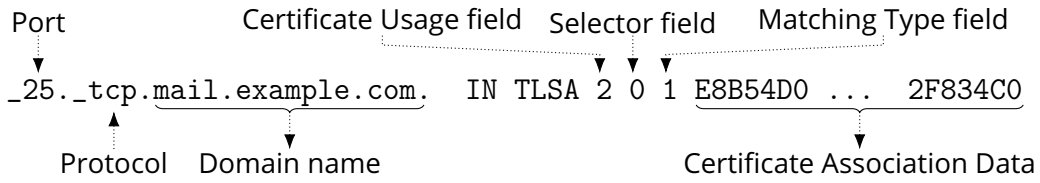


Figure 1.10: TLSA resource record.

handshake. This could be helpful if the end entity certificate is self-signed or if the owner of that certificate runs their own CA.

- **DANE-EE(3)**: Also referred to as *Domain-Issued Certificate*. This Certificate Usage value indicates that the Certificate Association Data contains a certificate that must match the certificate received during the TLS handshake. DANE-EE(3), unlike PKIX-EE(1), does not require validation of the certificate using a CA chain of trust.

- **Selector field**: 1 byte that determines the part of the certificate presented by the server during the TLS handshake which will be matched against the Certificate Association Data. It could have the following values:

  - **Cert(0)**: Full Certificate. The entire certificate presented by the server during the TLS handshake is used and compared against the Certificate Association Data of the TLSA RR.

  - **SPKI(1)**: SubjectPublicKeyInfo. The public key portion of the certificate presented by the server during the TLS handshake is used and compared against the Certificate Association Data of the

TLSA RR.

- **Matching Type field**: 1 byte that determines the method of comparison between the certificate presented by the server during the TLS handshake and the Certificate Association Data. It could have the following values:
  - **Full(0)**: Compare the full data extracted from the certificate presented by the server during the TLS handshake with the full data stored in the Certificate Association Data in the TLSA RR.
  - **SHA2-256(1)**: Compute the hash of the data extracted from the certificate presented by the server during the TLS handshake using SHA-256 and compare it with the hash value stored in the Certificate Association Data in the TLSA RR.
  - **SHA2-512(2)**: Compute the hash of the data extracted from the certificate presented by the server during the TLS handshake using SHA-512 and compare it with the hash value stored in the Certificate Association Data in the TLSA RR.
- **Certificate Association Data**: The data stored in the TLSA RR which are used to compare and match against the information provided in the certificate presented by the server during the TLS handshake.

Beyond DNSSEC and DANE, several other notable DNS security protocols have been developed to enhance privacy and security. Below, we briefly introduce some of these protocols.

DNSCurve DNSCurve [8], designed in 2009, adds link-level security to DNS using elliptic-curve cryptography. DNSCurve preserves confidentiality by encrypting DNS packets, protects the integrity of DNS responses through cryptographic authentication, and offers some protection against Denial of Service (DoS) attacks. It uses 256-bit public and secret keys, 192-bit nonces, and 128-bit authenticators. DNS servers using DNSCurve distribute their public keys by encoding them in an *NS (Name Server) RR*, ensuring that the public key distribution system is compatible with registries and nameserver software. On the other hand, clients share their public keys in the queries they send.

DNSCrypt DNSCrypt [7] acts as a security layer between DNS clients and recursive resolvers. It uses cryptographic signatures to verify that responses from the resolvers are authentic and have not been tampered with on the way. Anonymized DNSCrypt [1] was proposed in 2019 as an extension to further secure DNS traffic by not allowing the server to see the client's IP address.

DNS-over-TLS DNS-over-TLS (DoT) [179] [112] is one the few IETF-standardized approaches for securing DNS (the other notable one being DNS-over-HTTPS)). DoT replaces the traditional UDP transport with TCP and provides packet authentication and confidentiality for DNS traffic between clients and resolvers. This is achieved using TLS. A TLS session is established on TCP port 853, and DNS data is exchanged over the secure channel.

DNS-over-HTTPS DNS-over-HTTPS (DoH) [179] [110] is the other IETF-standardized protocol for securing DNS. Like DoT, it aims to preserve the integrity and confidentiality of DNS data. However, as the name suggests, DoH uses HTTPS, allowing DNS queries to benefit not only from TLS but also from the full HTTPS protocol. This enables web applications to use DNS securely. In DoH, a DNS query and its response are exchanged as part of an HTTPS session. The client encodes the DNS request into an HTTP request using either the GET or POST method.

QNAME Minimization One of the shortcomings of DNS is that the full Query Name (QNAME) and query type (QTYPE) are always sent during recursive DNS resolution, regardless of the stage of the process. However, the full QNAME and QTYPE are only needed when the request reaches the authoritative nameserver of the queried domain. For instance, when resolving `'www.afnic.fr'`, the root nameserver receives a query with the full QNAME `'www.afnic.fr'`, even though it only needs to know `'.fr'` and does not require the QTYPE. QNAME minimization [64] [80] aims to enable resolvers to send the minimum necessary information at each stage of the resolution process, following the principle: *the less data you send out, the fewer privacy problems you have* [75]. Therefore, when sending queries to nameservers that are not authoritative for the requested domains, resolvers implementing QNAME minimization send an altered QNAME to obscure the original one. Instead of the full QNAME, they send one level longer than what the nameserver is known to be authoritative for.

Oblivious DNS (ODNS) Oblivious DNS (ODNS) [180] addresses the fact that, in any privacy setup, there must always be a party trusted by default. This trusted party could be any entity, ranging from the Internet Service Provider (ISP) to large public DNS resolver companies. When a client sends a DNS request to a recursive resolver, the resolver has full access to both the domain name requested by the client and the client's IP address. Typically, the recursive resolver is trusted not to misuse or share this data with third parties. However, this trust is often unfounded, as there is no concrete reason to inherently trust the resolver. ODNS aims to eliminate the need for trust by preventing recursive resolvers from associating client identities with requested domain names. It leverages the existing DNS infrastructure, making its deployment more feasible. An ODNS stub is placed between the DNS client and the existing recursive resolver, while an ODNS recursive resolver is positioned between the existing recursive resolver and the DNS infrastructure, *i.e.*, the DNS nameservers.

Figure 1.11 shows the ODNS resolution process. The ODNS resolver is assumed to have a public/private key pair of which the public key is known to the ODNS stub.

1 The DNS client sends a regular query to fetch the *A RR* of `'www.afnic.fr'`.

2 The ODNS receives the query and generates a secret session key *k* which it uses to encrypt the domain name and appends the ODNS domain to the encrypted domain name. It also encrypts the session key *k* using $\mathrm{PbK}^{\texttt{ODNS Resolv.}}$, the public key of the ODNS resolver. The encrypted domain name and session key are then passed to the recursive resolver.

3 The recursive resolver forwards the query to the appropriate ODNS resolver according to the ODNS domain appended to the encrypted domain name is step 2.

4 The ODNS resolver decrypts *k* using its private key $\mathrm{PvK}^{\texttt{ODNS Resolv.}}$ and then decrypts the domain name using *k*. It then proceeds to resolve the domain using DNS.

5 The ODNS resolver receives the *A RR* of the domain name and encrypts it using the session key *k*.

6 The ODNS resolver forwards the encrypted query response to the recursive resolver.

7 The recursive resolver forwards the encrypted query response to the ODNS stub.

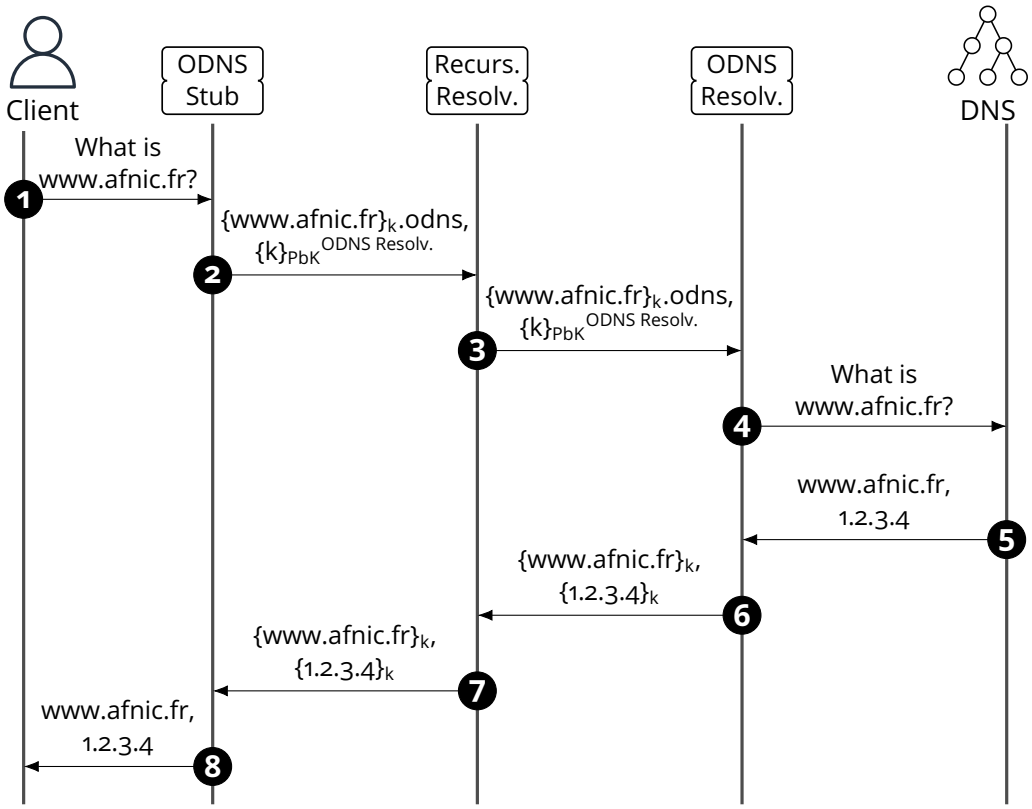


Figure 1.11: ODNS Resolution Process.

8 The ODNS stub decrypts the query response and passes it to the client.

Using ODNS, the recursive resolver can see the client's IP address but not the requested domain name. Conversely, the ODNS resolver can see the requested domain name but not the client's IP address.

1.4 Thesis Objectives

This thesis focuses on leveraging DNS and its security extensions to enhance security in IoT environments. With the increasing reliance on IoT devices and the predictions for even larger-scale adoption of IoT in the future, it becomes crucial to ensure secure communication for all connected devices. The distributed nature of DNS, along with its security and privacy extensions, positions it as a key component in addressing some of the significant challenges—particularly in security—faced by IoT networks. Thus, the primary objectives of this work are as follows:

- The objective is to identify, classify, and explain the challenges facing IoT environments, as well as the potential repercussions of failing to address these challenges.
- Leverage DNS and its security extensions to address some of the challenges in IoT, drawing conclusions about the added value of this approach and the associated overhead incurred.
- Implement, test, and evaluate the proposed solutions on real devices using open-source frameworks and tools.
- Leverage DNS-related datasets, such as real datasets of passive DNS traffic, and effectively use them to enhance IoT systems.

To address these objectives, several contributions are made that focus on improving the security and functionality of IoT environments through DNS and its extensions. We highlight these contributions next.

1.5 Thesis Contributions

Addressing the challenges faced by IoT environments using DNS and its security extensions first requires identifying these challenges and assessing the feasibility of utilizing DNS to mitigate them.

Thus, our **first contribution** in Chapter 2, which served as the foundation of this thesis, involved a comprehensive study of the IoT ecosystem. We identified and categorized the key challenges and conducted a systematic literature review to explore existing uses of DNS in IoT environments aimed at addressing the challenges initially identified.

The **second contribution** in Chapter 3 proposed using DANE, the DNS protocol designed to reinforce the PKI, to perform mutual authentication between the backend servers of an IoT network, securing the connection between the servers without relying on commercial CAs.

The **third contribution** in Chapter 4 introduced LoRaDANCE, a security mechanism that allows a LoRaWAN *ED* to join a LoRaWAN network without first needing to pre-share any secret keys with the backend servers, as required in standard LoRaWAN, and while mutually authenticating with the JS (Join Server). The mutual authentication is ensured using DANE, while asymmetric cryptography allows the device and server to generate the necessary secret, eliminating the need for pre-shared keys.

For our **fourth contribution** in Chapter 5, we conducted an in-depth study of IoT domain names and evaluated the differences between them and non-IoT domain names. For the purposes of this work, IoT domain names refer to those of IoT backend servers that IoT devices resolve via DNS, whereas non-IoT domain names correspond to backend servers accessed by generic devices and humans. This study was carried out in three phases: a statistical analysis, a DNS analysis, and a machine learning-based classification of the two domain name classes.

State of the Art: DNS to Address IoT Challenges

The first contribution serves as the state of the art for this thesis and is composed of two major parts.

First, we conducted a comprehensive study of the IoT ecosystem, which, due to the multitude of IoT technologies and the lack of standardization, remains fragmented and lacks unified definitions and standards. We adopted a vendor-neutral approach in defining IoT and listing its properties and requirements, drawing from documentation provided by regulatory bodies such as the International Telecommunication Union (ITU) and other vendor-neutral initiatives. This part involved categorizing the challenges facing IoT environments, resulting in four identified categories: the constrained nature of IoT devices, identification in IoT, IoT security, and the lack of interoperability between IoT technologies.

The second part comprised a literature review aimed at identifying how DNS has been used to address IoT challenges. We found instances where DNS was applied to address challenges in each of the identified categories.

Mutual Authentication of IoT Backend Servers

The second contribution involved utilizing DANE to implement a mutual authentication mechanism between two IoT backend servers.

DANE is a DNS protocol that allows a domain name owner to publish a TLSA RR in their DNS zones, binding their domain name to their raw public key or digital certificate. This enables validation of the

public key or certificate during TLS handshakes, similar to the role of traditional CAs. DANE was proposed as either an enhancement or a potential replacement for traditional CAs, depending on its usage. However, originally, DANE only validated server raw public keys or certificates, not those of clients.

To achieve mutual authentication between the TLS client and server, we implemented two Internet drafts issued by the IETF DANCE working group (DANE Authentication for Network Clients Everywhere). These drafts demonstrate how DANE can be used to validate TLS client raw public keys or digital certificates, thus enabling mutual authentication. The IoT technology we applied this to is LoRaWAN. We implemented our solution on a real LoRaWAN *ED*, using DANE to enable mutual authentication between the NS and JS during the join process. We evaluated the performance of our solution by analyzing the duration of the join process while varying the location of the DNS resolver, and studying the impact of DNS caching on overall performance.

LoRaDANCE: Using The DNS for Mutual Authentication Without Pre-shared Keys

The third contribution introduced LoRaDANCE, a mechanism that enables a LoRaWAN *ED* to join a LoRaWAN network without the need to pre-share the *AppKey*, which is the secret key used in LoRaWAN networks to generate session keys during the join process. LoRaDANCE also facilitates mutual authentication between the *ED* and the JS using DANE. As its name indicates, LoRaDANCE is inspired by DANCE, utilizing DANE for mutual authentication, although not in the conventional context of a TLS client and server.

We implemented our solution on a real LoRaWAN *ED*, utilizing DANE to enable mutual authentication between the *ED* and the JS during the join process. The secret key is derived using Elliptic Curve Diffie-Hellman. Finally, we evaluated the performance of LoRaDANCE by studying the duration of the join process and the power consumed during the join process.

IoT Backend Servers: Studying IoT Domain Names

The fourth contribution is a study on IoT backend servers, focusing on their domain names. IoT devices typically contact backend servers for tasks like relaying information or receiving updates, resolving their domain names via DNS. We compiled a list of real IoT domain names using datasets from prior studies and compared them with non-IoT domain names from two public lists of top-visited websites. The study was conducted in three phases: statistical comparison of domain name lengths and the number of labels, DNS analysis, and machine learning models to classify IoT and non-IoT domain names based on intrinsic differences.

Chapter 2

State of the Art: DNS to Address IoT Challenges

Contents

Chapter Overview

The chapter opens by outlining the inherent challenges faced by the Internet of Things (IoT) environments. The first challenge relates to the constrained nature of IoT devices, which often have limited processing power, memory, and power budget. The second challenge is identifying and managing the growing number of IoT devices, originating from lack of standardization and the diversity of technologies in the IoT landscape. The third challenge faced by IoT is security-related and stems mainly from the constraints IoT devices have, which deprive them from using effective security mechanisms. The fourth and final challenge identified in this chapter is the lack of interoperability between IoT technologies, where differing IoT technologies and standards hinder communication between devices from various manufacturers.

This chapter then explores how the Domain Name System (DNS) is leveraged as valuable tool to address the IoT challenges. The usages of DNS in IoT environments, both in research and industry, are presented, offering an overview of current approaches and implementations to alleviate some IoT challenges using DNS. The distributed nature of DNS and its existing infrastructure make it a potential solution for IoT device identification, name resolution, and ensuring interoperability across different platforms. Furthermore, DNS security extensions and protocols can enhance the security of IoT communications, helping to prevent attacks and ensure the integrity of data exchanged within IoT networks.

The chapter concludes by outlining the necessary precautions for integrating DNS into IoT environments and discussing the potential repercussions of such usage.

2.1 Introduction

If the average user had been asked 20 years ago to name a connected device, they would likely have mentioned personal computers, or servers for those more tech-savvy. Meanwhile, the concept of the Internet of Things would likely have been met with little recognition. At that time, the idea that cars, refrigerators, pets, and other everyday objects would one day require an Internet connection—something quite common today—was inconceivable. Even mobile phones were generally perceived as devices for voice calls and had limited functionality beyond that. Currently, however, the definition of connected devices has vastly changed. Personal computers are no longer exclusively the devices with network connectivity; they have become just one of many categories of connected devices.

One of the key catalysts for this change was the introduction of smartphones, which redefined our relationship with the Internet. Smartphone users were no longer confined to a desk to access the Internet. Pocket-sized mobile phones became the primary interface to the network, and users could hardly imagine being disconnected as many services moved online. This shift led to a growing appreciation of connected devices, sparking a boom in the IoT industry. The number of connected devices, along with their functions and underlying standards, increased significantly. Currently, with the deployment of 5G, IoT technologies are expected to benefit immensely. 5G offers faster data rates, lower latency, and greater network capacity. Sensor networks are already reaping substantial benefits from 5G's reliability [66][163]. As a result, many devices previously considered isolated are now connected.

The predictions about the global number of IoT devices vary widely, ranging from 25 billion [17] to 125 billion devices [16] in 2030, with total revenue estimates between 1 trillion 1 trillion [14] to 1.5 trillion USD [10]. These predictions illustrate the significance of IoT, which will be increasingly evident with time. Nevertheless, despite their widespread adoption, IoT technologies continue to face significant challenges. Figure 2.1 presents a taxonomy of the challenges confronting IoT technologies.

The **first challenge** relates to the nature of many IoT devices, which are constrained and have limited resources in terms of processing power, memory, and power budget. These devices are designed to perform specific tasks, such as measuring temperature or detecting motion. Such tasks do not necessitate complex components or circuitry and are typically carried out using a few sensors with simple connections. In addition, manufacturers of such devices aim to mass-produce while keeping the costs low. As a result, the design and construction are kept simple, with only the resources necessary to perform their intended tasks being provided. The constrained nature of IoT devices complicates their management. Mechanisms commonly used on the Internet today for regular, more powerful devices can not be directly used with constrained IoT devices. These mechanisms, for example, include encryption and decryption of data and receiving software updates or security patches. Such mechanisms, therefore, must be redeveloped to suit the less powerful IoT devices.

The **Second challenge** lies in the identification of these devices. Whether it is an IoT device or a generic

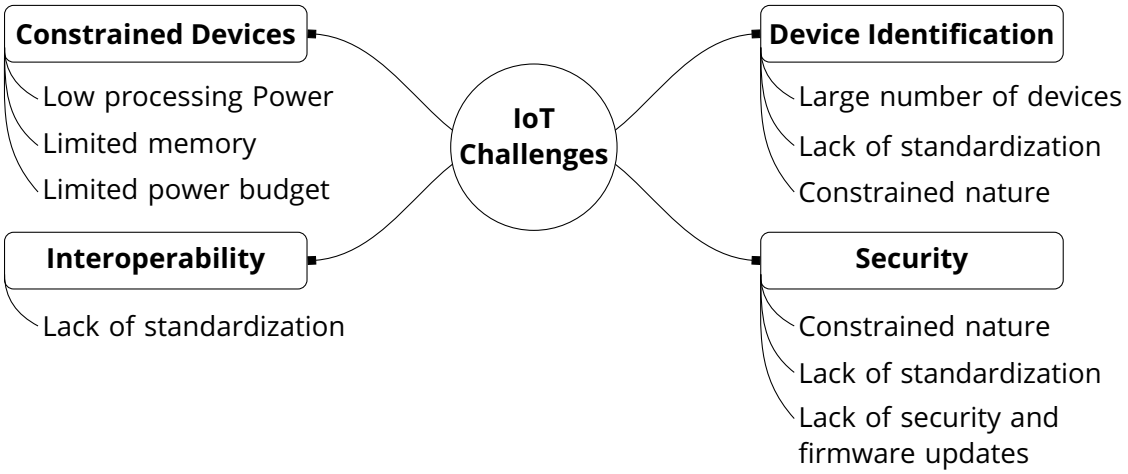


Figure 2.1: Taxonomy of challenges facing IoT environments.

device, any device connected to a network requires an identifier that uniquely identifies it, at least within the scope of its local network. Some common identifiers include IP addresses, which are used to identify devices on the Internet, and Media Access Control (MAC) physical addresses, which are used in the scope of local area networks (LANs). Identifiers serve at least one primary purpose: they allow the sender to specify the exact recipient of the data. Conversely, the recipient can use these identifiers to verify the source of the data or to determine whether the data is intended for them. The massive number of IoT devices, most of which are constrained, along with the lack of standardization, hinders finding a global identification scheme similar to IP addresses. The absence of global or sufficiently large-scale identification schemes makes it challenging to manage and track the large number of IoT devices, leading to potential security vulnerabilities and interoperability issues.

The **third challenge** is related to IoT security. The security of exchanged data is vital in any communication, and this holds especially true for IoT communications, where a significant amount of highly personal data is transmitted. However, contrary to regular Internet communications, where security and privacy mechanisms are generally standardized and broadly trusted, IoT technologies face significant security challenges. These devices have a relatively large attack surface, making them vulnerable to several cyber attacks. IoT devices, for example, are a primary target for attackers seeking to launch Denial of Service (DoS) attacks, as these devices are easily compromised. The constrained nature of IoT devices prevents them from using up-to-date security mechanisms, as these often require more memory and processing power resources than these devices have. In addition, the lack of standardization also impairs security since each manufacturer tends to use proprietary security mechanisms that are largely incompatible with those of other manufacturers. Lastly, these devices are usually not adequately maintained regarding software updates, especially critical security updates and patches.

Lastly, the **fourth challenge** we identify is the lack of interoperability between IoT manufacturers. The lack of global standards creates heterogeneity between different IoT technologies. For example, a motion sensor from one manufacturer may not be able to communicate with a motion sensor from a different manufacturer or with the backend server of that sensor, even if they belong to the same network. Moreover, devices from one network may not be able to communicate with devices from different networks if these networks are based on different IoT technologies. Heterogeneity arises in identification, data formatting, and security mechanisms used, among others. This causes the IoT environment to become vertically divided into separate silos where different technologies can not communicate. This lack of interoperabil-

ity disperses research and development efforts as a breakthrough in one technology is highly unlikely to be applicable to others. Interoperability between different IoT technologies is necessary for achieving a globally standardized IoT.

Given the spread of IoT technologies today and the projections of even larger-scale deployments in the future, identifying and addressing IoT challenges is imperative. While a single solution for all IoT drawbacks is unattainable, it is feasible to leverage existing tools to ameliorate IoT environments and their user experience. So, instead of developing new ones, it would be meaningful and efficient to consider existing tools and adapt them to constrained IoT devices to address these challenges.

A service that has stood the test of time as one of the cornerstones of the Internet is the Domain Name System (DNS). DNS, the Internet's naming service, has been fundamental since its inception in 1987 [27]. The role DNS plays, combined with its efficiency and robustness, has made it an indispensable service. The drawbacks of DNS, primarily stemming from its original design, which was not heavily focused on security [179], are being addressed through various initiatives. These efforts have led to several protocol advancements aimed at enhancing DNS security. For the billions of connected devices forming the IoT, DNS is also a necessary tool. Even the simplest IoT devices, such as thermometers or motion sensors, might need to use DNS to contact their backend servers [177]. IoT technologies could also utilize the DNS in other manners. For example, the functions of DNS and its distributed nature make it an ideal candidate for IoT naming and identification. Some proprietary IoT naming schemes are already using it [19]. Moreover, the various DNS security extensions and protocols help make IoT communications more secure if adapted to constrained devices.

However, even though DNS could be useful to use in IoT environments and could alleviate some of the challenges these environments face, the possible repercussions for DNS should not be overlooked. According to the projections discussed earlier, billions of IoT devices will be added annually. This raises questions regarding the scalability and security of DNS. There have already been some troubling impacts on DNS due to IoT, such as the infamous Mirai attack [5, 45].

This thesis revolves around using DNS in IoT environments to address some of the challenges these environments face. In this chapter, designated as the state of the art of our thesis, we explore how DNS is already being used to support and reinforce IoT. We first examine the challenges the IoT environments face. We primarily focus on the four classes of challenges we identified earlier: challenges due to the constrained nature of IoT devices and challenges related to the identification of these devices, security of IoT communications, and interoperability between different IoT technologies. We elaborate on each challenge and investigate how DNS is being used to address some of these challenges. We then illustrate the effects this usage might have on the DNS infrastructure. Overall, this chapter will give a better understanding of IoT challenges and the benefits DNS might have to address these challenges.

2.2 Challenges in IoT

In this section we explore the challenges the IoT environments face. These challenges could be divided into the following: the constrained nature of IoT devices, their identification, security, and the interoperability between different IoT technologies.

2.2.1 The Constrained IoT

The constrained nature of a large portion of devices in the IoT landscape is likely a major factor contributing to the challenges faced by these environments. These constraints, along with the vast number of IoT devices, set IoT devices apart from other generic devices and machines on the Internet. According to RFC

7228 [63], titled *'Terminology for Constrained-Node Networks'*, constrained devices are defined as devices that have:

- Limited Read-Only Memory (ROM)\Flash leading to limitations on the maximum code complexity
- Limited Random Access Memory (RAM) leading to constraints on the buffer sizes
- Limited processing power
- Batteries as sources of power
- Constraints on user interface and accessibility in deployment.

Table 2.1 shows the classes of constrained devices according to their data and code sizes. Class 0 devices are ultra-constrained and need intermediary devices such as gateways to communicate with designated backend servers. Class 1, on the other hand, could use protocols explicitly designed for constrained devices, such as Constrained Application Protocol (CoAP) over UDP [182]. Finally, Class 2 devices, which are more powerful, could use protocols used by regular devices on the Internet and benefit from lightweight specifically-designed protocols.

Table 2.1: Classes of Constrained Devices (KiB = 1024 bytes).

Name	**data size (e.g., RAM)**	**code size (e.g., flash)**
Class 0, C0	$\ll$ 10 KiB	$\ll$ 100 KiB
Class 1, C1	$\sim$ 10 KiB	$\sim$ 100 KiB
Class 2, C2	$\sim$ 50 KiB	$\sim$ 250 KiB

Such devices typically engage in gathering data from the physical world. The collected data are then relayed to one or more backend servers on the network for processing. The designers of such devices are limited in choosing their operating systems or security mechanisms due to the stringent constraints. The low processing power, memory, and power budget compel designers to abandon the well-maintained, popular operating systems used on personal computers and servers. Instead, they opt for operating systems that meet the constraints of IoT devices but are not highly regarded in terms of performance and code complexity. This, in turn, negatively affects the security performance of these devices, often leading to vulnerabilities that can be exploited for unauthorized access, data breaches, or even device manipulation. In addition, the way these devices are powered, mainly through batteries that are expected to last for years, adds further constraints. As a result, these devices are typically only active for short periods, operate at low bit rates, and often lack user interfaces, making them more difficult to maintain and monitor.

The limitations of IoT devices and the difficulty to closely monitor them due to their large number and typical lack of user interface make them the weakest link in any network setup.

2.2.2 Identification in IoT

Whenever a device needs to connect to a network, it requires an identifier that uniquely identifies it within that network. People use these identifiers daily, though they may often overlook their significance as it has become second nature. For example, the mobile phone number is one of the most widespread identification methods for devices connected to a network. It provides a unique global identifier for every user, and due to its uniqueness, a user can initiate contact with any other mobile phone number and be contacted

by others. This uniqueness guarantees that, by knowing a user's phone number, only the intended user is contacted and not any other user on the network. Regardless of the context in which an identifier is used, its structure must adhere to a specific set of rules, known as an identification scheme. Identification schemes establish the rules to be followed when creating and using an identifier. For instance, domain names are identifiers used in the context of the Internet to identify resources. However, creating a domain name is not arbitrary but follows specific rules and regulations, as outlined in the identification scheme in RFC 1035 [28].

IoT devices, as network-connected devices that send and receive data, also require identifiers to uniquely identify them within their respective networks. However, identification in IoT is not straightforward due to the vast number of IoT technologies and protocols, as well as the lack of interoperability between them. Different technologies use distinct identification schemes, which are often incompatible with one another. This lack of interoperability further fragments the IoT environment, as globally comprehensible identifiers are crucial for ensuring proper communication and interoperability between different IoT technologies.

The IoT landscape, composed of numerous technologies and protocols, is evidently divided and fragmented. This heterogeneity among IoT technologies, coupled with the lack of global standardization, prevents the creation of a single IoT identifier that can serve all variations within the IoT spectrum. As a result, it is difficult to specify how such an identifier should be formed or what identification scheme it should follow. Nevertheless, several vendor-neutral initiatives sought to establish a framework for IoT identifiers. These initiatives helped define a taxonomy and set requirements for IoT identifiers.

Taxonomy of IoT Identifiers

Two examples of the initiatives that sought to create a vendor-neutral and manufacturer-neutral perspective on IoT, particularly in the area of IoT identification, is the *EU-China Joint White Paper on Internet-of-Things Identification* issued by the EU-China IoT Advisory Group [72] and the *Identifiers in IoT* report issued by The Alliance for Internet of Things Innovation (AIoTI) [38]. These initiatives defined a taxonomy for IoT identifiers in an effort to organizes the various types of identifiers based on their function, purpose, or use case within IoT systems. Table 2.2 presents this taxonomy with the function and use cases for each category.

Requirements for IoT Identifiers

Additional examples of vendor-neutral and manufacturer-neutral initiatives that aimed to bring order to the fragmented landscape of IoT by establishing clear guidelines and standards for its identifiers are AIOTI [38] and the ITU-T [33] who suggested a set of requirements for IoT identifiers. Table 2.3 lists the most common requirements suggested by these two initiatives [38, 33].

Overview of Prominent IoT Identification Schemes

The identification schemes in IoT are diverse. Unlike machines using the TCP/IP protocol suite where each machine is identified by a unique IP addresse, IoT devices are identified by whichever identifier their manufacturer decides to use. This flexibility contributes to the wide variety of identification schemes in the IoT landscape. In the following, we introduce some of the most used ones. We begin with the IP address, which has been used as an identifier in 6LoWPAN. We follow that with the Digital Object Identifier, Electronic Product Code, and Object Identifier.

IP Addresses: We begin with one of the most common identifiers, the IP address. In the context of IoT, an IP address typically refers to an IPv6 address, as the 32-bit IPv4 addresses have already been exhausted.

Table 2.2: Taxonomy of IoT Identifiers.

Identifier Category	Function	Use case
Objects/Things Identifiers[38] [72]	Used to identify the entity of interest, which could be physical or virtual	Sensors, machines, humans, merchandise (physical) or data, files, metadata (virtual)
Communication Identifiers[38] [72]	Used to identify things in the scope of communicating with other devices, including Internet-based communications.	IP address, MAC address, E.164
Application and Service Identifiers[38] [72]	Used to identify applications/services used in the scope of IoT applications.	URL, URI, identifiers for different services on a single platform
User Identifier[38]	Used to identify physical or virtual objects that interact with IoT devices on the Internet.	ID for humans/animals (physical) or ID for software applications interacting with IoT devices (virtual)
Data Identifier[38]	Used to identify data instances and datatypes	Digital Twin, stored sensor measurements
Location Identifier[38]	Used to specify a location within a geographical region	Coordinates, postal codes
Protocol Identifier[38]	Used to identify protocols so that, for example, layers within a communication stack can identify what protocols are being used by other layers	Ethertype

The 128-bit IPv6, on the other hand, offers a massive address space (2^{128} addresses), which can theoretically accommodate both existing and future IoT devices. RFC 4919 [152] defines how IPv6 is used in Low-Power Wireless Personal Area Networks (LoWPANs), where devices align with our definition of constrained IoT devices. According to [152], these devices adhere to the IEEE 802.15.4-2003 standard, characterized by short-range communication, low bit rates, low power consumption, and limited computational and memory resources. In such networks, IPv6 functions as a unique identifier for each device within the network's scope. Beyond the large address space, IP addresses are favored due to their longstanding use, with readily available, well-known, and free-to-access standards and regulations. Additionally, IP addresses benefit from an established infrastructure that includes management, diagnostic, and commissioning tools [152].

In addition to its large address space, IPv6 offers several features that are particularly beneficial for IoT environments. One such feature is stateless address autoconfiguration [156], which allows IoT devices to automatically generate their own IP addresses without the need for manual configuration. Furthermore, IPv6 supports header compression through 6LoWPAN [189], reducing packet sizes and improving energy efficiency for constrained devices.

Digital Object Identifier: The Digital Object Identifier (DOI) is defined in the ISO 26324:2012 standard [30]. It was initially developed by the International DOI Foundation (IDF), which was formed by three publishing institutes [9] and later standardized by ISO 26324:2012 [30]. Later, the DOI system became the

Table 2.3: Requirements for IoT identifiers.

Requirement	**Definition**
Identify anything physical or virtual[33]	The identifier should be able to identify any physical or virtual thing as it is required that any physical or virtual thing can be connected to network infrastructure, which implies the necessity of having an identifier.
Communication between things[33]	Connection between things using identifiers, regardless if a particular thing needs to communicate or not, should be guaranteed.
Networking technology independence [33]	Identifiers should be independent of the underlying network technology used by the thing they identify.
Uniqueness [38]	Uniqueness is required within the specific application context. If a larger scope is needed where identifiers are no longer unique, a replacement or an extension of the identification scheme is always necessary to guarantee the identifier's uniqueness.
Security and privacy [38]	Identifiers used should preserve privacy and protect personal information. Ideally, they should not disclose any details about the entity they represent. The structure of the identifier should not reveal information about the identified object or entity.
Scalability [38]	The identification scheme should be able to accommodate the increasing number of things needing identification in the future.
Interoperability[38]	Even in the absence of a single identification scheme, the existing and any newly proposed identification schemes should account for interoperability between different schemes.

standard tool for identifying objects digitally. The word *Digital* in Digital Object Identifier refers to the identifier. DOIs are meant to be unique, persistent, and permanent digital identifiers for objects. Information about an object identified by a DOI can be retrieved upon resolving the DOI. DOIs are formed of two parts, prefix, and suffix, separated by a `'/'` and with no maximum length. The prefix identifies a unique naming authority that is responsible for assigning DOIs. The suffix is a unique identifier of objects which, for interoperability purposes, could be an existing identifier. A typical DOI, for example, is $10.100/20$.

The Handle System [131, 13] is responsible for the resolution of DOI. The Handle system is a set of distributed servers that allow the storage of handles that refer to digital objects. The system can efficiently and securely resolve the handles into information to locate and access the intended object. Additional services could be added, such as data confidentiality, data integrity, and non-repudiation.

The Electronic Product Code (EPC): The EPC is a universal identifier for physical objects operated by GS1 [12]. It is used whenever an object needs to be tracked or identified. EPC is predominantly known as the ID used in Radio Frequency Identification (RFID). RFIDs act as data carriers holding the object's EPC to which they are attached. However, this is not always the case. EPC is not exclusively used in RFID; the latter does not always contain an EPC. When stored on computer systems, EPCs are in the form of a Universal Resource Identifier (URI), often referred to as Pure Identity EPC URI. A typical Pure Identity EPC

URI in Uniform Resource Names (URN) notation, for example, is

$$urn : epc : id : sgtin : 0614141.112345.400$$

On RIFD tags, and due to their memory limitations, EPCs are encoded in binary form. The resolution of EPC codes is done using the Object Name Service (ONS) [19], which is based on DNS.

Object Identifier: The Object Identifier (OID) [18] was developed jointly by ITU-T, starting with the ITU-T X.660 series [29], and ISO through ISO/IEC 9834-1:2012 [31], with the goal of providing an unambiguous, persistent name for objects. The OIDs are organized in a hierarchical tree structure. The top-level is called the root; under it are nodes from which branch infinitely many arcs. An object's name is the path from the root downwards until the node related to that object is reached. A typical OID in dot notation is

$$1.2.840.113549.1$$

and in URN notation

$$urn : oid : 1.2.840.113549.1$$

For resolution, OID Resolution System (ORS) is a DNS-based system that accepts queries about OIDs and returns associated information. Various objects, big and small, could be identified using OIDs, such as countries, companies, X.509 certificates, standards, and Simple Network Management Protocol Management Information Bases (SNMP MIBs), to name a few.

2.2.3 IoT Security

IoT security is arguably one of the most heavily researched areas within the field of IoT. This is due to the vast number of interconnected devices, the sensitive nature of the data they handle, and the potential for significant vulnerabilities that could be exploited by attackers. RFC 8576 [95] titled *IoT Security: State of the Art and Challenges* gives a general overview of the security challenges facing IoT environments. IoT security drawbacks are not due to a lack of security mechanisms but have roots in the design of the IoT devices. The constrained IoT devices are meant to serve particular purposes, such as measuring temperature or detecting motion. Therefore, vendors of such devices seek to keep the designs simple and the prices low. So, most constrained IoT devices were designed without considering the security threats they may face. These facts prevent these devices from using modern security functions designed for more powerful ones. Therefore, they either abandon security or use weak protocols and implementations [95]. In addition, given their constrained nature and rapid development, some IoT devices do not receive necessary firmware updates as often as they should [200]. These firmware updates should be regular to avoid depriving these devices of possibly essential security updates. The Internet of Medical Things (IoMT) is one domain where security and privacy are paramount due to the sensitivity of the patient data. It is, however, vulnerable to the same attacks as other IoT devices [96] [86][166].

Figure 2.2 shows the classification of some IoT attacks [151, 178] based on the three-layer model for IoT architecture.

- **Perception Layer Attacks:**
  - **Node capture attack:** a physical attack against IoT nodes where the attacker captures the node and gains control over it. The attacker can then either impersonate the node, block incoming and outgoing traffic, or gain unauthorized access to the network associated with the node. Additionally, node capture attacks often allow the attacker to extract sensitive information, such as cryptographic keys, which can further compromise the security of the network.

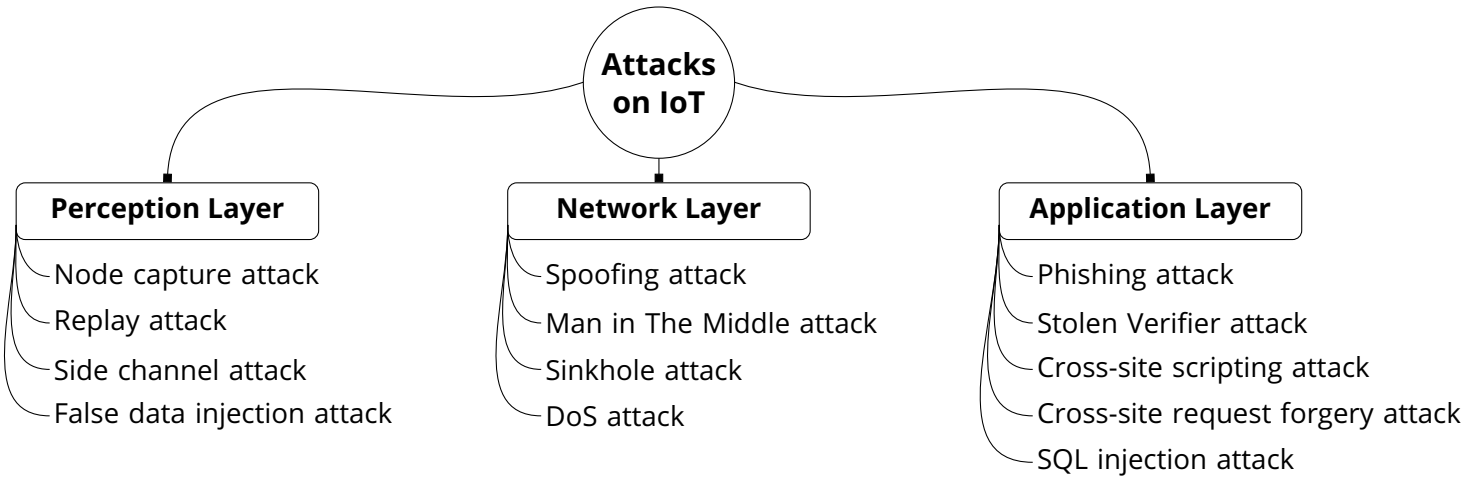


Figure 2.2: Attacks against IoT (three-layer architecture).

  - **Replay attack:** Replay attacks happen when the attacker captures a legitimate message destined for the device and saves it. The attacker later retransmits (replays) the same message to trick the devices about the sender's identity. Replay attacks could allow attackers to access the network and control the devices.
  - **Side-channel attack:** Side-channel attacks occur due to unintentional device information leakage. This includes power consumption, acoustic emissions, or timing information. Using this information, an attacker might be able to, for example, guess the device's key based on the power consumption during encryption or decryption.
  - **False data injection attack:** occur when attackers inject false data into devices. These attacks can be carried out manually on compromised devices, using Man-in-the-Middle or malware. False data injection attacks are particularly dangerous in critical applications such as healthcare.

- **Network Layer Attacks:**

  - **Spoofing attack:** occurs when an attacker impersonates legitimate devices or users on the network. This is achieved by altering data to make them look like they originated from other users or devices. Spoofing includes, for example, MAC spoofing, IP spoofing, or DNS spoofing.
  - **Man in The Middle attack:** Such attacks happen while data is transmitted between devices or between a device and its background servers. The attacker intercepts the data and could sniff, alter, or block the communication.
  - **Sinkhole attack:** occurs when attackers redirect IoT traffic to a compromised device or malicious server instead of the legitimate destination. This is done by manipulating the routing protocols used in IoT networks, causing data to flow through a compromised device. Once the traffic is rerouted, the attacker can drop, modify, or analyze the data.
  - **DDoS attack:** Denial of Service (DoS) attacks occur when a target server is overwhelmed by sending a large number of requests. The target could be a DNS server or a regular web server. Distributed Denial of Service (DDoS) attacks includes multiple compromised devices that are

used simultaneously to flood a target. IoT devices are particularly prone to being recruited for DDoS attacks due to their weak security properties, making them easily compromised and controlled by attackers, as seen in notable botnets like Mirai [45, 5].

- **Application Layer Attacks:**
  - **Phishing attack:** occurs when an attacker sends a malicious file or link to users of IoT devices posing as a legitimate entity such as a service provider or a manufacturer. Malware is installed upon clicking the link or opening the file, which might grant the attacker access to the network and control over its devices.
  - **Stolen Verifier attack:** occurs when an attacker obtains a password or an authentication token that grants them access to the network and the devices connected to it. The attacker can then impersonate the users or devices of the network, leading to possible information theft or data corruption.
  - **Cross-site scripting (XSS) attack:** is an injection attack where the attacker injects malicious code into a webpage or web application. This could be, for example, the web interface for managing IoT devices. As a result, the attacker can control the devices, steal sensitive information, or corrupt data.
  - **Cross-site request forgery (CSRF) attack:** occurs when an attacker tricks a user into performing a malicious action on the webpage the user is already authenticated to. The attacker can then gain access to the network, steal or corrupt data, or control the devices connected to that network.
  - **SQL injection attack:** occurs when an attacker inserts malicious SQL statements into an application's input field. As a result, the attacker can corrupt, steal, or modify data. In the context of IoT, an IoT device with a web interface that accepts user input, such as a security camera with a login page, may be vulnerable to SQL injection attacks.

The growing awareness of IoT vulnerabilities and the potential impact on security objectives, if exploited by attackers, has prompted significant attention from the research community [158][123][150]. IoT has also benefited from the development of new technologies like Blockchain [151] [149] [78], along with advances in Machine Learning (ML) [192][143], which have been employed to enhance security and mitigate potential attacks.

2.2.4 IoT Interoperability

In IoT, each vendor typically maintains its own infrastructure, devices, Application Programming Interfaces (APIs), and data formats [49]. The absence of a regulatory body to enforce standards for the creation and promotion of technology results in a notable lack of interoperability between products from different vendors. Interoperability is regarded by the ITU-T as one of its high-level requirements for IoT, stating that: *"interoperability is essential among heterogeneous and distributed systems for the provision and consumption of a variety of information and services"* [32].

RFC 8477 [121] attributes the lack of interoperability to the lack of an encoding-independent standardization and the link between the data formats and the technology that produces it. While there is no shortage of IoT standards, their abundance and the fact that they are developed by numerous organizations [133] make achieving interoperability even more challenging. Heterogeneity could be seen as device-level heterogeneity due to the various technologies and protocols used in devices, data-level heterogeneity

due to various formatting of data, and semantic heterogeneity related to how different technologies interpret data they receive from other technologies [167]. Semantic heterogeneity and achieving semantic interoperability have been studied extensively [167] [155] [94] [100]. Due to the lack of standardization, IoT technologies often send and receive data in proprietary formats. As a result, one IoT device may be unable to interpret the meaning (semantic) of data sent by another device if the two belong to different technologies. Solutions to address this semantic heterogeneity include the use of middleware, ontologies, and semantic web technologies.

Several organizations issued proprietary standards in an attempt to facilitate interoperability. We present two of them: The Web of Things and OneM2M.

The Web of Things: The Web of Things (WoT) [23][79], created by the World Wide Web Consortium (W3C), is a concept that aims to enable interoperability between existing IoT technologies by connecting things in these networks to the web. Rather than introducing new technologies, WoT aims to preserve and complement existing IoT systems, facilitating communication and integration across various platforms. The main building blocks of the WoT are:

- **Thing Description (TD):** The TD is a central building block of WoT. It contains metadata describing the thing, a set of Interaction Affordances indicating how the thing can be used, schemas for the data exchanged with the thing for machine understandability and web links to express the thing's relation with other things or documents.
- **Binding templates:** consist of reusable vocabulary and extensions to the TD format that enables a client to interact with diverse things exposing different protocols.
- **Scripting API:** an optional block that enables implementing the application logic of a thing using a common JavaScript API similar to web browser APIs.
- **Security and Privacy Guidelines:** guidelines for secure implementation and configuration of things.

Human-assisted semantics translation could extend the WoT, enabling IoT devices to interpret metadata from devices that use different semantics. This approach would further enhance semantic interoperability between heterogeneous devices by bridging the gaps in understanding between various technologies [160].

OneM2M: OneM2M [20] was established in 2012 as a partnership project between 8 standardization bodies. The goal was to address the interoperability problem in IoT and machine-to-machine communications by promoting a global IoT standard. OneM2M does not propose a new standard but builds on the existing standards and aims to facilitate interoperability. This is achieved by defining a common middleware between IoT devices, communication networks, and IoT applications. This middleware service layer contains a suite of common service functions (CSFs) exposed to IoT devices and applications via RESTful APIs. CSFs are general-purpose services unrelated to specific IoT domains or technologies. They can be looked at as generic operating system tools that various IoT devices and applications could use. OneM2M is reportedly designed to allow any IoT application to connect with any IoT device, promoting interoperability between isolated IoT silos.

The rapid development of IoT and its transition from a luxury to a necessity make it imperative to address its challenges. These challenges, as discussed above, are numerous and often interconnected. At the core of these issues is the constrained nature of IoT devices. When a device is built with such constrained specifications, it is automatically deprived of many tools and capabilities available to modern computers

and servers. Tools that deal, for example, with security, privacy, and identification are not compatible with constrained IoT. Another undeniable aspect of IoT is its diversity, driven by the wide range of technologies, each employing its own proprietary standards and tools. While developing new technologies is essential, the clear lack of interoperability between these diverse systems hinders IoT from evolving into a truly global network. Moreover, this lack of interoperability slows down research efforts, as progress becomes fragmented and specific to individual technologies rather than benefiting the entire IoT ecosystem. This dispersion of research efforts leads to slower, more isolated advancements rather than collective progress for the broader IoT environment.

The promising outlook for the future of IoT calls for action to ensure that its growth remains safe, secure, and beneficial to all stakeholders. Having more connected devices means, on the one hand, more load on network infrastructure and, on the other hand, more personal data entrusted to these devices. Hence, it is pivotal to assess the current challenges of IoT and to elaborate on possible solutions. Properly addressing these challenges will allow a safer and more prosperous future for IoT. When thinking about solutions, it is essential to find solutions that do not themselves create further challenges. For example, devising a name resolution mechanism for a single IoT technology that is not interoperable with other technologies is counterproductive since the IoT landscape as a whole will not benefit from that solution, only that specific technology. Hence, considering solutions encompassing as many IoT technologies as possible is of great benefit and importance. DNS could play a significant role in addressing the challenges of IoT. DNS was initially designed with the primary goal of mapping domain names to IP addresses. Since its inception, numerous extensions and functionalities have been added, giving DNS the maturity needed to become a reliable tool for securely storing and retrieving information for various applications. One example is DANE, where DNS is used as a complement to or even a complete replacement for certificate authorities in TLS handshakes, demonstrating that DNS can play roles in areas previously considered beyond its traditional scope [54].

2.3 DNS and IoT

Chapter 1 demonstrated the ability of DNS to evolve. Specifically, we explained how DNS transitioned from a basic name-to-address translation protocol to one capable of providing security and privacy for DNS traffic. Over the past forty years, DNS has proven to be an operationally viable and highly distributed infrastructure, supporting the backbone of the Internet. Up to this point in this chapter, we have discussed the challenges facing IoT environments, including the constrained nature of IoT devices, the complexities of identifying these devices, the lack of interoperability between the diverse technologies in the IoT landscape, and, finally, the security vulnerabilities inherent to IoT.

The remaining sections of this chapter explore state-of-the-art works that leverage the DNS infrastructure and its protocols to address IoT challenges, offering seamless identification, interoperability, security, and privacy for IoT devices. We conclude by evaluating the effectiveness of DNS in IoT and the impact IoT has on DNS. Table 2.4 summarizes the use cases of DNS in IoT environments.

2.3.1 DNS for constrained IoT

As described in Section 2.2.1, IoT devices are constrained in nature, with limited processing power, memory, and energy resources. Since DNS was originally designed without taking constrained devices into account, its memory and bandwidth requirements often exceed the capacities that IoT devices can handle

Even though most DNS responses fit in a 512-byte UDP packet[28], measurements between recursive resolvers and authoritative servers indicate that the network behavior is relatively uniform with IP packet

Table 2.4: Summary of DNS Use Cases in IoT environments.

DNS for constrained IoT	DNS for IoT name resolution
• QNAME minimization [64][80] and using elliptical curves [119] • DNS over CoAP (DoC) [182][138] • DNS over Datagram Transport Layer Security (DTLS) [172]	• Object Name Service (ONS) [19] • OID Resolution System (ORS) [18] • IoT name resolution [197][128] • Finding and localizing IoT devices [125][92] • Device autoconfiguration [136][137][135] • Device identification [162] • IoT roaming [59]
DNS for IoT security	**DNS for IoT interoperability**
• Secure IoT name resolution using DNS's security extensions. • PKI for IoT [59][54] • Authentication for device autoconfiguration [134][129] • Malicious activity detection through DNS traffic analysis [193][185]	• Overlay mechanisms for IoT naming [19][84][18][9] • PKI that encourages IoT Interoperability using DNS and its security extensions [59] • Interoperable service discovery using mDNS/DNS-SD [87][188][187]

sizes up to 1,500 bytes [6]. DNSSEC operating with RSA signatures leads to significantly higher memory requirements [35]. Such large messages increase CPU usage and require high bandwidth.

QNAME minimization [64] [80] and using elliptical curves [119] can considerably reduce the bandwidth and CPU usage. DNS over CoAP (DoC) is a protocol that enables DNS queries and responses to be transmitted over the Constrained Application Protocol (CoAP) [182], allowing DNS resolution to function efficiently in constrained IoT environments, where CoAP allows for HTTP-like communication on constrained nodes [138]. DNS over Datagram Transport Layer Security (DTLS) [172] is based on TLS protocol and provides encryption for queries and responses between DNS clients and servers. It is more suited for constrained IoT scenarios that support low latency and loss-tolerant communication.

The constrained nature of most IoT devices will allow adopting DNS to solve many of its challenges. The DNS accounts for constrained devices through several extensions and protocols that are adapted to function according to their constraints.

2.3.2 DNS for IoT name resolution

Given the primary role of DNS, which is to map domain names to IP addresses, it is expected to play a significant role in resolving IoT identifiers. Just like regular, non-IoT devices, IoT devices will require identifiers and systems to resolve these identifiers into addresses. These addresses can then be used to query IoT devices for their data, control and manage them, or redirect to locations containing relevant information about the device in question.

As explained in Section 2.2.2, the naming conventions for IoT are numerous and, in most cases, isolated from one another, *i.e.*, not interoperable. One potential solution to address the heterogeneity in identification schemes is for a standardization body to develop a globally unique naming convention and require different stakeholders in the IoT ecosystem to adopt it. This could be achieved from a purely technical perspective. For instance, the large IPv6 address space allows for every IoT device to have a unique identifier. However, establishing a single global naming convention for all IoT devices is nearly impossible. Industries such as retail, automotive, and defense have long relied on proprietary naming conventions, and altering these systems would significantly affect their infrastructure and operations.

A more feasible alternative is to preserve the existing naming conventions while enhancing the resolution processes they use. This is where DNS could be leveraged. For instance, DNS has been used with ENUM (Electronic NUmber Mapping) to map telephone numbers to web addresses [108]. Additionally, several DNS-based services, such as the ONS [19] and the ORS [18], allow for the registration and resolution of unique identifiers for IoT devices. This demonstrates the feasibility of DNS playing a more global role in name resolution for IoT, offering a familiar and robust solution for managing IoT devices and their namespaces.

The authors in [197] propose a new scheme for peer-to-peer equal name resolution. In the proposed method, names from various technologies are hashed into a string of bits. This not only preserves privacy by one-way hashing the name but also allows for resolution via DNS by simply adding a top-level domain (TLD), such as `'.iot'`. In [128], the authors studied the use of DNS architecture for IoT in the context of transport logistics. A hierarchical organization of DNS was presented, capable of scaling globally by translating unique URIs into network addresses, which can be used to extract information about the object of interest, such as its status or location.

The global reach and distributed nature of DNS were utilized in [125], where the existing DNS architecture was adapted to build a search engine for the WoT, specifically for devices and their offered services. A TLD such as `'.env'` is added, allowing users to use a regular browser to look up, for example, `'service.location.env'`. DNS resolution of `'service.location.env'` would return a list of devices offering the requested service at the specified location. To select a specific device, an additional level is appended to the name, specifying the desired device. The authors in [92] proposed a scheme to represent the semantic metadata of IoT devices and encode it into domain names, allowing devices to be discovered via DNS queries. The work also introduces the concept of *DNS as a Source of IoT Data*, suggesting that DNS could be used to store TXT records containing information about IoT devices.

In addition, DNS can assist with device autoconfiguration, which becomes particularly useful in scenarios where the number of IoT devices is too large to be individually named. Autoconfiguration using DNS allows devices to name themselves and register within their DNS zone. DNS Name Autoconfiguration (DNSNA) for IPv6-based IoT environments was proposed in [136] [137]. DNSNA provides a global framework for IoT autoconfiguration, including the definition of DNS name formats for such devices, name generation, and registration of the generated DNS names. DNSNA uses the IPv6 Neighbor Discovery Protocol (ND) to acquire the DNS search list through IPv6 Router Advertisement (RA) or DHCPv6. Once the DNS search list is obtained, the IoT device can generate its name using the DNS search list and its own information. The authors in [135] extended DNSNA to IPv4 IoT devices by proposing DNSNAv4, allowing IoT devices to register their DNS names using a DHCP server.

Moreover, the authors in [162] proposed IoTFinder, an IoT identification method based on DNS traffic analysis. IoTFinder is an ML-based multi-label classifier designed to automatically learn statistical DNS traffic fingerprints. Additionally, the authors in [59] introduced IoTRoam, a roaming setup for IoT devices. The use case for IoTRoam was demonstrated with a LoRaWAN network, where a device could locate and join its dedicated Join Server using DNS.

IoT is still far from having a global naming scheme with a unified name resolution mechanism, and a single solution to accommodate all technologies remains impractical at this time. However, this challenge remains manageable. Some approaches keep the existing naming schemes and work on an upper layer. For example, phone numbers differ in structure across countries but can still interoperate globally using international codes. In this case, while diversity would not be eliminated, it would be manageable. DNS can help many technologies interoperate while maintaining their individual naming schemes. As discussed earlier, this approach is already being used between individual technologies to achieve peer-to-peer interoperability via DNS.

2.3.3 DNS for IoT security

IoT presents several security risks to both consumers and businesses. As discussed in subsection 2.2.3, the security risks in IoT environments are numerous. However, despite these risks, the security mechanisms currently deployed in IoT environments are inadequate and often proprietary, as each IoT provider tends to develop its own closed security solution. Consequently, unlike the regular Internet, IoT lacks a global security mechanism with established trust anchors. Moreover, the constrained nature of IoT devices deprives them of the ability to use security mechanisms commonly employed on the Internet, such as complex cryptographic algorithms, which require more processing power and memory than these devices typically possess. As a result, bootstrapping trust, supporting secure communications, and ensuring privacy remain significant challenges for most highly constrained IoT devices.

IoT data communication often requires a gateway to translate between the IoT device (e.g., sensors communicating via protocols such as Bluetooth or LoRa) and the endpoints (e.g., cloud infrastructures using HTTP over IP) in the Internet core. One of the fundamental requirements in IoT is to control the terms under which an IoT device is permitted to onboard into the Internet core. Similar to regular devices on the Internet, IoT devices must be authenticated when joining a network. Such devices require an identifier and an authentication token to be admitted. However, IoT technologies employ different, predominantly non-interoperable, mechanisms for accepting new devices. These mechanisms are often weak, and their lack of compatibility further exacerbates interoperability challenges. DNS has been used to secure certain aspects of IoT communications, including both the process of IoT devices joining and registering with their networks, as well as their subsequent communications.

The authors in [59] propose a Public Key Infrastructure (PKI) based on DNS for the secure onboarding of IoT devices. The work in [134] builds on DNSNA by adding authentication. The resulting algorithm, Secure Domain Name System Name Autoconfiguration (SDNSNA) for IPv6 IoT devices, uses DNSNA for name generation and NFC-based authentication to verify and register devices. The concept allows users to communicate with an authentication server via their smartphone, which in turn communicates with IoT devices using NFC. More work on securing autoconfiguration and registration was done in [129] with the development of Advanced Secure DNS Name Autoconfiguration with Authentication and Authorization for enterprise IoT networks (ASDAI). In [85], the authors extended the EPC ONS, based on DNS, to dynamically support heterogeneous object code identification. The authors in [193] designed an IoT router that analyzes DNS traffic to detect whether IoT devices are consulting unknown DNS servers, a common behavior of compromised devices, such as those in botnets. Botnets, consisting of compromised devices controlled by a Command-and-Control (C&C) server, typically use DNS to connect to the server [141] [83]. DNS Filtering & Extraction Network System (D-FENS) was proposed in [185]. D-FENS operates between the client and the recursive DNS server, intercepting DNS requests. It then uses a deep learning model to accurately detect and blacklist unreported malicious domains that IoT devices might attempt to connect to.

Several protocols and extensions were devised to secure DNS, as discussed in Chapter 1, and these

could benefit IoT devices utilizing DNS. The benefits for IoT include enhanced security, privacy, and authentication.

2.3.4 DNS for IoT interoperability

The IoT infrastructure must integrate various IoT connectivity technologies, along with hardware, software, identification, security, privacy, and resolution services. These components and services are provided by multiple stakeholders. In addition to vertical integration, horizontal interoperability between devices and systems will be crucial for the success of an IoT network (*i.e.*, ensuring that devices from different ecosystems can work seamlessly together, in addition to integration within a single ecosystem).

Interoperability, for different technologies, refers to the ability to communicate seamlessly, allowing for the exchange and use of data, services, or functionality, regardless of the underlying technology or standards. The lack of interoperability remains one of the major challenges facing IoT today. As discussed in subsection 2.2.4, the absence of interoperability between IoT technologies has led to a highly fragmented environment composed of many incompatible systems. These technologies often develop and implement proprietary solutions, using communication protocols and data representation methods that few others can understand. As a result, users and devices within one IoT technology are typically restricted to communicating with those within the same ecosystem. This fragmentation has effectively turned IoT into isolated islands or silos, each with its own standards.

DNS could play a role in alleviating some of IoT's interoperability issues. Its robustness and distributed nature have already encouraged some organizations to resort to it when looking for IoT interoperability solutions.

The Object Name Service (ONS) [19] is the resolution system of EPC [12], utilizing DNS to resolve an EPC identifier to the location of the information associated with that identifier. The work in [84] proposes an enhancement to the ONS framework, enabling it to support heterogeneous object code identification. This approach shares similarities with established conventions such as OID [18] and DOI [9, 30]. In [59], the authors presented a PKI framework for IoT devices, built on DNS and its security extensions, which aims to facilitate interoperability between heterogeneous IoT technologies by providing a unified PKI infrastructure to enhance security and simplify management.

DNS-Based Service Discovery (DNS-SD) [70] demonstrates how DNS can be used to locate named entities. DNS-SD employs DNS Service records (SRV records) to locate services by specifying the service type and the domain it belongs to, in the format `'service.domain'`. The client requesting the service receives a list of available services matching the query in the form of `'instance.service.domain'`, from which a service instance is selected. DNS-SD is compatible with both standard unicast DNS and the DNS-like Multicast DNS (mDNS) [71]. DNS-SD was used in [87] to develop an interoperable service discovery mechanism for IoT environments. Other works [188, 187] also proposed solutions for service discovery of resource-constrained devices based on mDNS/DNS-SD.

DNS plays a vital role in enabling IoT interoperability. Beyond its primary function as a naming service, DNS, with its security extensions, offers a reliable and secure means to map IoT device names to network addresses or associated information. This capability encourages different IoT technologies to either consolidate their naming mechanisms or adopt DNS-based services to achieve interoperability across various naming standards. By doing so, a cohesive Web of Things can be built, featuring distributed registries that contain information about IoT devices and their services. Additionally, DNS supports dynamic updates, facilitating the automatic reconfiguration of device addresses, which makes it particularly well-suited for dynamic IoT environments.

2.3.5 Impact of IoT on DNS

So far, the discussions in this chapter promoted using DNS to address many of the challenges in IoT environments, demonstrating how DNS can help mitigate these challenges. However, Chapter 1 also highlighted the critical role of DNS as one of the cornerstones of the Internet. This importance warrants caution when exposing this vital piece of Internet infrastructure to IoT environments. The massive number of IoT devices, along with their constrained nature, poses a potential threat to DNS. IoT manufacturers often give little consideration to the security of their devices, as they are typically designed for specific functions with minimal security requirements. Furthermore, the constrained resources of IoT devices prevent them from implementing the advanced security solutions commonly used on the Internet. Additionally, many constrained IoT devices, such as sensors, lack user interfaces, making it difficult for users to detect if their devices have been compromised or are being exploited to launch attacks.

In [102], several challenges facing DNS in IoT environments are explored, with a focus on functionality, security, and availability issues. The work in [198] examines whether the current DNS infrastructure is prepared for IoT. A set of criteria that should be met before using DNS with IoT is laid out, followed by an analysis of whether these criteria are fulfilled. The conclusions drawn can be summarized as follows:

- **Security:** Security is a crucial enabler of IoT, and while DNS security has been enhanced, it remains too resource-intensive for constrained IoT devices.
- **Mobility:** An IoT naming service should support mobility and automatic name updates. While DNS is capable of handling automatic name updates, it was not originally designed with mobility in mind and therefore lacks inherent support for this feature at present.
- **Infrastructure Independence:** Name resolution should generally be independent of the underlying infrastructure. DNS can facilitate this by leveraging local link extensions and technologies such as cloud computing.
- **Localization:** All devices must be localizable and reachable. The authors argue that DNS is evolving to address service deployment and name format localization, making it better suited to meet these requirements.
- **Efficiency:** Efficiency is crucial for latency-sensitive IoT services. However, it remains a significant challenge for DNS, as its name resolution mechanism can introduce delays due to hierarchical delegation and unpredictable cache hits.

Even though using DNS with IoT is beneficial for the IoT environment, it must be approached with caution. The ability of IoT devices to access DNS servers presents significant risks to the DNS infrastructure. This applies to both public DNS servers used on the Internet and private DNS setups in isolated networks. Given the critical role DNS plays, these risks should not be underestimated.

The risks of using DNS with IoT can be summarized as follows:

- **Complex coding at the IoT layer:** The improper design and configuration of some IoT devices increase the likelihood of simple mistakes, creating vulnerabilities that can be exploited to launch DDoS attacks[107] [51].
- **DDoS attacks:** The large number of IoT devices, coupled with their security vulnerabilities, allow for complex and increased size DDoS attacks against Internet infrastructure, which includes the

DNS[107][51]. The Mirai botnet DDoS attack against the DYN DNS service provider was unprecedented at the time, reaching 1.2 Tbps[5, 45]. The massive volume of devices involved makes traditional filtering methods, such as IP address-based blocking, less effective, allowing these DDoS attacks to persist for extended periods.

- **DDoS amplification:** Also known as reflection attacks, these attacks exploit the fact that open DNS resolvers often respond with data much larger than the original query. Attackers abuse this by sending multiple DNS queries while using the victim's spoofed IP address as the source for these queries. The DNS servers then send their larger responses to the victim's machine, overwhelming its memory and CPU and potentially taking it out of service [51].
- **DNS vulnerability:** In 2021, a vulnerability was discovered in several popular TCP/IP stacks used in both IT and IoT firmware, known as 'Name: Wreck'. This vulnerability allows attackers to exploit devices for remote code execution and denial-of-service attacks [51].

2.4 Discussion

This chapter aimed to investigate the potential benefits of using DNS to address the challenges of IoT environments. As illustrated in Figure 2.1, IoT technologies face several challenges. The scale at which IoT integrates daily life and the prediction of larger-scale deployments in the future demand that its challenges are carefully examined and mitigated.

The constrained nature of IoT devices is both one of its major limitations and a factor that has contributed to its widespread adoption. These constrained devices are unable to use state-of-the-art protocols, which are designed for more powerful devices like personal computers with sufficient processing power and memory. As a result, IoT devices often rely on less sophisticated protocols, particularly in terms of security, putting the data they transmit and receive at risk. However, these constraints—related to processing, memory, and power—have also made IoT technology more affordable and accessible, fostering its rapid adoption. DNS, along with its constrained-friendly extensions specifically designed for IoT, offers a potential solution to help IoT devices benefit from modern tools while still accommodating their limitations.

The diverse IoT technologies, which are mostly isolated from one another, make it challenging to create a unified system for naming and identifying IoT devices. This is due to the diverse communication and data representation protocols employed by each manufacturer. Many of these protocols are proprietary and are not readily available for use by other technologies. The DNS, as the Internet's established naming system, is a natural candidate to address this issue. Its well-established infrastructure supports the idea of using DNS to overcome the IoT identification challenge. As discussed in Section 2.3, several initiatives already rely on DNS for IoT name resolution; however, a global solution has yet to be found.

Beyond identification, significant interoperability issues exist between IoT technologies. These technologies are often managed by independent authorities, follow their own standards, and are incompatible with one another. This lack of standardization results in a fragmented IoT environment, made up of isolated vertical silos where one technology cannot communicate with others. Addressing this lack of interoperability is critical to the future of IoT. Improving interoperability among most, if not all, IoT technologies will help transform IoT into a truly global network. Moreover, the lack of interoperability slows down research and development efforts, as progress becomes fragmented and specific to individual technologies, rather than benefiting the entire IoT ecosystem. Given DNS's role in turning the Internet into a cohesive network, it holds great potential to help transform the IoT environment from isolated silos into a more interconnected system.

Finally, IoT security is a major concern, particularly given the sensitive nature of the data flowing through IoT networks, which span applications from smart homes to healthcare and industrial environments. The increasing drive to connect everything promises a more convenient future but raises the stakes for security, as more data shared across the network increases the risk of security breaches. DNS can play a significant role in improving the security of IoT communications. DNSSEC, which is being progressively adopted across the Internet, can help ensure the integrity of DNS responses received by IoT devices, mitigating risks like DNS cache poisoning, spoofing, and other cyber threats. DNSSEC's role in safeguarding DNS response integrity is crucial, and its broader adoption will enhance the security of both IoT and traditional Internet communications.

In terms of privacy, DNS offers several extensions, including standardized options like DoT and DoH, although these need wider adoption. DNS can also be adapted for constrained IoT environments, as demonstrated by DoC. To prevent IoT networks from becoming the weakest link in cybersecurity, it is essential to implement DNS and its security extensions and protocols.

2.5 Conclusion

DNS is a valuable tool that has proven its worth as a foundational pillar of today's Internet. When applied in IoT environments, it holds the potential to address many of the challenges these environments face. Users, manufacturers, and administrators of IoT networks can reap significant benefits from leveraging DNS's capabilities. As outlined in this chapter, IoT faces numerous challenges, including device limitations, lack of standardization, security concerns, and interoperability issues. However, DNS can effectively address these challenges, promoting the widespread adoption and integration of IoT technologies.

DNS's flexibility allows it to be adapted for constrained devices, as demonstrated by its extensions like DNS over CoAP (DoC), enabling IoT technologies to take advantage of DNS's capabilities. Its scalability also makes DNS well-suited for accommodating the rapidly growing number of IoT devices. Additionally, DNS offers a universal standard for addressing devices, helping to alleviate some of the interoperability issues in IoT environments and enhancing efficiency and reliability in IoT systems.

Moreover, DNS's security extensions make it a critical tool in ensuring the security of IoT systems, particularly given the sensitivity of data transmitted within these networks. Despite its potential, caution must be exercised when using DNS in IoT environments, as this chapter has also highlighted the risks IoT may pose to the DNS infrastructure.

Chapter 3

Mutual Authentication of IoT Backend Servers

Contents

Chapter Overview

This chapter introduces the use of DNS-based Authentication of Named Entities (DANE) for mutual authentication, demonstrated through the setup of a mutual authentication mechanism between two backend servers in a LoRaWAN network. The chapter begins with a detailed explanation of the Public Key Infrastructure (PKI) using X.509 digital certificates (PKIX). It explores the differences between symmetric and asymmetric cryptography before introducing X.509 digital certificates and the role of Certificate Authorities (CAs) in issuing them. The chapter also explains the TLS handshake process for both TLS 1.2 and TLS 1.3. Following this, the drawbacks and potential vulnerabilities of PKIX are discussed, leading to our implementation and demonstration of how DANE can reinforce or even replace CAs. Finally, the system model and performance evaluation are presented.

3.1 Introduction

Among the security threats that could endanger online communications is connecting to a malicious server masquerading as a legitimate one. This could lead to severe repercussions such as exposing credentials and sensitive personal data. To mitigate this risk, connecting to a server on the Internet today is preceded by a few steps before exchanging any data. Initially, the client may query the Domain Name System (DNS) to resolve the domain name of the server into its IP address. After obtaining the IP address, the client initiates contact –but not data exchange– with the server to verify its identity. This step confirms to the client that the server is truly the intended destination for the connection.

The DNS resolution is facilitated by the DNS infrastructure, and the identity verification is enabled by the Public Key Infrastructure (PKI). While the DNS infrastructure is a well-defined system of nameservers with a specific resolution protocol, the PKI is more multifaceted, encompassing hardware, software, protocols, certification issuance and verification processes, and cryptographic methods. The different elements of the PKI function together to reinforce communication security, guaranteeing secure access and data exchange between clients and servers. This is achieved by implementing the necessary methods to bind certified entities to digital certificates, and more specifically, linking these entities to public keys. In summary, the PKI enables a client to verify the authenticity of the certificate (or public key) presented to it by a server, and to subsequently use this public key to establish a secure communication channel.

The most widely used protocol to secure Internet communication is the Transport Layer Security (TLS) protocol. The main objectives of TLS are confidentiality, integrity, and authenticity of the data exchange between clients and servers, making it a critical component of the PKI. During a process called the *TLS handshake*, a client can verify the identity of a server, and both parties could agree on security parameters for subsequent communications. The most common form of identification servers present to clients during the *TLS handshake* is the X.509 digital certificate. When X.509 certificates are used in the context of the PKI, the system is referred to as PKIX (Public Key Infrastructure using X.509 certificates).

Trusting the digital certificate of a server inherently depends on trusting the issuer of that certificate. It is possible that servers issue and sign their own certificates, known as self-signed certificates, which are rarely trusted. The most common and trusted approach is having a trusted third party, called a Certificate Authority (CA), issue the certificates.

In the context of PKIX, communications are typically client-server based, where the PKIX enables clients to verify the identity of servers in order to establish a secure connection. In this common model for web communications, clients are rarely required to prove their identity. Otherwise, clients would need to possess the equivalent of a public key certificate (*e.g.*, X.509 digital certificate). If clients, like servers, were to share a digital certificate and have it verified by the servers, then the authentication would be mutual, with each party verifying the identity of the other. Mutual authentication is important for reinforced communication security, as servers in this case could avoid malicious devices and only accept connections from legitimate, verified clients.

This form of authentication would be useful in Internet of Things (IoT) environments. Such networks typically consist of end devices (*EDs*) that communicate with IP-enabled backend servers either directly or through gateways that relay data between radio and IP spaces. Current security practices in IoT include having secret keys pre-shared and stored on *EDs* and backend servers to establish secure communication sessions between the two, *e.g.*, LoRaWAN. Additionally, the PKIX is used for authenticating the communication between the IP-enabled backend servers. Hard coding keys on the device and servers, printing them on the device, or even sending them by email, for example, are all key-sharing methods that pose a security risk. For communication between backend servers, setting up secure communication sessions requires using the PKIX where one backend server acts as a client while the other acts as a server. This

form of authentication is one-sided as only the identity of the backend server acting as server is verified while the identity of the other backend server, the one acting as a client, is not. Besides its cost entailed by paying a trusted CA to issue a certificate, this form of authentication has some drawbacks. It relies on multiple root CAs for issuing certificates. The root CAs have self-signed certificates and can sign and provide certificates to intermediary CAs responsible for issuing certificates to domains and servers. These CAs are distributed in nature and do not have a central trust anchor, and each major vendor, Google and Apple, for example, has its list of CAs (root store) that it trusts[73]. Moreover, these CAs could be compromised and issue rogue certificates [99]. On the other hand, resorting to self-signed certificates is not the optimal solution, especially when multiple stakeholders are involved.

The goal of this chapter is to use DNS, its security extensions (DNSSEC), and the DNS-based Authentification of Named Entities (DANE) protocol to reinforce the PKIX for IoT backend servers. The focus is on reinforcing the security of communication between two LoRaWAN backend servers using DANE.

3.2 The PKI using X.509 certificates (PKIX)

In this section, we review the PKIX currently used for establishing secure communications.

3.2.1 Symmetric vs Asymmetric Encryption

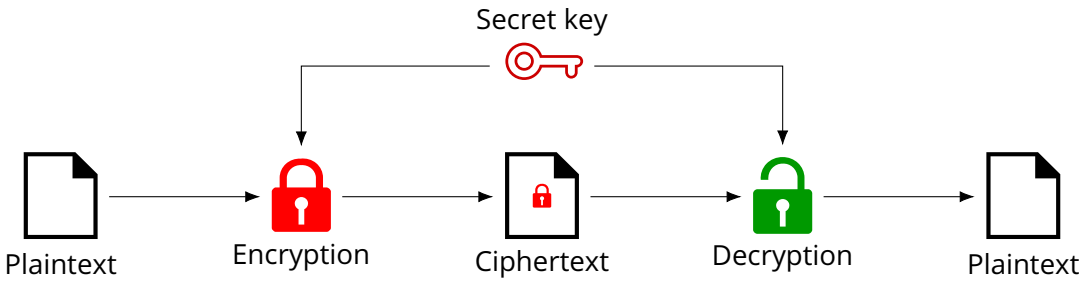
The goal of encryption is to transform data, making it unreadable to unauthorized parties. By running a piece of plaintext (unencrypted data) through an encryption algorithm, an equivalent unintelligible ciphertext is produced, which could be safely transported or stored. Reverting ciphertext back to the original plaintext is called decryption. There are two primary encryption techniques: symmetric key encryption and asymmetric key encryption.

Symmetric key encryption techniques use a single secret key for encryption and decryption. In secure communications between two parties, both parties must share the same secret key. See Figure 3.1a. Symmetric encryption techniques are fast and computationally efficient; however, sharing the secret key presents a challenge, as exposure of the key compromises the data. Asymmetric encryption techniques address this by removing the need of the communicating parties to have an identical pre-shared secret key.

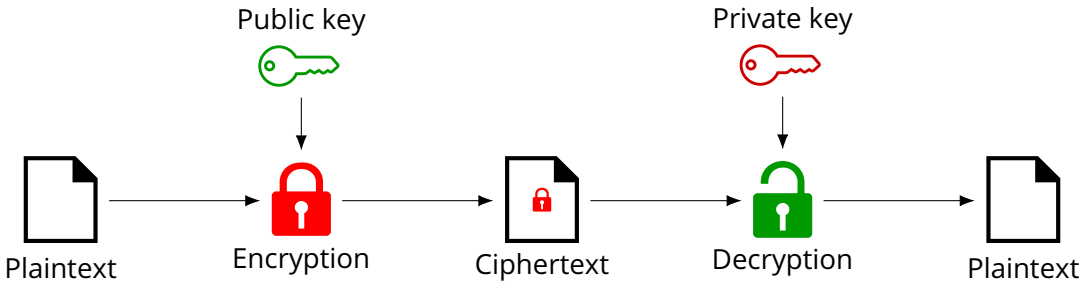
Asymmetric encryption relies on a pair of keys: a public key and a private key. Each one of the communicating parties has a key pair. The private key is secret and is never shared, while the public key can be shared safely in plaintext. Data encrypted with the public key can only be decrypted using the corresponding private key, and it is computationally infeasible to derive the private key from the public key. See Figure 3.1b. Compared to symmetric encryption, asymmetric techniques are slower and require larger key sizes.

Asymmetric encryption techniques are not limited to encrypting and decrypting data; they can also be used for key exchange. Two communicating parties, each with their own public and private key pair, can generate and agree on the same secret key without sharing anything other than their public keys. Therefore, given that symmetric encryption techniques are faster and more computationally-efficient, and asymmetric encryption techniques are slower but more secure, the less efficient asymmetric techniques are often used to safely exchange secret keys. Once the secret key is exchanged, the more efficient symmetric techniques are then employed for the actual data encryption.

One challenging aspect of asymmetric encryption is verifying that the public key truly belongs to the legitimate destination of the communication. For example, when connecting to server with whom the client might share sensitive data and credentials, the client should make sure that the public key of the



(a) Symmetric encryption.



(b) Asymmetric encryption.

Figure 3.1: Symmetric vs asymmetric encryption.

server is legitimate and truly belongs to the intended server. Otherwise, an adversary could impersonate the legitimate server and intercept the personal data of the client. One way of verifying the public keys of servers is through the use of digital certificates.

3.2.2 X.509 Digital Certificates

Digital certificates provide a solution for linking public keys to the entities that own them. A digital certificate acts like an electronic passport to bind a public key to the identity of its owner. Organizations, servers, and individuals can request digital certificates to prove the authenticity and ownership of their public key.

The most widely used standard for digital certificates is the X.509 standard, version 3 (v3) being the most recent [62]. The fields of the X.509 digital certificate are presented in Figure 3.2.

- **Version Number**: the X.509 version that applies to the certificate.
- **Serial Number**: a unique numerical identifier of the certificate assigned by the issuing authority.
- **Signature Algorithm ID**: specifies the public key and hashing algorithms used by the issuing authority to sign the certificate.
- **Issuer Name**: specifies the distinguished name of the issuing authority of the certificate.
- **Validity Period**: specifies the validity period of the certificate during which it can be trusted. It includes a *Not Before* and a *Not After* fields.
- **Subject Name**: specifies the subject name which is the name of the entity to which the certificate belongs *i.e.*, the entity which owns the private key corresponding to the public key included in the certificate.

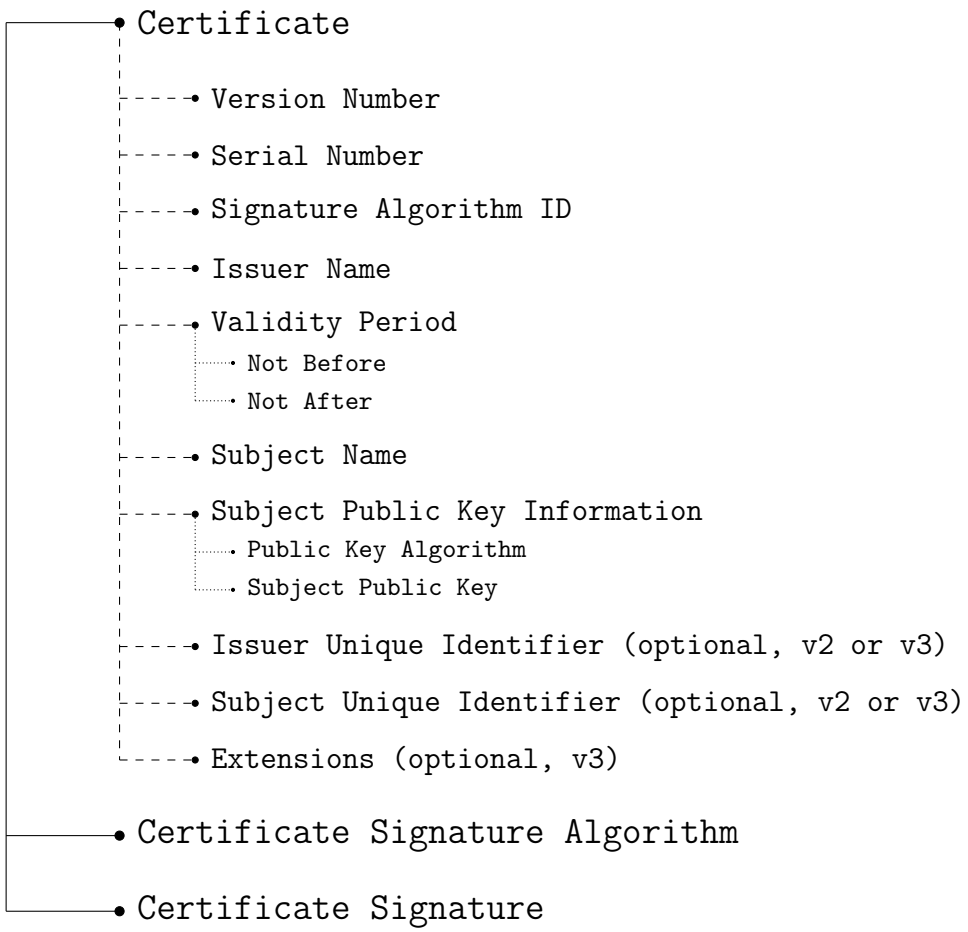


Figure 3.2: The structure of the X.509 Digital Certificate (v3).

- **Subject Public Key Information**: contains two important pieces of information: the *Subject Public Key* field which specifies the public key of the subject *i.e.*, to whom the certificate was issued, and the *Public Key Algorithm* field which specifies the algorithm that generated the public key and the hashing algorithm with which the public key is to be used.
- **Optional Fields**: some optional fields:
  - Issuer Unique Identifier (v2): specifies the unique name of the issuer.
  - Subject Unique Identifier (v2): specifies the unique subject name which is the name of the entity to which the certificate belongs.
  - Extensions (v3): introduced with v3. The *Extensions* field introduces a way to extend certificates to include additional information such as information about using the public key, information provided by the issuing authorities about the usage and interpretation of the certificate, and others.
- **Certificate Signature Algorithm**: specifies the public key and hashing algorithms used by the issuing authority to sign the certificate.
- **Certificate Signature**: contains the digital signature of the certificate.

In principle, the issuing authority could be the entity to which the certificate was issued, making it a self-signed certificate. However, self-signed certificates are not trusted, as the purpose of using certificates is to establish trust. A more secure approach is to rely on a trusted third party known as a Certificate Authority (CA).

3.2.3 Certificate Authorities

CAs are trusted organizations that are authorized to issue and sign digital certificates for end-entities such as individuals, servers, and organizations. An organization that wishes to become a CA must undergo strict security audits and must adhere to rigorous standards to acquire and maintain their trusted status.

In the context of PKIX, a chain of trust consisting of several layers of CAs is used to verify the legitimacy of digital certificates. The chain begins with root CAs which are at the top of the hierarchy. Root CAs use self-signed certificates and act as the trust anchor of the chain of trust. Operating systems and browsers are equipped with lists of root CA certificates in a *trust store* which allows them to validate certificates they receive. Root CAs rarely create individual leaf certificates but instead they create and sign intermediate CA certificates. Intermediate CAs are responsible for creating and signing leaf certificates for end-entities. Together, the root, intermediate, and leaf certificates form a chain of trust, where trusting the leaf certificate depends on the entire chain. When a client connects to a server, it receives the leaf certificate of the server and the intermediary certificate of the CA that issued it. After that, the client traces the chain of trust from the leaf certificate, through the intermediary CA, and finally to the root certificate stored in its *trust store*. Figure 3.3 illustrates the chain of trust, showing the relationships between the root CA, intermediate CA, and leaf certificates.

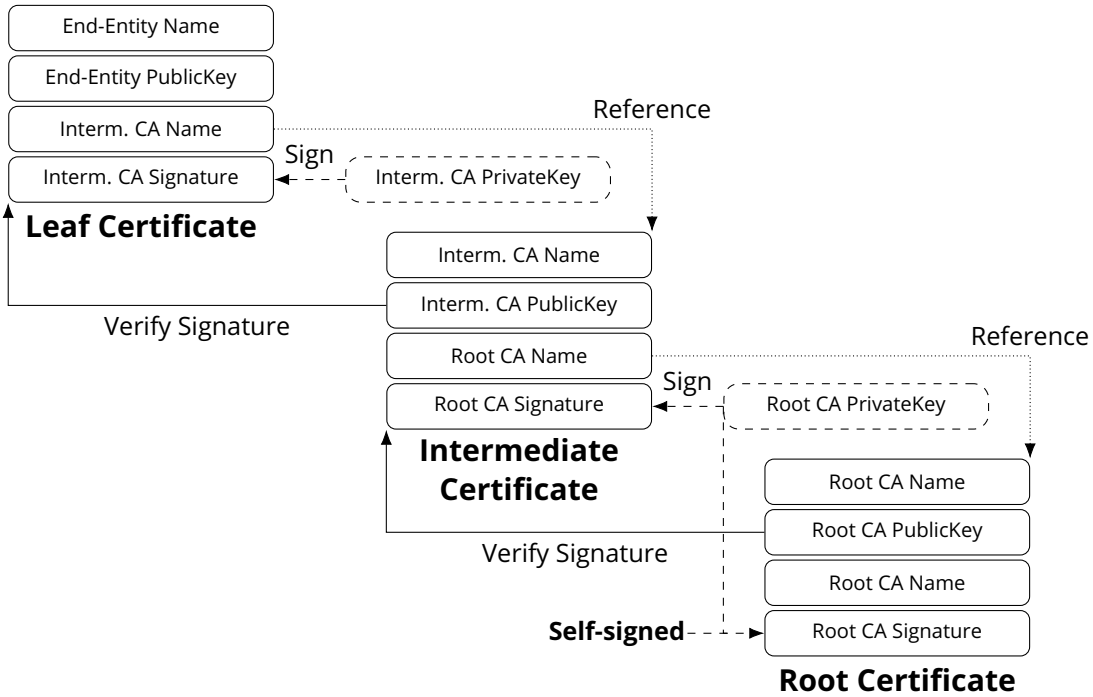


Figure 3.3: Digital certificates chain of trust.

The CA chain of trust plays a critical role in securing communications over the Internet. One protocol that leverages this chain of trust to ensure secure data exchange between clients and servers is the TLS protocol.

3.2.4 TLS Protocol

TLS is a protocol that makes use of the CA chain of trust to secure communications on the Internet by ensuring confidentiality, integrity, and authentication of the data exchanged between a client and a server. A major aspect of TLS involves the server sending its digital certificate such as the X.509 digital certificate to the client who verifies it using the CA chain of trust. Following the certificate verification, the client and the server can communicate securely. Authentication and agreement on the session security parameters are done during the TLS handshake, which occurs before connection establishment. Figure 3.4 illustrates the TLS handshake in TLS 1.2 and 1.3.

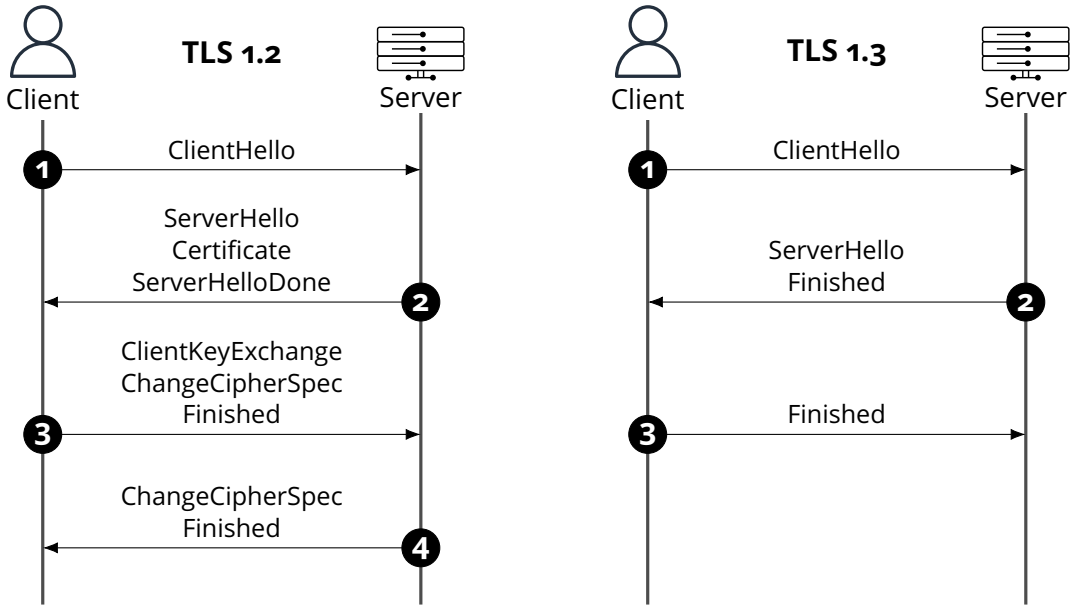


Figure 3.4: The TLS 1.2 and 1.3 handshake.

- **TLS 1.2**

  1. The client sends a *ClientHello*, specifying the supported TLS version, cipher suites, and *ClientRandom*, for key generation.
  2. The server replies with a *ServerHello*, indicating the chosen TLS version and cipher suite. It also sends a *ServerRandom* and its digital certificate with an optional certificate chain. Finally it sends a *ServerHelloDone*.
  3. After validating the server's certificate, the client sends a random *Pre-Master Secret* encrypted with the server's public key. Both generate the same *Master Key* using the *Pre-Master Secret*, *ClientRandom*, and *ServerRandom*, which they then use to generate the session keys for secure communication. The client then sends *ChangeCipherSpec*, signaling the switch to encryption, followed by a *Finished* message. If Diffie-Hellman (DH) or Elliptic Curve Diffie-Hellman (ECDH) is used, the client and server will have exchanged their DH parameters by this point.
  4. The server responds with *ChangeCipherSpec* and a *Finished* message, completing the handshake.

- **TLS 1.3**

  1. The client sends a *ClientHello*, specifying the supported TLS version, cipher suites, and a string of random bytes known as *ClientRandom* for key generation. The client also guesses the key exchange method (*e.g.*, DH or ECDH, RSA is not supported for key exchange) and sends the corresponding key share parameters.
  2. The server generates a *Master Secret* and replies with a *ServerHello*, indicating the selected TLS version, cipher suite, and its own random bytes (*ServerRandom*). The server also sends its key exchange method (matching the guessed key exchange method) and its digital certificate. The server also sends a *Finished* message.
  3. The client verifies the server certificate, generates the same *Master Secret*, and sends a *Finished* message, completing the handshake.

3.3 Reinforcing the PKIX Security with DANE

In the context of PKIX, digital certificates issued by one CA are not guaranteed to have universal trust. A new CA entering the market that wants its certificates to be universally trusted must be added to several *Root Stores* as different vendors have different *Root Stores*. For example, the Mozilla Firefox *Root Store* includes 152 CAs[153] while Apple's macOS root store contains 159 CA [46]. A root CA trusted by one vendor but not the other could publish millions of certificates that are only partially trusted on the Internet. This means that, in theory, a certificate, even though legitimate, could be trusted by some machines and not by others, depending on the manufacturer and whether or not the CA that issued it exists in its *Root Store*.

Moreover, digital certificates are not required to be verified by the CA that issued them. Consequently, any CA in the *Root Store* could validate any certificate which increases the attack surface for potential malicious activity. This expanded attack surface means that vulnerabilities in any root or intermediary CA can be exploited, allowing an attacker to issue fraudulent certificates. The root and intermediary CAs are not immune to security breaches [91, 76]. They could, in theory, issue certificates for any domain name like the root authority Diginotar did when it issued a fraudulent Google certificate in 2011 [99].

Finally, acquiring a digital certificate is not cheap and their cost accumulates when several certificates are required. For example, Figure 3.5 is a snippet from *globalsign.com* [97] showing the costs associated with buying digital certificates. These costs, while affordable for large companies, can be burdensome for smaller organizations or when multiple certificates are required. Additionally, the cost varies significantly depending on the type of certificate (*e.g.*, single domain, wildcard, or extended validation certificates) and the CA issuing it.

DANE (DNS-Based Authentication of Named Entities) [111, 202, 89] is a DNS protocol destined to address some of the shortcomings of the PKIX. DANE mainly aims to complement the role of CAs by giving control to zone owners regarding verifying their digital certificates. The main feature of DANE is the TLSA Resource Record (RR) that owners can add to their zone. TLSA RRs form an association between digital certificates or raw public keys and the domain names to which those certificates or keys were given.

The functionality of DANE within the context of PKI comes to light when we inspect the fields of the TLSA RR, especially the *Certificate Usage field*, revealing how it enhances the security of TLS connections [111]:

- **Certificate Usage field**: 1-octet value that represents the association that will be used to match the certificate provided in the TLS handshake. It could have the following values:
  - **PKIX-TA(0)**: Also referred to as *CA Constraint*. This *Certificate Usage* indicates that the *Certificate Association Data* contains the certificate or the public key of a CA or a *Trust Anchor* that can issue

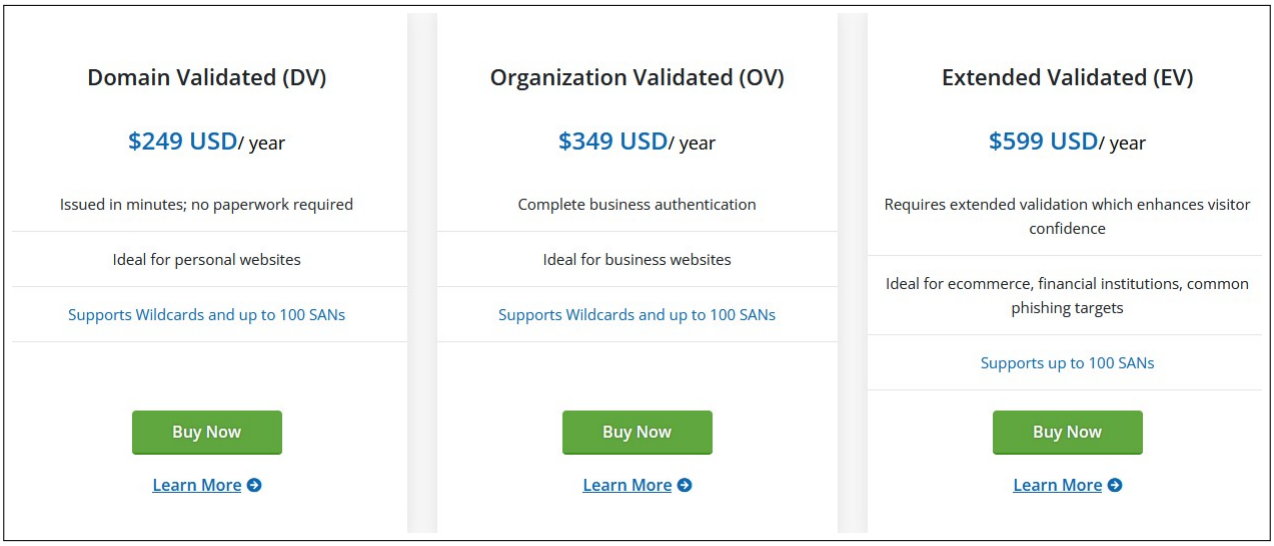


Figure 3.5: Cost of digital certificates. *https://shop.globalsign.com/en/ssl* [97]

certificates for a domain name associated with a server or service and that a CA certificate that matches the TLSA record must be included as part of a valid certification path, *i.e.*, the client must find this CA certificate in its *Trust Store* to validate the certificate presented by the server during the TLS handshake. *PKIX-TA(0)* is the strictest among the possible values of the *Certificate Usage field*, as it limits the CAs that can issue certificates for the domain name. This prevents rogue CAs from issuing fraudulent certificates.

- **PKIX-EE(1)**: Also referred to as *Service Certificate Constraint*. This *Certificate Usage* indicates that the *Certificate Association Data* contains the certificate or the public key of the end-entity that must be matched with the certificate provided by the server during the TLS handshake. The certificate must also be validated using a CA chain of trust. This *Certificate Usage* precises the certificate that the domain name of a service or a server must have. This prevents rogue CAs or even any CA from issuing a different certificate for the domain name, even if the certificate is legitimate. The certificate presented by the server during the TLS handshake must match the certificate in the TLSA RR and it must be validated by a CA chain of trust.
- **DANE-TA(2)**: Also referred to as *Trust Anchor Assertion*. This *Certificate Usage* indicates that the *Certificate Association Data* contains the certificate or the public key of a *Trust Anchor* that must be used to validate the end-entity certificate provided by the server during the TLS handshake. This can be helpful if the end-entity certificate is self-signed or if the owner of that certificate runs their own CA. This *Certificate Usage* allows the certificate owners to verify their certificate without necessarily relying on CA certificate, making it the first *Certificate Usage* that does not require validating the certificate received during the TLS handshake through a typical PKI hierarchy of globally recognized CAs as a chain of trust. It does, however, require the validation of the certificate using the *Trust Anchor* found in the *Certificate Association Data* of the TLSA RR.
- **DANE-EE(3)**: Also referred to as *Domain-Issued Certificate*. This *Certificate Usage* indicates that the *Certificate Association Data* contains a certificate that must match the certificate received during the TLS handshake. Similar to *DANE-TA(2)*, this *Certificate Usage* also does not require validating the certificate received during the TLS handshake through a typical PKI hierarchy of globally recognized CAs as a chain of trust, which differentiates it from PKIX-EE(1).

Examining the four possible values of the *Certificate Usage* field reveals two distinct facets of interaction between DANE and the CAs. When the *Certificate Usage field* is set to *PKIX-TA(0)* or *PKIX-EE(1)*, DANE coexists with the CAs, complementing their role by enhancing session security between clients and servers. Conversely, when the *Certificate Usage field* is set to *DANE-TA(2)* or *DANE-EE(3)*, DANE operates independently of the CAs, effectively assuming their role. This disparity between the two use cases provides DANE with the flexibility to be utilized in various PKI architectures.

A client connecting to a TLS server receives the server's certificate and performs a DNS query to fetch the TLSA RR of the server. To verify the certificate, the client compares the *Certificate Association Data* from the TLSA RR with the relevant part of the server's certificate. The specific part of the certificate that is compared, and how the comparison is performed, depends on the parameters specified in the TLSA RR. These parameters include the *Certificate Usage*, which defines the relationship between the certificate and the CA or DNS; the *Selector*, which determines whether the full certificate or just the public key is matched; and the *Matching Type*, which indicates whether the data should be matched exactly or using a hash (*e.g.*, SHA-256 or SHA-512). The integrity of the TLSA RR is guaranteed by DNSSEC [174, 176, 175].

3.4 DNS Complementing the PKIX in IoT Environments

3.4.1 Mutual Authentication via DANE - The DANCE WG

Up to this point, the discussions on DANE have focused on a client connecting to a server, retrieving its certificate, and fetching the corresponding TLSA RR for certificate verification. The scenario of clients possessing digital certificates or publishing TLSA RRs has not been explored, as DANE was designed for TLS server authentication, not client-side authentication. This is evident in the original specifications of DANE [111, 89], which primarily address the authentication of TLS servers rather than clients. This was the main motivation behind creating the DANE Authentication for Network Clients Everywhere (DANCE) IETF working group (WG) [4].

The DANCE WG aims to extend DANE to enable TLS client authentication. Like TLS servers, a TLS client would have a raw public key or a certificate and would publish a TLSA RR in DNS corresponding to that public key or certificate. This allows mutual authentication via DANE between TLS clients and servers where each one could verify the other's certificate or raw public key. The DANCE WG issued two Internet drafts, *TLS Client Authentication via DANE TLSA records* [114] and *TLS Extension for DANE Client Identity* [115].

TLS Client Authentication via DANE TLSA records[114]: The first draft describes how to publish TLSA RRs for TLS clients. Client identities are assumed to be represented by DNS domain names. Two formats for client identities are possible:

- **Service specific client identity:** The client identity has the following format:

  `[_service].[client-domain-name]`

  The first label (`_service`) is the application service name while the remaining ones are the domain name of the client.

- **DevId: IoT Device Identity:** The client identity has the following format:

  `[devicename]._device.[org-domain-name]`

The first label (`[devicename]`) refers to the name of the IoT device. The second label (`_device`) allows delegating a group of devices to a DNS subzone for easier management. The remaining labels are the domain name of an organization.

Similar to verifying servers, a TLS client that wishes to be authenticated via DANE must have a raw public key or a certificate binding their identity to a public key. The public key or certificate has a corresponding TLSA RR allowing verifying it via DANE. Clients should always signal that they have DANE identities to spare the server from sending unnecessary DNS queries. This could be done either using the *DANE Client Identity TLS extension* presented in the second draft [115] or, if the TLS extension is not used, the client certificate must have the client's DNS name specified in the Subject Alternative Name extension's DNSName type.

TLS Extension for DANE Client Identity [115]: The second draft defines a TLS extension that allows clients to express their support for DANE and their intent to be verified and allows them to share their DANE *ClientID* with the server. The extension could be empty to indicate to the server that the client wishes to be verified via DANE and that its identity could be extracted from its certificate. Otherwise, if the client is using a raw public key, then the extension must be used and must have the full *ClientID* in the form of a domain name that the server could use to fetch the client's TLSA RR. In TLS 1.2, the extension is sent in the *ClientHello* message. In TLS 1.3, the extension is sent in the *Client Certificate* message.

3.5 System Model

In this section, we implement the DANCE WG drafts [114, 115] to enable mutual authentication between two LoRaWAN backend servers. These servers frequently communicate and would benefit from mutual authentication via DANE, which ensures that each server can verify the identity of the other.

We modified the Golang TLS library to add support for the DANE *ClientID* extension in TLS 1.3 (both client and server sides). Additionally, we adapted the implementation to support TLS 1.2, where the extension is included in the *ClientHello* message to allow the *ClientID* to be transmitted. The source code for the modified TLS library, DANE library, DANCE library, and the adapted Chirpstack software is available at *https://gitlab.rd.nic.fr/dance*.

Our setup is presented in Fig. 3.6.

For a LoRaWAN *End-Device (ED)* to join the network, it first sends a *Join-request* to the *gateway*, which forwards it to the *Network Server (NS)*. The NS then connects to the *Join Server (JS)* to verify the request. After verification, a *Join-accept* is sent back to the gateway, which then delivers it to the *ED*. Before the *NS* contacts the *JS*, the two servers perform mutual authentication via a TLS handshake that includes a certificate exchange. In our implementation, each server verifies the other's certificate using DANE. Each server has already published its TLSA RR in its DNS zone and retrieves the TLSA RR of the other to verify the received certificate. We test several scenarios to evaluate this setup.

Our setup is based in France, and we vary the location of the DNS resolver to study its effect on performance. We begin with a local resolver on both the *NS* and *JS*, and examine the impact of enabling and disabling caching. Caching allows the recursive resolver to store a RR for a specified duration, known as the Time-to-Live (TTL), enabling it to respond directly to requests without querying any nameservers. Additionally, we study the use of remote resolvers, including Google's public DNS resolver `'8.8.8.8'` and a resolver we deployed in Singapore to maximize the distance between the DNS clients and the resolver. These scenarios are compared to the baseline case where DANE is not used for certificate verification, and a single Certificate Authority (CA) is used instead. The CA, owned by our company, has a self-signed certificate that allows it to issue certificates for both servers, acting as a trusted CA for both.

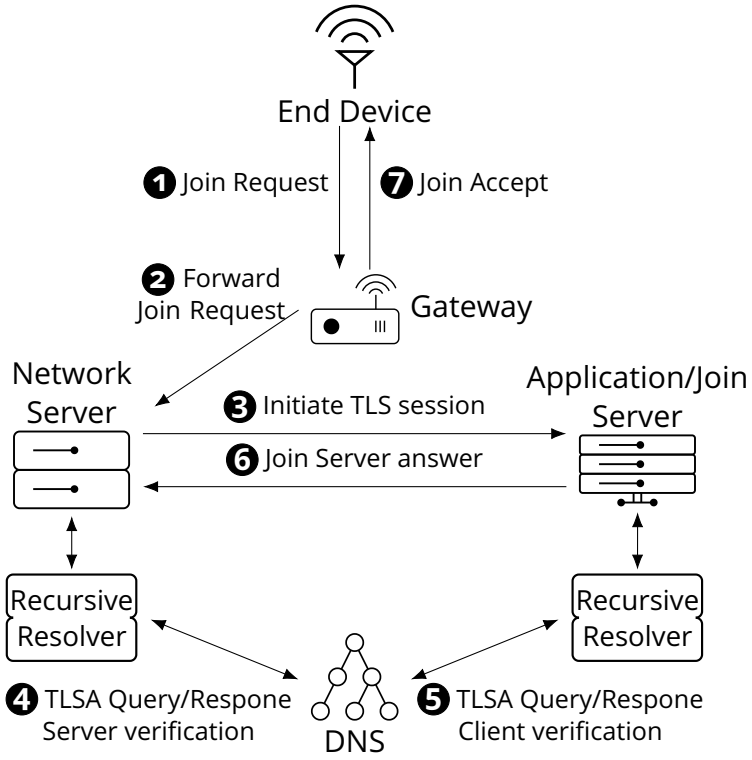


Figure 3.6: System model.

3.6 Evaluation

In this section, we present the results of the measurements conducted across various scenarios: the baseline case, where mutual authentication is performed without DANE and our company acts as the CA using a self-signed certificate, and several scenarios involving mutual authentication via DANE, with varying configurations and locations of the recursive resolver. We start by using a local resolver on each server and examine the effect of caching. Next, we use Google's public DNS resolver `'8.8.8.8'` and a resolver located in Singapore as remote resolvers. The primary metric we consider is the time elapsed from when the gateway receives the *Join-request* from the *ED* to when it receives the *Join-accept* from the *NS*. We analyze the impact of using DANE for mutual authentication on the duration of the *Join-request* process. In each scenario, the *ED* sent a *Join-request* every 10 seconds over a period of 7 hours. However, due to duty cycle restrictions (1% in our region), not all join requests were successful.

Fig. 3.7 is the histogram of the duration of successful join requests in the baseline case where DANE is not used. The average *join request duration* is around 0.412 seconds.

3.6.1 Using a Local DNS Resolver

In the first scenario, a resolver is installed separately on both the *NS* and the *JS*. Each server's resolver retrieves the TLSA RR of the other server, *i.e.*, the server it needs to authenticate. We conducted measurements both with and without caching enabled. Fig.3.8a presents the histogram of the duration of successful join requests when a local DNS resolver without caching is used, resulting in increased delays, with the *average Join-request duration* reaching approximately 1.037 seconds. After enabling caching on both servers, with a TLSA RR TTL of 20 minutes, the average delay dropped to around 0.442 seconds, as

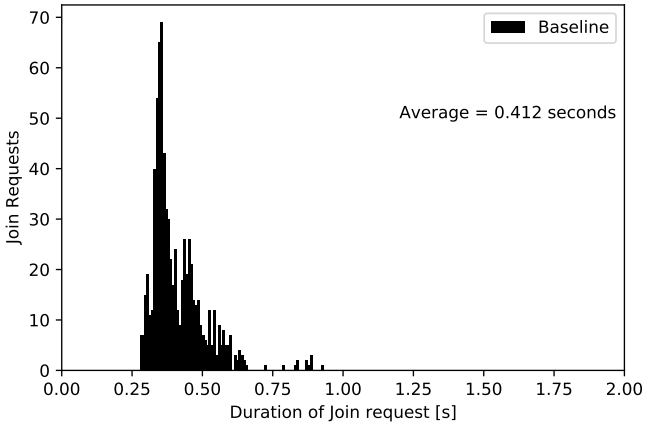


Figure 3.7: Duration of join requests [s] (Baseline).

shown in Fig.3.8b.

Figures 3.8a and 3.8b illustrate that using DANE for mutual authentication without caching introduces significant delay, with the *average Join-request duration* more than doubling, increasing from 0.412 seconds in the baseline case to 1.037 seconds. However, enabling caching at the recursive resolver significantly reduced latency, bringing the *average Join-request duration* down to 0.442 seconds.

3.6.2 Using Remote DNS Resolvers

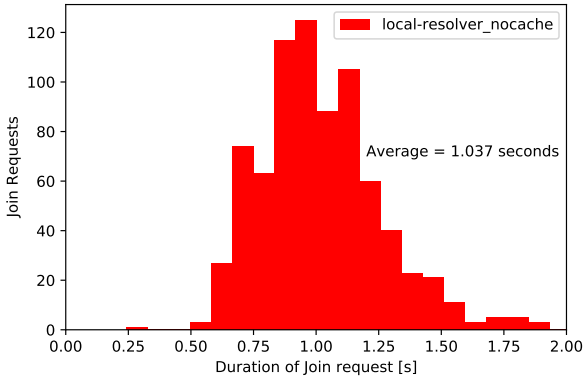
Caching proved to be effective in reducing the latency introduced by DANE, even for remote resolvers. Fig.3.8c presents the histogram of the duration of successful join requests when using Google's public DNS resolver `'8.8.8.8'`, with the *average Join-request duration* around 0.489 seconds. Fig.3.8d shows the duration of successful join requests when the resolver is located in Singapore, where the *average Join-request duration* is approximately 0.887 seconds.

Figures 3.8c and 3.8d highlight the importance of caching in reducing the latency introduced by DANE. The *average Join-request duration* in both cases (0.489 seconds for Google's public DNS and 0.887 seconds for the Singapore DNS resolver) is shorter than the *average Join-request duration* when using a local resolver without caching.

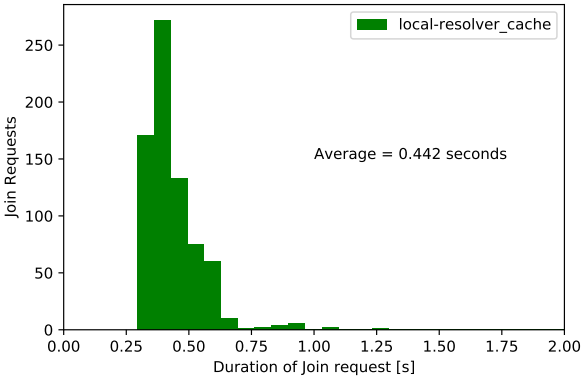
3.7 Conclusion

In this chapter, we implemented the DANCE WG drafts to use DANE to perform mutual authentication between a LoRaWAN *NS* and *JS*. The implementation showed that DANE could be used for authenticating certificates by adding appropriate TLSA RRs in the DNS zone of the domain name whose certificate is to be verified. The performance evaluation showed that the DNS resolution adds latency, which could be overcome by enabling caching at the resolver level.

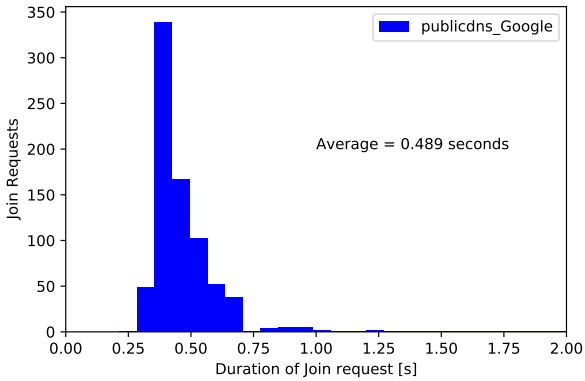
Mutual authentication between backend servers in IoT environments is essential to enhance the security of networks, allowing servers to verify the identity of other servers to which they are connecting. By integrating DANE, the authentication model reduces the reliance on traditional PKIX mechanisms and commercial CAs, mitigating the security risks associated with rogue certificates and compromised CAs,



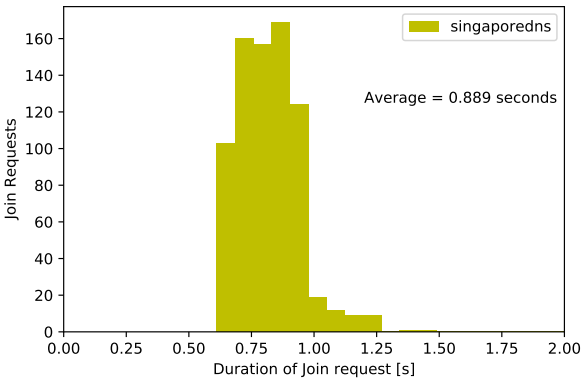
(a) Local resolver without caching.



(b) Local resolver with caching.



(c) Remote DNS resolver - Google.



(d) Remote DNS resolver - Singapore.

Figure 3.8: Duration of Join Requests in different scenarios.

which are common vulnerabilities in traditional PKIX setups, while also reducing the costs associated with using commercial CAs.

Chapter 4

LoRaDANCE: Using The DNS for Mutual Authentication Without Pre-shared Keys

Contents

Chapter Overview

This chapter proposes LoRaDANCE, a security mechanism that enables a LoRaWAN *End-Device (ED)* to join a LoRaWAN network while guaranteeing two two key aspects: first, the device does not need to pre-share any secret keys, unlike in standard LoRaWAN, for example, LoRaWAN 1.0.x, which requires pre-sharing the secret *AppKey*. Second, the *ED* and the *Join Server (JS)* mutually authenticate using the DNS-based Authentication of Named Entities (DANE) protocol.

4.1 Introduction

Mutual authentication that does not rely on static pre-shared secret keys is paramount to enhancing the security of communications. This form of authentication would be particularly beneficial for Internet of Things (IoT) technologies, allowing IoT devices to mutually authenticate with their backend servers. These technologies would also benefit from eliminating the need to pre-share secret keys. For example, in LoRaWAN, when a LoRaWAN *End-Device (ED)* joins a LoRaWAN network, a secret key, namely the *AppKey*, must

be provisioned at the *ED* and the *Join Server* (*JS*) to be used during the join process to authenticate the *Join-request*, encrypt the *Join-accept*, and derive the session keys [142]. A *Join-request* could not be completed without having the *AppKey* shared in advance between the *ED* and the *JS*. Pre-sharing the *AppKey*, a secret key, is risky as inadequate provisioning jeopardizes the security of the join process and any future messages between the *ED* and its backend servers.

In this chapter, we propose and implement LoRaDANCE, a mutual authentication mechanism between LoRaWAN *EDs* and their backend servers, which does not require pre-shared secret keys. LoRaDANCE uses asymmetric cryptography to allow a LoRaWAN *ED* to join a network without having to pre-share any secret keys with its backend servers. Instead, the *ED* and the *JS* each have a public/private key pair, and they use the Elliptic-curve Diffie–Hellman (ECDH) key exchange mechanism to generate the same secret, which is then used during the join process. In addition, the DNS-based Authentication of Named Entities (DANE) protocol is used to enable mutual authentication between the *ED* and the *JS* where each one verifies the identity of the other.

In Chapter 3, we used DANE to provide mutual authentication between the LoRaWAN backend servers. The work in [148] suggests a new rejoin mechanism in LoRaWAN that uses ECDH to refresh the secret root keys periodically. The work in [154] suggests PK-OTAA, which uses ECDH for key exchange in LoRaWAN to dynamically generate secret root keys without relying on pre-shared keys. Both works, however, do not explicitly illustrate a method of authenticating the exchanged public keys. LoRaDANCE evades the use of pre-shared keys and allows the authentication of public keys.

4.2 Prerequisite

4.2.1 DNSSEC, DANE, & DANCE WG

Enabling DNSSEC [174, 176, 175] allows DNS resolvers to verify that the DNS responses they receive from DNS nameservers are authentic. DNSSEC-enabled zones contain some DNSSEC-specific Resource Record (RR) types, which include: *DNSSKEY*, which contains a public key, *RRSIG*, which contains a digital signature, and *DS*, which contains a hashed value of a *DNSKEY* RR of a child zone. DNSSEC uses an authentication chain that starts at a trust anchor (usually the root nameserver) whose public key is known by all DNSSEC-enabled resolvers. It descends along the DNS namespace, with every level vouching for the level below it. DANE [111, 89] uses DNSSEC to bind X.509 digital certificates or raw public keys to domain names. The main feature of DANE is the TLSA RR. See Figure 4.1.

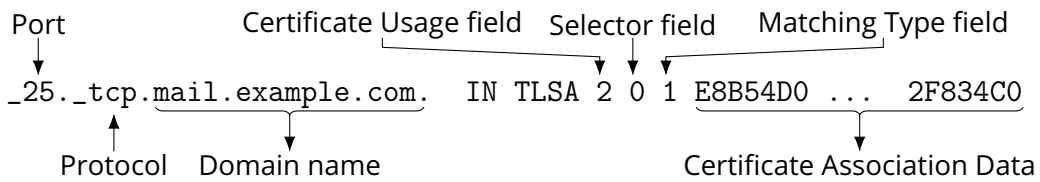


Figure 4.1: Example of a TLSA resource record.

When using raw public keys, the certificate usage DANE-EE(3) with selector SPKI(1) would be used [89]. See Chapter 1.

In the context of DANE, only TLS servers are thought to have an X.509 digital certificate or a raw public key to be verified. The DANE Authentication for Network Clients Everywhere (DANCE) IETF working group (WG) aims to extend the use of DANE to authenticate not exclusively TLS servers but also TLS clients. The DANCE WG issued several drafts, most notably [114] that demonstrates using DANE to authenticate TLS

clients and [115] that defines a TLS extension that allows clients to indicate that they have a TLSA RR that could be used to validate their certificate or public key.

4.2.2 Key Elements of LoRaWAN OTAA

The LoRaWAN backend servers are:

- **The Network Server (NS)**: The *NS* verifies if devices are allowed in the network during the join procedure and manages device identification and data rate regulation afterward.
- **The Join Server (JS)**: The *JS* is responsible for the authorization of *EDs* to join the network and the generation of session keys.
- **The Application Server (AS)**: The *AS* receives and processes data sent by the *EDs*. It has the application logic that end-users can interact with.

An *ED* joins a LoRaWAN network using Activation by Personalization (ABP) or Over-The-Air Activation (OTAA). OTAA is the most secure method [142] and is the method we consider in this work. Before the join procedure begins, a secret key and other identifiers should be provisioned on the *ED* and the backend servers. The provisioned items are:

- **DevEUI**: a 64-bit unique Extended Unique Identifier in the IEEE EUI64 address space for each *ED* that globally identifies the device.
- **JoinEUI**: a 64-bit unique Extended Unique Identifier in the IEEE EUI64 address space that globally identifies a *JS*.
- **AppKey**: an AES-128 root (secret) key specific to each *ED* and is used by the *ED* and the *JS* to derive the session keys.

The provisioning is done as follows:

- **Provisioned on the *ED***: *DevEUI*, *JoinEUI*, and *AppKey*.
- **Provisioned on the *NS***: *DevEUI*.
- **Provisoned on the *JS***: *DevEUI* and *AppKey*.

The OTAA procedure requires having the *AppKey*, a secret key, provisioned on the *ED* and the *JS*. The *AppKey* is involved in nearly all steps of the join process. It also serves as the basis for generating session keys, namely *NwkSKey* and *AppSKey*, which are used later to authenticate and secure the communications between the *ED* and its backend servers, respectively the *NS* and *JS*. Given the absence of a completely risk-free method for pre-sharing the *AppKey* between the *ED* and *JS*, this remains a weak security link.

In our work, we remove the need to pre-share the *AppKey* by having it securely generated at the *ED* and the *JS*. We also have the *ED* and the *JS* verify each other's identities by mutually authenticating using DANE.

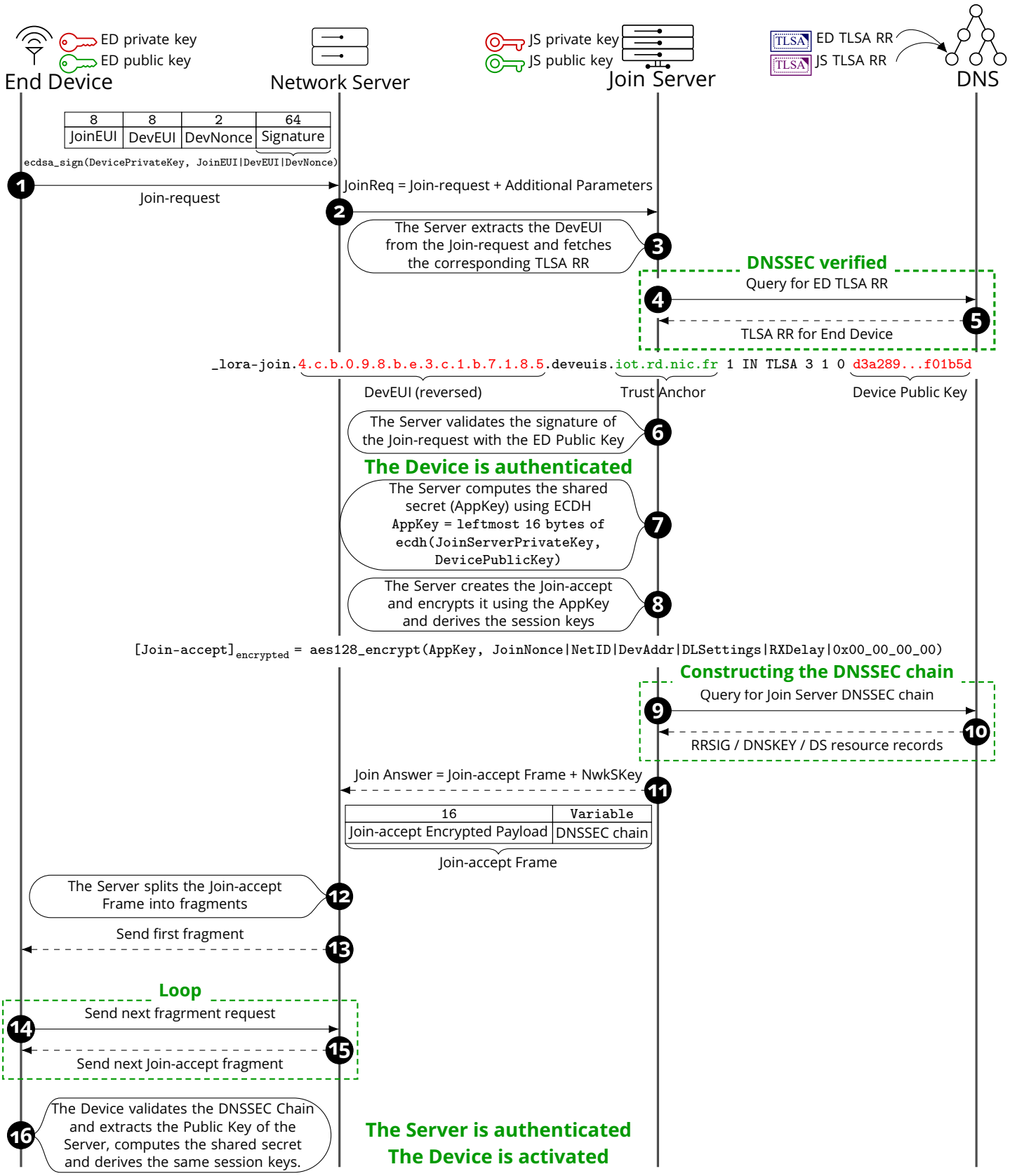


Figure 4.2: Illustration of LoRaDANCE.

4.3 LoRaDANCE

In this section, we present LoRaDANCE, a mechanism that allows a LoRaWAN *ED* to join a LoRaWAN network without the need to have a pre-shared *AppKey* or any other secret identifier between the *ED* and the backend servers. Our method allows the *ED* to be activated and allows both the *ED* and the *JS* to generate the session keys without the need to pre-share secret keys. LoRaDANCE also includes having the *ED* and the *JS* perform a DANCE-inspired mutual authentication using DANE.

In LoRaDANCE, we use DANE to allow mutual authentication between the *ED* and the *JS*. However, we do not use TLS or digital certificates. The *ED* and the *JS* each have a public/private key pair. Each one publishes a TLSA RR containing their raw public key. To mutually authenticate, each should receive the TLSA RR corresponding to the other's public key. Given that the *ED* does not have Internet access, the *JS* itself sends its TLSA RR along with the DNSSEC trust chain that allows the *ED* to verify it.

The *JS TLSA RR* is published in the `joineuis.iot.rd.nic.fr` zone, which acts as a placeholder for LoRa Alliance's `joineuis.lorawan.net`. Meanwhile, the *ED TLSA RR* is published in the `deveuis.iot.rd.nic.fr` zone. The hierarchical structure of *DevEUIs* following the EUI64 standard allows for delegating parts of the `deveuis` zone to manufacturers. The TLSA options must be DANE-EE(3) with selector SPKI(1) (see Chapter 1) and matching type Full(0), as in our case, the TLSA RRs contain the raw public keys.

In addition, the provisioning is done as follows:

- **Provisioned on the *ED***: *DevEUI*, *JoinEUI* and *ED* private key.
- **Provisioned on the *NS***: *DevEUI*.
- **Provisoned on the *JS***: *DevEUI* and *JS* private key.

LoRaDANCE is presented in Figure 4.2 and is as follows:

1 The *ED* constructs and sends a *Join-request* which includes the *JoinEUI*, *DevEUI*, and *DevNonce*. Instead of adding a MIC, the *ED* signs the *Join-request* using the Elliptic Curve Digital Signature Algorithm (ECDSA) and adds the digital signature to the *Join-request*. The *Join-request* is received by the *gateway*, which forwards it to the *NS*.

2 The *NS* receives the *Join-request* and verifies if it is the home *NS* of the *ED* using the *DevEUI*. If so, the *NS* queries DNS to get the IP address of the *JS* associated with the *ED* using the *JoinEUI*. The *NS* then forwards the *JoinReq* comprised of the *PHYPayload* of the *Join-request* and additional parameters to the *JS*.

3 The *JS* receives the *JoinReq* and extracts the *DevEUI*.

4 The *JS* sends a DNS query to obtain the *ED TLSA RR* based on the *DevEUI* extracted from the *Join-request* in **Step 3**.

5 The *JS* receives the DNSSEC-verified *ED TLSA RR*. The Certificate Association Data of the received RR contains the public key of the *ED*.

6 The *JS* verifies the signature received in the *Join-request* using the public key of the *ED*. After successful validation, **the authentication of the *ED* is complete**.

7 The *JS* derives the *AppKey*, taking the leftmost 16 bytes of the result from the ECDH operation between the private key of the *JS* and the public key of the *ED*.

8 The *JS* constructs a *Join-accept* which it encrypts using the *AppKey* and derives the session keys *NwkSKey* and *AppSKey*. We add 4 bytes of padding (`'0x00_00_00_00'`) to the 12-byte data to make it 16 bytes long, ensuring it meets the input size requirement for AES-128 encryption.

9 & 10 The *JS* sends multiple queries to construct the DNSSEC chain that verifies –its own– *JS TLSA RR*. Choosing an intermediate DNSSEC trust anchor between the zone to be verified and the root nameserver lets us select the DNSSEC algorithm, such as ECDSAP256SHA256 (Algorithm 13). Otherwise, we would be limited to RSA-based algorithms used in the root zone and common TLDs, which have larger key and signature sizes. In our case, we use `iot.rd.nic.fr` as the trust anchor. The *JS* constructs the DNSSEC chain as follows:

- The *JS* queries DNS and retrieves its own TLSA RR and its associated RRSIG RR.
- The *JS* then queries DNS and retrieves the Start of Authority (SoA) RR to determine the authoritative zone, `joineuis.iot.rd.nic.fr` in our case.
- Afterwards, the *JS* queries DNS and retrieves the DNSKEY RRs of the authoritative zone and their associated RRSIG RR.
- The *JS* then queries DNS and retrieves the Delegation Signer (DS) RR of the authoritative zone and its associated RRSIG RR.
- The last three steps are repeated until the trust anchor is reached.
- Finally, the *JS* queries DNS and retrieves the DNSKEY RRs of the trust anchor, `iot.rd.nic.fr` in our case.

11 The *JS* sends the *Join Answer* composed of the *Join-accept Frame* = *Join-accept* encrypted payload + the DNSSEC chain encoded in Concise Binary Object Representation (CBOR) [140] and the *NwkSKey* to the *NS*.

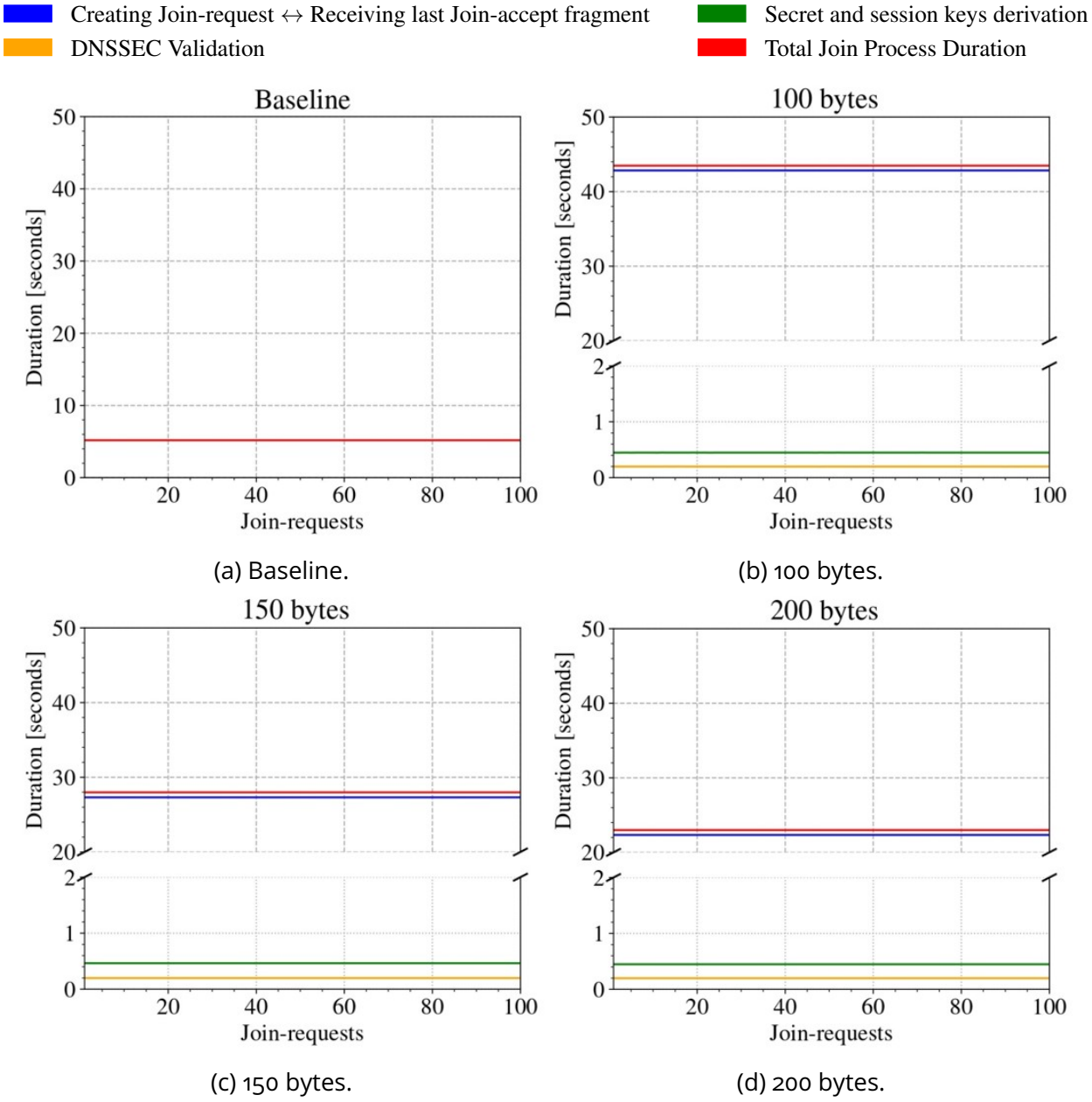
12 The *NS* splits the *Join-accept Frame* into fragments of predetermined size and indexes the fragments. When deployed, however, the fragment size would be variable and must adapt according to the data rate in the network.

13 The *NS* sends the first fragment to the *ED*.

14 After receiving the fragment, the *ED* sends a *Next Fragment Request*, which triggers the sending of the next fragment by the *NS*.

15 **Step 14** repeats until the *JS* signals that the current fragment is the last fragment.

16 The *ED* reassembles the *Join-accept Frame*, verifies that the name associated with the *JS TLSA RR* matches the *JoinEUI* from the *Join-request*, validates the DNSSEC chain, and then extracts the Public Key of the server. After that, **the server authentication is complete, and the *ED* is activated**.



(a) Baseline.

(b) 100 bytes.

(c) 150 bytes.

(d) 200 bytes.

Figure 4.3: Join process duration of baseline LoRaWAN 1.0.x and LoRaDANCE with fragment size = 100, 150, and 200 bytes.

4.4 Evaluation

In this section, we evaluate the performance of LoRaDANCE and compare it to the standard LoRaWAN 1.0.x, acting as our baseline, focusing on two aspects: the duration of the join process and the power consumption during the join process. The *ED* consists of a NUCLEO-L476RG development board equipped with a SX1276MB1MAS LoRa shield. We utilize RIOT OS [57] on the *ED* and Chirpstack for the backend servers.

4.4.1 Join Process Duration

We first compare the duration of the Join process, starting when the *ED* sends the *Join-request* and ending when the *Join-accept* is received and processed, *i.e.*, when the *ED* is activated. We test several scenarios where the device sends 100 *Join-requests* using LoRaDANCE, varying the fragment sizes between 100, 150, and 200 bytes. These measurements are compared against the baseline case. The results are presented in Figure 4.3.

Figure 4.3a shows the duration of 100 *Join-requests* for the baseline case. Figures 4.3b, 4.3c, and 4.3d illustrate the *Join-request* duration when using LoRaDANCE with fragment sizes of 100, 150, and 200 bytes, respectively. In the case of LoRaDANCE, in addition to measuring the total duration of the *Join-requests*, we also visualize the individual components of this duration and use a broken y-axis to emphasize better the lower values we obtained. Specifically, we illustrate the following:

- The time between creating the *Join-request* and receiving the last fragment of the *Join-accept* from the *NS*.
- The time the *ED* takes to validate the DNSSEC chain.
- The time the *ED* takes to derive the shared secret and session keys.

The results indicate an increase in the duration of the join process when LoRaDANCE is used. Specifically, the total join process duration increases from an average of 5.18 seconds in the baseline case to 43.48 seconds with a 100-byte fragment size, 27.98 seconds with a 150-byte fragment size, and 22.99 seconds with a 200-byte fragment size. This increase is primarily due to the fragmentation and transmission of the *Join-accept* message and its reassembly at the *ED*. Conversely, the DNSSEC chain validation and key derivation add minimal overhead. It is also observed that increasing the fragment size significantly decreases the incurred delay, as larger fragments reduce the number of transmissions required.

4.4.2 Power Consumption

We measure the computation power consumed by the NUCLEO-L476RG board during the join process, testing several scenarios where the device sends a *join-request* using LoRaDANCE with fragment sizes of 100, 150, and 200 bytes. These measurements are compared against the baseline case. The results are presented in Figure 4.4.

Consistent with the results observed for the total duration of the join process, LoRaDANCE appears to increase the overall power consumption. In the case of LoRaDANCE, the most power-intensive phases are sending the *join-request*, verifying the DNSSEC chain, and configuring the *ED*. However, these power spikes occur over short periods of time. On the other hand, the *ED* consumes a moderately low amount of power (around 22 mW) for the duration of receiving and reassembling the fragments of the *Join-accept*. We observed earlier that increasing the fragment size significantly decreases the delay caused by the transmission of fragments, as larger fragments reduce the number of transmissions required. This extends to the overall power consumption, where shorter transmission durations consume less power. For the baseline case, the most power-intensive phases are, as expected, sending the *Join-request* and receiving the *Join-accept*.

4.5 Discussion

The practicality of LoRaDANCE The overhead introduced by LoRaDANCE is primarily due to the fragmentation and transmission of the *Join-accept* message and its reassembly at the *ED*. For security-

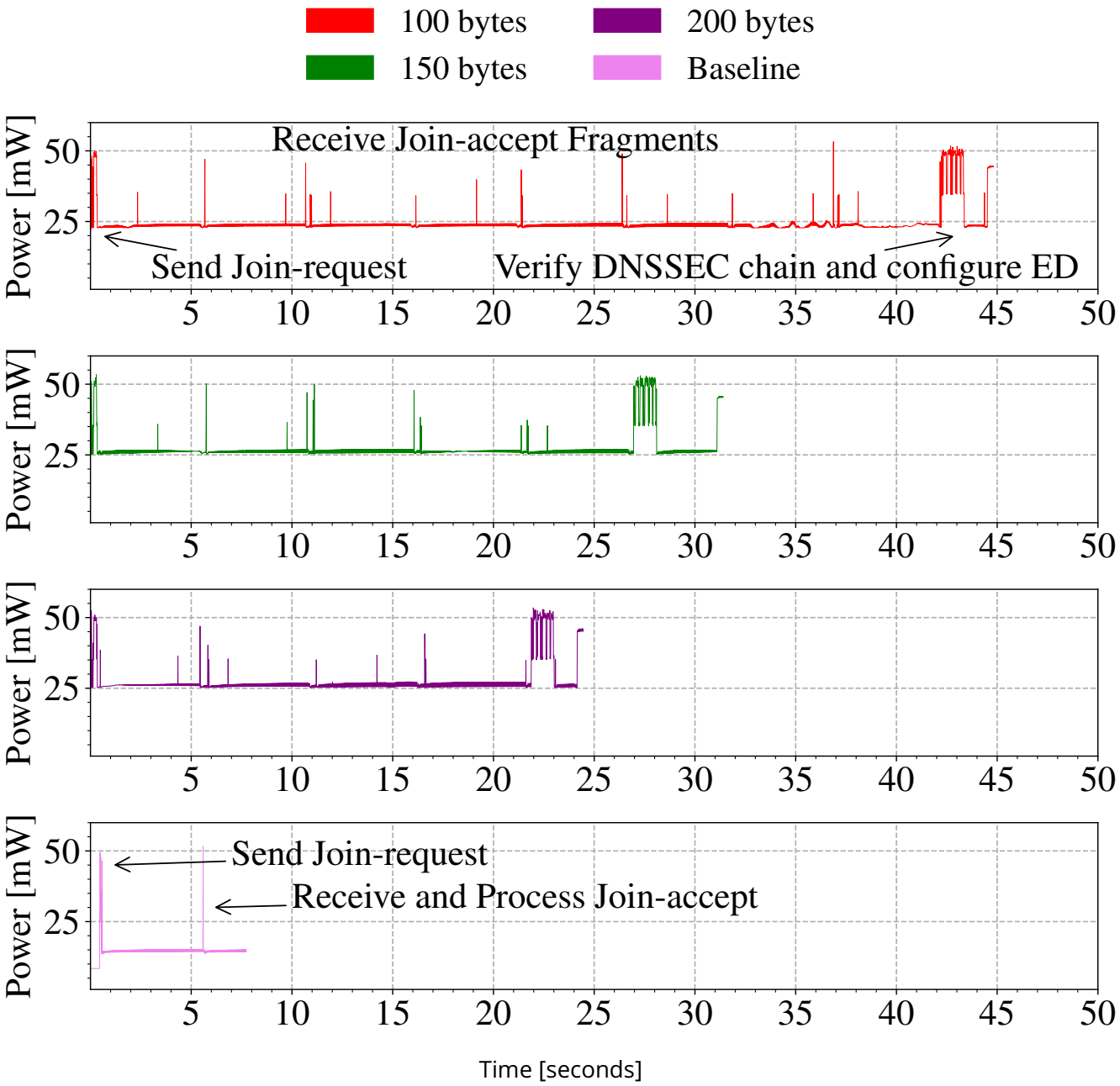


Figure 4.4: Power consumption during the join process for the baseline LoRaWAN 1.0.x and LoRaDANCE with fragment size = 100, 150, and 200 bytes.

critical applications, the increased resource consumption is an acceptable trade-off for enhanced security. Conversely, in resource-constrained environments where power efficiency and quick join times are crucial, further optimizations of LoRaDANCE may be necessary. Finally, it should be noted that the join process does not occur frequently in typical LoRaWAN networks, which mitigates the impact of the increased join process duration and power consumption.

Extending LoRaDANCE to LoRaWAN 1.1 LoRaDANCE can be easily extended to LoRaWAN 1.1, where, in addition to the *AppKey*, which is used to generate the *AppSKey* session key, the secret *NwkKey* is employed

to generate three additional session keys: *FNwkSIntKey*, *SNwkSIntKey*, and *NwkSEncKey* [37]. Within the context of LoRaDANCE, the *NwkKey* could be derived by taking the rightmost 16 bytes of the result from the ECDH operation.

Considerations when Choosing a DNSSEC trust anchor When selecting a DNSSEC trust anchor other than the root nameservers, it is crucial to choose a trustworthy entity (*e.g.*, the LoRa Alliance). However, it's important to note that using a new trust anchor may prevent the use of Canonical Name (CNAME) records, as these records could redirect to domain names with trust anchors still tied to the root nameservers.

4.6 Conclusion

In this chapter, we presented LoRaDANCE, a novel mechanism that leverages DANE and asymmetric cryptography to allow a LoRaWAN *End Device* to join the network without the need to pre-share any secret keys with the backend servers while mutually authenticating with the *Join Server*. We implemented and tested our solution on real devices. While LoRaDANCE introduces an overhead in terms of the join process duration and consumed power, the trade-offs are acceptable, given the security benefits. Looking forward, we plan to explore optimizations to further reduce overhead and extend the applicability of LoRaDANCE to more constrained environments.

Chapter 5

IoT Backend Servers: Studying IoT Domain Names

Contents

Chapter Overview

This chapter provides an in-depth analysis of IoT domain names, specifically those associated with IoT backend servers. These servers are typically contacted by IoT devices to relay data, receive commands, or download software updates. The devices are usually pre-configured with the domain names of these servers, which are resolved through the Domain Name System (DNS). The chapter examines the differences between IoT and non-IoT domain names, beginning with the compilation of an IoT dataset based on previous studies and the preparation of a corresponding non-IoT dataset. The study is conducted in three phases. The first phase focuses on analyzing the composition and statistical properties of the domain names. The second phase involves a DNS analysis, comparing how DNS zones for IoT domain names are structured compared to those of non-IoT domain names. Finally, the third phase involves using machine learning to classify the two types of domain names, identifying inherent differences between them.

5.1 Introduction

IoT devices are rarely standalone and often contact dedicated backend servers [177]. Interacting with these backend servers is essential for the functioning and maintenance of these devices. To reach one of its backend servers, an IoT device is usually equipped with the domain name of this server which it resolves using the Domain Name System (DNS). Afterwards, a connection can be established and the device could for example, relay data to the server or receive commands and firmware updates.

We aim to gain a better understanding of IoT backend servers using their domain names. For simplicity, we refer to these domain names as IoT domain names. We conduct a three-phase study over IoT domain names and compare them to non-IoT domain names, *i.e.*, domain names of servers catering to non-IoT devices and human users.

The study will allow us to investigate the structure of IoT domain names and if they differ from their non-IoT counter parts, allowing us to determine any patterns or identifiers that may appear in one class of domain names but not in the other. We will also analyze the DNS zones of IoT and non-IoT domain names. A DNS analysis gives an insight into the DNS practices in each class of domain names and reveals information regarding, for example, the security and dynamism of the average DNS zone from each class of domain names. The final phase of this study involves using machine learning to classify IoT and non-IoT domain names. This will allow detecting the subtle differences between IoT and non-IoT domain names, differences that may not be traceable visually or using by studying the statistical properties.

We compile two datasets of domain names. First, using packet captures from 12 past studies that included testbeds with real IoT devices, we construct a dataset of IoT domain names. We call this dataset the IoT dataset. Second, we use two datasets of non-IoT domain names. Previous works [195, 165, 199, 169] use lists of top-visited websites as a negative class in domain name classification problems. We also use two top lists, namely Cisco Umbrella 1 Million [74] and Tranco Top 1M [190, 132].

5.2 Motivation

In this chapter, we do not study specific IoT technologies or certain properties of these technologies. Instead, This chapter studies IoT from a new perspective by studying the domain names of the servers contacted by IoT devices. We aim to investigate the differences between the domain names of these servers and the domain names of servers contacted by non-IoT devices and human users.

We first study the structure of IoT and non-IoT domain names. This includes studying the composition and the statistical properties of these domain names and drawing conclusions about the patterns and particularities of the average domain name of each class. This phase will clarify if the choice of domain names depends on the types of devices the underlying server caters to. In addition, we will be able to extrapolate any possible current conventions in naming IoT domain names and if any special attention is given when assigning domain names to such servers. This could be helpful, for example, to model and generate IoT domain names that align with the average domain name of that type—for instance, aiding protocol design such as name compression for constrained devices[140] and in data augmentation in the context of machine learning.

Second, we study the DNS properties of the IoT and non-IoT domain names. This includes resolving several DNS resource records (RRs) for each domain name of each dataset and calculating for each RR the percentage of domain names that have it, the average resolution time, the average Time-To-Live (TTL), which tells the resolver how long to cache a query before requesting a new one, and the average response size. This study will help identify how the DNS zones of each class of domain names are set up. We will be able, for example, to identify the dominant RRs used in each class, whether security-related RRs such as Domain Name System Security Extensions (DNSSEC)-related and Certification Authority Authorization (CAA) RRs appear in IoT DNS zones, how static IoT DNS zones are by comparing their TTL values, and the average size of DNS responses from these zones are to indicate if IoT zones, belonging generally to constrained devices, have smaller DNS responses to enhance performance.

The third phase of our study evaluates the feasibility of classifying between the two classes of domain names based solely on the domain name. We investigate if popular machine learning models often used in domain name classification problems can successfully classify IoT and non-IoT domain names based on their inherent differences. This phase of our study provides a deeper insight into the differences between the two classes of domain names. This includes differences that are not detectable by visually inspecting or statistically studying the domain names. We can also conclude through this study the parts of the domain name that are most indicative of class, *i.e.*, which section of the domain name holds the most information and therefore differentiates one class from the other. Besides drawing conclusions about structural differences between the two classes, successfully classifying IoT and non-IoT domain names using machine learning allows the detection of possible IoT traffic in mixed traffic containing traffic of both classes. This capability could aid in detecting outliers in IoT networks. For example, non-IoT traffic in networks consisting solely of IoT devices could be detected.

5.3 Background

5.3.1 IoT Domain Names

IoT devices usually contact servers on the Internet to relay information about the physical world or to receive commands and firmware updates [177]. These devices rely on domain names as an indirection mechanism to connect to IP endpoints, simplifying maintenance. This allows, for example, a transparent change of server addresses, as only the mapping in the DNS would need to be changed. The IoT devices are typically pre-configured with the domain names of the servers they might need to contact, and they obtain the addresses of these servers by resolving the domain names via DNS. Beyond address resolution, future IoT devices might also use DNS to identify the service bindings of these servers, *e.g.*, whether they use the TCP-based HTTP/2, the QUIC-based HTTP/3, or other services such as CoAP, using SVCB (Service Binding) resource records [181, 41].

The domain names of such servers may exhibit distinct construction characteristics influenced by var-

ious factors. An IoT backend server, for example, might have a name that correlates with its high-level function. For example, a collection of IP cameras might have a backend server whose domain name is `cam.example.com`. Moreover, large IoT backend service providers tend to adopt a naming convention for their servers, which follows the pattern below [177]:

`<subdomain>.<region>.<second-level-domain>`,

where `<subdomain>` could be the name of the IoT service or the protocol name, `<region>` refers to the location of the server, and `<second-level-domain>` could be the second-level domain of the service provider or a name related to the IoT service.

Last, being involved in machine-to-machine (M2M) communications, some IoT domain names may contain machine-friendly character sequences that do not prioritize legibility or memorability and are challenging for humans to comprehend.

Meanwhile, non-IoT domain names do not follow the same patterns. Since servers with such domain names serve a wide range of devices and human users, the legibility and memorability of their domain names are prioritized.

5.3.2 Domain Name Classification

Domain name classification leverages machine learning techniques to classify two or more categories of domain names. It enables, for example, efficiently detecting phishing or domain names generated by domain generation algorithms (DGAs) [88, 165, 195, 69]. Phishing domain names are used by malicious servers that pose as legitimate ones and lure users into providing sensitive information and credentials. On the other hand, DGAs run on malware-infected devices and generate domain names to help the infected devices contact the Command & Control (C&C) servers. These domain name classification techniques could also be applied in IoT environments, aiding in classifying IoT and non-IoT domain names.

5.3.3 Brief Overview of Machine Learning

Supervised machine learning is a subset of artificial intelligence that allows models to improve their performance by learning from data. Supervised machine learning involves having a labeled dataset with correct (target) values. This dataset is split into training and testing datasets. The models are trained using the training dataset and their performance is evaluated using the testing dataset. In unsupervised machine learning, the datasets are not labeled. Instead, the models identify patterns and relationships in the data. Reinforcement learning is a third type of machine learning that trains an agent. The agent interacts with its environment and calculates a reward based on its actions. The agent aims to maximize the reward. Supervised machine learning techniques could be useful in the context of classification between IoT and non-IoT domain names.

Machine learning models have demonstrated their power in classification, be it binary (two classes) or multi-class. For example, they have been used for domain name classification to detect phishing attacks [88, 165, 195, 69]. To achieve this, labeled datasets containing phishing and benign domain names are used to train the models.

The machine learning models usually expect numerical data which necessitates obtaining the real-valued vector representation of the domain names. Depending on the method used, the real-valued vector representation can capture the semantic information and context of words in the textual data.

There are several options to achieve this. One way is through Natural Language Processing (NLP). NLP methods aim to obtain the real-valued vector representation of the textual data. This includes, for

example, character level embedding [186, 199, 165], which obtains a fixed-size real-valued vector representation of each character. Another NLP method is the Term Frequency-Inverse Document Frequency (TF-IDF), which assigns an importance value to each element of the text based on its frequency of appearance [169]. A different way of processing textual data is feature extraction. Feature extraction methods study the text and try to extract properties (*e.g.*, domain name length, number of hyphens, number of dots) and use those properties as a real-valued vector representation of the textual data [169, 67].

5.3.4 Machine Learning Models Overview

We use supervised machine learning techniques to perform the binary classification between IoT and non-IoT domain names. The following is a brief overview of the six models we use, which can be used for binary and multi-class classification, except Logistic Regression, which is used for binary classification but can be extended to perform multi-class classification.

Naïve Bayes (NB):

NB [170] is a parametric classifier based on Bayes's theorem. It is referred to as *naïve* since it assumes the mutual independence of features. This means that the value of one feature does not affect the value of any other feature in the input vector. While this assumption might not hold in real-world scenarios, it simplifies computation.

Logistic Regression (LR):

LR [105] is a parametric classifier. It is referred to as *logistic* since it uses the *logistic function* (sigmoid function) to map a linear combination of features to a probability score.

K-Nearest Neighbors (KNN):

KNN [77] is a non-parametric and instance-based classifier. It is a simple model that does not involve the traditional training and testing phases. Instead, it assumes that similar data points are placed closer to each other, so the class of a particular data point is similar to the class of its K *nearest neighbors*.

Support Vector Machine (SVM):

SVM [106] is a parametric classifier whose primary objective is to find a hyperplane that best separates the data points of different classes. The closest points to the hyperplane from the different classes are called *support vectors*.

Decision Tree (DT):

DT [120] is a non-parametric classifier. A DT is a *tree-like structure* where every node represents a feature, every branch is a decision, and every leaf node is a class. DTs are trained through a process known as recursive splitting and the goal is to have most features represented as nodes with their branches and leaf nodes. A downside of DTs is their tendency to experience overfitting as the size of the dataset grows.

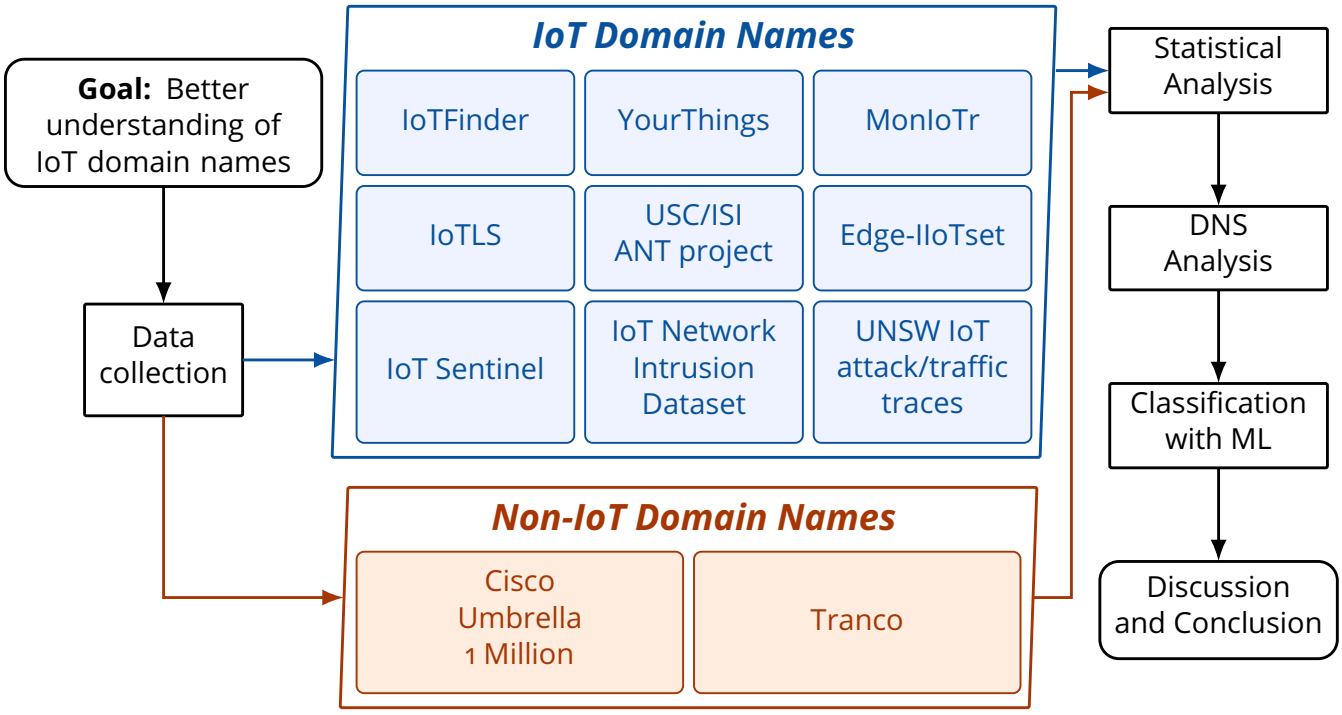


Figure 5.1: Data collection and phases of this study.

Random Forest (RF):

RF [65] is a non-parametric classifier. It is referred to as an *ensemble method* because it leverages the power of a group (an ensemble) of individual Decision Trees to improve predictive accuracy and reduce overfitting.

5.3.5 Word2vec for Word Embedding

In this chapter, we use a Word2vec model to obtain the real-valued vector representation of the domain names. In the context of NLP, this type of representation is known as a word embedding. Word2vec [147, 146] is one of several word embedding techniques, and it uses a shallow neural network to convert each word to a vector of real numbers. Word2vec captures semantic relationships between words by learning from large amounts of text data, and the resulting real-valued vectors depend on the context in which each word appears within the text. Two possible architectures for Word2vec exist: Continuous Bag-of-Words (CBOW) and Skip-gram. CBOW estimates the vector representation of a target word based on the context (*i.e.*, surrounding words) of this word. The number of surrounding words considered within the context is specified by a *window size*. Two words frequently appearing together in similar contexts within the text will have vector representations that are geometrically close to each other in the vector space. Consequently, words that do not appear regularly together in the text are assigned vector representations that are distant geometrically. For Skip-gram, the target word is used to predict the surrounding context words within a specified window size. Between CBOW and Skip-gram, we chose to go with CBOW as it is less expensive computationally and faster to train [146].

In the following sections, we demonstrate the different phases of our study, the statistical analysis, DNS analysis, and the classification between IoT and non-IoT domain names using machine learning. Figure 5.1 summarizes our steps. We first discuss how the data were collected to construct the lists of IoT and non-IoT

domain names.

5.4 Data Collection

In our analysis, we use two categories of domain name datasets. The first is the IoT dataset, which includes a list of domain names of servers contacted by IoT devices. The second category must contain domain names of servers that cater to generic devices and human users, namely non-IoT domain names. For this purpose, we use two lists of top-visited websites, the Cisco Umbrella 1 Million [74] and Tranco [190, 132].

5.4.1 IoT Dataset

The IoT dataset should contain domain names of servers on the Internet contacted by IoT devices, *e.g.*, IoT backend servers that provide services to IoT devices such as a server that saves the footage from IP cameras or servers from which IoT devices receive commands and software updates. This list may also include domain names of servers that are not IoT-specific but are nevertheless contacted by IoT devices. To construct the IoT dataset, we used 12 datasets from previous studies: IoTFinder [162], YourThings [40], MonIoTr [173], IoTLS [161], three datasets from the USC/ISI ANT project [43, 44, 42], Edge-IIoTset [93], IoT Sentinel [145], IoT Network Intrusion Dataset [126], UNSW IoT traffic traces [184], and UNSW IoT attack traces [101]. These datasets contain packet captures collected in testbeds that included real IoT devices. Each dataset is available as a set of PCAP files, including DNS messages sent from and received by the devices. Table 5.1 presents the devices used in each testbed, and below is a brief overview of the studies:

- **IoTFinder** [162]: IoTFinder is a multi-label classifier for detecting IoT devices that uses passively collected DNS traffic. The data were collected between August 1, 2019, and September 30, 2019.
- **YourThings** [40]: A study of home-based IoT devices to assess their security properties. The data were collected between April 10 and April 19, 2018.
- **MonIoTr** [173]: A study of information exposed in the traffic of consumer IoT devices. The data were collected between March 28 and May 8, 2019, and on September 1, 2019.
- **IoTLS** [161]: A study about the use of TLS in consumer IoT devices. The data were collected between January 2018 and March 2020.
- **USC/ISI ANT project** [43, 44, 42]: The ANT Lab is an Internet research group at the University of Southern California (USC) that has published several datasets related to various network topics, *e.g.*, traffic, outage, and DNS. We used three datasets from the USC/ISI ANT project. Two datasets contain the bootup traces of several IoT devices. The third dataset contains traffic observed in an IoT network of several devices over ten days.
- **Edge-IIoTset** [93]: An IoT traffic dataset that includes benign and attack traffic. The benign traffic we used in this chapter was collected between November 21, 2021, and January 10, 2022.
- **IoT SENTINEL** [145]: IoT SENTINEL is a security system that identifies devices present in the network and monitors traffic from vulnerable ones. The traffic was collected during the setup of each device.
- **IoT Network Intrusion Dataset** [126]: An IoT traffic dataset that includes benign and attack traffic. We used the benign traffic in this chapter.

Table 5.1: The IoT devices in the datasets used in this study

IoTFinder & YourThings	Nest Camera	D-Link Mov Sensor	Xiaomi Cam	Smartlife Bulb	Soil Moisture Sensor	Amazon Echo
Echo Dot Gen3	Nest Guard	Echo Dot	Xiaomi Cleaner	Smartlife Remote	Sound Sensor	August Cam
Amazon Echo Gen1	Nest Protect	Echo Plus	Xiaomi Hub	Smartthings Hub	Stepper Motor	Awair air monitor
Amazon Fire TV	Nest Thermostat	Echo Spot	Xiaomi Rice Cooker	Switchbot Hub	Temperature Sensor	Belkin Camera
Android Tablet	Netgear Arlo Camera	Fire TV	Xiaomi Strip	TP-Link Bulb	Ultrasonic Sensor	Wemo Motion Sensor
Apple HomePod	Nintendo Switch	Flux Bulb	Yi Cam	TP-Link Plug	Water Level Sensor	Belkin Switch
Apple TV (4thGen)	Nvidia Shield	GE Microwave	ZModo Doorbell	Wemo Plug	**IoT SENTINEL**	Blipcare BP Meter
August Door Cam	Philips Hue	Google Home	**IoTLS**	Wink Hub 2	D-Link Home Hub	Canary Camera
AVTech IP Cam	Piper NV	Google Home Mini	Amazon Cloudcam	Yi Camera	D-link Day Camera	Dropcam
Belkin Netcam	Play Station 4	Honeywell T-stat	Amazon Echo Dot	Zmodo Doorbell	D-Link Door Sensor	Google Chromecast
Belkin Crockpot	Rachio 3	Insteon Hub	Amazon Echo Dot 3	**ISI**	D-link HD IP Camera	Hello Barbie
Belkin WeMo Link	Ring Doorbell	Invoke with Cortana	Amazon Echo Plus	Amazon Dash Button	D-link Motion Sensor	HP Printer
Wemo Motion Sensor	Roku 4	Lefun Cam	Amazon Echo Spot	Amazon Echo Dot	D-link Siren	iHome PowerPlug
Belkin WeMo Switch	Roku TV	LG TV	Amcrest Camera	Amazon Fire TV	D-link Smart Plug	LiFX Bulb
Bose SoundTouch 10	Roomba	Lightify Hub	Apple HomePod	Amcrest IP Cam	D-Link Water Sensor	NEST Smoke Sensor
Canary	Samsung Hub	Luohe Cam	Apple TV	Belkin SmartPlug	Edimax Cam	Netatmo Camera
Caseta Wireless Hub	Samsung Smart TV	Magichome Strip	Behmor Brewer	D-Link IP Cam	Edimax Smart Plug	Netatmo station
Chamberlain Opener	Securifi Almond	Microseven Cam	Blink Camera	Dyson Purifier	Edimax Smart Plug 2	Phillip Hue Lightbulb
Chinese Webcam	Sonos	Nest T-stat	Blink Hub	Foscam IP Cam	Ednet Cam	Pixstart photo frame
D-Link DCS-5009L	Sonos Beam	Netatmo Weather	D-Link Camera	Foscam IP Cam 2	Ednet Gateway	Ring Door Bell
Facebook Portal	TP-Link WiFi Bulb	Philips Bulb	Fire TV	Google Speaker	Fitbit Aria	Samsung Smart Cam
Google Home Hub	TP-Link WiFi Plug	Philips Hue Hub	GE Microwave	HP Envy 4500 Printer	Homematic switch	Smart Things
Google Home Mini	Ubuntu Desktop	Ring Doorbell	Google Home Mini	Philips Hue	Lightify Gateway	TP-Link Camera
Google OnHub	Wink Hub	Roku TV	Harman Invoke	Renpho Humidifier	MAX! Gateway	TP-Link Plug
Google Home	Wink Hub 2	Samsung Dryer	Insteon Hub	Samsung IP Cam	Philips Hue Bridge	Triby Speaker
Harman Invoke	Withings Home	Samsung Fridge	LG Dishwasher	Tenvis IP Cam	Philips Hue Switch	Withings Monitor
Insteon Hub	Xbox One X	Samsung TV	LG TV	TPLink Smart Bulb	Smarter iKettle	Withings Scale
iPad	**MonIoTr**	Samsung Washer	Meross Dooropener	TPLink Smart Plug	Smarter Coffee	Withings sleep sensor
iPhone	Allure with Alexa	Sengled Hub	Nest Thermostat	Wansview IP Cam	TP-Link Smart Plug	Amazon Echo
Koogeek Lightbulb	Amazon Cloud Cam	Smarter Brewer	Philips Hub	Wyze IP Cam	TP-Link Smart Plug 2	Chromecast Ultra
LG WebOS TV	Amcrest Cam	Smarter iKettle	Ring Doorbell	**Edge-IIoTset**	WeMo Insight Switch	iHome Smart plug
LIFX Virtual Bulb	Anova Sousvide	Smart Things Hub	Roku TV	DC Motor	WeMo Bridge	LiFX bulb
Harmony Hub	Apple TV	TP-Link Bulb	Samsung Dryer	Flame Sensor	Wemo Switch	Netatmo camera
Logitech Logi Circle	Behmor Brewer	TP-Link Plug	Samsung Fridge	Heart Rate Sensor	Withings Scale	Phillips Hue bulb
VeraLite controller	Blink Cam	Wansview Cam	Samsung TV	Humidity Sensor	**Kang**	Samsung smartcam
My Cloud EX2 Ultra	Blink Hub	WeMo Plug	Samsung Washer	IR Receiver Sensor	EZVIZ WiFi Camera	TP-Link smart plug
Nest Bell	Bosiwo Cam	WiMaker Camera	Sengled Hub	pH Sensor	SKT NUGU speaker	WeMo motion
Nest Cam IQ	D-Link Cam	Wink 2 Hub	Smarter iKettle	Servo Motor	**UNSW\Attack**	WeMo Switch

- **UNSW IoT traffic traces** [184]: A study about the classification of IoT devices in Smart Home environments. The traffic was collected between October 2016 and April 2017.

- **UNSW IoT attack traces** [101]: A study about detecting volumetric attacks against IoT devices and the dataset includes benign and attack traffic. The traffic was collected for 16 days.

We filtered the PCAP files and extracted the unique DNS responses received by each device. Some datasets also contained captures from generic non-IoT devices such as desktop PCs, smartphones, or gaming consoles. These devices are shaded in Table 5.1, and we removed their packet captures. Finally, we extracted the queried domain names from the resulting DNS responses. The number of unique IoT domain names obtained from each dataset is presented in Figure 5.2. The resulting IoT dataset contained 4145 unique domain names.

5.4.2 Non-IoT datasets

The non-IoT dataset should contain domain names of servers used by generic, non-IoT devices or those used by humans directly. For that purpose, we use two lists of top-visited websites as non-IoT datasets:

- **Cisco Umbrella 1 Million** [74], which we refer to as the Cisco dataset, is a daily published list of one million websites. Any domain name could be included in the list. The ranking of each domain name is based on the number of unique client IPs that visited it [74]. The list for our evaluation was gathered on July 15, 2024.

- **Tranco** [190], which we refer to as the Tranco dataset, is a research-oriented list of one million domain names. The ranking of each domain name is based on its average rank over the past 30 days from four other popular domain name lists [132]. The Tranco list for our evaluation was gathered on July 15, 2024, and thus covers the period from June 15 to July 14, 2024.

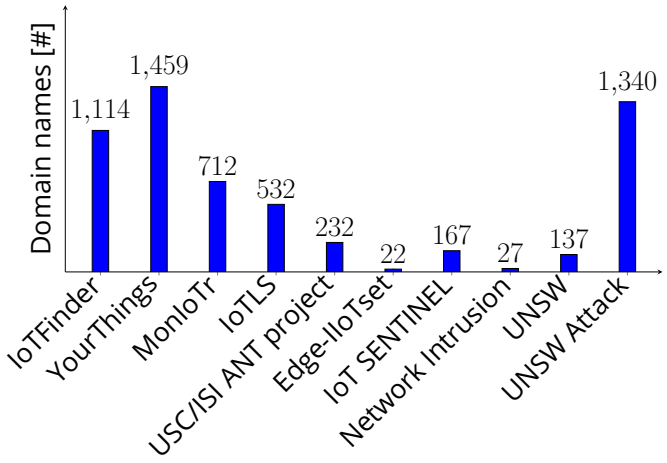
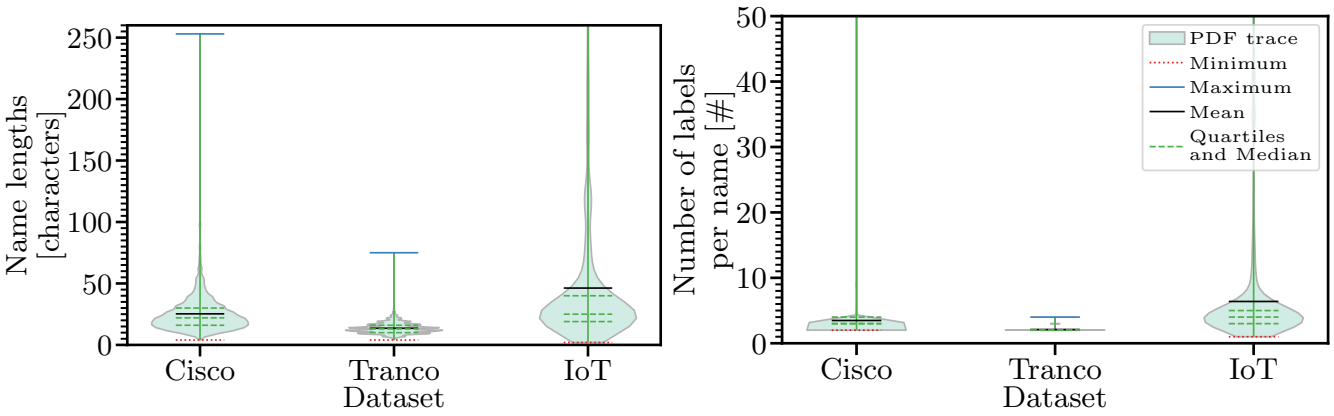


Figure 5.2: The number of unique IoT domain names extracted from each dataset.



(a) Name lengths.

(b) Number of labels.

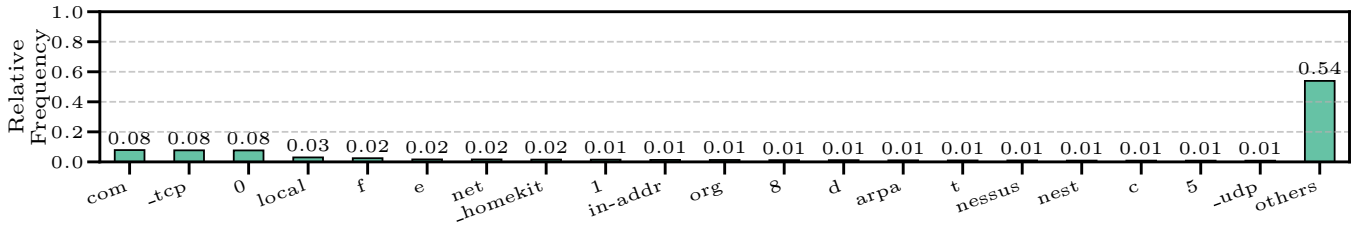
Figure 5.3: Violin plots for name properties found for each domain name in our datasets.

Such top-lists are usually used in research, especially in domain name classification problems [195, 165, 199, 169]. Some IoT domain names might appear in the top-visited websites lists, and we can account for that by removing these domain names from the Cisco and Tranco datasets.

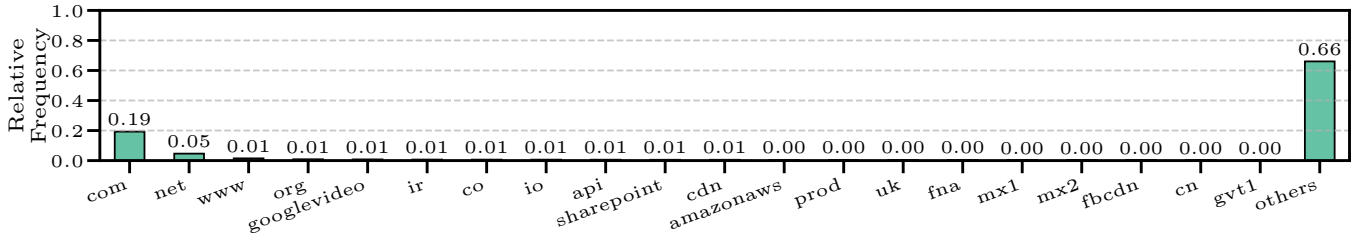
5.5 Statistical Study

We perform a statistical analysis of the domain name lengths and number of labels in each dataset, for which the results can be seen in the violin plots in Figure 5.3. Violin plots are similar to box plots, showing key statistical properties. However, they also estimate the probability density function (PDF) as a trace that forms the "body" of the "violins" around the properties.

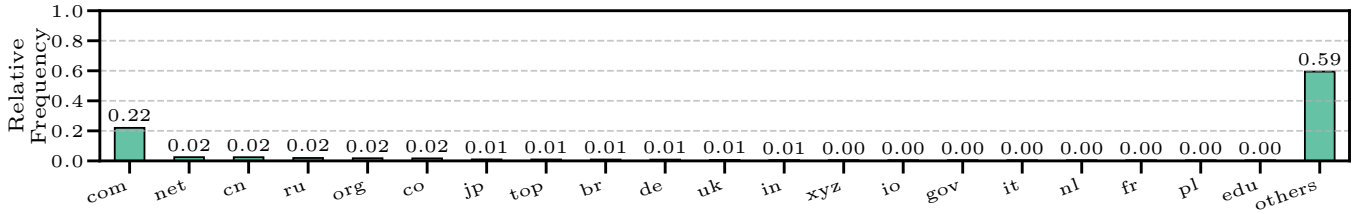
The violin plots allow us to easily spot a similarity between domain names in the IoT and Cisco datasets regarding domain name length and number of labels per domain. This is due to the way each dataset is constructed. The two datasets contain domain names as observed in the DNS requests and are, therefore, more representative. The Tranco dataset, on the other hand, is different from the others. The average Tranco domain name has fewer characters and labels than the average domain name from the other datasets. The Tranco dataset mainly contains second-level domains in the form of *domain.tld*, while the



(a) IoT dataset



(b) Cisco.



(c) Tranco.

Figure 5.4: Violin plots for name properties found for each domain name in our datasets

IoT and Cisco datasets do not have that limitation.

Moreover, it is observed that IoT domain names do not adhere to the restriction imposed by RFC 1035 [28], which limits the maximum domain name size to 253 bytes and the maximum size of the label to 63 bytes. In fact, among the 4145 domain names present in the IoT dataset, 73 domain names exceed 253 bytes with a maximum length of 652 characters. Meanwhile, the domain names from the Cisco and Tranco datasets are at most 253 bytes, with maximum lengths of 253 and 75 characters for the Cisco and Tranco datasets, respectively. This suggests that size restrictions are less stringent when choosing an IoT domain name. However, considering that a significant portion of IoT devices are constrained and have limited resources, reducing the length of the domain names these devices resolve could reduce the overall size of the DNS messages and potentially lower power consumption.

In addition to studying the length and number of labels, we plot the relative frequencies of the top labels in each dataset in Figure 5.4. The goal is to assess how common –or uncommon– the labels between IoT and non-IoT domain names are. Hence, we plot the top 20 labels for each dataset and aggregate the remaining labels under *others*.

The label "com" comes first in the three datasets, accounting for approximately 20% of the labels in the Cisco and Tranco datasets and 8% of the labels in the IoT dataset. Beyond "com", the IoT and non-IoT domain names diverge in their subsequent labels. The most frequent labels in the two non-IoT datasets, Cisco and Tranco, are mostly Top-Level Domains (TLDs) such as "net", "org", and "co". See Figures 5.4b and 5.4c. Meanwhile, the most frequent labels in the IoT dataset do not follow the same trend. We observe in Figure 5.4a that, contrary to the non-IoT datasets, the top 20 labels in the IoT dataset are not mostly TLDs. Instead, we observe protocol-related labels such as "tcp" and "udp" and IoT-technology-specific and brand-

Table 5.2: The results (Averages) of the DNS analysis of the IoT and non-IoT datasets

	A				AAAA				MX				CNAME				DNSKEY				DS				TXT				SRV				CAA			
	Record existence [%]	Query duration [s]	TTL [s]	Response size [bytes]	Record existence [%]	Query duration [s]	TTL [s]	Response size [bytes]	Record existence [%]	Query duration [s]	TTL [s]	Response size [bytes]	Record existence [%]	Query duration [s]	TTL [s]	Response size [bytes]	Record existence [%]	Query duration [s]	TTL [s]	Response size [bytes]	Record existence [%]	Query duration [s]	TTL [s]	Response size [bytes]	Record existence [%]	Query duration [s]	TTL [s]	Response size [bytes]	Record existence [%]	Query duration [s]	TTL [s]	Response size [bytes]	Record existence [%]	Query duration [s]	TTL [s]	Response size [bytes]
IoT	**46.66**	77.97	**1165846.37**	134.05	**14.60**	60.62	**6583.37**	175.23	**1.01**	187.70	15532.50	125.40	28.97	76.12	8540.39	96.53	**0.12**	38.79	2943.40	276.60	**0.12**	166.17	69180.0	88.80	10.74	148.96	4242.48	184.07	0.0	0.0	0.0	0.0	**0.53**	40.85	18204.50	190.82
Cisco	**78.41**	51.27	**999.08**	138.04	**29.67**	52.03	**564.32**	164.04	**12.30**	60.41	12911.55	126.97	40.97	69.20	6986.38	96.92	**1.83**	65.99	1622.97	275.84	**1.64**	125.06	54651.0	94.26	19.47	74.88	7274.32	598.11	0.07	159.15	58860.0	115.67	**5.33**	56.76	15958.43	182.28
Tranco	**79.88**	76.21	**4164.59**	75.18	**24.34**	53.57	**3190.43**	108.61	**71.70**	99.69	19324.49	119.80	0.55	180.26	635.22	76.26	**10.11**	51.31	40029.60	370.47	**8.69**	121.86	55525.03	99.79	83.43	109.70	9388.62	738.17	0.29	111.77	8810.0	118.42	**20.46**	75.58	18361.85	225.75

specific labels such as "_homekit", "nessus", and "nest". This indicates that the domain names in IoT environments are often tailored specifically for these environments. Furthermore, the presence of "local" indicates a prevalence of local domains, highlighting the focus on local communications within IoT environments. Finally, the appearance of "in-addr" and "arpa" in the top labels indicates significant reverse DNS activity.

The statistical study demonstrates the characteristics of both classes of domain names and allows us to draw a few conclusions about the differences between IoT and non-IoT domain names. First, we noticed a deviation from RFC 1035 [28] guidelines concerning domain name size in the IoT dataset, a phenomenon not present in the non-IoT datasets, namely the Cisco and Tranco datasets. The DNS activity in IoT environments mainly involves machine-to-machine communications, with many of these domain names configured using auto-configuration schemes. When combined with the significant local DNS activity in IoT environments –evidenced by the strong presence of labels like "local" in the IoT dataset–, it becomes apparent why longer domain names might appear in such networks. Second, when studying the most frequent labels in each dataset, several protocol-related or technology-related labels such as "tcp" and "nest" appeared among the top labels in the IoT dataset, signaling an inclination to give explicit, self-explanatory domain names to IoT-related backend servers. On the other hand, the top 20 most frequent labels in the non-IoT datasets were predominantly TLDs. This demonstrates the stark diversity of these domain names in labels other than TLDs, which, unlike IoT domain names, do not indicate specific protocols, technologies, or providers.

5.6 DNS Analysis

In this section, we perform a DNS analysis of the different datasets. The goal is to compare the IoT, Cisco, and Tranco datasets by resolving several RRs and recording their availability, the query duration, the Time To Live (TTL), and the response size for each RR. This study gives insights into any possible differences between the DNS zones of the average domain name from each dataset. If found, we highlight these differences. The RRs we resolve are:

- **A**: IPv4 address record. It maps a domain to an IPv4 address.

- **AAAA**: IPv6 address record. It maps a domain name to an IPv6 address.
- **MX**: Mail exchange record. It routes emails to specific mail servers.
- **CNAME**: Canonical name record. It is used when a domain name is an alias for another domain name.
- **DNSKEY**: It holds a public key that resolvers use to verify DNSSEC signatures.
- **DS**: Delegation Signer record. It is used to verify DNSKEY records of child DNS zones.
- **TXT**: Stores text notes.
- **SRV**: Service record. It specifies the server/port number of a specific service.
- **CAA**: Certificate Authority Authorization record. It allows domain name owners to specify which certificate authorities are allowed to issue TLS certificates for their domain name.

We use Google's public DNS resolver [11] (`'1.1.1.1'`) and the study is done on the 4145 domain names of the IoT dataset and the top 4145 domain names of the Cisco and Tranco datasets. The results are presented in Table 5.2. The *Record existence [%]* is the percentage of domain names in each dataset with that record. The rest of the values, namely *Query duration [s]*, *TTL [s]*, and *Response size [bytes]*, are average values over each dataset.

Record existence The IoT dataset has the lowest existence rate for all the resource record types. In particular, only 46.66% of the IoT domain names have an A resource record, and only 14.60% of them have a AAAA resource record. This is partly due to the nature of domain names in the IoT dataset, which includes local (Multicast DNS) addresses that do not resolve to A and AAAA resource records in the global DNS, *i.e.*, outside the local network, but may have such records in a local network context. As for MX records, only 1.01% of IoT domain names have this resource record, while it is 12.30% and 71.70% for the Cisco and Tranco datasets, respectively. The low presence of MX records in the DNS zones of IoT domain names indicates that these domain names are less likely to be associated with email services. Security-related resource records are also sparsely present in IoT-related DNS zones with only 0.12% of IoT domain names having DNSSKEY or DS RRs, and only 0.53% of them having a CAA RR. Meanwhile, the non-IoT datasets show a higher –but nevertheless low– percentage for these RRs. The Cisco dataset has a 1.83%, 1.64%, and 5.33% presense of DNSKEY, DS, and CAA RRs, respectively, among its domain names, while the percentages are 10.11%, 8.69%, 20.46% for the Tranco dataset. This signifies security concerns in IoT environments which already face several security vulnerabilities due to the constrained nature of a lot of IoT devices [95].

Time-To-Live (TTL) Other than resource records that are inherently not adopted by IoT domain names, as demonstrated in the previous paragraph, and whose TTL values are not accounted for, the TTL values of the resource records of IoT domain names exceed those of domain names from the Cisco and Tranco datasets, or are at least comparable. This indicates a more static DNS environment for IoT domain names with less frequent updates than non-IoT domain names. In addition, high TTL values could prolong the impact of compromised records, such as in the case of DNS cache poisoning.

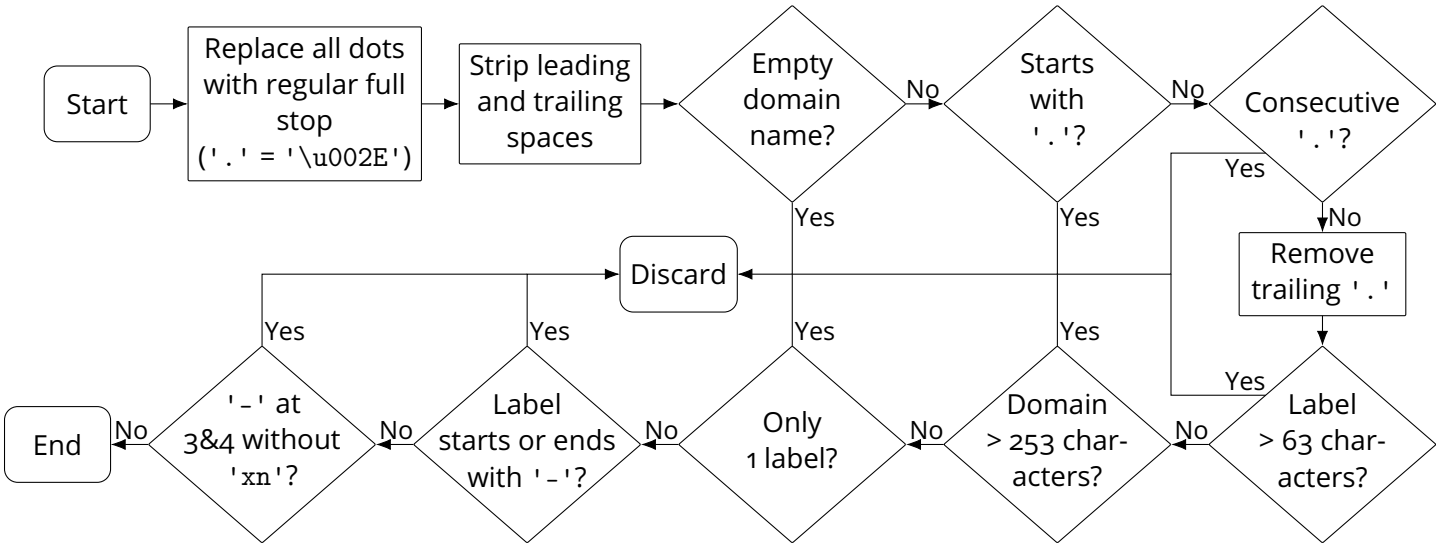


Figure 5.5: Domain name syntax check used by Zonemaster.

Query duration & Response size No notable differences are observed when comparing the query durations and the response sizes between the resource records across the three datasets. This is interesting, considering that a large portion of IoT technologies include constrained devices with a limited power budget. It seems, however, that DNS response size is not yet optimized to suit the limited resources some of these devices have. IoT-friendly DNS protocols such as DNS-over-CoAP (DoC) [139] could be useful to address this issue.

After the statistical and DNS analysis, we move in the next section to classifying IoT and non-IoT domain names using machine learning.

5.7 Classifying IoT and non-IoT domain names: Preprocessing and Word Embedding

5.7.1 Data Preprocessing

We preprocess the data to ensure the validity and consistency of the domain names in each dataset. We perform the following tests:

Syntax Check. The first step includes a syntax check to ensure our datasets are consistent, meaning that the domain names across the datasets respect the same syntax rules. For this purpose, we use the syntax checking used by Zonemaster [34]. The process starts with a normalization procedure that replaces all the dots with the regular full stop of Unicode `'\u002E'` (or `'.'` as a character). The next step removes leading and trailing spaces. Next, a sequence of tests is conducted.

- Check if the domain name starts with a dot,
- Check if the domain name has consecutive dots,
- Remove trailing dots if found,
- Check if any label in the domain name is longer than 63 characters,

- Check if the total length of the domain name is more than 253 characters,
- Check if the domain name has only one label,
- Check if any label starts or ends with a hyphen (`'-'`), and, finally,
- Check if the domain name has double hyphen (`'-'`) at positions 3 and 4 without it starting with `'xn'`[1].

If one of the checks fails, the domain name is discarded. The results of the syntax check are presented in Table 5.3.

Table 5.3: Number of unique domain names after the syntax check

Dataset	*IoT*	*Cisco*	*Tranco*
Accepted [#]	3 953	993 065	1 000 000
Discarded [#]	192	6 935	0
Total [#]	4 145	1 000 000	1 000 000

Remove commons Some domain names from the IoT dataset might appear in the Cisco or Tranco datasets. Therefore, we remove the common domain names between the IoT and other datasets from the other datasets. The resulting dataset sizes can be seen in Table 5.4.

Table 5.4: Number of unique domain names after removing the IoT domain names from the non-IoT datasets

Dataset	*Cisco*	*Tranco*
Common with IoT Dataset [#]	1 565	32
Remaining [#]	991 500	999 968
Total [#]	993 065	1 000 000

Final lists After data preprocessing, we obtain the final datasets, which will be used in the following steps. The final datasets can be seen in Table 5.5.

Table 5.5: Number of unique domain names in the final datasets.

Dataset	*IoT*	*Cisco*	*Tranco*
Domain Names [#]	3 953	991 500	999 968

[1] *i.e.*, not an Internationalized Domain Name (IDN)

5.7.2 Word2vec: Real-Valued Vector Representation of Domain Names

Word2vec expects prose text as input, *i.e.*, in the form of full documents with connected sentences and ideas where it can be used to capture the semantic relations. The challenge we face when using Word2vec with domain names is that these domain names do not form prose text. Instead, they are individual labels separated by periods. As such, each domain name is treated as a sentence, and each label is treated as a word. For example, `iot.backend.org` contains three labels and is transformed to "iot", "backend", and "org". Another challenge is the limited size of domain names, which results in limited context and explains our choice of a *window size* of 3. After the Word2vec algorithm is done, we obtain a real-valued vector representation of each label. The dimensions of each vector are set in advance. Before applying Word2vec, and to have a consistent dataset in terms of size for training the machine learning models, we pad the domain names by adding `'*'` as a dummy label on the left of each domain name. We pad all the domain names to have 120 labels to account for the maximum number of labels per domain in the three datasets, which is 117.

The parameters we used are as follows:

- **Padding:** To each domain name, we added `'*'` on the left. Each `'*'` was treated as a dummy label (*i.e.*, a word), and they were added until all the domain names were of length 120 labels (words).
- **Word2vec:** We used CBOW (Continuous Bag-of-Words Model) with a *window size* of 3.
- **Vectors:** Each word was represented by a vector $\in \mathbb{R}^{32}$.

This vector representation can then be used to map each domain name to a 32×120 real-valued vector ($\in \mathbb{R}^{32\times 120}$). The Word2vec process is depicted in Figure 5.6.

5.8 Results: Domain Name Classification

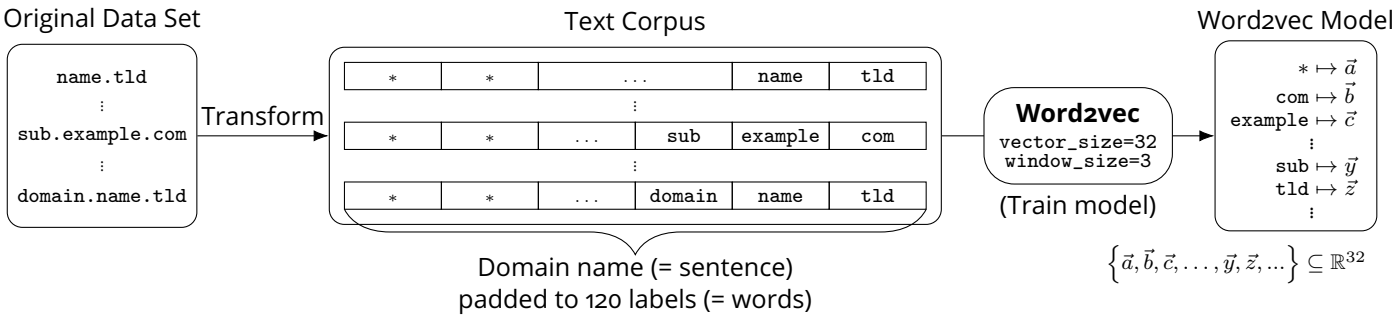
We train six machine learning models to classify IoT and non-IoT domain names. We use the following models and refer to them using the acronyms in parentheses: Naïve Bayes (NB), Logistic Regression (LR), K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Decision Tree (DT), and Random Forest (RF).

After preprocessing the data, The IoT dataset contains 3953 domain names. From the Cisco and Tranco datasets, we then pick 3953 domain names individually. We also create an additional list of 3953 domain names by uniformly sampling a Mix of the Cisco and Tranco datasets.

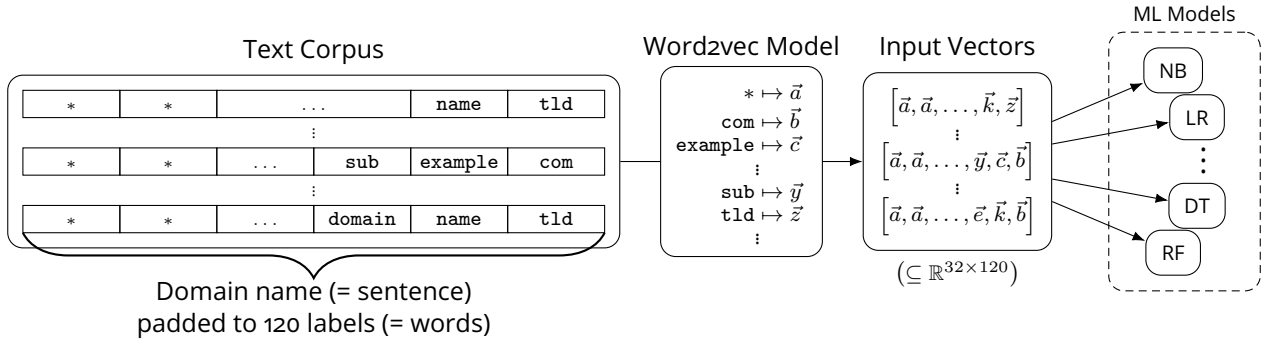
We select the 3953 domain names from the Cisco and Tranco datasets in two ways:

- We take the top 3953 domain names or
- randomly choose them, uniformly distributed, from the whole list.

After selecting 3953 domain names from each dataset, the domain names are labeled accordingly, and a combined dataset is constructed. The combined dataset is then processed via Word2vec to obtain the real-valued vector representation of each domain name. These real-valued vectors are used to train the machine learning models. The models are trained as binary classifiers between two classes, namely IoT and non-IoT domain names, where the non-IoT domain names come from the Cisco dataset, Tranco dataset, or a Mix of them. To evaluate the performance of each of the models, we calculate the resulting accuracy, precision, recall, and the F_1 score in subsection 5.8.1. Moreover, we perform in subsection 5.8.2 cross-validation to assess the robustness of the models and their ability to generalize to unseen data. Lastly, we perform in subsection 5.8.3 an ablation test to analyze the impact of the different labels of the domain names on the performance of the models.



(a) Step 1: Generate Word2vec model as label to real-valued vector mapping.



(b) Step 2: Use Word2vec model to generate input for machine learning models from text corpus.

Figure 5.6: Word embedding: After prepending `'*'` to each domain name until it has 120 labels, Word2vec is used to generate a real-valued vector representation of 32×120 real numbers of each domain name.

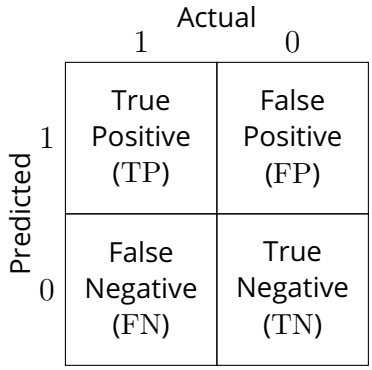


Figure 5.7: The 2×2 confusion matrix for binary classifiers.

5.8.1 Performance Evaluation

In this section, we present our results after training several machine learning models to classify IoT and non-IoT domain names. In each scenario, each dataset was processed with Word2vec to obtain the real-valued vector representation of each domain name of size 32×120. We used an 80-20 train-test split.

We train the machine learning models: NB, LR, KNN, SVM, DT, and RF. For each model, we calculate four parameters: Accuracy, precision, recall, and the F_1 score. We first calculate the confusion matrix to identify true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). See Figure 5.7. The

values for our parameters are then calculated using the following formulas:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{5.1}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5.2}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{5.3}$$

$$F_1 = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = \frac{\text{TP}}{\text{TP} + \frac{\text{FN}+\text{FP}}{2}} \tag{5.4}$$

Accuracy measures the ratio of positive predictions (TP and TN) to all the predictions made by the model, see Equation 5.1. Precision measures the ratio of true positive predictions (TP) to all the positive predictions (TP and FP) made by the model, see Equation 5.2. Recall measures the ratio of true positive predictions (TP) to all the actual positive instances in the dataset (TP and FN), see Equation 5.3. Finally, the F_1 score is the harmonic mean of precision and recall, see Equation 5.4.

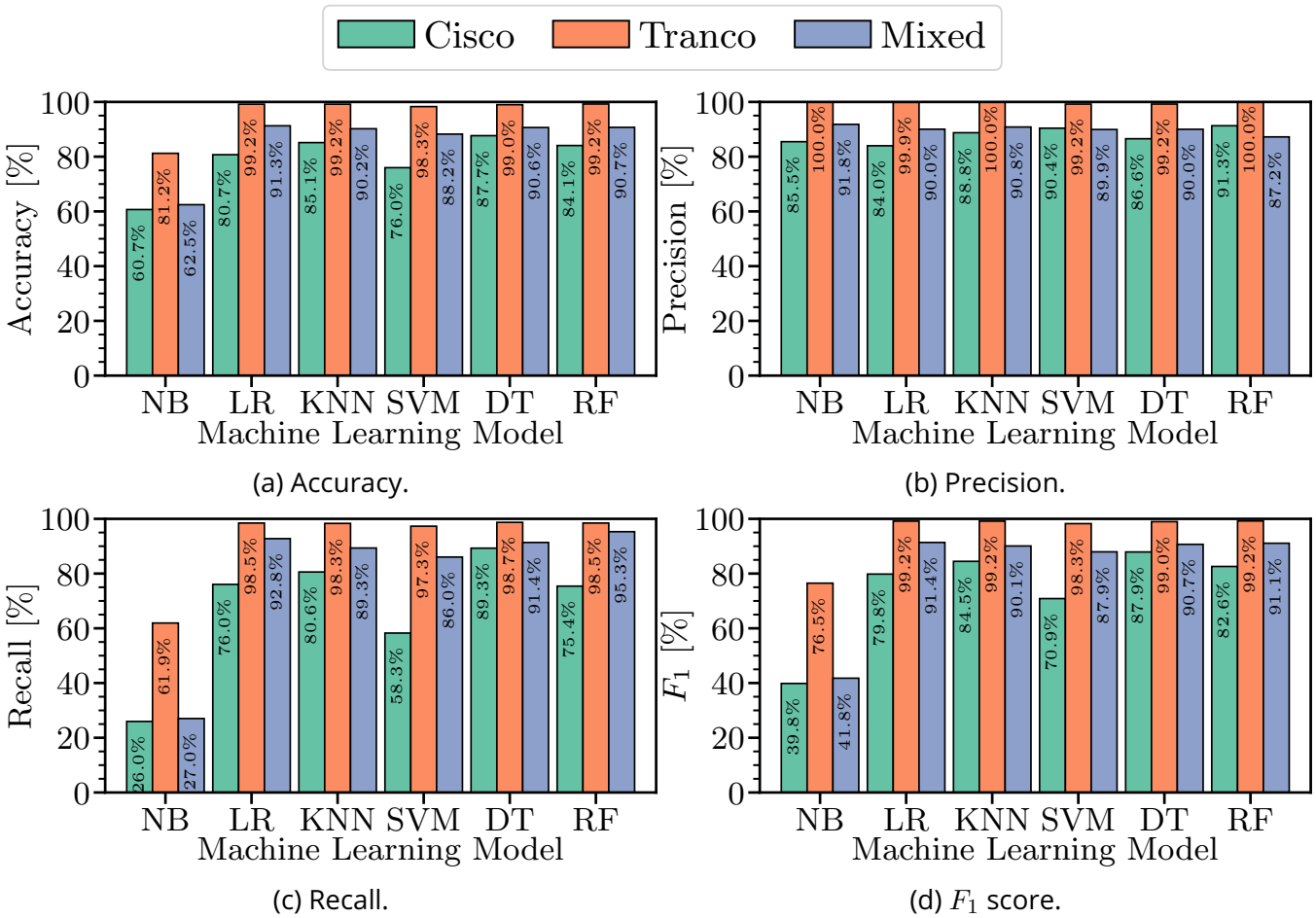
The results for using the top 3953 domain names can be seen in Figure 5.8. For the random selection of domain names, see Figure 5.10.

Results when using top domain names from the Cisco and Tranco datasets, and a Mix of the two datasets

The results of training the models using the top 3953 domain names from the Cisco and Tranco datasets are presented in Figure 5.8. Each graph represents one of the four parameters obtained by the different models. The six models we trained exhibited the strongest performance when the Tranco dataset was used. The lowest performing model when the Tranco dataset was used was NB, while the rest of the models achieved values between 97% and 100% for the four parameters. The lowest-performing model, regardless of the non-IoT dataset used, is NB. NB exhibited a random-classifier-like performance when the Cisco and Mix datasets were used and low recall and F_1 scores with the Tranco dataset while maintaining high precision. Among the best-performing models are DT and RF. Both models achieved close to 99% for the four parameters when the Tranco dataset was used. RF is usually preferred between the two models as DT tends to overfit and not perform as well when exposed to unseen data. The performance of the models is also visualized in Figure 5.9, which shows the Receiver Operating Characteristic (ROC) curves plotted for every model when the Cisco dataset is used. ROC curves show the performance of the models at different classification thresholds. The performance of the models can be compared by comparing the Area Under the Curve (AUC) of each one. In our case, the ROC curves in Figure 5.9 further show the superiority of RF compared to the other models where its AUC = 0.93. As expected from the previous measurements, NB has the lowest AUC of 0.69 and, therefore, has the lowest performance between the six models.

Results when using random domain names from the Cisco and Tranco datasets, and a Mix of the two datasets

The results of training the models using random 3953 domain names from the Cisco and Tranco datasets are presented in Figure 5.10. Each graph represents one of the four parameters obtained by the different models. We trained the six models by randomly choosing 3953 domain names from each list to generalize our results further. This is particularly interesting, as the Cisco and Tranco datasets each contain nearly one million domain names. For each list, 100 random picks of 3953 domain names were made, and the results presented are the average of each of the four parameters over the 100 random picks.



(a) Accuracy.

(b) Precision.

(c) Recall.

(d) F_1 score.

Figure 5.8: Accuracy, precision, recall, and F_1 score of each ML model for the top 3953 domain names from the Cisco and Tranco datasets, plus a uniformly sampled Mix of 3953 domain names from the two datasets, each vs. the 3953 domain names from the IoT dataset.

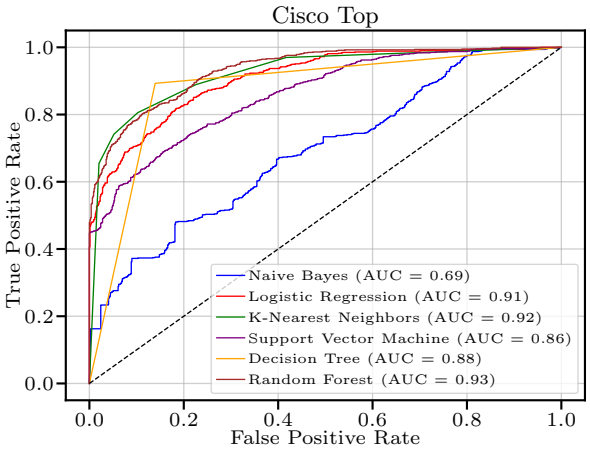
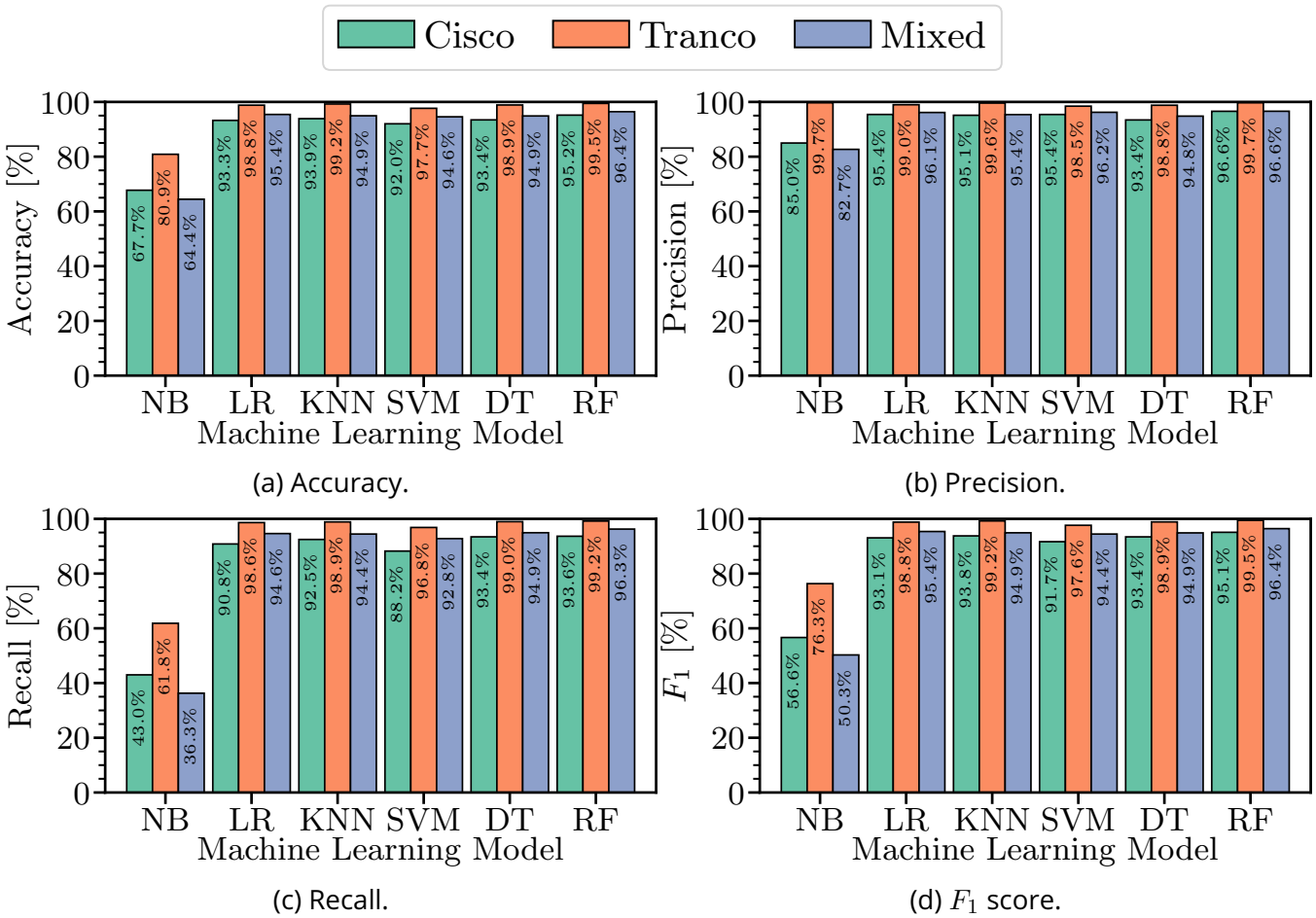


Figure 5.9: Receiver Operation Characteristic (ROC) Curves for the top 3953 domain names from the Cisco dataset. The Area Under the Curve (AUC) is provided in the legend.



(a) Accuracy.

(b) Precision.

(c) Recall.

(d) F_1 score.

Figure 5.10: Average accuracy, precision, recall and F_1 score of each ML model for 100 random picks of 3953 domain names from the Cisco and Tranco datasets, plus a uniformly sampled Mix of 3953 domain names from the two datasets, each vs. the 3953 IoT domain names.

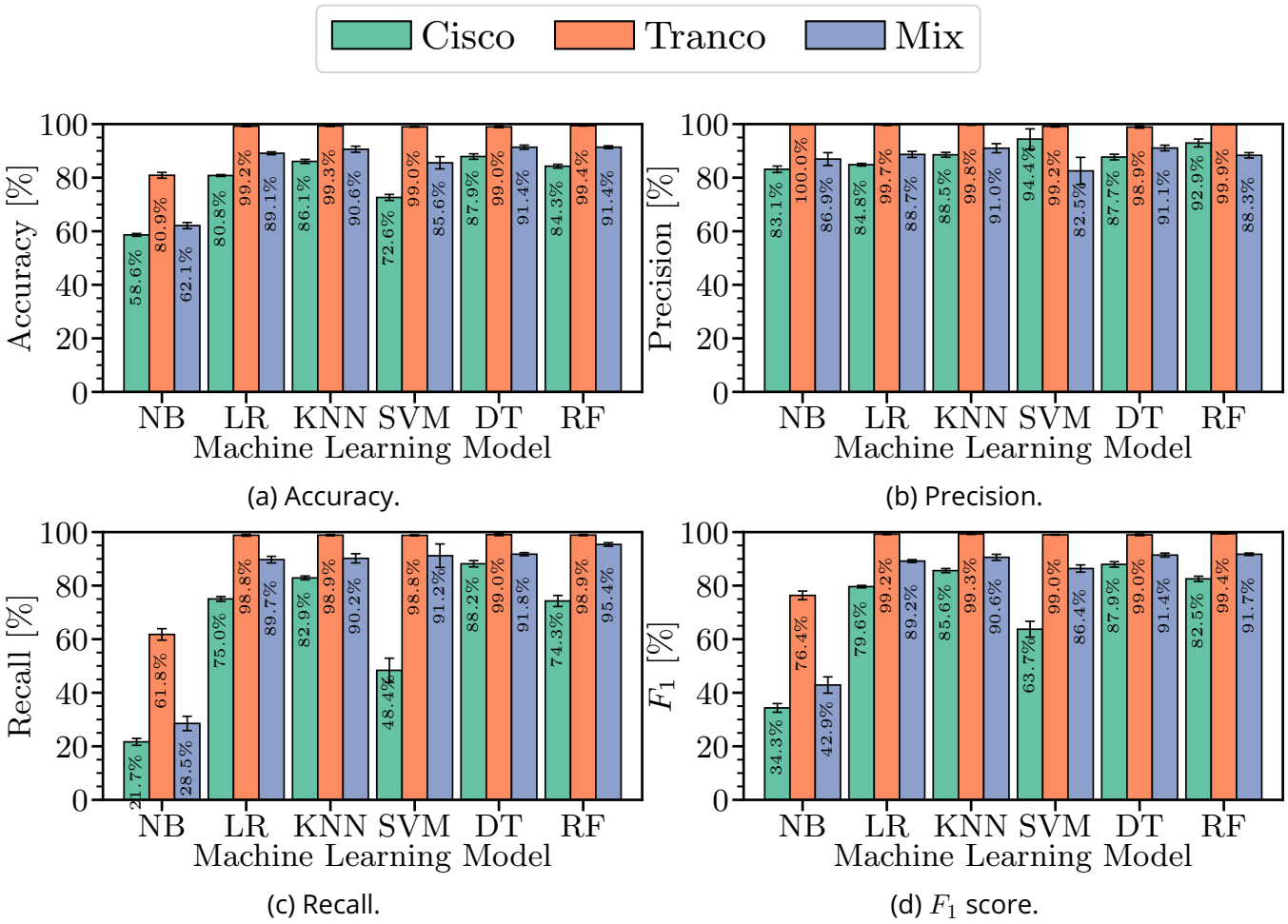
The results over the 100 random picks are consistent with the previous results obtained when using the top domain names of the Cisco and Tranco datasets. The models performed best when the Tranco dataset was used, with NB having the lowest performance and the rest of the models achieving values $> 90\%$ for the four parameters. The lowest-performing model, regardless of the dataset used, is still NB, particularly when the Cisco and Mix datasets are used as non-IoT datasets, where it exhibited a random-classifier-like performance. The best-performing model, regardless of the dataset used, is still RF.

5.8.2 Cross Validation

We use cross-validation to assess the robustness of the models and their ability to generalize to unseen data. When assessing a model using cross-validation, the dataset containing all the classes is divided into K folds or subsets, and the model is trained K times. One of the K folds is used as a testing dataset during every training instance, while the remaining $K - 1$ are used for training. We use Stratified K-fold cross-validation to ensure that the distribution of classes in the folds is similar to their distribution in the original dataset. Given the size of the IoT dataset, we used K = 5 to ensure that each fold contains enough entries to provide a reliable performance estimate. K = 5 allows each model to be trained five times. From the Cisco and Tranco datasets, we choose the top 3953 domain names, which are added to the 3953 domain

names of the IoT dataset. We show the results in Figure 5.11 as averages and standard deviation values of the evaluation parameters over the five folds.

The colored bars in Figure 5.11 represent the mean of the four parameters over the five folds, and the error bars at the top of each colored bar represent the standard deviation. We notice that the means of the four parameters over the five folds are consistent with the results from the performance evaluation we performed in Section 5.8.1 while having a low standard deviation, which indicates that the models are stable across the folds and that they are likely to generalize well to unseen data.



(a) Accuracy.

(b) Precision.

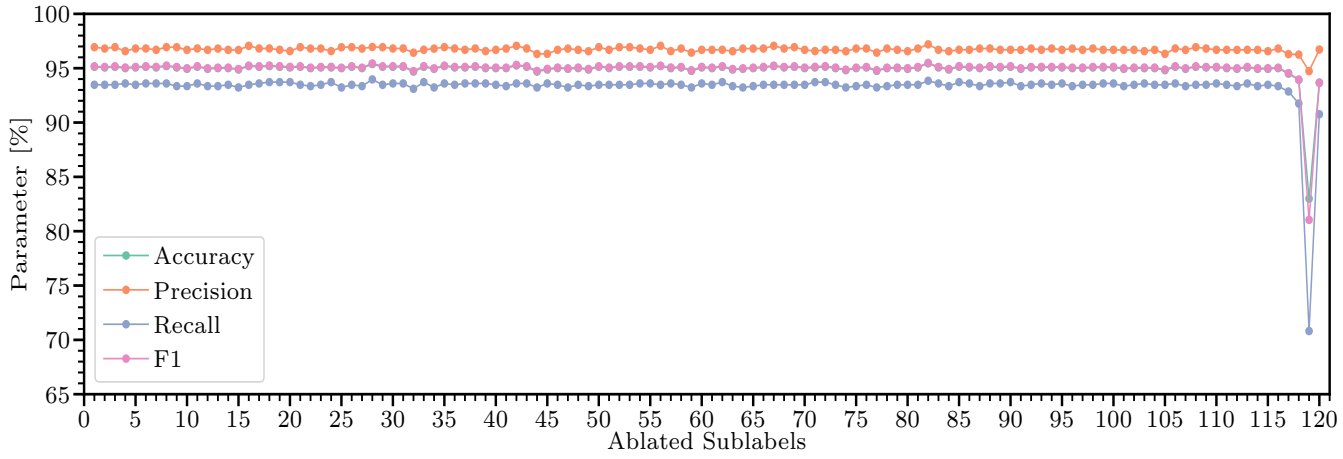
(c) Recall.

(d) F_1 score.

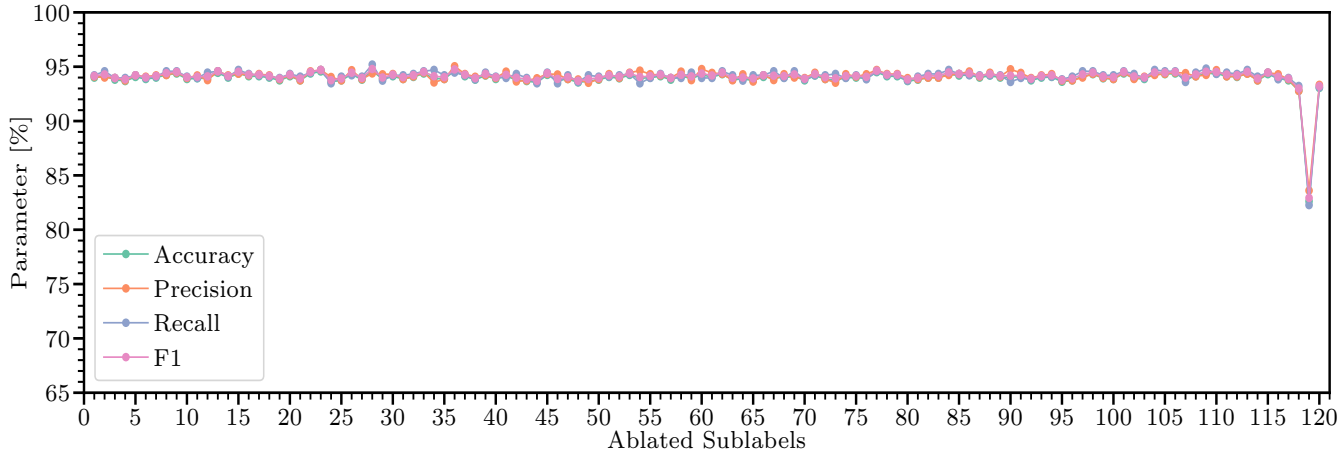
Figure 5.11: Mean (colored bars) and standard deviation (error bars) of accuracy, precision, recall and F_1 score of the ML models over five folds for the top 3953 domain names from the Cisco and Tranco datasets, plus a uniformly sampled Mix of 3953 domain names from the two datasets, vs. the 3953 IoT domain names.

5.8.3 Ablation Test

An ablation test includes removing elements from the ML model or suppressing a subset of the features to study their possible effect on performance. We perform the ablation test by removing one label at a time by replacing the 32-dimensional vector representing the label with zeros. Since we padded each domain name up to 120 labels, we perform 120 training and testing sessions, ablating one label in both the training



(a) Cisco.



(b) Tranco.

Figure 5.12: Ablation Test with Random Forest (RF).

and the testing datasets every time, training, and evaluating the models. The results for RF when used with the Cisco and Tranco datasets are presented in Figures 5.12a and 5.12b.

For both figures, the stable performance observed when ablating the dummy labels (`'*'`) demonstrates that the padding we added to the left of each domain name held no information and did not alter the performance of the models. The performance starts dropping as the last labels (starting from label 115) are ablated. The biggest drop in the performance is when label 119 is ablated. The second-to-last label, *i.e.*, the second-level domain (label 119), appears to have the highest impact on the performance as the values drastically dropped in both figures. For example, the accuracy, recall and F_1 score in Figure 5.12a dropped by around 15 and 25 percent, respectively. When the last label—the TLD of the domain name (label 120)—was ablated, however, values of the parameters did not experience the same drastic decrease, and the effect of ablating label 120 seemed almost equivalent to ablating the dummy labels. This shows that the second-level domain of a domain name is the most indicative of its class and that third-level domains and above gradually become less informing about the class of a domain name until ablating them becomes equivalent to ablating a dummy label. Moreover, the TLD of the domain names also seems not to provide significant information about the class of the domain name as ablating it (label 120) only slightly affected the performance.

5.9 Discussion

The size of the IoT dataset Despite the large amounts of raw data we started with, the size of the IoT dataset remained relatively modest. This is primarily due to the scope of our study, which mainly covers IoT devices that engage in machine-to-machine communications. Such devices exhibit limitations in their functionalities compared to generic devices and IoT devices that are not strictly machine-to-machine. This explains the low number of servers on the Internet these devices contact. Hence, we noticed a low number of frequently contacted servers instead of numerous servers that are less regularly or rarely contacted.

Using top-lists as non-IoT datasets We used two known lists of top-visited websites, namely the Cisco and Tranco datasets, to evaluate the validity of using such lists as negative classes in similar contexts. We noticed that the Tranco dataset is the least representative between the two of real non-IoT domain names, as the majority of domain names in it are second-level domains, which does not reflect how domain names actually appear in DNS traffic. The Cisco dataset, however, is suitable as its entries are not limited to second-level domains and are included in the dataset as seen in DNS traffic. The Cisco dataset also statistically resembles the domain names in the IoT dataset. The difference between the Cisco and Tranco datasets was most visible when training and testing the machine learning models. The Tranco dataset is easily distinguishable, so the models almost achieved perfect scores. The performance was different with the Cisco dataset which achieved a lower performance. The Cisco dataset is the better option for use as a negative class in domain name classification problems with similar contexts.

Better sources of data The domain names in the IoT dataset were extracted from packet captures of testbeds that had real devices. A better data source requires larger, more diverse testbeds with more IoT devices. On the other hand, changing the scope to include devices other than strictly M2M ones, would definitely enlarge and diversify the list of domain names.

5.10 Conclusion

In this chapter, we studied the domain names of IoT backend servers. We constructed a dataset of IoT domain names using 12 public datasets of domain names from past studies and used the two lists of top-visited websites Cisco and Tranco as non-IoT datasets. We conducted a three-phase comparative study between IoT and non-IoT domain names. Our results showed that IoT domain names usually contain indicative keywords related to the protocols or technologies used by the IoT servers. In addition, the DNS zones related to IoT displayed less dynamism and adoption of security measures compared to non-IoT domain names. The machine learning models we trained, on the other hand, were successful in classifying IoT and non-IoT domain names, with Random Forest having the best (overall) performance.

Chapter 6

Conclusion and Perspective

6.1 Summary of Contributions

In this thesis, we leveraged the Domain Name System (DNS) to address key challenges in IoT environments. The contributions presented here introduce an additional layer of security to IoT systems and lay the groundwork for promising future research. The following is a summary of these contributions.

The first contribution aimed to address two questions: What are the challenges facing IoT environments? And how is DNS being used by the research community and industry to address these challenges? The answers to these questions were the result of a comprehensive literature review, through which we identified four primary challenges faced by IoT environments. These challenges include the constrained nature of IoT devices, which have limited processing power, memory, and energy resources, and the identification of IoT devices, where diverse technologies in the IoT ecosystem often rely on proprietary identification schemes that are rarely compatible with one another. The third challenge relates to IoT security. Security is a critical concern in IoT because the constrained nature of IoT devices limits their ability to implement advanced security mechanisms, which significantly increases their attack surface. The fourth and final challenge is the lack of interoperability between IoT technologies. This issue arises directly from the absence of standardization, which has led to the development of numerous technologies, each with its own protocols, resulting in a vertically fragmented IoT ecosystem. As for addressing these challenges, the literature review revealed that DNS can help resolve some of them. For example, DNS protocols like DNS-over-CoAP (DoC), which are specifically designed for constrained IoT devices, enable such devices to resolve domain names and access services and resources on the web. In terms of IoT identification, DNS plays a major role through its core function of mapping domain names to IP addresses. Many IoT identification schemes, such as the Electronic Product Code (EPC) and Object Identifier (OID), rely on DNS for resolving their identifiers. IoT also benefits from DNS in terms of security. For example, through the use of DANE (DNS-based Authentication of Named Entities), DNS can act as a Certificate Authority (CA) for such devices. Lastly, DNS already serves as the basis for several systems designed to facilitate interoperability between IoT technologies.

The second contribution involved using DANE to implement a mutual authentication mechanism between two IoT backend servers. DANE gives the power back to domain name owners to decide how their digital certificate is validated and who gets to validate it, if needed. We used DANE to enable mutual authentication between two LoRaWAN backend servers, namely the *Network Server (NS)* and the *Join Server (JS)*, during the join process. However, in its original form, DANE only validates TLS server certificates and

does not consider that clients might have certificates, preventing the setup of a mutual authentication scheme. The DANE Authentication for Network Clients Everywhere (DANCE) IETF working group extends DANE to support the validation of client certificates as well. We implemented two Internet drafts issued by the DANCE working group and successfully established mutual authentication scheme. The evaluation of our solution showed that setting up a mutual authentication scheme via DANE introduces overhead due to the DNS resolution involved in validating certificates. However, this overhead can be mitigated by selecting a higher Time-to-Live (TTL) value for DNS records.

The third contribution is LoRaDANCE, a mechanism that enhances the security of LoRaWAN Over-the-Air Activation (OTAA) in two ways: it enables a LoRaWAN *End-Device (ED)* to join a network without the need to pre-share any secret keys, as required in standard LoRaWAN. Additionally, it allows the *ED* and the *Join Server (JS)* to perform a DANCE-inspired mutual authentication using DANE. We implemented LoRaDANCE and evaluated its performance. The *End-Device (ED)* and *Join Server (JS)* successfully performed mutual authentication, and the secret key, namely the *AppKey*, was derived on both devices using Elliptic Curve Diffie-Hellman (ECDH).We found that, while LoRaDANCE increases the average duration of the join process and consequently the power consumption, its security advantage justifies the trade-off.

The fourth contribution is a study on IoT backend servers, focusing on the domain names of these servers. IoT devices are rarely standalone. They typically need to contact backend servers to relay information, receive commands, or obtain firmware updates. These devices are generally equipped with the domain names of these servers, which they resolve via DNS. In this context, an IoT domain name refers to the domain name of a server contacted by IoT devices, specifically the domain name of an IoT backend server. For example, an IoT backend server could be a server that receives and stores footage from IP cameras. We compiled a list of real IoT domain names using datasets from previous studies that used testbeds with real IoT devices. Additionally, we used two public lists of top-visited websites as non-IoT domain names. Our study was carried out in three phases. The first phase is a statistical comparison between IoT and non-IoT domain names, based on their maximum, minimum, and average lengths (in characters) and the number of labels. This phase also included an analysis of the most frequent labels in each dataset. The second phase consisted of a DNS analysis. For each dataset, we resolved several RRs. For each dataset and each RR, we calculated the percentage of domain names having that RR, the average query duration, the average Time-to-Live (TTL), and the average response size. The third and final phase involved training several machine learning models to classify IoT and non-IoT domain names. The goal of this phase was to identify intrinsic differences between the two classes of domain names that cannot be detected visually or through statistical analysis.

6.2 Future Perspectives

6.2.1 Short-Term: Revisiting Contributions

In the short term, we plan to address any outstanding questions from our contributions, including resolving issues that were not fully addressed due to time constraints or remained ambiguous by the conclusion of our work. When implementing and testing solutions with real software and hardware, newer versions often emerge, promising improved performance. In some cases, it's not just about better tools, but rather a lack of knowledge at the time or logistical constraints that prevented us from exploring certain options during the initial implementation. Revisiting some of our contributions and testing them with the latest software and hardware, or re-evaluating them with a fresh perspective after completing the thesis could

provide valuable insights and potential improvements.

When we implemented the second contribution, *i.e.*, *Mutual Authentication of IoT Backend Servers*, the LoRaWAN *End-Device (ED)* we used was a MultiTech mDot, whereas for the third contribution, *i.e.*, *LoRaDANCE*, we used a NUCLEO-L476RG with an SX1276MB1MAS shield, along with the latest version of RIOT OS, the open-source operating system for constrained devices. It would be interesting to re-implement earlier solutions on potentially more efficient software and hardware. Furthermore, for both contributions, we used ChirpStack open-source LoRaWAN server v3, while ChirpStack v4 has been released for some time. ChirpStack v4 now uses Rust instead of Go, which was incompatible with the implementation of our second contribution, as it used Go. Therefore, it would be interesting to test our solutions with the latest version of ChirpStack to explore potential improvements.

In the fourth contribution, *IoT Backend Servers: Studying IoT Domain Names*, we used 12 datasets from previous studies that employed testbeds of real IoT devices to construct our IoT dataset, which had 4,145 domain names. Our objective is to increase both the size and diversity of this dataset, for which two options are available. The less costly option is to search for additional datasets, *i.e.*, datasets we may have overlooked or that have been released recently. While this option requires less effort, it is limiting as we rely on the availability and content provided by dataset owners. The alternative is to build our own testbed. Although more costly, and potentially a medium -or long-term- goal if financial and logistical constraints arise, this option offers greater flexibility in tailoring the datasets to our specific requirements.

6.2.2 Medium-Term: IoT-Friendly DNSSEC?

One of the major breakthroughs in securing DNS was the introduction of DNSSEC, which ensures the integrity of DNS responses received by resolvers. However, the main challenge with DNSSEC in IoT environments is the large size of the digital signatures. In our implementation of LoRaDANCE, despite using Algorithm 13, which generates relatively small signatures, and encoding the DNSSEC chain using CBOR (Concise Binary Object Representation), the resulting chain was still too large and required fragmentation. The receptions of these fragments at the device increased the power consumption considerably.

A medium-term goal would be to devise a DNSSEC-like protocol for constrained IoT environments. An appropriate first step would be to identify an efficient DNS protocol suited for IoT, such as DNS-over-CoAP (DoC) [138], which uses CoAP, a lightweight alternative to HTTP. In addition, exploring data compression and representation techniques, similar to CBOR, to compress DNS messages, could further optimize the protocol for constrained networks, though the compression of signatures may be limited.

6.2.3 Long-Term

Looking to the future, DNS is expected to become more integrated into IoT environments. A promising area of exploration involves developing a global naming system, similar to domain names, for constrained IoT devices that also preserves the privacy of IoT data.

Another potential research direction is the development of new DNS extensions specifically designed for IoT environments. These extensions could enable more efficient and secure resolution of device names.

Additionally, DNS could play a significant role in providing services such as location-based functionalities and personalized content delivery for IoT technologies. In the context of Social IoT (SIoT)[50], which envisions a human-centric IoT with increased interaction between humans and devices, DNS could facilitate connections between IoT devices and social media platforms, as well as enable autoconfiguration of device profiles.

Moreover, the integration of blockchain technologies offers enhanced security, transparency, and decentralization, which can greatly benefit IoT applications. On the one hand, Distributed Sensor Networks (DSNs) leveraging blockchain technologies could benefit from decentralized data storage, secure data transmission, transparency, and scalability. On the other hand, DNS could adopt blockchain technology to provide a secure, decentralized name resolution and registration system [196][168].

Another relevant research axis is the integration of edge computing. Leveraging edge computing can offload some of the DNS resolution tasks from constrained IoT devices. DNS queries can be processed at the edge, reducing latency and improving response times. This approach can also reduce the energy consumption of IoT devices since they do not need to communicate directly with a remote DNS server.

Furthermore, there is a crucial need to investigate Energy-Efficient DNS algorithms. Algorithms for DNS query processing and caching that minimize energy consumption and maximize battery life for constrained devices need to be explored. Hence, edge-Assisted caching can reduce the need for direct queries to external DNS servers. This minimizes energy consumption and network latency. In that scope, using ML models to predict DNS patterns and pre-fetch responses, ensuring the relevant DNS entries are cached before they are requested. This is especially relevant in scenarios where IoT devices generate repetitive queries.

In summary, novel DNS algorithms for constrained IoT devices need to incorporate techniques such as adaptive querying, caching, lightweight and energy-aware message processing. These approaches ensure that DNS operations do not drain battery life or introduce prohibitive latency, allowing IoT devices to function efficiently within the upcoming 6G ecosystem.

Publications

Conferences

[55] Ibrahim Ayoub, Gaël Berthaud-Müller, Sandoche Balakrichenan, Kinda Khawam, and Benoît Ampeau. Loradance: Using the dns for mutual authentication without pre-shared keys. In *2024 20th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2024. doi:10.23919/(Accepted)

[54] Ibrahim Ayoub, Gaël Berthaud-Müller, Sandoche Balakrichenan, Kinda Khawam, and Benoît Ampeau. The dns to reinforce the pkix for iot backend servers: Implementation and evaluation. In *2022 14th IFIP Wireless and Mobile Networking Conference (WMNC)*, pages 80–84, 2022. doi:10.23919/WMNC56391.2022.9954304

[58] Sandoche Balakrichenan, Ibrahim Ayoub, and Benoît Ampeau. Pki for iot using the dns infrastructure. In *2022 IEEE International Conference on Public Key Infrastructure and its Applications (PKIA)*, pages 1–8, 2022. doi:10.1109/PKIA56009.2022.9952253

Journals

[56] Ibrahim Ayoub, Martine S. Lenders, Benoît Ampeau, Sandoche Balakrichenan, Kinda Khawam, Thomas C. Schmidt, and Matthias Wählisch. What domain names is my device resolving? a study of iot backend servers. *IEEE Access*, 2024. (Submitted). doi:10.1109/ACCESS.2024.TBD

[53] Ibrahim Ayoub, Sandoche Balakrichenan, Kinda Khawam, and Benoît Ampeau. Dns for iot: A survey. *Sensors*, 23:4473, 05 2023. doi:10.3390/s23094473

Bibliography

[1] Anonymized dnscrypt. Accessed: 2023. URL: https://github.com/DNSCrypt/dnscrypt-protocol/blob/master/ANONYMIZED-DNSCRYPT.txt.

[2] Bluetooth® Low Energy (LE). Accessed: 2023. URL: https://www.bluetooth.com/learn-about-bluetooth/tech-overview/.

[3] Cloudflare public dns. Accessed: 2023. URL: https://www.cloudflare.com/dns/.

[4] Dane authentication for network clients everywhere (dance). URL: https://datatracker.ietf.org/wg/dance/about/.

[5] Ddos attack that disrupted internet was largest of its kind in history, experts say. Accessed: 2023. URL: https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet.

[6] Dns 2xl. Accessed: 2023. URL: https://labs.apnic.net/?p=1386.

[7] Dnscrypt. Accessed: 2023. URL: https://www.dnscrypt.org/.

[8] Dnscurve: Usable security for dns. Accessed: 2023. URL: https://dnscurve.org/.

[9] Doi handbook. Accessed: 2023. URL: https://www.doi.org/hb.html.

[10] Global iot market to grow to $1.5trn annual revenue by 2030. Accessed: 2023. URL: https://www.iot-now.com/2020/05/20/102937-global-iot-market-to-grow-to-1-5trn-annual-revenue-by-2030/.

[11] Google public dns. Accessed: 2023. URL: https://developers.google.com/speed/public-dns.

[12] Gs1 epc tag data standard. Accessed: 2023. URL: https://www.gs1.org/standards/rfid/tds.

[13] Handle.net registry. Accessed: 2023. URL: https://www.handle.net/.

[14] Internet of things (iot) total annual revenue worldwide from 2019 to 2030. Accessed: 2023. URL: https://www.statista.com/statistics/1194709/iot-revenue-worldwide/.

[15] Lora alliance®. Accessed: 2023. URL: https://lora-alliance.org/.

[16] Number of connected iot devices will surge to 125 billion by 2030. Accessed: 2023. URL: https://sst.semiconductor-digest.com/2017/10/number-of-connected-iot-devices-will-surge-to-125-billion-by-2030/.

[17] Number of internet of things (iot) connected devices worldwide from 2019 to 2030. Accessed: 2023. URL: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/.

[18] Object identifier (oid) repository. Accessed: 2023. URL: http://oid-info.com/.

[19] Object name service (ons). Accessed: 2023. URL: https://www.gs1.org/standards/epcis/epcis-ons/2-0-1.

[20] Onem2m. Accessed: 2023. URL: https://www.onem2m.org/.

[21] Sigfox. Accessed: 2023. URL: https://www.sigfox.com/en.

[22] Standardization of nb-iot completed. Accessed: 2023. URL: https://www.3gpp.org/news-events/1785-nb_iot_complete.

[23] Web of things (wot) architecture. Accessed: 2023. URL: https://www.w3.org/TR/wot-architecture/.

[24] Zigbee the full-stack solution for all smart devices. Accessed: 2023. URL: https://csa-iot.org/all-solutions/zigbee/.

[25] The Domain Naming Convention for Internet User Applications. RFC 819, August 1982. URL: https://www.rfc-editor.org/info/rfc819, doi:10.17487/RFC0819.

[26] Domain requirements. RFC 920, October 1984. URL: https://www.rfc-editor.org/info/rfc920, doi:10.17487/RFC0920.

[27] Domain names - concepts and facilities. RFC 1034, November 1987. URL: https://www.rfc-editor.org/info/rfc1034, doi:10.17487/RFC1034.

[28] Domain names - implementation and specification. RFC 1035, November 1987. URL: https://www.rfc-editor.org/info/rfc1035, doi:10.17487/RFC1035.

[29] Information technology – procedures for the operation of object identifier registration authorities: General procedures and top arcs of the international object identifier tree, 2011. ITU-T Recommendation X.660.

[30] Iso 26324:2012: Information and documentation — digital object identifier system, 2012. ISO 26324:2012 Standard.

[31] Iso/iec 9834-1:2012: Information technology — procedures for the operation of object identifier registration authorities: General procedures and top arcs of the international object identifier tree — part 1, 2012. ISO/IEC 9834-1:2012 Standard.

[32] Overview of the internet of things, 2012. ITU-T Recommendation Y.4000.

[33] Requirements and common characteristics of the iot identifier for the iot service, 2014. ITU-T Recommendation Y.4801.

[34] Afnic and The Swedish Internet Foundation. Zonemaster: Requirements and normalization of domain names in input. URL: https://github.com/zonemaster/zonemaster/blob/4ae8a6e/docs/specifications/tests/RequirementsAndNormalizationOfDomainNames.md.

[35] Bernhard Ager, Holger Dreger, and Anja Feldmann. Predicting the dnssec overhead using dns traces. In *2006 40th Annual Conference on Information Sciences and Systems*, pages 1484–1489, 2006. doi:10.1109/CISS.2006.286699.

[36] Sarah A. Al-Qaseemi, Hajer A. Almulhim, Maria F. Almulhim, and Saqib Rasool Chaudhry. Iot architecture challenges and issues: Lack of standardization. In *2016 Future Technologies Conference (FTC)*, pages 731–738, 2016. doi:10.1109/FTC.2016.7821686.

[37] LoRa Alliance. Lorawan 1.1 specification. Accessed on: September 2024. URL: https://resources.lora-alliance.org/technical-specifications/lorawan-specification-v1-1.

[38] Alliance for Internet of Things Innovation. Identifiers in internet of things (iot), 2018. URL: https://aioti.eu/wp-content/uploads/2018/03/AIOTI-Identifiers_in_IoT-1_0.pdf.pdf.

[39] Gianluca Aloi, Giancarlo Fortino, Raffaele Gravina, Pasquale Pace, and Claudio Savaglio. Simulation-driven platform for edge-based aal systems. *IEEE Journal on Selected Areas in Communications*, 39:446–462, 2021.

[40] Omar Alrawi, Chaz Lever, Manos Antonakakis, and Fabian Monrose. SoK: Security Evaluation of Home-Based IoT Deployments. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1362–1380, 2019. doi:10.1109/SP.2019.00013.

[41] Christian Amsüss and Martine Sophie Lenders. CoAP Transport Indication. Internet-Draft draft-ietf-core-transport-indication-06, Internet Engineering Task Force, July 2024. Work in Progress. URL: https://datatracker.ietf.org/doc/draft-ietf-core-transport-indication/06/.

[42] ANT Lab. 10-day Operantional IoT Traces, PREDICT ID: USC-LANDER/IoT_Operation_Traces-20200127. Provided by the USC/LANDER project. URL: http://www.isi.edu/ant/lander.

[43] ANT Lab. IoT devices' First-Time Bootup Traces, PREDICT ID: USC-LANDER/IoT_Bootup_Traces-20161207. Provided by the USC/LANDER project. URL: http://www.isi.edu/ant/lander.

[44] ANT Lab. IoT devices' First-Time Bootup Traces, PREDICT ID: USC-LANDER/IoT_Bootup_Traces-20181107. Provided by the USC/LANDER project. URL: http://www.isi.edu/ant/lander.

[45] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J. Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, Chaz Lever, Zane Ma, Joshua Mason, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas, and Yi Zhou. Understanding the Mirai Botnet. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1093–1110, Vancouver, BC, August 2017. USENIX Association.

[46] Apple. List of available trusted root certificates in ios 17, ipados 17, macos 14, tvos 17, and watchos 10, September 24, 2024. URL: https://support.apple.com/en-us/105116.

[47] Roy Arends, Geoffrey Sisson, David Blacka, and Ben Laurie. DNS Security (DNSSEC) Hashed Authenticated Denial of Existence. RFC 5155, March 2008. URL: https://www.rfc-editor.org/info/rfc5155, doi:10.17487/RFC5155.

[48] Kevin Ashton. That 'internet of things' thing. Accessed: 2023. URL: https://www.itrco.jp/libraries/RFIDjournal-That%20Internet%20of%20Things%20Thing.pdf.

[49] Mohammed Atiquzzaman, Mahda Noura, and Martin Gaedke. Interoperability in internet of things: Taxonomies and open challenges. *Mobile Networks and Applications*, 07 2018. doi:10.1007/s11036-018-1089-9.

[50] Luigi Atzori, Antonio Iera, Giacomo Morabito, and Michele Nitti. The social internet of things (siot) - when social networks meet the internet of things: Concept, architecture and network characterization. *Comput. Networks*, 56:3594–3608, 2012.

[51] Kaileshwar Aucklah, Avinash Mungur, Sheeba Armoogum, and Sameerchand Pudaruth. The impact of internet of things on the domain name system. In *2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 449–454, 2021. doi:10.1109/ICICCS51141.2021.9432217.

[52] Muhammad Ayaz, Mohammad Ammad-Uddin, Zubair Sharif, Ali Mansour, and El-Hadi M. Aggoune. Internet-of-things (iot)-based smart agriculture: Toward making the fields talk. *IEEE Access*, 7:129551–129583, 2019. doi:10.1109/ACCESS.2019.2932609.

[53] Ibrahim Ayoub, Sandoche Balakrichenan, Kinda Khawam, and Benoît Ampeau. Dns for iot: A survey. *Sensors*, 23:4473, 05 2023. doi:10.3390/s23094473.

[54] Ibrahim Ayoub, Gaël Berthaud-Müller, Sandoche Balakrichenan, Kinda Khawam, and Benoît Ampeau. The dns to reinforce the pkix for iot backend servers: Implementation and evaluation. In *2022 14th IFIP Wireless and Mobile Networking Conference (WMNC)*, pages 80–84, 2022. doi:10.23919/WMNC56391.2022.9954304.

[55] Ibrahim Ayoub, Gaël Berthaud-Müller, Sandoche Balakrichenan, Kinda Khawam, and Benoît Ampeau. Loradance: Using the dns for mutual authentication without pre-shared keys. In *2024 20th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2024. doi:10.23919/(Accepted).

[56] Ibrahim Ayoub, Martine S. Lenders, Benoît Ampeau, Sandoche Balakrichenan, Kinda Khawam, Thomas C. Schmidt, and Matthias Wählisch. What domain names is my device resolving? a study of iot backend servers. *IEEE Access*, 2024. (Submitted). doi:10.1109/ACCESS.2024.TBD.

[57] Emmanuel Baccelli, Cenk Gündoğan, Oliver Hahm, Peter Kietzmann, Martine S. Lenders, Hauke Petersen, Kaspar Schleiser, Thomas C. Schmidt, and Matthias Wählisch. Riot: An open source operating system for low-end embedded devices in the iot. *IEEE Internet of Things Journal*, 5(6):4428–4440, 2018. doi:10.1109/JIOT.2018.2815038.

[58] Sandoche Balakrichenan, Ibrahim Ayoub, and Benoît Ampeau. Pki for iot using the dns infrastructure. In *2022 IEEE International Conference on Public Key Infrastructure and its Applications (PKIA)*, pages 1–8, 2022. doi:10.1109/PKIA56009.2022.9952253.

[59] Sandoche Balakrichenan, Antoine Bernard, Michel Marot, and Benoit Ampeau. IoTRoam: design and implementation of an open LoRaWAN roaming architecture. In *IEEE Global Communications Conference (GLOBECOM)*, Madrid, Spain, December 2021. URL: https://hal.archives-ouvertes.fr/hal-03100628.

[60] Mohamed Ben-Daya, Elkafi Hassini, and Zied Bahroun. Internet of things and supply chain management: a literature review. *International Journal of Production Research*, 57:1–24, 11 2017. doi:10.1080/00207543.2017.1402140.

[61] Giulio Maria Bianco, Romeo Giuliano, Gaetano Marrocco, Franco Mazzenga, and Abraham Mejia-Aguilar. Lora system for search and rescue: Path-loss models and procedures in mountain scenarios. *IEEE Internet of Things Journal*, 8(3):1985–1999, 2021. doi:10.1109/JIOT.2020.3017044.

[62] Sharon Boeyen, Stefan Santesson, Tim Polk, Russ Housley, Stephen Farrell, and David Cooper. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280, May 2008. URL: https://www.rfc-editor.org/info/rfc5280, doi:10.17487/RFC5280.

[63] Carsten Bormann, Mehmet Ersue, and Ari Keränen. Terminology for Constrained-Node Networks. RFC 7228, May 2014. URL: https://www.rfc-editor.org/info/rfc7228, doi:10.17487/RFC7228.

[64] Stéphane Bortzmeyer, Ralph Dolmans, and Paul E. Hoffman. DNS Query Name Minimisation to Improve Privacy. RFC 9156, November 2021. URL: https://www.rfc-editor.org/info/rfc9156, doi:10.17487/RFC9156.

[65] L Breiman. Random Forests. *Machine Learning*, 45:5–32, 10 2001. doi:10.1023/A:1010950718922.

[66] Andrew Bulashenko, Stepan Piltyay, Alina Polishchuk, and Oleksandr Bulashenko. New traffic model of m2m technology in 5g wireless sensor networks. In *2020 IEEE 2nd International Conference on Advanced Trends in Information Theory (ATIT)*, pages 125–131, 2020. doi:10.1109/ATIT50783.2020.9349305.

[67] Andrei Butnaru, Alexios Mylonas, and Nikolaos Pitropakis. Towards Lightweight URL-Based Phishing Detection. *Future Internet*, 13(6):154, Jun 2021. doi:10.3390/fi13060154.

[68] Zhanghua Cai, Yantao Zhou, Yihong Qi, Weihua Zhuang, and Lei Deng. A millimeter wave dual-lens antenna for iot-based smart parking radar system. *IEEE Internet of Things Journal*, 8(1):418–427, 2021. doi:10.1109/JIOT.2020.3004403.

[69] Cagatay Catal, Görkem Giray, Bedir Tekinerdogan, Sandeep Kumar, and Suyash Shukla. Applications of deep learning for phishing detection: a systematic literature review. *Knowledge and Information Systems*, 64(6):1457–1500, June 2022. doi:10.1007/s10115-022-01672-x.

[70] Stuart Cheshire and Marc Krochmal. DNS-Based Service Discovery. RFC 6763, February 2013. URL: https://www.rfc-editor.org/info/rfc6763, doi:10.17487/RFC6763.

[71] Stuart Cheshire and Marc Krochmal. Multicast DNS. RFC 6762, February 2013. URL: https://www.rfc-editor.org/info/rfc6762, doi:10.17487/RFC6762.

[72] China Academy of Telecommunication Research (CATR)and Research Cluster on the Internet-of-Things (IERC). Eu-china joint white paper on internet-of-things identification, 2018. URL: https://iot6.eu/sites/default/files/imageblock/EU-China_IOT-ID-White-Paper-V1.0-Final.pdf.

[73] Chris Kemmerer. What is a root store?, 2019. URL: https://www.ssl.com/faqs/what-is-a-root-store/.

[74] Cisco Umbrella. Cisco Umbrella 1 Million. URL: https://s3-us-west-1.amazonaws.com/umbrella-static/index.html.

[75] Alissa Cooper, Hannes Tschofenig, Dr. Bernard D. Aboba, Jon Peterson, John Morris, Marit Hansen, and Rhys Smith. Privacy Considerations for Internet Protocols. RFC 6973, July 2013. URL: https://www.rfc-editor.org/info/rfc6973, doi:10.17487/RFC6973.

[76] Bill Corbitt. Do you "trust" me?, 2013. URL: https://www.intersecworldwide.com/blog/do-you-trust-me.

[77] Pádraig Cunningham and Sarah Jane Delany. K-Nearest Neighbour Classifiers - A Tutorial. *ACM Comput. Surv.*, 54(6), jul 2021. doi:10.1145/3459665.

[78] Hong-Ning Dai, Zibin Zheng, and Yan Zhang. Blockchain for internet of things: A survey. *IEEE Internet of Things Journal*, 6(5):8076–8094, 2019. doi:10.1109/JIOT.2019.2920987.

[79] Soumya Kanti Datta and Christian Bonnet. Advances in web of things for iot interoperability. In *2018 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW)*, pages 1–2, 2018. doi:10.1109/ICCE-China.2018.8448890.

[80] Wouter de Vries, Quirin Scheitle, Moritz Müller, Willem Toorop, Ralph Dolmans, and Roland Rijswijk-Deij. A first look at qname minimization in the domain name system. pages 147–160, 03 2019. doi:10.1007/978-3-030-15986-3_10.

[81] Anthony S. Deese, Joe Jesson, Thomas Brennan, Steven Hollain, Patrick Stefanacci, Emily Driscoll, Connor Dick, Keith Garcia, Ryan Mosher, Brian Rentsch, Andrew Bechtel, and Efrain Rodriguez. Long-term monitoring of smart city assets via internet of things and low-power wide-area networks. *IEEE Internet of Things Journal*, 8(1):222–231, 2021. doi:10.1109/JIOT.2020.3005830.

[82] Swati Dhingra, Rajasekhara Babu Madda, Amir H. Gandomi, Rizwan Patan, and Mahmoud Daneshmand. Internet of things mobile–air pollution monitoring system (iot-mobair). *IEEE Internet of Things Journal*, 6(3):5577–5584, 2019. doi:10.1109/JIOT.2019.2903821.

[83] 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 *Proceedings of the 2011 Seventh European Conference on Computer Network Defense*, EC2ND '11, page 9–16, USA, 2011. IEEE Computer Society. doi:10.1109/EC2ND.2011.16.

[84] Duo Ding, Minbo Li, and Zhu Zhu. Object naming service supporting heterogeneous object code identification for iot system. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, volume 01, page 545–554, 7 2018. doi:10.1109/COMPSAC.2018.00084.

[85] Duo Ding, Minbo Li, and Zhu Zhu. Object naming service supporting heterogeneous object code identification for iot system. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, volume 01, pages 545–554, 2018. doi:10.1109/COMPSAC.2018.00084.

[86] Yi Ding, Guozheng Wu, Dajiang Chen, Ning Zhang, Linpeng Gong, Mingsheng Cao, and Zhiguang Qin. DeepEDN: A deep-learning-based image encryption and decryption network for internet of medical things. *IEEE Internet of Things Journal*, 8(3):1504–1518, feb 2021. URL: https://doi.org/10.1109%2Fjiot.2020.3012452, doi:10.1109/jiot.2020.3012452.

[87] Badis Djamaa and Mark Richardson. Towards scalable dns-based service discovery for the internet of things. Lecture Notes in Computer Science, page 432–435, 2014. doi:10.1007/978-3-319-13102-3_70.

[88] Nguyet Quang Do, Ali Selamat, Ondrej Krejcar, Enrique Herrera-Viedma, and Hamido Fujita. Deep Learning for Phishing Detection: Taxonomy, Current Challenges and Future Directions. *IEEE Access*, 10:36429–36463, 2022. doi:10.1109/ACCESS.2022.3151903.

[89] Viktor Dukhovni and Wes Hardaker. The DNS-Based Authentication of Named Entities (DANE) Protocol: Updates and Operational Guidance. RFC 7671, October 2015. URL: https://www.rfc-editor.org/info/rfc7671, doi:10.17487/RFC7671.

[90] Dmitry Elkin and Valeriy Vyatkin. *IoT in Traffic Management: Review of Existing Methods of Road Traffic Regulation*, pages 536–551. 08 2020. doi:10.1007/978-3-030-51974-2_50.

[91] European Union Agency for Cybersecurity (ENISA). Operation black tulip: Certificate authorities lose authority, 2011. URL: https://www.enisa.europa.eu/media/news-items/operation-black-tulip/.

[92] Simon Fernandez, Michele Amoretti, Fabrizio Restori, Maciej Korczynski, and Andrzej Duda. Semantic identifiers and DNS names for IoT. In *2021 International Conference on Computer Communications and Networks (ICCCN)*. IEEE, jul 2021. URL: https://doi.org/10.1109%2Ficccn52240.2021.9522285, doi:10.1109/icccn52240.2021.9522285.

[93] Mohamed Amine Ferrag, Othmane Friha, Djallel Hamouda, Leandros Maglaras, and Helge Janicke. Edge-IIoTset: A New Comprehensive Realistic Cyber Security Dataset of IoT and IIoT Applications: Centralized and Federated Learning, 2022. doi:10.21227/mbc1-1h68.

[94] Maria Ganzha, Marcin Paprzycki, Wieslaw Pawlowski, Paweł Szmeja, and Katarzyna Wasielewska. *Towards Semantic Interoperability Between Internet of Things Platforms*, pages 103–127. 07 2018. doi:10.1007/978-3-319-61300-0_6.

[95] Oscar Garcia-Morchon, Sandeep Kumar, and Mohit Sethi. Internet of Things (IoT) Security: State of the Art and Challenges. RFC 8576, April 2019. URL: https://www.rfc-editor.org/info/rfc8576, doi:10.17487/RFC8576.

[96] Ali Ghubaish, Tara Salman, Maede Zolanvari, Devrim Unal, Abdulla Al-Ali, and Raj Jain. Recent advances in the internet-of-medical-things (iomt) systems security. *IEEE Internet of Things Journal*, 8(11):8707–8718, 2021. doi:10.1109/JIOT.2020.3045653.

[97] GlobalSign. Globalsign ssl certificates, September 24, 2024. URL: https://shop.globalsign.com/en/ssl.

[98] Pradyumna Gokhale, Omkar Bhat, and Sagar Bhat. Introduction to iot. 5:41–44, 01 2018. doi:10.17148/IARJSET.2018.517.

[99] Google Security Blog. An update on attempted man-in-the-middle attacks, 2011. URL: https://security.googleblog.com/2011/08/update-on-attempted-man-in-middle.html.

[100] Amelie Gyrard, Soumya Kanti Datta, and Christian Bonnet. A survey and analysis of ontology-based software tools for semantic interoperability in iot and wot landscapes. In *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, pages 86–91, 2018. doi:10.1109/WF-IoT.2018.8355091.

[101] Ayyoob Hamza, Hassan Habibi Gharakheili, Theophilus A. Benson, and Vijay Sivaraman. Detecting Volumetric Attacks on LoT Devices via SDN-Based Monitoring of MUD Activity. In *Proceedings of the 2019 ACM Symposium on SDN Research*, SOSR '19, page 36–48, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3314148.3314352.

[102] Almira Hamzic and Isabel Olofsson. DNS and the Internet of Things: Outlining the challenges faced by DNS in the Internet of Things. Master's thesis, KTH Royal Institute of Technology, Sweden, 2016.

[103] Hua Han, Wenjin Ma, MengChu Zhou, Qiang Guo, and Abdullah Abusorrah. A novel semi-supervised learning approach to pedestrian reidentification. *IEEE Internet of Things Journal*, 8(4):3042–3052, 2021. doi:10.1109/JIOT.2020.3024287.

[104] Jiliang Han, Na Lin, Junhu Ruan, Xuping Wang, Wei Wei, and Huimin Lu. A model for joint planning of production and distribution of fresh produce in agricultural internet of things. *IEEE Internet of Things Journal*, 8(12):9683–9696, 2021. doi:10.1109/JIOT.2020.3037729.

[105] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009. doi:10.1007/978-0-387-21606-5.

[106] Marti Hearst, S.T. Dumais, E. Osman, John Platt, and B. Scholkopf. Support vector machines. *Intelligent Systems and their Applications, IEEE*, 13:18 – 28, 08 1998. doi:10.1109/5254.708428.

[107] Cristian Hesselman, Merike Kaeo, Lyman Chapin, Kimberly Claffy, Mark Seiden, Danny McPherson, Dave Piscitello, Andrew McConachie, Tim April, Jacques Latour, and Rod Rasmussen. The dns in iot: Opportunities, risks, and challenges. *IEEE Internet Computing*, 24(4):23–32, 2020. doi:10.1109/MIC.2020.3005388.

[108] Bernie Hoeneisen and Alexander Mayrhofer. ENUM Validation Architecture. RFC 4725, November 2006. URL: https://www.rfc-editor.org/info/rfc4725, doi:10.17487/RFC4725.

[109] Paul E. Hoffman. Cryptographic Algorithm Identifier Allocation for DNSSEC. RFC 6014, November 2010. URL: https://www.rfc-editor.org/info/rfc6014, doi:10.17487/RFC6014.

[110] Paul E. Hoffman and Patrick McManus. DNS Queries over HTTPS (DoH). RFC 8484, October 2018. URL: https://www.rfc-editor.org/info/rfc8484, doi:10.17487/RFC8484.

[111] Paul E. Hoffman and Jakob Schlyter. The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA. RFC 6698, August 2012. URL: https://www.rfc-editor.org/info/rfc6698, doi:10.17487/RFC6698.

[112] Zi Hu, Liang Zhu, John Heidemann, Allison Mankin, Duane Wessels, and Paul E. Hoffman. Specification for DNS over Transport Layer Security (TLS). RFC 7858, May 2016. URL: https://www.rfc-editor.org/info/rfc7858, doi:10.17487/RFC7858.

[113] Christian Huitema and Eric Rescorla. Issues and Requirements for Server Name Identification (SNI) Encryption in TLS. RFC 8744, July 2020. URL: https://www.rfc-editor.org/info/rfc8744, doi:10.17487/RFC8744.

[114] Shumon Huque and Viktor Dukhovni. TLS Client Authentication via DANE TLSA records. Internet-Draft draft-ietf-dance-client-auth-05, Internet Engineering Task Force, January 2024. Work in Progress. URL: https://datatracker.ietf.org/doc/draft-ietf-dance-client-auth/05/.

[115] Shumon Huque and Viktor Dukhovni. TLS Extension for DANE Client Identity. Internet-Draft draft-ietf-dance-tls-clientid-03, Internet Engineering Task Force, January 2024. Work in Progress. URL: https://datatracker.ietf.org/doc/draft-ietf-dance-tls-clientid/03/.

[116] Thien Huynh-The, Cam-Hao Hua, Nguyen Anh Tu, and Dong-Seong Kim. Physical activity recognition with statistical-deep fusion model using multiple sensory data for smart health. *IEEE Internet of Things Journal*, 8(3):1533–1543, 2021. doi:10.1109/JIOT.2020.3013272.

[117] IANA. List of top-level domains. Accessed: 2024. URL: https://data.iana.org/TLD/tlds-alpha-by-domain.txt.

[118] IANA. Root servers. Accessed: 2024. URL: https://www.iana.org/domains/root/servers.

[119] Kevin Igoe, David McGrew, and Margaret Salter. Fundamental Elliptic Curve Cryptography Algorithms. RFC 6090, February 2011. URL: https://www.rfc-editor.org/info/rfc6090, doi:10.17487/RFC6090.

[120] Yacine Izza, Alexey Ignatiev, and Joao Marques-Silva. On Explaining Decision Trees, 2020. arXiv:2010.11034.

[121] Jaime Jimenez, Hannes Tschofenig, and Dave Thaler. Report from the Internet of Things (IoT) Semantic Interoperability (IOTSI) Workshop 2016. RFC 8477, October 2018. URL: https://www.rfc-editor.org/info/rfc8477, doi:10.17487/RFC8477.

[122] Simon Josefsson. Storing Certificates in the Domain Name System (DNS). RFC 4398, March 2006. URL: https://www.rfc-editor.org/info/rfc4398, doi:10.17487/RFC4398.

[123] Anca Jurcut, Pasika Ranaweera, and Lina Xu. *Introduction to IoT Security*, pages 1–39. 12 2019. doi:10.1002/9781119471509.w5GRef260.

[124] Latika Kakkar, Gupta Deepali, Sapna Saxena, and Sarvesh Tanwar. *IoT Architectures and Its Security: A Review*, pages 87–94. 01 2021. doi:10.1007/978-981-15-9689-6_10.

[125] Andreas Kamilaris, Koula Papakonstantinou, and Andreas Pitsillides. Exploring the use of dns as a search engine for the web of things. 03 2014. doi:10.1109/WF-IoT.2014.6803128.

[126] Hyunjae Kang, Dong Hyun Ahn, Gyung Min Lee, Jeong Do Yoo, Kyung Ho Park, and Huy Kang Kim. IoT network intrusion dataset, 2019. doi:10.21227/q70p-q449.

[127] Matthieu Kanj, Vincent Savaux, and Mathieu Le Guen. A tutorial on nb-iot physical layer design. *IEEE Communications Surveys Tutorials*, 22(4):2408–2446, 2020. doi:10.1109/COMST.2020.3022751.

[128] Bill Karakostas. A dns architecture for the internet of things: A case study in transport logistics. *Procedia Computer Science*, 19:594–601, 12 2013. doi:10.1016/j.procs.2013.06.079.

[129] Tae Hyun Kim, Douglas Reeves, and Rudra Dutta. Advanced secure dns name autoconfiguration with authentication for enterprise iot network. In *2021 IEEE Global Communications Conference (GLOBECOM)*, pages 01–06, 2021. doi:10.1109/GLOBECOM46510.2021.9685237.

[130] Nikhil Kumar, Debopam Acharya, and Divya Lohani. An iot-based vehicle accident detection and classification system using sensor fusion. *IEEE Internet of Things Journal*, 8(2):869–880, 2021. doi:10.1109/JIOT.2020.3008896.

[131] Larry Lannom, Lt. Col. Brian P. Boesch, and Sam Sun. Handle System Overview. RFC 3650, November 2003. URL: https://www.rfc-editor.org/info/rfc3650, doi:10.17487/RFC3650.

[132] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. Tranco: A research-oriented top sites ranking hardened against manipulation. In *Proceedings of the 26th Annual Network and Distributed System Security Symposium*, NDSS 2019, February 2019. doi:10.14722/ndss.2019.23386.

[133] Euijong Lee, Young-Duk Seo, Se-Ra Oh, and Young-Gab Kim. A survey on standards for interoperability and security in the internet of things. *IEEE Communications Surveys Tutorials*, 23(2):1020–1047, 2021. doi:10.1109/COMST.2021.3067354.

[134] Keuntae Lee, Hyungsuk Kang, Jaehoon Paul Jeong, Hyoungshick Kim, and Jung-Soo Park. Secure dns name autoconfiguration for ipv6 internet-of-things devices. In *2016 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 564–569, 2016. doi:10.1109/ICTC.2016.7763534.

[135] Keuntae Lee, Seokhwa Kim, and Jaehoon Paul Jeong. Dnsnav4: Dns name autoconfiguration for internet-of-things devices in ipv4 networks. In *2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pages 347–351, 2017. doi:10.1109/WAINA.2017.117.

[136] Sejun Lee, Jaehoon Jeong, and Jungsoo Park. Dns name autoconfiguration for iot home devices. In *2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops*, pages 131–134, 2015. doi:10.1109/WAINA.2015.104.

[137] Sejun Lee, Jaehoon Paul Jeong, and Jung-Soo Park. Dnsna: Dns name autoconfiguration for internet of things devices. In *2016 18th International Conference on Advanced Communication Technology (ICACT)*, pages 410–416, 2016. doi:10.1109/ICACT.2016.7423412.

[138] Martine S. Lenders, Christian Amsüss, Cenk Gündogan, Marcin Nawrocki, Thomas C. Schmidt, and Matthias Wählisch. Securing name resolution in the iot: Dns over coap. *Proc. ACM Netw.*, 1(CoNEXT2), September 2023. doi:10.1145/3609423.

[139] Martine Sophie Lenders, Christian Amsüss, Cenk Gündoğan, Thomas C. Schmidt, and Matthias Wählisch. DNS over CoAP (DoC). Internet-Draft draft-ietf-core-dns-over-coap-07, Internet Engineering Task Force, June 2024. Work in Progress. URL: https://datatracker.ietf.org/doc/draft-ietf-core-dns-over-coap/07/.

[140] Martine Sophie Lenders, Carsten Bormann, Thomas C. Schmidt, and Matthias Wählisch. A Concise Binary Object Representation (CBOR) of DNS Messages. Internet-Draft draft-lenders-dns-cbor-07, Internet Engineering Task Force, May 2024. Work in Progress. URL: https://datatracker.ietf.org/doc/draft-lenders-dns-cbor/07/.

[141] Wanting Li, Jian Jin, and Jong-Hyouk Lee. Analysis of botnet domain names for iot cybersecurity. *IEEE Access*, 7:94658–94665, 2019. doi:10.1109/ACCESS.2019.2927355.

[142] LoRa Alliance. TS001-1.0.4 LoRaWAN® L2 1.0.4 Specification. Accessed on: February 2024. URL: https://resources.lora-alliance.org/technical-specifications/ts001-1-0-4-lorawan-l2-1-0-4-specification.

[143] Zhihan Lv, Liang Qiao, Jinhua Li, and Houbing Song. Deep-learning-enabled security issues in the internet of things. *IEEE Internet of Things Journal*, 8(12):9531–9538, 2021. doi:10.1109/JIOT.2020.3007130.

[144] Gang Mei, Nengxiong Xu, Jiayu Qin, Bowen Wang, and Pian Qi. A survey of internet of things (iot) for geohazard prevention: Applications, technologies, and challenges. *IEEE Internet of Things Journal*, 7(5):4371–4386, 2020. doi:10.1109/JIOT.2019.2952593.

[145] Markus Miettinen, Samuel Marchal, Ibbad Hafeez, N. Asokan, Ahmad-Reza Sadeghi, and Sasu Tarkoma. IoT SENTINEL: Automated device-type identification for security enforcement in IoT. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, jun 2017. doi:10.1109/icdcs.2017.283.

[146] Tomas Mikolov, Kai Chen, G.s Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. *Proceedings of Workshop at ICLR*, 2013, 01 2013. doi:10.48550/arXiv.1301.3781.

[147] Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. Exploiting Similarities among Languages for Machine Translation, 2013. doi:10.48550/arXiv.1309.4168.

[148] Stefano Milani and Ioannis Chatzigiannakis. Design, analysis, and experimental evaluation of a new secure rejoin mechanism for lorawan using elliptic-curve cryptography. *Journal of Sensor and Actuator Networks*, 10(2), 2021. URL: https://www.mdpi.com/2224-2708/10/2/36, doi:10.3390/jsan10020036.

[149] Daniel Minoli and Benedict Occhiogrosso. Blockchain mechanisms for iot security. *Internet of Things*, 1-2:1–13, 09 2018. doi:10.1016/j.iot.2018.05.002.

[150] Nivedita Mishra and Sharnil Pandya. Internet of things applications, security challenges, attacks, intrusion detection, and future visions: A systematic review. *IEEE Access*, 9:59353–59377, 2021. doi:10.1109/ACCESS.2021.3073408.

[151] Bhabendu Kumar Mohanta, Debasish Jena, Somula Ramasubbareddy, Mahmoud Daneshmand, and Amir H. Gandomi. Addressing security and privacy issues of iot using blockchain technology. *IEEE Internet of Things Journal*, 8(2):881–888, 2021. doi:10.1109/JIOT.2020.3008906.

[152] Gabriel Montenegro, Christian Schumacher, and Nandakishore Kushalnagar. IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals. RFC 4919, August 2007. URL: https://www.rfc-editor.org/info/rfc4919, doi:10.17487/RFC4919.

[153] Mozilla. Ca certificates in firefox, September 24, 2024. URL: https://ccadb.my.salesforce-sites.com/mozilla/CACertificatesInFirefoxReport.

[154] Fredrik Mårlind and Ismail Butun. Activation of lorawan end devices by using public key cryptography. In *2020 4th Cyber Security in Networking Conference (CSNet)*, pages 1–8, 2020. doi:10.1109/CSNet50428.2020.9265530.

[155] Soulakshmee Devi Nagowah, Hatem Ben Sta, and Baby Ashwin Gobin-Rahimbux. An overview of semantic interoperability ontologies and frameworks for iot. In *2018 Sixth International Conference on Enterprise Systems (ES)*, pages 82–89, 2018. doi:10.1109/ES.2018.00020.

[156] Dr. Thomas Narten, Tatsuya Jinmei, and Dr. Susan Thomson. IPv6 Stateless Address Autoconfiguration. RFC 4862, September 2007. URL: https://www.rfc-editor.org/info/rfc4862, doi:10.17487/RFC4862.

[157] Musa Ndiaye, Stephen S. Oyewobi, Adnan M. Abu-Mahfouz, Gerhard P. Hancke, Anish M. Kurien, and Karim Djouani. Iot in the wake of covid-19: A survey on contributions, challenges and evolution. *IEEE Access*, 8:186821–186839, 2020. doi:10.1109/ACCESS.2020.3030090.

[158] Nataliia Neshenko, Elias Bou-Harb, Jorge Crichigno, Georges Kaddoum, and Nasir Ghani. Demystifying iot security: An exhaustive survey on iot vulnerabilities and a first empirical look on internet-scale iot exploitations. *IEEE Communications Surveys Tutorials*, 21(3):2702–2733, 2019. doi:10.1109/COMST.2019.2910750.

[159] Muhammad Nouman, Sui Yang Khoo, M. A. Parvez Mahmud, and Abbas Z. Kouzani. Recent advances in contactless sensing technologies for mental health monitoring. *IEEE Internet of Things Journal*, 9(1):274–297, 2022. doi:10.1109/JIOT.2021.3097801.

[160] Oscar Novo and Mario Francesco. Semantic interoperability in the iot: Extending the web of things architecture. *ACM Transactions on Internet of Things*, 1:1–25, 03 2020. doi:10.1145/3375838.

[161] Muhammad Talha Paracha, Daniel J. Dubois, Narseo Vallina-Rodriguez, and David Choffnes. IoTLS: Understanding TLS Usage in Consumer IoT Devices. In *Proceedings of the 21st ACM Internet Measurement Conference*, IMC '21, page 165–178, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3487552.3487830.

[162] Roberto Perdisci, Thomas Papastergiou, Omar Alrawi, and Manos Antonakakis. Iotfinder: Efficient large-scale identification of iot devices via passive dns traffic analysis. In *2020 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 474–489, 2020. doi:10.1109/EuroSP48549.2020.00037.

[163] Stepan Piltyay, Andrew Bulashenko, and Ivan Demchenko. Wireless sensor network connectivity in heterogeneous 5g mobile systems. In *2020 IEEE International Conference on Problems of Infocommunications. Science and Technology (PIC S&T)*, pages 625–630, 2020. doi:10.1109/PICST51311.2020.9468073.

[164] Diego G.S. Pivoto, Luiz F.F. de Almeida, Rodrigo da Rosa Righi, Joel J.P.C. Rodrigues, Alexandre Baratella Lugli, and Antonio M. Alberti. Cyber-physical systems architectures for industrial internet of things applications in industry 4.0: A literature review. *Journal of Manufacturing Systems*, 58:176–192, 2021. URL: https://www.sciencedirect.com/science/article/pii/S0278612520302119, doi:10.1016/j.jmsy.2020.11.017.

[165] Yanchen Qiao, Bin Zhang, Weizhe Zhang, Arun Kumar, and Hualong Wu. DGA Domain Name Classification Method Based on Long Short-Term Memory with Attention Mechanism. *Applied Sciences*, 9:4205, 10 2019. doi:10.3390/app9204205.

[166] Abdur Rahman, M. Shamim Hossain, Nabil A. Alrajeh, and Fawaz Alsolami. Adversarial examples—security threats to covid-19 deep learning systems in medical iot devices. *IEEE Internet of Things Journal*, 8(12):9603–9610, 2021. doi:10.1109/JIOT.2020.3013710.

[167] Hafizur Rahman and Md. Iftekhar Hussain. A comprehensive survey on semantic interoperability for internet of things: State-of-the-art and research challenges. *Transactions on Emerging Telecommunications Technologies*, 31(12):e3902, 2020. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.3902, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/ett.3902, doi:10.1002/ett.3902.

[168] Balaji Rajendran, Gopinath Palaniappan, Dijesh R, Bindhumadhava Bapu S, and Sudarsan S D. A universal domain name resolution service – need and challenges - study on blockchain based naming services. In *2022 IEEE Region 10 Symposium (TENSYMP)*, pages 1–6, 2022. doi:10.1109/TENSYMP54529.2022.9864361.

[169] Routhu Rao, Tatti Vaishnavi, and Alwyn Pais. CatchPhish: detection of phishing websites by inspecting URLs. *Journal of Ambient Intelligence and Humanized Computing*, 11, 02 2020. doi:10.1007/s12652-019-01311-4.

[170] Sebastian Raschka. Naive Bayes and Text Classification I - Introduction and Theory, 2017. arXiv:1410.5329.

[171] Nafiul Rashid, Manik Dautta, Peter Tseng, and Mohammad Abdullah Al Faruque. Hear: Fog-enabled energy-aware online human eating activity recognition. *IEEE Internet of Things Journal*, 8(2):860–868, 2021. doi:10.1109/JIOT.2020.3008842.

[172] Tirumaleswar Reddy.K, Dan Wing, and Prashanth Patil. DNS over Datagram Transport Layer Security (DTLS). RFC 8094, February 2017. URL: https://www.rfc-editor.org/info/rfc8094, doi:10.17487/RFC8094.

[173] Jingjing Ren, Daniel J. Dubois, David Choffnes, Anna Maria Mandalari, Roman Kolcun, and Hamed Haddadi. Information Exposure for Consumer IoT Devices: A Multidimensional, Network-Informed Measurement Approach. In *Proc. of the Internet Measurement Conference (IMC)*, 2019. doi:10.1145/3355369.3355577.

[174] Scott Rose, Matt Larson, Dan Massey, Rob Austein, and Roy Arends. DNS Security Introduction and Requirements. RFC 4033, March 2005. URL: https://www.rfc-editor.org/info/rfc4033, doi:10.17487/RFC4033.

[175] Scott Rose, Matt Larson, Dan Massey, Rob Austein, and Roy Arends. Protocol Modifications for the DNS Security Extensions. RFC 4035, March 2005. URL: https://www.rfc-editor.org/info/rfc4035, doi:10.17487/RFC4035.

[176] Scott Rose, Matt Larson, Dan Massey, Rob Austein, and Roy Arends. Resource Records for the DNS Security Extensions. RFC 4034, March 2005. URL: https://www.rfc-editor.org/info/rfc4034, doi:10.17487/RFC4034.

[177] Said Jawad Saidi, Srdjan Matic, Oliver Gasser, Georgios Smaragdakis, and Anja Feldmann. Deep dive into the iot backend ecosystem. In *Proceedings of the 22nd ACM Internet Measurement Conference*, IMC '22, page 488–503, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3517745.3561431.

[178] Eryk Schiller, Andy Aidoo, Jara Fuhrer, Jonathan Stahl, Michael Ziörjen, and Burkhard Stiller. Landscape of iot security. *Computer Science Review*, 44, 05 2022. doi:10.1016/j.cosrev.2022.100467.

[179] Giovanni Schmid. Thirty years of dns insecurity: Current issues and perspectives. *IEEE Communications Surveys Tutorials*, 23(4):2429–2459, 2021. doi:10.1109/COMST.2021.3105741.

[180] Paul Schmitt, Anne Edmundson, Allison Mankin, and Nick Feamster. Oblivious dns: Practical privacy for dns queries. *Proceedings on Privacy Enhancing Technologies*, 2019:228–244, 04 2019. doi:10.2478/popets-2019-0028.

[181] Benjamin M. Schwartz, Mike Bishop, and Erik Nygren. Service Binding and Parameter Specification via the DNS (SVCB and HTTPS Resource Records). RFC 9460, November 2023. URL: https://www.rfc-editor.org/info/rfc9460, doi:10.17487/RFC9460.

[182] Zach Shelby, Klaus Hartke, and Carsten Bormann. The Constrained Application Protocol (CoAP). RFC 7252, June 2014. URL: https://www.rfc-editor.org/info/rfc7252, doi:10.17487/RFC7252.

[183] Ravi Pratap Singh, Mohd Javaid, Abid Haleem, and Rajiv Suman. Internet of things (iot) applications to fight against covid-19 pandemic. *Diabetes & Metabolic Syndrome: Clinical Research & Reviews*, 14(4):521–524, 2020. URL: https://www.sciencedirect.com/science/article/pii/S1871402120301065, doi:10.1016/j.dsx.2020.04.041.

[184] Arunan Sivanathan, Hassan Habibi Gharakheili, Franco Loi, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics. *IEEE Transactions on Mobile Computing*, 18(8):1745–1759, 2019. doi:10.1109/TMC.2018.2866249.

[185] Jeffrey Spaulding and David Mohaisen. Defending internet of things against malicious domain names using d-fens. pages 387–392, 10 2018. doi:10.1109/SEC.2018.00051.

[186] Sriram Srinivasan, Vinayakumar Ravi, Ajay Arunachalam, Mamoun Alazab, and Soman Kp. *DURLD: Malicious URL Detection Using Deep Learning-Based Character Level Representations*, pages 535–554. Springer, 01 2021. doi:10.1007/978-3-030-62582-5_21.

[187] Milosh Stolikj, Pieter J. L. Cuijpers, Johan J. Lukkien, and Nina Buchina. Context based service discovery in unmanaged networks using mdns/dns-sd. In *2016 IEEE International Conference on Consumer Electronics (ICCE)*, page 163–165, 1 2016. doi:10.1109/ICCE.2016.7430565.

[188] Milosh Stolikj, Richard Verhoeven, Pieter J. L. Cuijpers, and Johan J. Lukkien. Proxy support for service discovery using mdns/dns-sd in low power networks. In *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, pages 1–6, 2014. doi:10.1109/WoWMoM.2014.6918925.

[189] Pascal Thubert and Jonathan Hui. Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks. RFC 6282, September 2011. URL: https://www.rfc-editor.org/info/rfc6282, doi:10.17487/RFC6282.

[190] Tranco. Tranco. URL: ttps://tranco-list.eu/list/K273W.

[191] Soe Tun, Samaneh Madanian, and Farhaan Mirza. Internet of things (iot) applications for elderly care: a reflective review. *Aging Clinical and Experimental Research*, 33, 04 2021. doi:10.1007/s40520-020-01545-9.

[192] Aashma Uprety and Danda B Rawat. Reinforcement learning for iot security: A comprehensive survey. *IEEE Internet of Things Journal*, PP:1–1, 11 2020. doi:10.1109/JIOT.2020.3040957.

[193] Thales L. von Sperling, Francisco L. de Caldas Filho, Rafael Timóteo de Sousa, Lucas M. C. e Martins, and Rodrigo L. Rocha. Tracking intruders in iot networks by means of dns traffic analysis. In *2017 Workshop on Communication Networks and Power Systems (WCNPS)*, pages 1–4, 2017. doi:10.1109/WCNPS.2017.8252938.

[194] Tim Wicinski. DNS Privacy Considerations. RFC 9076, July 2021. URL: https://www.rfc-editor.org/info/rfc9076, doi:10.17487/RFC9076.

[195] Jonathan Woodbridge, Hyrum Anderson, Anjum Ahuja, and Daniel Grant. Predicting Domain Generation Algorithms with Long Short-Term Memory Networks. *arXiv preprint arXiv:1611.00791*, 11 2016. doi:10.48550/arXiv.1611.00791.

[196] Pengcheng Xia, Haoyu Wang, Zhou Yu, Xinyu Liu, Xiapu Luo, Guoai Xu, and Gareth Tyson. Challenges in decentralized name management: The case of ens. In *Proceedings of the 22nd ACM Internet Measurement Conference*, IMC '22, page 65–82, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3517745.3561469.

[197] Zhiwei Yan, Ning Kong, Ye Tian, and Yong-Jin Park. A universal object name resolution scheme for iot. In *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, pages 1120–1124, 2013. doi:10.1109/GreenCom-iThings-CPSCom.2013.193.

[198] Zhiwei Yan, Hongtao Li, Sherali Zeadally, Yu Zeng, and Guanggang Geng. Is dns ready for ubiquitous internet of things? *IEEE Access*, 7:28835–28846, 2019. doi:10.1109/ACCESS.2019.2901801.

[199] Luhui Yang, Guangjie Liu, Jinwei Wang, Huiwen Bai, Jiangtao Zhai, and Yuewei Dai. Fast3DS: A real-time full-convolutional malicious domain name detection system. *Journal of Information Security and Applications*, 61:102933, 2021. doi:10.1016/j.jisa.2021.102933.

[200] Koen Zandberg, Kaspar Schleiser, Francisco Acosta, Hannes Tschofenig, and Emmanuel Baccelli. Secure firmware updates for constrained iot devices using open standards: A reality check. *IEEE Access*, 7:71907–71920, 2019. doi:10.1109/ACCESS.2019.2919760.

[201] Guijuan Zhang, Dianjie Lu, and Hong Liu. Iot-based positive emotional contagion for crowd evacuation. *IEEE Internet of Things Journal*, 8(2):1057–1070, 2021. doi:10.1109/JIOT.2020.3009715.

[202] Ólafur Guðmundsson. Adding Acronyms to Simplify Conversations about DNS-Based Authentication of Named Entities (DANE). RFC 7218, April 2014. URL: https://www.rfc-editor.org/info/rfc7218, doi:10.17487/RFC7218.