# PKI for IoT using the DNS infrastructure

Sandoche Balakrichenan, Benoit Ampeau, Ibrahim Ayoub

# PKI for IoT using the DNS infrastructure

1st Sandoche BALAKRICHENAN
*Afnic*
*sandoche.balakrichenan@afnic.fr*

2nd Ibrahim Ayoub
*Afnic*
*ibrahim.ayoub@afnic.fr*

3rd Benoît Ampeau
*Afnic*
*benoit.ampeau@afnic.fr*

***Abstract*—The main challenge facing IoT today is security. The constrained nature of IoT devices deprives them of using modern security solutions. This leads them to use aged and more vulnerable security mechanisms that expose them to high risks. When compared to more powerful devices, constrained IoT devices cannot use the Public Key Infrastructure with X.509 certificates to establish secure sessions. Moreover, the idea of self-signed certificates and having one trusted CA does not seem that popular. The Domain Name System (DNS) using the DNS-based Authentication of Named Entities protocol (DANE) and DNS's security extensions (DNSSEC) can help create the sought-after Public Key Infrastructure (PKI) for IoT. With a concrete example, this article explains how DNS can deliver IoT PKI functions based on DANE, backed by DNSSEC.**

***Index Terms*—DNS, PKI, DNSSEC, DANE, LoRaWAN**

## I. Introduction

Internet of Things (IoT) is becoming one of the most important market segments and the target of all ICT key players. IoT security mechanisms are still in their infancy and have failed to keep up with the technology's rapid growth. The increasing role of IoT devices in security breaches raises the issue of enforcing proper security on IoT devices. High-profile attacks such as the one by the Mirai botnet that exploited the vulnerabilities of IoT devices illustrate the importance of having security solutions.

Similar to the Internet, security mechanisms in the IoT should ensure confidentiality, integrity, privacy and availability of the services offered. The IoT landscape has different networking topologies : the prominent ones being mesh, point-to-point and star topology. In this article, we will focus on the security issues in the star topology where all IoT devices are connected to a central gateway. Communication between the IoT devices and the Cloud servers in the Internet is made through this central gateway.

In the scope of the star topology, the most common threat vectors include: physical attacks on the IoT devices such as tampering with the cryptographic keys; compromising the cloud servers in the Internet to which the IoT device connects to via the gateway; man-in-the-middle attacks, where malicious actors eavesdrop and possibly alter the communication between the source and the destination; exploiting vulnerabilities in IoT devices to organize a Distributed Denial of Service (DDoS) attacks.

In the Internet, security is enforced by the Public Key Infrastructure (PKI) through digital certificates. PKI enables secure authentication and communication between Internet devices (such as home computers and server) without needing tokens, password policies or other cumbersome user-initiated factors. But the PKI infrastructure cannot be directly applied in the current state in the IoT due to various challenges.

Current security mechanisms in IoT are based on proprietary closed solutions, which translates to an increased cost for end-users and hinders the possibility of secure communication between different security solution providers. A ***second*** challenge is that most low-end IoT devices are highly constrained: they have little memory, limited processing capacity and power. The ***third*** challenge is to provide open authentication support and trusted anchors with scalable key distribution required to secure communication channels like in the current Internet so that IoT devices can bootstrap application-specific security mechanisms. The ***fourth*** challenge is bootstrapping trust.

The traditional use of PKI does not fit constrained IoT devices: since it required sufficient computing power, storage for the chain of trust and sufficient bandwidth for sending and receiving certificates, encrypted data using large block ciphers and signatures, as well as obtaining revocation lists, are technically and economically infeasible for this class of devices. This article explains the concept of replacing the trust and security schemes based on the traditional PKI with a novel approach that relies on the DNS (Domain Naming Service) [1] [2] infrastructure and builds all the required functionalities upon DNS. DNS brings the advantage of a single trust anchor with lightweight authentication schemes suitable for constrained IoT devices and easily automated for large-scale IoT deployments.

In this article, we start with the challenges of IoT security State of the art description in II, followed by a detailed illustration of how the Internet communication security is complemented using the DNS-based PKI in III and finally conclude on how the DNS based PKI can be applied to one of the most constrained IoT networks in V.

## II. State of the Art – Challenges of IoT security

A recent NIST report [3] recommends to address cybersecurity and privacy risks for IoT devices with three high-level mitigation goals:

- protect device security (prevent a device from conducting attacks)
- protect data security (guarantee confidentiality, integrity and/or availability of data)
- protect privacy (prevent disclosure of personally identifiable information)

This work was supported by the French National Research Agency (ANR) - PIVOT (ANR-20-CYAL-0002).

Most low-end IoT devices are highly constrained: they have limited memory, processing capacity, and power. Managing Public Key security mechanisms as deployed in the Internet on such devices and transferring them over bandwidth-constrained IoT networks is too heavy and therefore represents a significant challenge.

Bootstrapping trust when an IoT device connects to the network and starts to operate is a security concern. The device is usually equipped with an identifier and a Pre-Shared Key (PSK) to contact some servers on the Internet associated with the IoT device for onboarding. PSK needs to be shared between stakeholders in the supply chain—from the OEM (Original Equipment Manufacturer) to the device owner, the network service provider, the application server provider, etc. PSK is often shared insecurely, such as printing the keys on the back of devices, sending via mail or printing on the invoice. There have been reports of massive breaches of PSK provisioning systems, which are vulnerable to passive pervasive monitoring. There are secure ways in which PSK could be shared, but they rely on proprietary cloud services or secure key elements, which could increase the cost of IoT services.

Most of the existing IoT security and data protection solutions are not generic. Proposed approaches will have to balance improving security, trust, and privacy and allow scope for innovation and evolution within the market. Open security standards like those defined for the Internet should be adapted for IoT to satisfy the resource-constrained requirements, thus reducing costs. Finally, the provided solutions should be immediately deployable into legacy or new IoT technologies.

Current security mechanisms in IoT are based on proprietary closed solutions, which translates to an increased cost for end-users and businesses. Weak IoT security has its roots in economic factors because of the tension between cost and security objectives. Including adequate security and privacy in IoT costs money and slows the product development process. In addition, security requires specialized skills and experience that manufacturers may not have at hand, requiring either new stable or external consulting, both of which increase costs. The proposed IoT security solutions should not incur additional costs hindering innovation and evolution in the IoT market.

The essential aspect of IoT concerns heterogeneous types of devices and communication networks with different requirements, thus creating closed independent silos and leading to interoperability issues. Past efforts to create a single architecture to provide security, privacy and trust in the heterogeneous IoT ecosystem did not succeed.

## III. Key Management in the Internet

In web browsing communication, ***first*** (Fig. 1), the browser obtains the IP address of a domain name using the DNS infrastructure and ***secondly*** it connects to the domain's web server using the IP address via Hyper Text Transfer Protocol (HTTP) connection.

For the first operation, securing the communication during DNS resolution could be provided by DNS Security

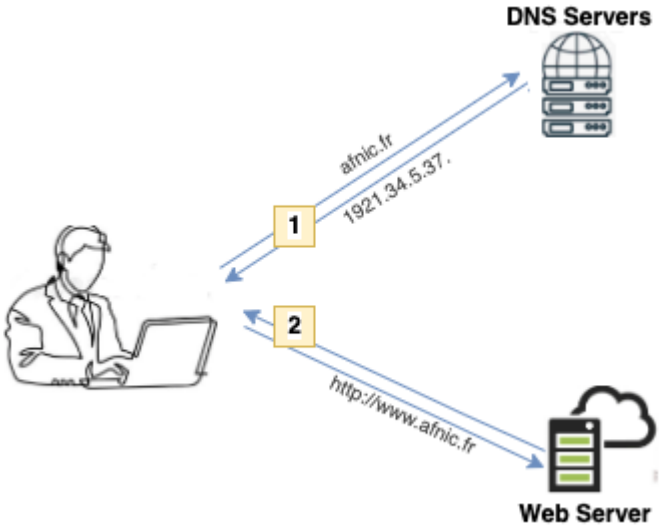


Figure 1. Internet communication without any security

(DNSSEC) [4] described in subsection III-B. For the second operation, the Transport Layer Security (TLS) protocol comes to the rescue, allowing the client and the server to authenticate each other and negotiate an encryption algorithm and cryptographic keys before the data is exchanged. TLS ensures that data cannot be tampered with during transit since the data is encrypted. Details of the second operation is explained in subsection III-A

### A. Public-Key Infrastructure X.509 (PKIX)

Encrypting and decrypting the data in the TLS protocol is done by a matching pair of cryptographic keys: ***public*** and ***private key***. The Data encrypted by a public key can be decrypted only by the corresponding private key and vice versa, enabling secure communication with unknown users.

A website (e.g., a bank) publishes its public key for anyone to download. An account holder in the bank, Alice, encrypts a message using the public key and sends it to the bank. Only the bank can decrypt the message using its private key. Thus, Alice is sure that her message is accessed only by the bank and not by anyone else.

On the other hand, there is a possibility that an impersonator publishes their public key posing as Alice's bank. Alice will encrypt the message using the public key and send it to the impersonator, thinking she is communicating with her bank. The impersonator could do a man in the middle and copy the message. As the impersonator is the owner of the public and the private key, it will enable them to decrypt and read the message sent by Alice. Thus, there arises a possibility that anyone can create a public key for accessing any domain name.

Hence, binding between the identity (e.g., the domain name) and the public key is necessary. The X.509 standard [5] proposed by the ITU and ISO provides a mechanism to bind a particular public key to a specific identity. The domain holder can do this binding, and in that case, it is called a

self-signed certificate. If the self-signed certificate is obtained from a trusted source by the application using the certificate for authentication, then it is accepted. Otherwise there is no guarantee of the certificate's authenticity.

*1) The Certification Authorities (CAs) role:* This is where the need for a ***trusted third party*** arises. It is just like the passport wherein it can be issued only by a trusted third party delegated by a country's government. The trusted authority attests that the person in the photo is identified by a particular name, surname and other credentials.

In web browsing, a ***digital certificate*** issued for a domain name, is like the passport. In the PKIX ecosystem, the role of the trusted authority is played by organizations called CAs. A certificate issued by a given CA binds the given domain name with information such as the certificate assignee, the entity which has requested the certificate, its validity period etc. The CA is used to assert to the browser that the web server represents the domain asked by the client. A TLS connection is established between the browser and the web server on successful authentication of the certificate. With the TLS connection established, the traffic between the browser and the web server is encrypted and is protected against any third-party eavesdropping.

Similar to how a passport attested by one country's trusted authority is accepted by other countries as a validated document for authenticating a person, browser vendors (such as Firefox, Chrome, Internet explorer, Safari etc.,) accept digital certificates created only by certain CAs. The browser vendors authorize an organization to be a CA only after understanding that they are trustworthy and they follow strict principles and procedures to provide certificates only for correct domain holders. Once the browser vendors authorize an organization to be a CA, the latter's digital certificate is added to the list of trusted CAs in the browser library. Thus, once a client using a browser accesses a domain name which has a digital certificate generated by one of the CAs among its pre-installed list, the certificate is implicitly trusted, as shown in Fig. 2.

### *B. Using DNS infrastructure and its security extensions as PKI*

PKIX provides data origin authentication, confidentiality and data integrity. The DNS resolution, as illustrated in Fig. 1 doesn't have such security features. DNSSEC enables the possibility to provide origin authentication of DNS data, data integrity and authenticated denial of existence for the DNS resolution.

In PKIX, data origin authentication is provided by the digital certificate. TLS provides confidentiality and data integrity. Data is encrypted only during transmission, and the data stored in the server end is not encrypted. In the case of DNSSEC, the data at the server end, i.e. in the DNS zone, is encrypted using the public and private keys. Hence data in the DNS zone is encrypted before being transmitted in the network. In the PKIX model at the client end, an HTTP user agent such as a web browser is used for validation. In DNSSEC, a validating resolver is used to verify the integrity of the transmitted data.

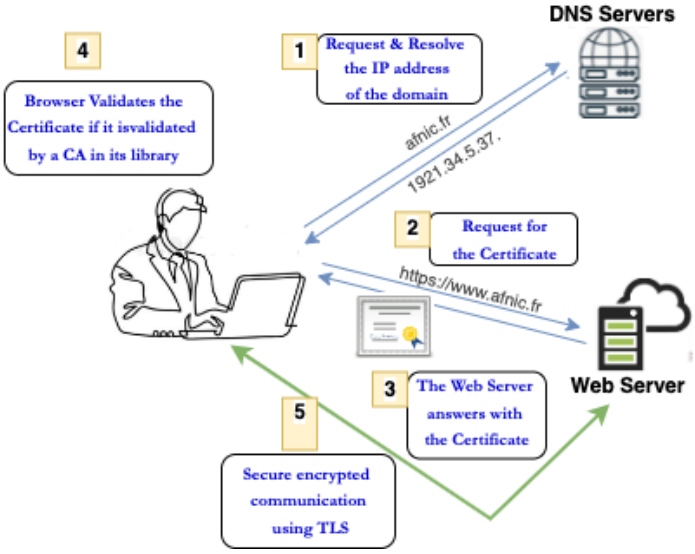


Figure 2. Secured Communication between the browser and the web server using the PKIX ecosystem and TLS

*1) A DNSSEC primer:* Since DNSSEC is an extension of DNS which acts upon a DNS zone, it is important to have some basic idea of a DNS zone.

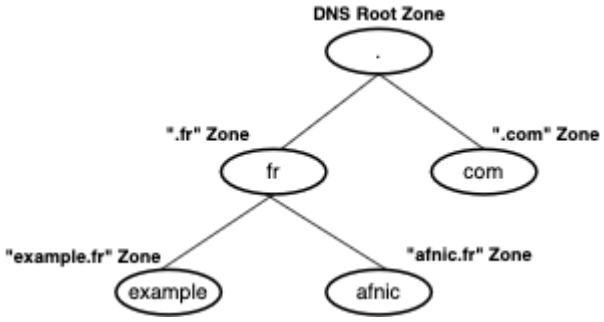


Figure 3. DNS Tree example

As shown in Fig. 3, the Top Level Domain (TLD) ".fr" has two sub domains "example.fr" and "afnic.fr". But, ".fr", "afnic.fr" and "example.fr" are three separate zones. Each of these zones contains all the data for their specific domains. For example, the ".fr" zone contains data specific to the ".fr" domain managed by one entity. Similarly, "example.fr" contains data specific to the "example.fr" domain, which will/could be managed by a different entity. A fictitious example of a DNS zone file for a fictitious domain "example.fr" is as follows:

```
; Zone file for www.example.fr
$TTL 1h                 ; Time To Live
example.fr IN SOA ns.example.fr.
                host.example.fr.
(
            2022060304; Serial number
            3h          ; Refresh
            1h          ; Retry
```

```
            1h          ; expire
            1h          ; Negative cache
)
example.fr.   IN  NS     dns1.examplehost.fr.
example.fr.   IN  NS     dns2.examplehost.fr.
example.fr.   IN  A      192.168.0.100
```

To enable DNSSEC, using the Public-Key cryptographic technique, the zone administrator (for a zone such as "example.fr") creates a public and private key pair. The private key is accessible only to the domain owner, and all can access the public key.

DNSSEC zone administrators are recommended to create two pairs of public and private keys, wherein one is called the Zone Signing Key (ZSK), and the other pair is called the Key Signing Key (KSK). Both public keys are published in the domain's DNS zone, and they are referred to as "DNSKEY". A sample of DNSKEY record in a signed DNS zone is as follows:

```
; ZSK public key
example.fr.   IN   DNSKEY 256 3 5
        AwEAAda013Wp4CQaUBrExCIRZCYpThKqyr
        pAaC7rAm2Jn+VlYnzIqmwELmn0EqIsdf03
        /e7cV8Bao94dX3xdcK+kZ6t5Of1hOLal5q
        yn/nsKZlH247VsEE62lHQNBnxPBHIpwUL7

; KSK Public key
example.fr.   IN   DNSKEY 257 3 5
        A9Vze/B+hmwDJ+83cZ1JWW2G9geiboeMy
        iuuoXB7FVavuIHJtiux+WjseJeQ4XYUGV
        DZkPyXiJWQ/rL7azGiZB2CKPMxHyr3L5P
        d1rjC50DQS45TFDvemAmvezCBs6sUtlD8
```

In a DNSSEC-signed zone, every single Resource Record set (***RR set*** ) [6] has one (sometimes more ) corresponding RR signature (***RRSIG)*** record [7]. The RRSIG is the digital signature of the RRset. The RRSIG is produced by hashing the RRset and encrypting the hash with the concerned private key.

The private key of the ZSK is used to produce the RRSIG for all the contents of the DNS zone, except for the two DNSKEY records. It is important to note that the signed DNSSEC zone contains the RRSIG and the original unsigned data. The below example shows the contents of the "A" type RR for a DNSSEC, signed zone "example.fr".

```
; Unsigned 'A' type RR
example.fr.  1   A   192.168.0.100

; RRSIG for 'A' type RR
example.fr. 1 RRSIG A 5 5 1
(
    202206300640 202206030640 3960 example.fr.
    s8dMOWQjoTKEo1bsK+EYUY+32Bd84300FcJfl
    00FcJflqthv1u60DVDVobllhqt0AaiD/dlnn7
    32Bd84300FcJflqthv1u60DVDVobllhqt0Aai
)
```

### C. Building the Chain of trust

DNSSEC adds a level of security wherein if the original IPv4 address (e.g., "192.16.0.100") in the zone "example.fr" is modified to "198.51.100.1" by an impersonator during DNS resolution, a DNSSEC validating client application will find it out. The reason is that the corresponding RRSIG will not match with the hash of the original RR on un-signing.

But there is a possibility that the impersonator successfully sent the false IP address and the corresponding RRSIG, signed by their own generated private/public key pair before the valid response. And, if the public key is trusted, a DNSSEC validating client application would not be able to find out that the data has been tampered with.

In the PKIX model, to counter such a compromise, the certificate generated by the web server was authenticated by a trusted third party, i.e. the CA. In the DNSSEC case, the validation of the public key is based on a cryptographic ***chain of trust***.

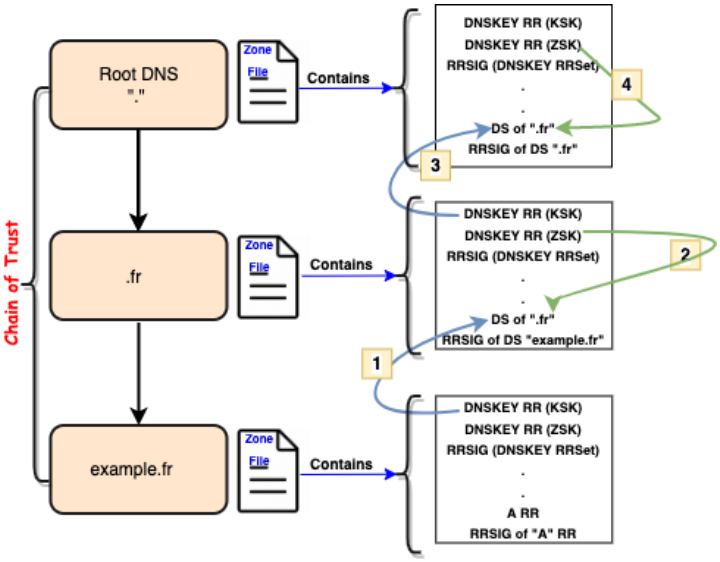


Figure 4. An example of how the chain of trust is established

Fig. 4 illustrates the process of building the chain of trust:

1) From the KSK DNSKEY of the "zone example.fr", a Delegation Signer (DS) RR [9] is created. The DS RR is the hash of the KSK DNSKEY. This DS RR is published in the parent zone of "example.fr", which is ".fr".
2) This record has to be signed by the parent zone ZSK private key, i.e. by the ZSK private key in ".fr" to get the RRSIG of the "example.fr" DS.
3) Similarly the DS record for the ".fr" zone is published in its parent zone, which is the DNS root ".".
4) As in Step '2', the RRSIG of ".fr" DS is obtained by by signing the ".fr" DS with the "." KSK as shown in step 4.

This is how the chain of trust is established.

### D. Verification of the data using the chain of trust

Subsection III-C explained the process of building the chain of trust. This subsection describes how a DNSSEC-aware resolver validates a DNSSEC-enabled DNS query/response. For DNSSEC resolution, the ”public key” of the DNS zone must be configured with a DNS resolver that validates with DNSSEC. Usually, most resolvers have the public key (KSK DNSKEY) of the DNS root ”. ”as ***”Trust Anchor”***.

The resolver can be on the local computer or at a remote location (such as at the ISP). Let’s assume that the DNSSEC validating resolver is at the ISP and doesn’t have the IP address of ’example.fr’ in its cache. It begins a recursive query to identify the DNS server that holds the authoritative information for ’example.fr’.

The DNS resolution process starts top-down from the ”.” to ”.fr” to ”example.fr” authoritative servers. Each of these authoritative servers will add additional DNSSEC data to the DNS responses. This additional data, in effect, is the digital signature of the DNS data contained in the response, i.e. the RRSIG.

1) The ’A’ record of ’example.fr’ is validated through its RRSIG (in the same reply) using the DNSKEY of ’example.fr’
2) The DNSKEY of ’example.fr’ is validated through its RRSIG using the DS of ’example.fr’
3) The DNSKEY of ’.fr’ is validated through its RRSIG using the DS of ’.fr’
4) The DNSKEY of the root zone ’.’ is validated through its RRSIG using the trust-anchor

### E. DANE (DNS Authentication of Named Entities) [8] - Augmenting the security in PKIX

*1) The problem of many:* At a glance, if we look at the size of the list of CAs accepted by popular browsers such as Chrome, Firefox, Internet explorer, etc., it varies but is in the range of hundreds. For example, a browser such as Firefox trusts 1,482 CA Certificates (as per EFF SSL observatory in 2010) provided by 651 organizations. Complementing the issue is that in the CA ecosystem, there is a practice of a CA providing authorization to other organizations or its branches to create certificates on its behalf. They are called subordinate CAs. A browser will trust the digital certificate produced by the subordinate CA also.

Even if one CA among the list of CAs, or its subordinates are compromised, it can generate a certificate for any domain name, which could then be authenticated by a browser, thereby compromising a secure web communication. For instance, two different CAs (where one is a compromised CA) can issue two separate certificates for the same domain, which the browser will trust. The lacunae here is that the domain owner has no way of telling the browser which CA certificate to be used to authenticate to connect to the server of the particular domain.

*2) Limiting the attack surface, what are the options?:* Different techniques were proposed to reduce the attack probability in the PKIX model such as Trust on First Use (ToFU) [9], Certificate Transparency (CT) [10], Certificate Authentication and Authorization (CAA) [11] and DANE. Of the different technologies proposed to limit the attack surface, ToFU is the easiest to implement because it needs only a browser to install the ToFU compatible browser add-on. Perspectives and CT are a system of Notary services which does not completely coexist with the current PKIX model and need additional services acting as notary services. CAA is like a hack which does not need any modifications and, in the short term, looks like a better option for limiting the attack surface. But looking at security from an end-to-end perspective and providing more options to the users (such as self-signed certificates), DANE ranks high.

*3) DANE Primer:* DANE standardizes the usage of the TLSA (TLS Authentication) RR as shown in Fig. 5. TLSA RR for a domain name is published in the domain’s DNS zone to indicate the certificate information that corresponds to a specific service on a specific port of a name in that zone.


Figure 5. TLSA RR explained

As shown in Fig. 5, the TLSA resource record consists of four fields: the ”certificate Usage”, ”a Selector”, ”a Matching type”, and the ”Certificate for association” data. The application must match the ’certificate for association data field’ in the TLSA RR with the target certificate (i.e. the certificate obtained from the domain’s web server) based on the other values (certificate usage, selector and matching type) in the TLSA resource record.

For DANE to augment the security in the existing PKIX model, the domain’s DNS zone should be DNSSEC signed, and the DNS resolution validated using DNSSEC. DANE protocol enables the domain owner to pinpoint the CA certificate the browser should use to validate the certificate obtained from the web server, thus reducing the attack probability.

## IV. IoT System Overview

IoT connectivity technologies could be classified broadly into three categories: Short Range (Bluetooth, Zigbee, Zwave), Medium range (WiFi) and Long-range (LoRa, NB-IoT, Wi-Sun, Sigfox).

We plan to validate our hypothesis of using DNS-based PKI on LPWAN (Low-Power Wide Area Networks), considered the most resource-constrained IoT networks, and then extend them to other types of networks. If the DNS-based PKI work for LPWAN, it should also work for other less constrained networks.

In the LPWAN category, we short-listed LoRaWAN since, compared to other IoT connectivity technologies, the LoRaWAN ecosystem provides the freedom to its stakeholders to make their own choices in choosing the ED manufacturers Service providers, and applications providers. Since the radio

connectivity uses a license-free spectrum, the freedom of choice in LoRaWAN extends to deployment options. Public LoRaWAN has nationwide coverage; private LoRaWAN focuses on specific use-cases and community networks that can be used for free by end-users.

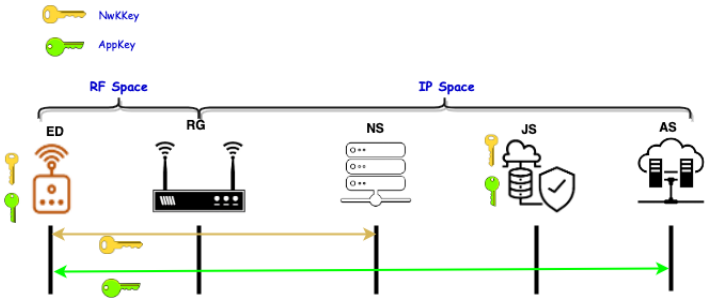


Figure 6. LoRaWAN Key Distribution

LoRaWAN is an asymmetric protocol with a star topology as shown in the Fig. 6. Data transmitted by the IoT ED (End-Device) is received by a Radio Gateway (RG), which relays it to a Network Server (NS). The NS decides on further processing the incoming data based on the ED's unique identifier (DevEUI). The NS has multiple responsibilities like forwarding the uplink from the ED to the Application Server (AS), queuing the downlink from the AS to the ED, forwarding the ED onboarding request to the appropriate AA (Authentication Authorization ) servers, named as Join Server (JS) in LoRaWAN terminology. While the ED is connected to the RG via LoRa modulated ***RF messages***, the connection between the RG, the NS and the AS is done through ***IP traffic*** and can be backhauled via Wi-Fi, hardwired Ethernet or Cellular connection.

The JS acting as the AA server controls the terms on how the ED gets activated (i.e., onboarded) to a selected LoRaWAN. There are two types of ED activation: ***Over the Air Activation (OTAA)*** and ***Activation by Personalization (ABP)***. With ABP, the ED is directly connected to a LoRaWAN by hardcoding the cryptographic keys and other parameters required for secured communication. With OTAA, the parameters necessary to create a secured session between the ED and the servers in the Internet are created dynamically. This secured session is similar to the TLS handshake used in the HTTPS connection. OTAA is preferred over ABP since it is dynamic, decouples the ED and the backend infrastructure, and doesn't need session keys to be hardcoded.

The ED performs a Join procedure (i.e. the onboarding process) with the JS during OTAA by sending the Join Request (JR). The JR payload contains the ED's unique identifier (i.e. the DevEUI), the cryptographic AES-128 root keys: NwkKey, AppKey and JoinEUI (unique identifier pointing to the JS).

The JS associated with the ED has preliminary information such as the ED's DevEUI, the cryptographic keys: ***NwkKey*** and ***AppKey*** (as shown in the Fig. 6) required for generating session keys to secure the communication between the ED and the NS and AS.

### *A. Key Sharing challenge between multiple Stakeholders*

As shown in the Fig. 7, the IoT ED manufacturer injects the root keys - NwkKey and AppKey into the ED. Since LoRaWAN has the potential of multiple stakeholders, there is a need for these root keys to be pre-shared with them. The process of sharing the root keys is currently done by printing the keys behind the device, sending them via mail etc. which is not secure.

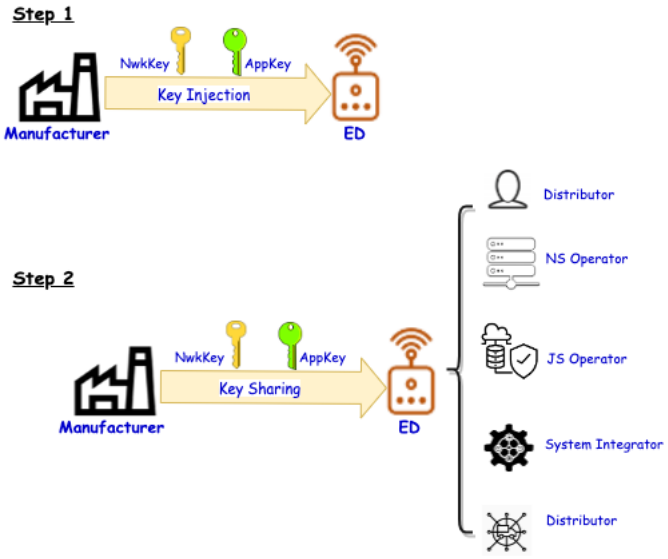


Figure 7. LoRaWAN Key injection sharing

One method of solving this operational nightmare is to use asymmetric keys based on a PKI as used to secure web traffic.

### *B. PKI Challenges in IoT*

*1) Constrained IoT devices and network:* LoRa-based IoT devices are highly constrained: they have little memory, limited processing capacity, and limited power. We plan to experiment with our hypothesis on Class 0 and 1 [12], which are very constrained with RAM size much less than 10 KB and flash memory much less than 100 KB. For Class 1, they are around 10 KB and 100 KB, respectively.

The LoRaWAN specification [13] defines a maximum data payload for each worldwide region. This full payload size varies by DataRate (DR) because of the maximum on-air transmission time allowed for each regional specification. While 52 bytes is the maximum payload in Europe, in the U.S. and other regions that operate in the 900MHz ISM band, the lowest DR is restricted to 11 bytes. These limits are chosen to meet the requirements of the regulatory agencies of the region.

Due to the ED and the network constraints, The CA model for issuing the X.509 digital certificates is not operationally feasible for LoRaWAN. The primary issue with the X.509 digital certificates is its size. Thus not compatible with resource-constrained IoT networks. Asymmetric Keys using PKI, which have worked well for secure Internet communication, cannot be used due to its size. It is impossible to send a 2048 byte

X.509 digital certificate over a LoRaWAN Communication which has an MTU (Maximum Transfer Unit) of 52 bytes.

*2) Non Availability of dedicated CA infrastructure for IoTs:* In the web, the browser client (such as Chrome or Firefox) has a certificate store containing thousands of Root CA certificates. The browser authenticates any server that delivers an X.509 certificate digitally signed by anyone of the Root CA in its certificate store. Such certificate store infrastructure is unavailable in the LoRaWAN backend network elements or any IoT backend infrastructures.

*3) Cost:* Even if we assume the infrastructure exists, the digital certificates come at a cost, which is not viable for most IoT services. We tried "Let's encrypt", which provides X.509 digital certificates for free. However, it was not possible to benefit since they do not offer certificates for domain names with more than ten labels (JoinEUI has more than 16 labels). A viable solution to resolve the operational and cost issue is to generate our Self-Signed certificates.

## V. Using DNS-based PKI in IoT

We propose designing an open PKI for IoT using the DNS protocol and its security extensions - DNSSEC and DANE.

For secure ED onboarding, the interface between the ***back-end network elements*** (the NS, JS and the AS) in the IP space (Fig. 6) should be mutually authenticated (i.e., both the client and the server authenticate each other), as per the LoRaWAN backend specifications [14]. Nevertheless, how mutual authentication should be done is left to the implementer's choice and is not normative.

A viable solution to resolve the operational and cost issue is to generate Self-Signed certificates.

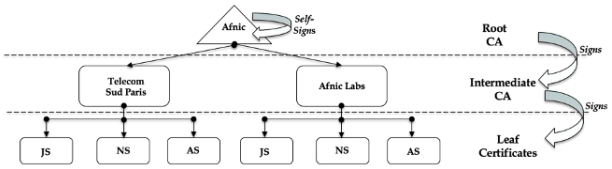


Figure 8. Certificate provisioning infrastructure

A CA provisioning infrastructure (Fig. 8) was set up where Afnic emulated the Root CA role and generated intermediate certificates for two LoRaWAN - TSP (Telecom Sud Paris) and Afnic Labs. Complete details on setting up the infrastructure is provided in the Quick Start guide [15].

During testing, we identified that combining the intermediate and the server leaf certificate (a combined trust chain - Fig. 9) during a TLS handshake could bypass the need for having a certificate store with all intermediate certificates. The validating server needs to store only the root CA certificate. The certificate validation process is done by sending the combined trust chain to the server's IP address. On receiving the combined trust chain, the server first verifies the leaf certificate in the combined trust chain. When the leaf certificate is unknown, it checks the following certificate in the chain, the intermediate certificate. Since the root CA signs the intermediate certificate, the combined certificate chain becomes trusted. Thus, the backend network elements (NS, AS and the JS) could be mutually authenticated even if they are in different networks since they have a common root CA at the top of the chain of trust.

Since the infrastructure uses self-signed certificates, there is no cost involved.

### A. Need for DANE?

The issue with the certificate provisioning infrastructure using a single root CA (Fig. 8) creates a single private PKI. This approach fails from an operational feasibility perspective as number of stakeholders are not willing to be restricted to a single CA.

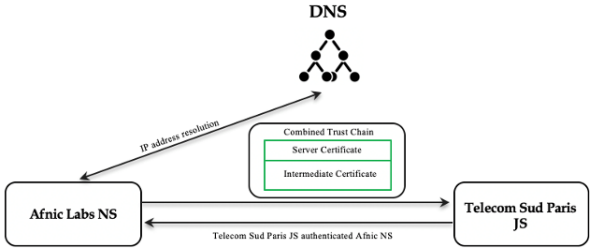


Figure 9. Certificate Validation Process

The IETF DANCE (DNS Authentication of Named Clients Everywhere) WG (Working Group) [16] seeks to make PKI-based IoT device identity universally discoverable, more broadly recognized, and less expensive to maintain by using DNS as the constraining namespace and lookup mechanism. DANCE builds on patterns established by the original DANE RFCs [8] [17] to enable client and sending entity certificate, public key, and trust anchor discovery. DANCE allows entities to possess a first-class identity, which, thanks to DNS, may be trusted by any application also relying on the DNS. A first-class identity is an application-independent identity.

The IETF DANCE WG is discussing two Internet drafts [18] and [19] based on DANE, which possibly solves the single root CA issue. For this to work, a TLS Client should have a signed DNS TLSA record (as in Fig. 5) published in the DNS zone corresponding to its DNS name and X.509 certificate or public key.

[19] specifies a TLS extension to convey a DANE Client Identity to a TLS server. The extension contain the client identity in the form of the DNS domain name that is expected to have a DANE TLSA record published for it as shown in the example below:

```
light_sensor._device.example.com. IN TLSA (
       3 1 2
       0f8b48ff5fd94117f21b6550aaee89c8
       d8adbc3f433c8e587a85a14e54667b25
       f4dcd8c4ae6162121ea9166984831b57
       b408534451fd1b9702f8de0532ecd03c )
```

During the TLS handshake, the server requests a client certificate (via the "Client Certificate Request" message). The server then extracts the DANE client identity, constructs the DNS query name for the corresponding TLSA record and authenticates the client's certificate or public key. During mutual authentication, both the client and the server could be authenticated as shown in the Fig.10

DANE-based mutual authentication enables using a self-signed certificate with different Root CA's. Thus, each institution can choose its Root CA to sign the certificates and validate dynamically based on DANE client identity.

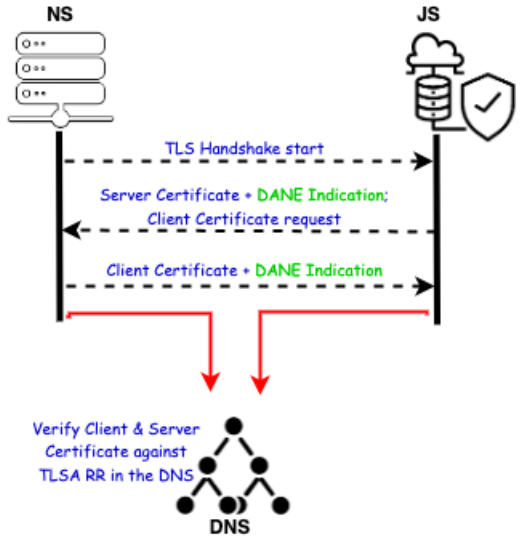


Figure 10. Mutual authentication facilitated by DANE

## VI. Conclusion

In this paper, we demonstrated how the DNS-based PKI infrastructure could be a fitting solution to construct the IoT PKI. DNS-based PKI enables the use of self-signed certificates, thus solving the cost issue and a federated set up with the possibility of each entity choosing its own Root CA's. It is true that we still do not have a solution to extend the PKI to the RF space (Fig. 6). It is a work in process. If we could have end-to-end communication between the IoT ED and its back elements by either compressing or fragmenting the X.509 digital certificate, then it could solve several operational issues in IoT key management.

## References


[1] P. Mockapetris. ""Domain names - concepts and facilities", STD 13, RFC 1034". In: *IETF Legacy RFC* (November 1987). https://www.rfc-editor.org/info/rfc1034.

[2] P. Mockapetris. ""Domain names - implementation and specification", STD 13, RFC 1035". In: *IETF Legacy RFC* (November 1987). https://www.rfc-editor.org/info/rfc1035.

[3] Katie B. et al. *Considerations for Managing Internet of Things (IoT) Cybersecurity and Privacy Risks*. Tech. rep. NIST, 2019.

[4] Simon Josefsson. *Storing Certificates in the Domain Name System (DNS)*. RFC 4398. Mar. 2006.

[5] *X.509: Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks*. 2008.

[6] Paul A. Vixie et al. *Dynamic Updates in the Domain Name System (DNS UPDATE)*. RFC 2136. Apr. 1997.

[7] Scott Rose et al. *Resource Records for the DNS Security Extensions*. RFC 4034. Mar. 2005.

[8] P. Hoffman and J. Schlyter. ""The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698". In: *IETF PROPOSED STANDARD* (August 2012). https://www.rfc-editor.org/info/rfc6698.

[9] Yaron Sheffer and Daniel Migault. *TLS Server Identity Pinning with Tickets*. RFC 8672. Oct. 2019.

[10] Ben Laurie, Adam Langley, and Emilia Kasper. *Certificate Transparency*. RFC 6962. June 2013.

[11] Phillip Hallam-Baker and Rob Stradling. *DNS Certification Authority Authorization (CAA) Resource Record*. RFC 6844. Jan. 2013.

[12] Carsten Bormann, Mehmet Ersue, and Ari Keränen. *Terminology for Constrained-Node Networks*. RFC 7228. May 2014.

[13] *LoRaWAN® Specification v1.1*.

[14] *LoRaWAN Backend specs*. https://lora-alliance.org/sites/default/files/2018-04/lorawantm_specification_-v1.1.pdf.

[15] *IoTRoam-QuickStart*. https://github.com/AFNIC/IoTRoam-Tutorial/blob/master/QuickStart.md.

[16] *Charter for the IETF DANCE WG*. https://datatracker.ietf.org/wg/dance/about//. 2022.

[17] Viktor Dukhovni and Wes Hardaker. *The DNS-Based Authentication of Named Entities (DANE) Protocol: Updates and Operational Guidance*. RFC 7671. Oct. 2015.

[18] Huque S., Dukhovni V., and Wilson A. *TLS Extension for DANE Client Identity, draft-huque-tls-dane-clientid-06*. Tech. rep. IETF, 2022.

[19] Huque S., Dukhovni V., and Wilson A. *TLS Client Authentication via DANE TLSA records, draft-huque-dane-client-cert-08*. Tech. rep. IETF, 2022.