A comparison between cryptography libraries used in BRSKI protocol for constrained devices

Ricardo R. Ehlert[1], **Laura R. Soares**[1], **Jéferson C. Nobre**[1]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

{rrehlert,lrsoares,jcnobre}@inf.ufrgs.br

Abstract. *The lack of a standard protocol for bootstrapping constrained devices is still a challenge in the management of Internet of Things (IoT) and Healthcare Internet of Things (HIoT). The Bootstrapping Remote Secure Key Infrastructure (BRSKI) protocol is a standard proposed by the Internet Engineering Task Force (IETF) for non-constrained devices, with cBRSKI as a constrained variant. In this work we review the current state of authentication protocols for constrained environments, emphasizing use-cases in the healthcare scenario. Then, we compare the memory usage and execution time of the cryptography library used in the reference implementation of both BRSKI and cBRSKI with lightweight alternatives. A test implementation was written using WolfSSL to perform the Cryptographic Message Syntax (CMS) signing function of the protocol, which is performed by OpenSSL in the reference implementation. Our experiments show that the lightweight library results in reduced bootstrap time and memory usage, without harming functionality. These findings highlight alternative BRSKI implementations suitable for constrained devices, and demonstrate that using lightweight cryptography libraries is recommended for IoT and HIoT.*

1. Introduction

Information and communication technologies can improve all dimensions of quality of life, and the healthcare industry is no different. Healthcare Internet of Things (HIoT) and Wireless Body Area Networks (WBAN) help healthcare practitioners to remotely monitor patients, which enables faster diagnosis and agile decision-making (Soares et al. 2023).

The on-boarding and bootstrapping of constrained devices is still a challenge for IoT and HIoT devices. Without a standardized solution, most on-boarding and authentication protocols are proprietary, which poses a challenge to connect devices that belong to different manufacturers. The current standard solution for device bootstrapping is the Bootstrapping Remote Secure Key Infrastructure (BRSKI) protocol (Pritikin et al. 2021). BRSKI was designed for conventional devices, therefore, applying it directly to constrained devices can be challenging. The Constrained BRSKI, or cBRSKI, is a counterpart to the BRSKI protocol currently in development (Richardson et al. 2025) which aims to adapt the original protocol to constrained environments. However, cBRSKI has not yet reached the status of standard protocol.

There are several efforts in the literature evaluating the current state of authentication protocols in constrained networks. Peltonen et al. (Peltonen et al. 2020) investigate the use of the Extensible Authentication Protocol (EAP) (Vollbrecht et al. 2004) and whether it can be applied to resource-constrained networks or not. Carrillo and

Lopez (Carrillo and Lopez 2016) present a novel bootstrapping service using EAP, interacting with AAA infrastructures and defining a new EAP lower layer using Constrained Application Protocol (CoAP) (Shelby et al. 2014). In the healthcare field, Cheng et al. (Cheng et al. 2019) discuss authentication in a heterogeneous IoT environment and propose a reliable mechanism to update authentication and session keys. Aghili et al. (Aghili et al. 2019) analyze the Lightweight RFID Protocol for Medical Privacy Protection in IoT (LRMI) (Fan et al. 2018), an RFID-based solution, and propose an enhanced authentication protocol called SecLAP, to overcome the security concerns of LRMI. Xu et al. (Xu et al. 2019) propose a three-phase mutual authentication and key agreement scheme. A blockchain-based WBAN was proposed for wearable IoT devices in healthcare by Baucas et al. (Baucas et al. 2023). The lack of a standardized protocol and implementation for bootstrapping constrained devices is evident across the state-of-art literature.

The absence of standardized protocols for constrained devices poses interoperability and security challenges in IoT and HIoT. Without a common protocol, different sensors are incapable of communicating and working together. Work-in-progress documents, such as the cBRKSI, propose a series of changes in the protocol and are challenging to already existing implementations. However, changes in the reference BRSKI implementation can make the protocol more suitable for IoT devices. This paper proposes an alternative to the current reference implementation of the BRSKI protocol, regarding the technologies used in the client (*pledge*) application. The evaluation uses different cryptographic libraries to compare the functionality of the client application. Metrics such as memory usage and total bootstrap time are considered. The successful implementation of a lightweight cryptography library, resulting in an effective bootstrap process, demonstrates that these changes can result in a protocol suitable for constrained devices.

This paper is organized as follows. Background information on the BRSKI protocol is presented in Section 2. Then, Section 3 presents the related works on bootstrapping of constrained devices, with special attention to use case scenarios in the healthcare field. Section 4 shows the proposed alternative to the standard BRSKI implementation, and Section 5 evaluates it. Finally, Section 6 concludes the paper.

2. Background

The BRSKI protocol is a current Internet Engineering Task Force (IETF) proposed standard for device bootstrapping, as defined in RFC 8995. This section explains the bootstrapping process and the components used in the reference implementation.

2.1. Bootstrapping Remote Secure Key Infrastructure (BRSKI)

Bootstrapping is the process where a simple system initiates a more complex one. In the networking field, bootstrapping typically refers to the initial distribution of settings and configurations such as identification and security information to establish trust and enable communication between different parties (Reinfurt et al. 2017). The BRSKI protocol (Pritikin et al. 2021) is part of the Autonomic Networking Integrated Model and Approach (ANIMA) working group inside the IETF, which develops and maintains specifications for interoperable protocols for automated network management and control (Jiang and Eckert 2020). The BRSKI standard defines an Autonomic Control Plane (ACP) (Eckert et al. 2021) that controls and executes the bootstrapping process.

Figure 1 shows the essential components of the BRSKI architecture. The *Domain* (the area in gray) is the set of entities sharing a common local trust anchor, such as a local network. The *Pledge* is the unconfigured device trying to connect to the new network. When entering this network, the pledge contacts a *Registrar*–a domain representative which decides whether a new device is allowed to join, via the *Join Proxy*–a domain entity that facilitates this communication. The pledge authenticates itself using its *IDevID* certificate. The registrar then contacts a *Manufacturer Authorized Signing Authority (MASA)*, configured by the device's manufacturer, to verify the authenticity of the pledge using the *MASA certificate*.

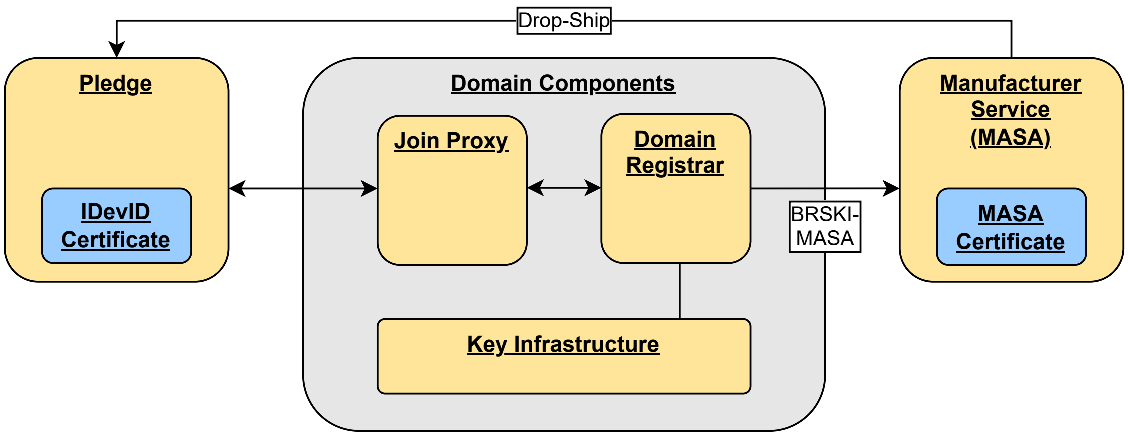


Figure 1. BRSKI Protocol Architecture.

Having an IETF standard bootstrap protocol allows devices from different manufacturers to use the same protocol, which enables effective communication and interoperability. However, constrained devices do not yet have a common protocol, despite bootstrapping being essential in the lifecycle of an IoT device (Garcia-Morchon et al. 2019). On top of that, IoT devices have various application areas, which make them unlikely to be produced by the same manufacturer. The lack of a standard bootstrap protocol for constrained devices raises challenges on how to approach this operation while maintaining device interoperability between nodes (Malik et al. 2019).

2.2. Bootstrapping Process

When joining a new network, the *Pledge* starts the bootstrapping process by discovering a local *Join Proxy* to establish a connection to the *Domain Registrar*. This communication begins with a TLS Handshake, followed by a TLS server authentication by the registrar. Then, the pledge formulates a voucher-request, a message to be sent to the *MASA* to authenticate the device. This request contains a timestamp, the pledge serial number, and the certificate of the registrar that the pledge is communicating to, obtained during the server authentication. This message is digitally signed by the pledge and sent to the registrar, who decides whether the device is accepted to join the network or not. This decision can be based on the device type or manufacturer specified in the signing certificate. If the device is accepted, the registrar then sends the voucher-request to the MASA service.

The digital signing of the voucher-request and voucher messages, made by the pledge and the MASA respectively, is a key part of the bootstrapping process. Within the pledge, the messages are signed with the X.509 IDevID certificate (IEEE 2018) and validated by the MASA to certify that the manufacturer indeed fabricated that device. The process of digitally signing the voucher is done using CMS (Cryptographic Message Syntax)(Housley 2009), which is a standard syntax for storing signed data, using a X.509 certificate. Upon receiving the voucher-request, the MASA service determines if the registrar claim to own that pledge device is accepted. If the claim is accepted, the MASA service issues a new voucher containing the device serial number and the certificate of the domain registrar who is the new device owner. This new voucher is sent to the registrar, who can audit and verify it based on the MASA audit logs, and then passed to the pledge device to complete the bootstrapping process.

Figure 2 shows the bootstrapping communication flow. The transparent join proxy mediates the initial communication, followed by the authentication between the pledge and the registrar. Then, the registrar decides to either accept the device or not, and communicates with the MASA. Finally, the voucher is sent to the registrar by the MASA, and then handed to the Pledge.

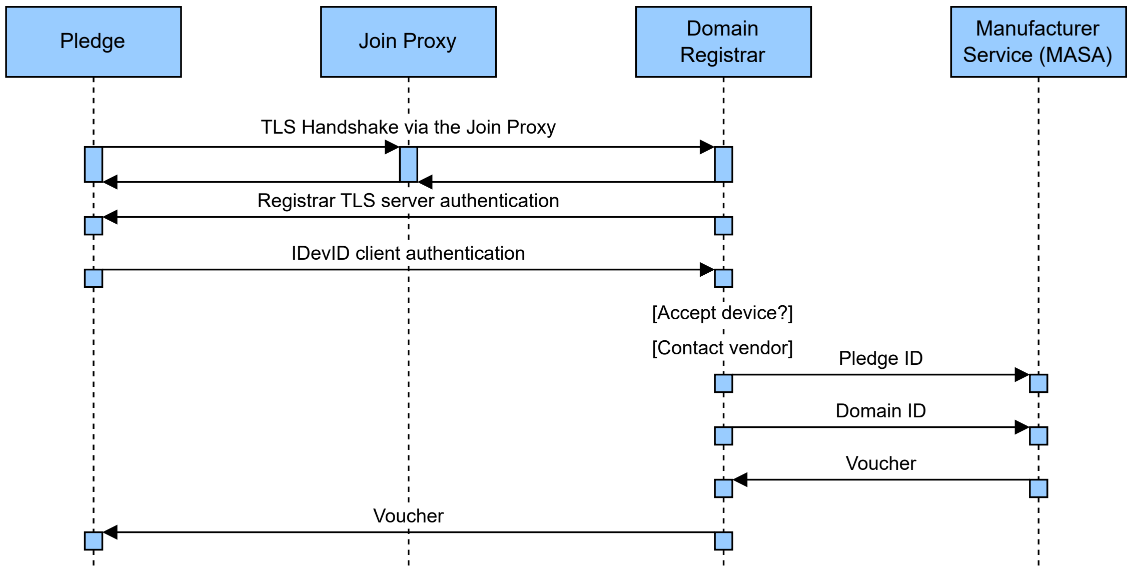

Pledge
Join Proxy
Domain Registrar
Manufacturer Service (MASA)
TLS Handshake via the Join Proxy
Registrar TLS server authentication
IDevID client authentication
[Accept device?]
[Contact vendor]
Pledge ID
Domain ID
Voucher
Voucher

Figure 2. BRKSI Protocol's Bootstrapping Communication Flow.

2.3. Reference Implementation

The open-source reference implementation of BRSKI used in RFC 8995 is *Minerva*[1], which follows the ANIMA concepts inside the *Minerva Ecosystem*. The ecosystem consists of four main components: *Highway*, *Fountain*, *ChariWT*, and *Reach*. *Highway*, *Fountain*, and *Reach* are under the MIT license, and *ChariWT* is under the Apache 2.0 license. *Highway* is the IDevID creator/signer, and the MASA Authorized Signing Authority. It is run by the manufacturer to create the IDevID certificates and as a public-facing service to generate vouchers in response to signed voucher-requests. The *Fountain* is the Minerva domain owner, the Join Registrar/Coordinator, and it also operates as an integrated Certification Authority, issuing IDevIDs to the new nodes. *Reach* is a pledge

[1]https://github.com/AnimaGUS-minerva/

simulator written in ruby, and *ChariWT* is a voucher library used by all other components to create and validate vouchers. All four components are written in Ruby. During the bootstrap phase, *Reach* signs the voucher-request message digitally, however, the OpenSSL library available for Ruby does not support this operation. A modified version[2] of the library is used instead, which can cause some drawbacks.

RFC 8995 describes that pledge authentication and pledge voucher-request signing are done via a PKIX-shaped certificate installed during manufacturing. This certificate is known as the 802.1AR IDevID and is a globally unique manufacturer-provided certificate used to identify the device (IEEE 2018). When crafting a voucher-request, the pledge must authenticate it using CMS and sign it with its own IDevID certificate, creating a signed voucher-request. Within the BRSKI protocol, CMS is used to digitally sign the MASA vouchers and the pledge voucher-requests, and also to authenticate the signed messages. The signed voucher messages are managed and validated by the *ChariWT* library and are a key part of the BRSKI reference implementation.

3. Related Work

This section presents the current state of authentication protocols and schemes within constrained networks, with particular attention to HIoT and WBAN devices—the healthcare field has a substantial demand for services and applications using constrained sensor nodes. For the scope of this work, the terminology regarding constrained devices follows the definitions in RFC 7288 (Bormann et al. 2014). A constrained node has some of its characteristics limited by size, weight, available power and energy, making the limitation in energy and network bandwidth a dominating consideration in all design requirements. Also, some layer-2 services such as full connectivity and broadcast/multicast may be lacking. This section also discuss IETF work-in-progress documents on the subject, and compare the main components of the solutions.

3.1. Authentication in Constrained Networks

Peltonen et al. (Peltonen et al. 2020) investigate whether EAP, which is widely used for bootstrapping enterprise Wi-Fi devices, is suitable for resource-constrained devices or not. They based the choice on the design principles of the protocol. It offers a centralized AAA (Authentication, Authorization and Accounting) infrastructure to monitor and manage all devices with fine-granted access control, supporting individual authentication, key provisioning, and the possibility of isolating devices after the initial bootstrapping. A cloud-hosted environment ensures better security and frequent updates, and the credentials issued can be scaled depending on the capability of the devices. The authors evaluated that most existing EAP methods require credentials to be pre-provisioned on the device. As many IoT devices have limited User Interfaces available, they opted to use Nimble out-of-band authentication for EAP (EAP-NOOB) (Aura et al. 2021), which surpasses this problem by allowing device authentication and bootstrapping without preconfigured credentials.

Also using EAP as bootstrapping service, Carrillo and Lopez (Carrillo and Lopez 2016) presented a EAP lower layer using CoAP and interacting with AAA infrastructures. After the bootstrapping is completed, the security

[2]https://github.com/mcr/ruby-openssl

association between the Smart Object and the Controller in an IoT network can be established in two ways, using a new CoAP Option (AUTH Option) defined for integrity protection, or done through DTLS (Rescorla and Modadugu 2012). The proposed solution improved the original protocol thanks to a shorter message size, decreased bootstrapping time and energy consumption, and with increased success percentage–the probability of finishing the bootstrapping. The work did not discuss the use of different cryptography libraries in the implementation.

From the industry standpoint, Wi-Fi Easy Connect, or Device Provisioning Protocol (DPP), is a proprietary provisioning protocol issued by the Wi-Fi Alliance (Wi-Fi Alliance 2022) aiming to replace the Wi-Fi Protected Setup (WPS). Bootstrapping and authentication can be configured in various user-friendly ways, such as QR code, NFC, Bluetooth, and others, making it suitable for variety of network sizes and devices–including IoT. DPP uses elliptic curve cryptography for key establishment and the Advanced Encryption Standard (AES) for symmetric encryption, and is able to provision credentials from passwords to X.509 certificates. However, recent security analysis of the Easy Connect/DPP show that some features introduced security vulnerabilities such as the risk of connected users impersonating the Configurator device and distributing malicious files to other devices (Chatzisofroniou and Kotzanikolaou 2025). $hostapd$, a popular open-source implementation of DPP, uses OpenSSL as the default cryptographic library but has optional internal TLS/cryptographic libraries that can be used in place of external ones[3].

3.2. HIoT and WBAN Environments

In the healthcare field, HIoT and WBAN devices also have authentication problems to be solved. Cheng et al. (Cheng et al. 2019) discuss the authentication within the Community Medical Internet of Things (CMIoT), a network with heterogeneous nodes. They design an efficient node secure two-way identity authentication method, and propose a secure and reliable update mechanism for updating authentication keys and session keys. The scheme is proven to be correct, secure, and efficient. Authentication and security solutions in HIoT have also been proposed based on Radio-Frequency Identification (RFID), for its secure and lightweight safety mechanisms. A lightweight RFID mutual authentication (LRMI) protocol is proposed by Fan et al. (Fan et al. 2018). However, Aghili et al. (Aghili et al. 2019) analyze the LRMI protocol and proposes an enhanced authentication protocol called SecLAP, to overcome the security concerns of the LRMI. The enhanced protocol reduces the flows between tag and reader, the server does not store their identity, and the freshness of all messages is preserved.

Most of the existing authentication schemes are based on asymmetric encryption. However, the sensor devices in WBAN and HIoT are typically resource-constrained, therefore, their computing resources can hardly afford to use asymmetric encryption efficiently. Xu et al. (Xu et al. 2019) propose a mutual authentication and key agreement scheme consisting in a initialization phase and a registration phase, both executed by the system administrator in a secure environment, and an authentication phase, where the sensor nodes exchanges information with the server over an insecure channel. This scheme uses a hash of the previous session key to generate the new one, which helps guarantee

[3]https://w1.fi/wpa_supplicant/

forward secrecy. Using a private blockchain, Baucas et al. (Baucas et al. 2023) propose a WBAN plataform for wereable HIoT devices. The private blockchain addresses security issues thanks to its immutability properties. The authors did a security analysis using the STRIDE threat model (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege) (Hu et al. 2022). The scheme can also limit the amount of data being disclosed by creating access levels to keep sensitive data written and protected. However, there is no emphasis on the cryptography libraries in use and how they impact the constrained devices.

3.3. State-of-art BRSKI

At the time of the writing of this paper, several Internet Drafts based on the BRSKI protocol are in discussion within the IETF aiming to adapt the original protocol for different use cases. BRSKI Cloud Registrar (Friel et al. 2025) extends the pledge functionalities to situations when there is no local infrastructure available, such as home or remote office. The device uses a *call-home* mechanism to find the operator-maintained infrastructure, contact a Cloud Registrar, and proceed with the bootstrapping process. BRSKI with Pledge in Responder Mode (BRSKI-PRM) (Fries et al. 2025) aims to enable bootstrapping in domains with limited or nonexistent connectivity between the pledge and the domain registrar. It achieves that by changing the interaction model from a pledge-initiated mode, as used in BRSKI, to a pledge-responding mode, where the pledge is in a server role. It also adds a new component, the Registrar-Agent, to facilitate the communication between the pledge and the domain registrar by providing a store and forward communication path between them. The BRSKI-PRM is agnostic to enrollment protocols (such as domain Certification Authorities). Finally, BRSKI with Alternative Enrollment (BRSKI-AE) (von Oheimb et al. 2025) reached the IETF Standard status (RFC 9733) while extending BRSKI to support certificate enrollment mechanisms such as the Certificate Management Protocol (CMP) (Brockhaus et al. 2023). CMP uses authenticated self-contained signed objects for certification messages, for use cases where BRSKI original enrollment mechanism over Secure Transport is not suitable.

Particularly relevant to the scope of this work, the Constrained Bootstrapping Remote Secure Key Infrastructure (cBRSKI) (Richardson et al. 2025) Internet-draft provides a solution for secure zero-touch onboarding of resource-constrained IoT devices. The protocol is also designed for constrained networks, which may have limited data throughput or may experience frequent packet loss. Because cBRSKI is a variant of BRSKI, the architecture is very similar. Some of the changes are the use of CBOR-encoded vouchers, instead of JSON; the Enrollment over Secure Transport (EST) protocol, used in BRSKI, is replaced with EST-over-CoAPS; and HTTPS used in BRSKI is replaced with DTLS-secured CoAP (CoAPS).

3.4. Literature Gap

The ongoing standardization efforts towards a bootstrapping protocols for constrained devices is evident after reviewing the related literature presented in this section. The amount of works on the topic highlights the need for early discussion about implementation choices, such as the appropriate cryptographic libraries. Table 1 compares all the works, standards and drafts discussed so far regarding their target devices, standardization status, and the authentication mechanisms they employ. Within these works, we found no discussion regarding the use of different cryptography libraries and how they can affect each implementation.

Table 1. Comparison between works on bootstrapping protocols

Work	Targeted Devices	Standard	Authentication
BRSKI (Pritikin et al. 2021)	Regular	IETF Standard	IDevID (x.509) Certificates
cBRSKI (Richardson et al. 2025)	Constrained	IETF Draft	IDevID (x.509) Certificates
BRSKI-PRM (Fries et al. 2025)	Regular	IETF Draft	IDevID (x.509) Certificates
BRSKI Cloud (Friel et al. 2025)	Regular	IETF Draft	IDevID (x.509) Certificates
BRSKI-AE (von Oheimb et al. 2025)	Regular	IETF Standard	IDevID (x.509) Certificates, CMP
EAP-NOOB (Aura et al. 2021)	Constrained	No	EAP-NOOB
CoAP EAP (Carrillo and Lopez 2016)	Constrained	No	DTLS/CoAP Option
Easy Connect/DPP (Wi-Fi Alliance 2022)	Regular and Constrained	Wi-Fi Alliance Proprietary Standard	Public Key-based Identities
CMIoT (Cheng et al. 2019)	Constrained	No	Two-way Identity
LRMI (Fan et al. 2018)	Constrained	No	RFID
SecLAP (Aghili et al. 2019)	Constrained	No	RFID
Mutual Auth (Xu et al. 2019)	Constrained	No	Symmetric Keys
Blockchain (Baucas et al. 2023)	Constrained	No	Blockchain

Recently, the US National Institute of Standards and Technology (NIST) initiated a project on *Trusted Internet of Things (IoT) Device Network-Layer Onboarding and Lifecycle Management* (Fagan et al. 2024). Currently on draft stage, the project is part of a series of NIST Cybersecurity Practice Guides aiming to facilitate the adoption of standards-based approaches to cybersecurity issues–such as how to provide network credentials to IoT devices in a trusted manner and maintain a secure device posture throughout the device lifecycle. One of the goals of NIST SP 1800-36 is to develop examples of trusted network-layer onboarding solutions for IoT devices that provide private network credentials, enable the devices and the network to mutually authenticate, and use encrypted channels. Some of the builds used as example in the current draft use BRSKI as trusted network-layer onboarding protocol. It emphasizes BRSKI efficiency on handling the onboarding of a large number of devices, however, it discusses the reliance on the device manufacturer to provide the MASA server in order to properly and securely provision the devices. Concerns about the MASA infrastructure are related to the possibility of the manufacturer ceasing to provide the service, whether temporarily or permanently–due to DDoS attacks, power outages, server maintenance, going out of business, and so on. If the MASA server becomes unavailable, the onboarding services will no longer function. These concerns are also addressed in depth in RFC 8995.

4. Lightweight Alternative BRSKI Implementation

RFC 8995 states that the BRSKI Protocol is aimed at general non-constrained devices. As HIoT and IoT devices tend to have constrained resources, it is challenging to meet the necessary conditions to guarantee the proper execution of the BRKSI protocol. cBRSKI, while proposing several changes in the protocol components, still uses parts of the original reference implementation. In this section, we compare cryptography library alternatives for the authentication of the pledge device and its communication with the registrar, while keeping the remaining structure of the reference implementation. A lightweight library can guarantee the functionality of the bootstrapping protocol and is better suited for constrained devices.

The BRKSI protocol uses the cryptography library for CMS signing vouchers and voucher-requests. As shown in Section 2, the reference implementation uses the OpenSSL library, one of the most popular and widely used implementations of TLS (de Ruiter 2016). However, the OpenSSL library is not designed with IoT devices in mind, with alternatives targeted at embedded systems offering smaller footprints in comparison due to their increased modularity. In OpenSSL, the dependencies between modules make it harder to have an efficient modular design if compared to lightweight libraries (Khlebnikov and Adolfsen 2022).

To select the ideal library for the scope of this work, we evaluated the commonly used lightweight libraries and compared them in functionality and build size. The most well-known lightweight TLS libraries are WolfSSL (formerly CyaSSL, also known as Yet Another SSL) and Mbed TLS (formerly PolarSSL). The latest versions of WolfSSL have assembly optimizations and are very performant. According to the benchmarks on the WolfSSL website, WolfSSL raw encryption performance is often on par and, in some cases, can be even 50% faster than OpenSSL (Khlebnikov and Adolfsen 2022). Due to its modularity, the library has a typical footprint size that ranges from 20 to 100 kB, which makes it up to 20 times smaller than OpenSSL (Qutqut et al. 2018). To cut down the library sizes, unnecessary modules can be disabled during the compilation process, which can help reduce the overall build size and suit the needs of the target system. We built all three alternatives inside a Linux machine to compare their build sizes: 1.5 MB for WolfSSL, 1.6 MB for Mbed TLS, and 15.5 MB for OpenSSL. Our analysis used WolfSSL version 5.7.0, Mbed TLS 3.6.0, and OpenSSL version 3.0.2. Figure 3 shows the difference between build sizes.

Comparing the build sizes, it is evident that OpenSSL has a much bigger build size in comparison to the other lightweight libraries. Another important feature necessary for the BRSKI protocol is the CMS signing, used in the *ChariWT* voucher library. With further research, we found that the Mbed TLS does not support this operation (Nyman and Elliott 2024). Despite Mbed TLS having a similar build size to WolfSSL, the lack of the support for CMS signing necessary for the BRSKI protocol makes it unsuitable for the implementation. Therefore, we selected the WolfSSL library for the proposed alternative implementation of the BRSKI protocol. Table 2 presents this comparison.

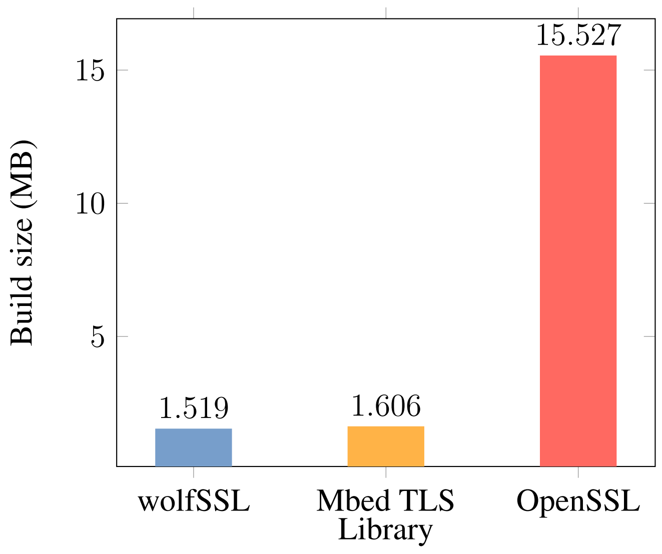


Figure 3. Difference in Build Sizes

Table 2. Comparison between TLS libraries

	WolfSSL	**Mbed TLS**	**OpenSSL**
IoT Targeted	Yes	Yes	No
CMS Signing	Yes	No	Yes
Modular Compilation	Easy	Easy	Hard (Modular dependencies)
Build Size	1.519 MB	1.606 MB	15.527 MB

In addition, as mentioned in Subsection 2.3, the reference BRSKI implementation uses a patched version of the *ruby-openssl* library for the signatures and verification. This comes with a series of setbacks. Using a specific patched version of a library means there will be no future updates, even security ones, which can cause problems in the future. Since the main reason for using this patched library version is to run the CMS signing functions, and since this work searches alternatives to make the BRSKI protocol lighter for the new pledge device, it adds to the motivation behind using an alternative lightweight cryptography library with support for CMS.

The proposed work targets only the pledge application, not interfering with the already established implementation of the registrar and MASA components. This will allow constrained devices to be used with the already operational implementation, without the need for any modifications in the environment. Figure 4 shows the communication flow of the BRSKI protocol and highlights the step of the bootstrapping process that will be affected. As no change is proposed in the protocol itself, this study can help work-in-progress variants such as cBRSKI by demonstrating that a lightweight library such as WolfSSL can make for an even lighter protocol, more suitable for IoT and HIoT devices.

5. Evaluation

To evaluate the alternative lightweight cryptographic library in the BRSKI implementation, we prepared a custom pledge application to be integrated into the existing *Minerva*

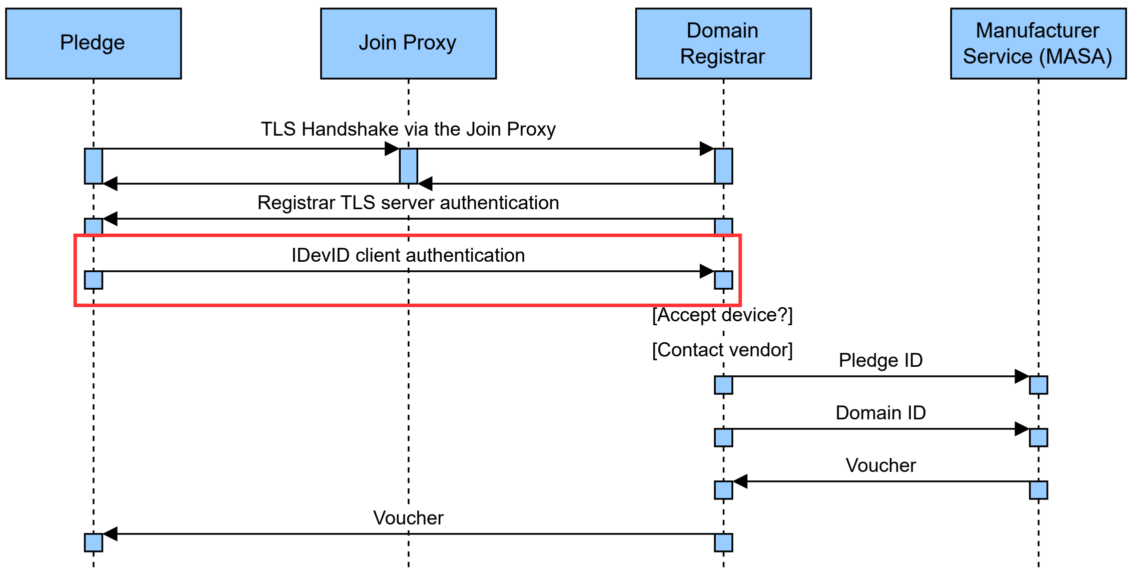

Pledge
Join Proxy
Domain Registrar
Manufacturer Service (MASA)
TLS Handshake via the Join Proxy
Registrar TLS server authentication
IDevID client authentication
[Accept device?]
[Contact vendor]
Pledge ID
Domain ID
Voucher
Voucher

Figure 4. Targeted Step of BRKSI Protocol Bootstrapping Communication Flow.

ecosystem. First, we compare the performance of both original and alternative versions only while executing the CMS signing function. Then, we compare them inside the complete bootstrapping process.

5.1. Emulation Environment

As discussed in Section 2.1, the BRSKI protocol has three main components: the pledge device to be authenticated; the domain registrar in charge of the authentication; and the MASA service that validates the pledge authenticity. With that in mind, the testing environment was set up using three different systems, one for each of the components. We used two Linux virtual machines running Ubuntu 22.04 for the MASA server and the Domain Registrar each, with the pledge application executing in the host machine.

Within the *Minerva* ecosystem, the MASA role is performed by the *Highway* component, the registrar by the *Fountain*, and the pledge application by the *Reach*. With the *Highway* and *Fountain* infrastructures set up within the virtual machines and the *Reach* application in the host machine, we simulate an environment where the local pledge reaches and completes the enrollment function by communicating with the MASA and the Join Registrar over the network, which should resemble a realistic environment.

5.2. Implementation

The CMS signing function is the target of our changes, as discussed in Section 4. This function is implemented using the WolfSSL lightweight cryptography library, instead of OpenSSL library used in the reference implementation. We aim to demonstrate that using a lightweight library in the BRSKI protocol is possible and contributes to making the protocol lighter.

In the reference implementation source code the signing is executed after the initial handshake between the pledge and the registrar, which results in the pledge formulating a voucher-request and digitally signing it. We isolated the digital signing step of the application executed by OpenSSL from the reference implementation so it is possible to compare the CMS signing function between different libraries. Then, we prepared a different pledge application[4] using WolfSSL[5] for the digital signing of the voucher-request. There are no changes to the registrar and MASA components and the messages they receive and send are exactly the same.

5.3. Experiments

The experiments first compare the two isolated applications performing the CMS signing function of an example voucher request–one using the OpenSSL library, as in the BRSKI reference implementation, and the other using the WolfSSL library as a lightweight counterpart. The comparison between libraries used WolfSSL 5.7.0 and OpenSSL 3.0.2, the same versions evaluated in Section 4. Comparing the two isolated applications allows for a better view of metrics such as execution time and memory usage during the message signing, as it prevents other functions executed by the pledge during the bootstrapping process from showing up in the results. Then, the two pledge applications are executed within the bootstrap enrollment process, in order to compare the execution time of the CMS signing between pledge and registrar inside the whole bootstrapping operation. Each experiment ran each application up to 20 times, which means 40 runs per test and 80 runs across both tests.

5.4. Results

We evaluate the isolated pledge applications in terms of memory usage and execution time, and then measure the execution time of the whole bootstrapping operation for each cryptographic library alternative. Memory usage data is collected by using Linux *memusage* script, and execution time is collected by using the *time* script.

The two pledge applications execute the CMS signing function of the bootstrapping process. This function runs after the pledge and registrar have their communication established, and the pledge has elaborated the voucher-request. For the scope of this test, the proper voucher-request is already formulated and the applications execute just the message signing phase. Figure 5 shows the execution time metrics for each pledge application running the signing function, measured in milliseconds (ms). In Figure 5a it is possible to see that, even in the maximum case, WolfSSL still takes only about 46% of the execution time of OpenSSL. Also, the WolfSSL application had more consistent results than the OpenSSL application. Figure 5b compares the average time results of 1.16 ms for WolfSSL and 1.72 ms for OpenSSL, while also showing the standard deviation, which was much smaller with the WolfSSL library.

Figure 6 shows the of memory usage of each pledge application, measured in kilobytes (kB). The measured values remained consistent across all the runs, which is correct since the same actions were being executed every time. The difference, however, is significant between the two applications. The lightweight cryptographic library alternative, WolfSSL, used only 23 kB of memory, while OpenSSL used over 30 times more space, 722 kB, to execute the same signing function.

We also evaluated the role played by both cryptographic libraries in the complete bootstrap process. The process starts with the pledge discovering the registrar server, constructing and signing a voucher-request, and finishes when the pledge receives the final

[4]https://github.com/rrehlert/reach_tests
[5]https://github.com/wolfssl

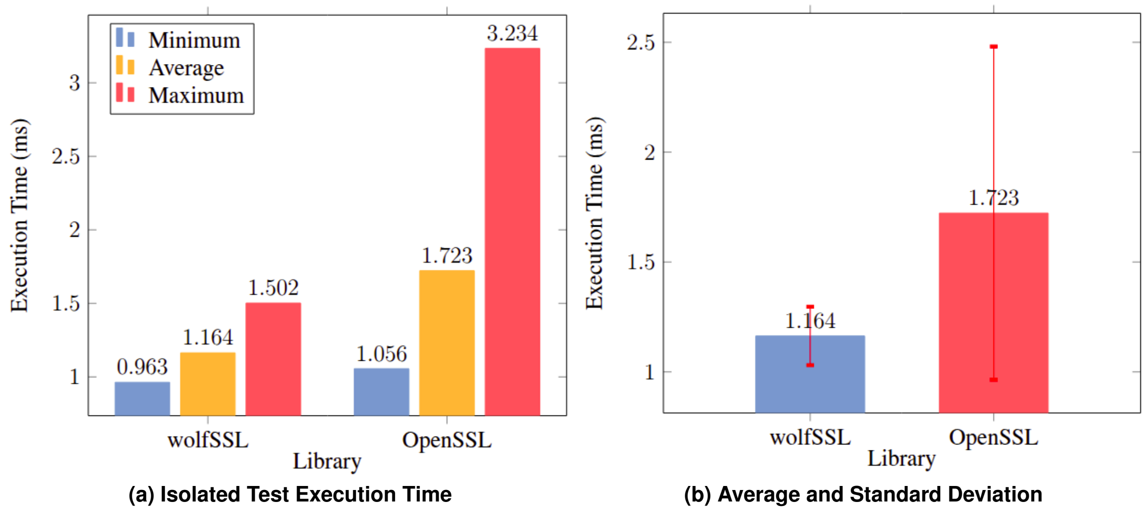


Figure 5. Execution time metrics of each pledge application

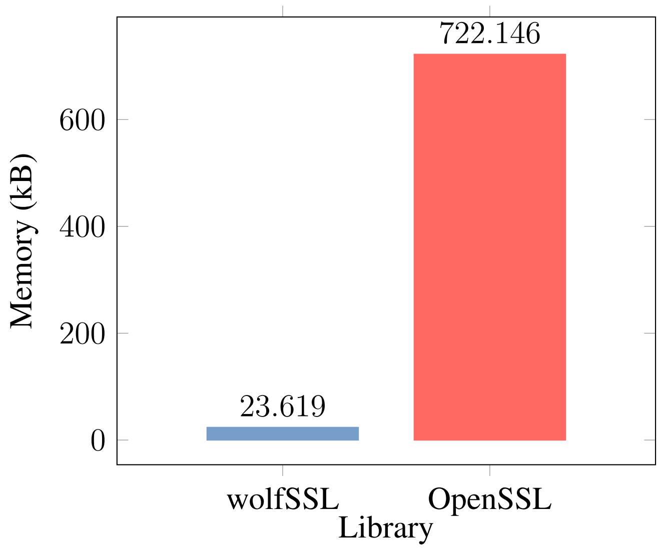
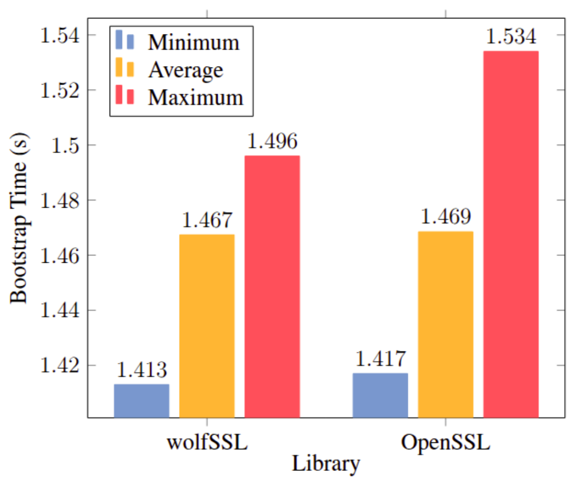


Figure 6. Isolated Test Memory Usage

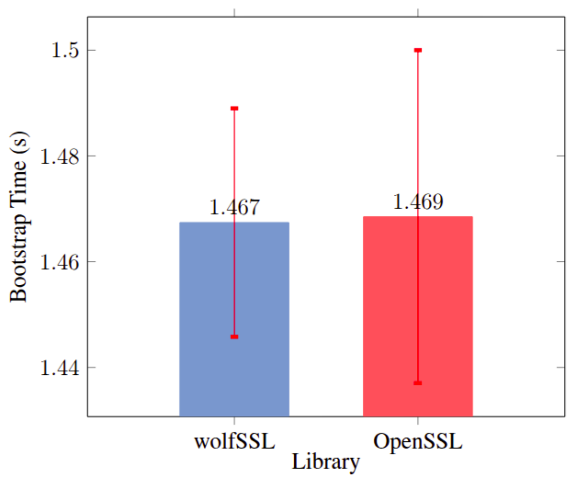
voucher, which authenticates the connection. Figure 7 compares the pledge application using WolfSSL and the one using OpenSSL regarding the total bootstrap time, measured in seconds (s). The results show a slightly faster total time for the WolfSSL implementation. The WolfSSL version executed the bootstrap in an average of 1.467s, versus 1.469s for OpenSSL version (Figure 7a). Despite the small difference in the average times, the WolfSSL alternative had substantially more consistent results compared to the OpenSSL implementation. The standard deviation for the tests running the WolfSSL application was 0.0216s, and 0.0315s for OpenSSL. Figure 7b compares the average of both tests. Similar to the previous isolated test, we see that using WolfSSL results in a more predictable and consistent implementation alternative regarding total execution time.

5.5. Discussion

The tests with the isolated signing application show that the WolfSSL library achieves a more predictable scheme, which can be helpful for constrained devices. Regarding



(a) Total Bootstrap Time



(b) Average and Standard Deviation

Figure 7. Execution time metrics of the whole bootstrap operation

memory usage, while the difference of 700 kB between both libraries can be considered very small for regular devices and computers, IoT and constrained devices have limited memory and processing resources (Abdul-Qawy et al. 2015). Despite the time difference between both WolfSSL and OpenSSL implementations being less apparent in the full bootstrapping process, they corroborate the results presented within the isolated applications–which points to WolfSSL as a faster and more consistent alternative.

Between the pledge applications, the average bootstrap time had a difference of 0.001 seconds (or 1 ms). This can be explained in part by the results obtained by the isolated applications, which had the OpenSSL implementation being, on average, 0.5 ms slower than the WolfSSL counterpart. The successful implementation and execution of the bootstrap process within the BRSKI protocol using a lightweight cryptography library shows that changes in the implementation can make the protocol more efficient and suitable for constrained devices. Additionally, the implementation does not affect components other than the pledge application, therefore changes to the already standardized Registrar and MASA infrastructures are not necessary.

6. Final Considerations

The absence of standard protocols for constrained devices is apparent, even more so in areas which these devices are broadly used–such as HIoT and WBAN systems in healthcare. The bootstrap process in particular, for when a device joins a network for the first time, does not have IoT-focused standards. While RFC8995 defines and describes the BRSKI standard solution for regular devices, it acknowledges that it is not designed for constrained devices.

This work proposes changes in the reference implementation of the BRSKI protocol, resulting in a more suitable implementation aimed towards IoT devices. These changes focus on the cryptography library used in the reference implementation, OpenSSL, and compare its efficiency with an implementation using WolfSSL as a lightweight alternative. Our results demonstrate that using of a lightweight TLS library in the pledge application is possible and does not affect the protocol functionalities. Using the lightweight library results in a slightly faster bootstrap process and more efficient memory usage in the message signing function. This process can make the BRSKI protocol more suitable for IoT environments without changing the protocol itself.

Future work concerning the use of the BRKSI protocol in constrained environments should focus on implementing the whole pledge client using lightweight libraries, not just the bootstrap function. Additionally, the alternative implementations should be deployed to IoT and HIoT devices in real-world scenarios to validate that such changes have real effects. Finally, other security protocols can be analyzed the same manner as this paper did to the BRSKI protocol, with the goal of finding lightweight implementation alternatives that move towards formats suitable for IoT and HIoT devices.

Regarding IETF drafts yet to be standardized, the cBRSKI describes a protocol based on BRSKI, but with a few changes such as using CoAP instead of HTTPS and the use of CBOR encoding for the voucher and voucher-requests. In addition, the use of a CBOR encoding of X.509 certificates through profiling is being evaluated (Mattsson et al. 2024), resulting in a C509 certificate. Both documents are still work in progress, but, if standardized, may pose big advancements in the security of HIoT devices, the first one introducing a protocol fully designed for constrained environments, and the second one introducing a lighter counterpart of the X.509 certificate used, for instance, within the IDevID certificates of the devices.

References

[Abdul-Qawy et al. 2015] Abdul-Qawy, A. S., Pramod, P., Magesh, E., and Srinivasulu, T. (2015). The internet of things (iot): An overview. *International Journal of Engineering Research and Applications*, 5(12):71–82.

[Aghili et al. 2019] Aghili, S. F., Mala, H., Kaliyar, P., and Conti, M. (2019). SecLAP: Secure and lightweight RFID authentication protocol for Medical IoT. *Future Generation Computer Systems*, 101:621–634.

[Aura et al. 2021] Aura, T., Sethi, M., and Peltonen, A. (2021). Nimble Out-of-Band Authentication for EAP (EAP-NOOB). RFC 9140, Internet Engineering Task Force.

[Baucas et al. 2023] Baucas, M. J., Spachos, P., and Gregori, S. (2023). Private Blockchain-Based Wireless Body Area Network Platform for Wearable Internet of Thing Devices in Healthcare. In *IEEE International Conference on Communications*, pages 6181–6186.

[Bormann et al. 2014] Bormann, C., Ersue, M., and Keranen, A. (2014). Terminology for Constrained-Node Networks. RFC 7228, Internet Engineering Task Force.

[Brockhaus et al. 2023] Brockhaus, H., von Oheimb, D., and Gray, J. (2023). Certificate Management Protocol (CMP) Updates. RFC 9480, Internet Engineering Task Force.

[Carrillo and Lopez 2016] Carrillo, D. G. and Lopez, R. M. (2016). Lightweight CoAP-Based Bootstrapping Service for the Internet of Things. *Sensors*, 16(3).

[Chatzisofroniou and Kotzanikolaou 2025] Chatzisofroniou, G. and Kotzanikolaou, P. (2025). Security analysis of the wi-fi easy connect. *International Journal of Information Security*, 24(2):1–11.

[Cheng et al. 2019] Cheng, X., Zhang, Z., Chen, F., Zhao, C., Wang, T., Sun, H., and Huang, C. (2019). Secure Identity Authentication of Community Medical Internet of Things. *IEEE Access*, 7:115966–115977.

[de Ruiter 2016] de Ruiter, J. (2016). A tale of the OpenSSL state machine: A largescale black-box analysis. In *Secure IT Systems: 21st Nordic Conference, NordSec 2016, Oulu, Finland, November 2-4, 2016. Proceedings 21*, pages 169–184. Springer.

[Eckert et al. 2021] Eckert, T., Behringer, M. H., and Bjarnason, S. (2021). An Autonomic Control Plane (ACP). RFC 8994, Internet Engineering Task Force.

[Fagan et al. 2024] Fagan, M., Marron, J., Watrobski, P., Souppaya, M., Barker, W., Deane, C., Klosterman, J., Rearick, C., Mulugeta, B., Symington, S., et al. (2024). Trusted Internet of Things (IoT) Device Network-Layer Onboarding and Lifecycle Management. *NIST Special Publication 1800-36.*

[Fan et al. 2018] Fan, K., Jiang, W., Li, H., and Yang, Y. (2018). Lightweight RFID Protocol for Medical Privacy Protection in IoT. *IEEE Transactions on Industrial Informatics*, 14(4):1656–1665.

[Friel et al. 2025] Friel, O., Shekh-Yusef, R., and Richardson, M. (2025). BRSKI Cloud Registrar. Internet-draft, Internet Engineering Task Force. Work in Progress: `https://datatracker.ietf.org/doc/draft-ietf-anima-brski-cloud/14/`. Access on May 9th, 2025.

[Fries et al. 2025] Fries, S., Werner, T., Lear, E., and Richardson, M. (2025). BRSKI with Pledge in Responder Mode (BRSKI-PRM). Internet-draft, Internet Engineering Task Force. Work in Progress: `https://datatracker.ietf.org/doc/draft-ietf-anima-brski-prm/21/`. Access on May 9th, 2025.

[Garcia-Morchon et al. 2019] Garcia-Morchon, O., Kumar, S., and Sethi, M. (2019). Internet of Things (IoT) Security: State of the Art and Challenges. RFC 8576, Internet Engineering Task Force.

[Housley 2009] Housley, R. (2009). Cryptographic Message Syntax (CMS). RFC 5652, Internet Engineering Task Force.

[Hu et al. 2022] Hu, X., Cheng, D., Chen, J., Jin, X., and Wu, B. (2022). Multiontology Construction and Application of Threat Model Based on Adversarial Attack and Defense Under ISO/IEC 27032. *IEEE Access*, 10:117955–117972.

[IEEE 2018] IEEE (2018). IEEE Standard for Local and Metropolitan Area Networks - Secure Device Identity. *IEEE Std 802.1AR-2018 (Revision of IEEE Std 802.1AR-2009)*, pages 1–73.

[Jiang and Eckert 2020] Jiang, S. and Eckert, T. (2020). Autonomic Networking Integrated Model and Approach (anima). `https://datatracker.ietf.org/group/anima/about/`. Access on May 09th, 2025.

[Khlebnikov and Adolfsen 2022] Khlebnikov, A. and Adolfsen, J. (2022). *Demystifying Cryptography with OpenSSL 3.0: Discover the best techniques to enhance your network security with OpenSSL 3.0.* Packt Publishing.

[Malik et al. 2019] Malik, M., Dutta, M., and Granjal, J. (2019). A Survey of Key Bootstrapping Protocols Based on Public Key Cryptography in the Internet of Things. *IEEE Access*, 7:27443–27464.

[Mattsson et al. 2024] Mattsson, J. P., Selander, G., Raza, S., Höglund, J., and Furuhed, M. (2024). CBOR Encoded X.509 Certificates (C509 Certificates). Internet-draft, Internet Engineering Task Force. Work in Progress: `https://datatracker.ietf.org/doc/draft-ietf-cose-cbor-encoded-cert/13/`. Access on May 9th, 2025.

[Nyman and Elliott 2024] Nyman, B. and Elliott, P. (2024). GitHub Issue 8768: Better support for PKCS#7. Archived at `https://web.archive.org/`

web/20250509220002/https://github.com/Mbed-TLS/mbedtls/issues/8768 on May 09th, 2025.

[Peltonen et al. 2020] Peltonen, A., Inglés, E., Latvala, S., Garcia-Carrillo, D., Sethi, M., and Aura, T. (2020). Enterprise Security for the Internet of Things (IoT): Lightweight Bootstrapping with EAP-NOOB. *Sensors*, 20(21).

[Pritikin et al. 2021] Pritikin, M., Richardson, M., Eckert, T., Behringer, M. H., and Watsen, K. (2021). Bootstrapping Remote Secure Key Infrastructure (BRSKI). RFC 8995, Internet Engineering Task Force.

[Qutqut et al. 2018] Qutqut, M. H., Al-Sakran, A., Almasalha, F., and Hassanein, H. S. (2018). Comprehensive survey of the IoT open-source OSs. *IET Wireless Sensor Systems*, 8(6):323–339.

[Reinfurt et al. 2017] Reinfurt, L., Breitenbücher, U., Falkenthal, M., Leymann, F., and Riegg, A. (2017). Internet of Things patterns for device bootstrapping and registration. In *Proceedings of the 22nd European Conference on Pattern Languages of Programs*, pages 1–27.

[Rescorla and Modadugu 2012] Rescorla, E. and Modadugu, N. (2012). Datagram Transport Layer Security Version 1.2. RFC 6347, Internet Engineering Task Force.

[Richardson et al. 2025] Richardson, M., der Stok, P. V., Kampanakis, P., and Dijk, E. (2025). Constrained Bootstrapping Remote Secure Key Infrastructure (cBRSKI). Internet-draft, Internet Engineering Task Force. Work in Progress: https://datatracker.ietf.org/doc/draft-ietf-anima-constrained-voucher/27/. Access on May 9th, 2025.

[Shelby et al. 2014] Shelby, Z., Hartke, K., and Bormann, C. (2014). The Constrained Application Protocol (CoAP). RFC 7252, Internet Engineering Task Force.

[Soares et al. 2023] Soares, L. R., Bastos, L., Martins, B., Medeiros, I., Rosário, D., Nobre, J. C., and Cerqueira, E. C. (2023). A Continuous Heart-Based Biometric Authentication for Healthcare Internet of Things. In *Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais (SBSeg)*, pages 43–54. SBC.

[Vollbrecht et al. 2004] Vollbrecht, J., Carlson, J. D., Blunk, L., Aboba, D. B. D., and Levkowetz, H. (2004). Extensible Authentication Protocol (EAP). RFC 3748, Internet Engineering Task Force.

[von Oheimb et al. 2025] von Oheimb, D., Fries, S., and Brockhaus, H. (2025). BRSKI with Alternative Enrollment (BRSKI-AE). RFC 9733, Internet Engineering Task Force.

[Wi-Fi Alliance 2022] Wi-Fi Alliance (2022). *Wi-Fi Easy Connect Specification v3.0*. Archived at https://web.archive.org/web/20231031231735/https://www.wi-fi.org/system/files/Wi-Fi_Easy_Connect_Specification_v3.0.pdf.

[Xu et al. 2019] Xu, Z., Xu, C., Liang, W., Xu, J., and Chen, H. (2019). A Lightweight Mutual Authentication and Key Agreement Scheme for Medical Internet of Things. *IEEE Access*, 7:53922–53931.