Internet of Things 34 (2025) 101762

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot

Research article

Integration of Hardware Security Modules into BLE Beacons: Fundamentals and Use in a Secure and Private Geofencing Application

Miguel Mesa-Simón [a,*], Antonio Escobar-Molero [a], Luis Parrilla [b], Diego P. Morales [b], José Antonio Álvarez-Bermejo [c], Francisco J. Romero [b,*]

[a] Infineon Technologies AG, Am Campeon 1-15, Neubiberg, 85579, Germany
[b] Department of Electronics and Computer Technology, University of Granada, Faculty of Sciences, Av. Fuente Nueva s/n, Granada, 18071, Spain
[c] Department of Information Technology, University of Almeria, Cañada de San Urbano, Almería, 04120, Spain

ARTICLE INFO

Keywords:
Authentication
Bluetooth low energy beacons
Geofencing
Hardware security modules
Over-the-air activation
Wireless nodes

ABSTRACT

Bluetooth Low Energy (BLE) is a wireless technology designed for creating personal area networks in low-power applications. In the context of BLE, Beacon devices are widely used to transmit small packets of data with unique identifiers at regular intervals to be detected by surrounding devices. These devices enable a wide range of applications, including indoor navigation, marketing, and asset tracking. However, BLE Beacons suffer from multiple security issues and privacy concerns since the transmissions are unencrypted and do not include authentication mechanisms. While many implementations try to provide security to the Beacons packet, they often rely on external servers, static keys, synchronization for key derivation, or use difficult to maintain and to operate Public Key Infrastructure (PKI). In this work, we propose a solution to enhance Beacon security through the integration of Secure Elements (SEs), establishing a Root of Trust. Our approach is based on the over-the-air activation of the BLE beacons incorporating an authentication mechanism and a key derivation technique to safeguard privacy and data integrity in the communication. We demonstrate that this implementation incurs minimal delays and power consumption compared to traditional Beacons while avoiding the added complexity of solutions based on Certificates and Public Key Infrastructure (PKI). The feasibility of the proposed approach is also illustrated through a secure and privacy-preserving geofencing application. In summary, this method supports a low-power and secure point-to-point communication suitable not only for BLE beacon networks, but also for other IoT scenarios where data privacy is critical.

1. Introduction

The Internet of Things (IoT) is nowadays more and more present in our environment, finding new applications in fields such as smart cities, e-health, industry monitoring or smart homes [1]. The number of IoT devices, which is already in the realm of billions, is expected to keep increasing for the foreseeable future [2]. These IoT networks focus on transmitting, receiving and processing data, and generally consist of embedded devices, commonly referred to as motes, that can communicate wirelessly, and

* Corresponding authors.
E-mail addresses: miguelangel.mesasimon@infineon.com (M. Mesa-Simón), franromero@ugr.es (F.J. Romero).

https://doi.org/10.1016/j.iot.2025.101762
Received 11 December 2024; Received in revised form 20 August 2025; Accepted 6 September 2025
Available online 13 September 2025

that have long-lasting battery life. Given the broad field of IoT applications, each with different needs and expectations, there are multiple solutions that try to deal with the different challenges of technical implementations, such as physical size constraints, area of coverage or power consumption. This way, we can find from small wearable devices focused on health monitoring that communicate in wireless body area networks (WBAN) [3], to bigger size motes that communicate via Low-power wide-area network (LP-WAN) protocols over big distances [4].

In this context, Bluetooth and its low-power version, Bluetooth Low Energy (BLE), have become widely adopted for short-range wireless communication between devices, especially in power and size constrained IoT applications. Among the different approaches, BLE beacons have gained significant attention as they are continuously advertising data wirelessly to nearby devices at periodic intervals. They are perfect candidates for low data rate applications that require small, energy-efficient and cost-effective devices. Because of that, the use of BLE beacons have attracted the attention of multiple industrial corporations, such as Google or Apple, for diverse applications, especially in indoor localization and proximity detection. However, BLE Beacons suffer from important security issues and privacy concerns associated with cracking, spoofing, piggybacking or hijacking [5,6].

It is common that, due to the previously mentioned constraints and other restrictions, security challenges are not addressed in IoT. These restrictions arise due to several causes, mainly due to the heterogeneity of nodes and servers with different technical specifications that are expected to work together. These devices need to interconnect wirelessly, which often restricts the security measures to be implemented in order to cover all possible devices and adapt to those with greater technical limitations. Additionally, the ubiquity of the motes, usually being distributed in different locations and close to the users, increases the possibility of unwarranted modifications and physical attacks [7]. This leaves IoT architectures exposed to security threats, which may be fatal, especially in critical sectors like energy generation, health monitoring or traffic control.

In the case of BLE beacons, a common approach to increase the security of the communication is based on the use of IDs that can only be deciphered by a cloud server. However, since this method is implemented at firmware level, once this algorithm is discovered, the system can be abused [5]. This is also a common vulnerability of most IoT protocols that rely on public-key encryption schemes since if the keys are compromised, the security of the entire system can be exposed [8].

Recently, with the appearance of low power and low footprint Secure Elements (SEs) specifically designed to be integrated with IoT devices, such as Hardware Security Modules (HSM), some of these security issues can be addressed. The SEs can be used to offload security functionality from the main processor, whilst maintaining full functionality and not affecting the performance or consumption of the device [9]. In this context, we propose the use of such SEs to authenticate the nodes and secure the communication of BLE beacons. In particular, this work proposes an over-the-air authentication scheme, similar to the one used in LoRaWAN, to demonstrate how the use of these devices can add security to a communication scenario without compromising the latency or power requirements of the network. The integrity of the system is guaranteed by the use of SEs, and the generation of session keys for the communication ensures the integrity and privacy of data. To prove the relevance of the developed mechanism, we also provide an example of use in the context of BLE beacons for geofencing applications. Furthermore, we demonstrate how the addition of SEs does not significantly affect performance nor consumption of the nodes by experimentally measuring their current consumption and execution time.

This paper is structured as follows. Following this introduction, Section 2 presents an overview of the relevant IoT communication protocols upon which the work is based. This section also presents the security measures they incorporate, as well as the background regarding Bluetooth Beacons and Secure Elements. After that, Section 3 introduces a literature review of security in Bluetooth Beacons, which includes the specific contributions provided in this work. Section 4 includes a detailed description of the different elements used in our implementation. Section 5 provides the implementation details of the proposed authentication mechanism, commonly referred to as End Device activation process. The validation of the proposed approach is presented in Section 6, including experimental results associated with execution time, power consumption, scalability and regulatory compliance. After that, the practicability of the proposed solution for enhanced security and privacy BLE beacon communication is demonstrated by means of a geofencing use case in Section 7. Finally, the main conclusions are drawn in Section 8.

2. Background

This section provides an overview of the relevant IoT communication protocols in which the implementation is based, as well as their security features. The necessary background regarding Bluetooth Beacons, and Secure Elements is also provided.

2.1. Relevant communication protocols and their security measures

Given the heterogeneity of IoT applications, diverse solutions are available to meet specific needs within this context. These include from small wearable bracelets for health tracking to bigger nodes for smart city applications or weather monitoring devices in agriculture. In any case, devices in these applications form networks that may extend across various geographic areas, categorized by the network size, which ranges from short-range Body Area Networks (BAN) to Local Area Networks (LAN), and ultimately to Wide Area Networks (WAN). Different communication protocols can be applied at each network level, with possible overlaps as a feasible approach. In this section, we focus on the relevant protocols for the implementation, specifically BLE and its security, Bluetooth Beacons and LP-WAN protocols, like LoRaWAN.

1. Bluetooth: it is a wireless communication technology originally designed to transmit data over a short distance, but that has been extended over subsequent revisions to be able to cover areas of hundreds of meters, making it effectively able to build LAN networks [10]. Bluetooth operates in the 2.4 GHz Industrial, Scientific and Medical (ISM) band. In the particular case of BLE beacons, advertisement packets are used to transmit small quantities of information with low power consumption [11]. Security wise, Bluetooth presents four different security levels, numbered from 1 to 4 in increasing level of security, as well as three security modes [12]:

   (a) Security Level 1: no encryption, no pairing, no security.
   (b) Security Level 2: supports AES-128 encryption during communication but devices do not need to be paired.
   (c) Security Level 3: supports AES-128 encryption and requires pairing.
   (d) Security Level 4: supports ECDHE encryption.

   Then, for the security modes:

   (a) Security Mode 1: no data signing.
   (b) Security Mode 2: includes signing of the data for both paired and unpaired connections.
   (c) Mixed Security Mode: devices that need to support both previous cases.

   Different combinations of both security levels and modes can be implemented depending on the particular application. In Bluetooth, the pairing process is used to achieve a level of peer entity authentication as well as encryption [13]. The pairing process involves the generation of temporary keys that are used during the pairing to then generate the session keys that will eventually be used during the communication. This process generally requires the confirmation of the user via a PIN displayed on the display. If the device lacks a way of showing it, the PIN is set automatically to a certain value, generally all zeroes. There is also the possibility of doing an Out of Band pairing, for example via Near-Field Communication (NFC). As it will be shown in the following section, advertisement packets in Bluetooth are generally not encrypted nor authenticated. This is designed in order to not generate unwanted overhead and delays, as it would cause a significant power consumption to check every advertisement packet that the device receives.
2. LP-WAN: here, two LP-WAN protocols can be distinguished: LoRaWAN and MIoTy [14]. Both can cover large distances and share a similar network structure, but with a different physical layer implementation. While MIoTy uses Telegram Splitting, LoRaWAN uses Spread Spectrum Modulation. This causes MIoTy to be able to cover longer distances at lower consumption. Due to LoRaWAN having a higher rate of adoption, the focus will be on its security mechanism. Security in LoRaWAN is achieved with the End Device Activation process, where a set of parameters, including cryptographic nonces and device identifiers, are exchanged to generate a pair of session keys. One of these session keys is used for data encryption, the so called Application Session Key (AppSKey), while the other one is used to provide integrity to the data, the Network Session Key (NwkSKey). The use of two different keys allows LoRaWAN gateways to receive packets from multiple End Devices, verify their integrity, and then send them to the corresponding Application Server over the Internet, preserving the privacy of the data. This process is shown in Fig. 1, which specifically corresponds to the OTAA (Over the Air Activation) of the End Device [15].

Now that the relevant protocols in which the implementation is based have been shown, the concept of BLE Beacons and how they use BLE advertisement packets to transmit small packets of information can be presented.

2.2. BLE Beacons

Bluetooth Beacons are a class of BLE devices that use the advertising mode of BLE to broadcast information to all nearby devices with Bluetooth capabilities. Typically, these packets are used in the peripheral role of BLE to be found by other devices and establish a connection. However, in the case of BLE beacons, the information is directly codified into these packets, which are usually sent as non-connectable and non-scannable undirected mode advertisement. Bluetooth Beacons do not include a mechanism to address particular devices, therefore all the information is advertised to surrounding BLE enabled devices. The custom information that can be codified into the advertising packet is called Manufacturer Specific Data and for BLE beacons it is limited to 28 bytes [16]. In this way, although the data rate is more limited, the power consumption of the devices can be considerably reduced, as they can wake up, send the advertising packet with the specific information, and return to sleep mode [17,18].

Over the last few years, BLE beacons have been proposed for multiple applications that rely on the use of these advertising packets to provide users with a particular experience. Use cases of this technology include tracking customers in a store, trigger location-based actions in a museum, or check-in systems for workplaces, as shown in the example of Fig. 2 [19]. Most common applications of BLE beacons are based around the user localization, which is possible thanks to the proximity sensing capabilities of the BLE beacons, where the location of the emitter can be determined based on the RSSI (Received Signal Strength Indicator) value of the received packet and the power at which the advertisement packet was sent (value typically included in the frame) [5]. From all the previously mentioned wireless technologies, this indoor user location is uniquely provided by these Beacons.

The format for the three most popular Bluetooth Beacons packets is shown in Fig. 3. The first frame corresponds to iBeacons by Apple Inc [20], which is the most supported format by the BLE Beacons available on the market. Other alternatives to iBeacons include Eddystone [21], released by Google, and the AltBeacon [22], authored by Radius Networks. This work is focused on the

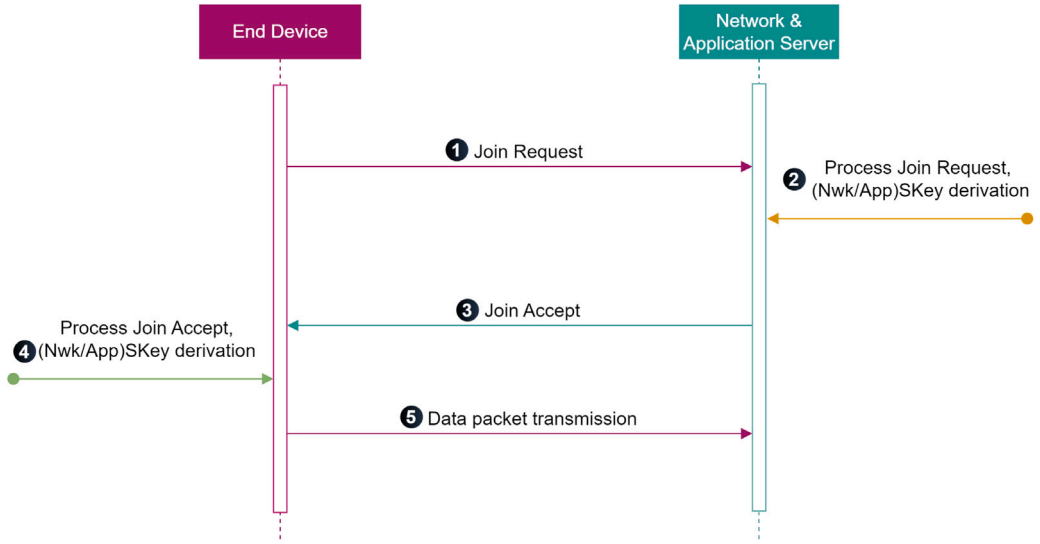

Fig. 1. OTAA LoRaWAN message flow during End Device Activation.

Fig. 2. Beacon system for user tracking and positioning in a museum [19].

iBeacon format given its popularity, but it is important to note that what has been implemented in this paper could be ported to the other Beacon format by simply adapting the data to the available fields of the other Beacons.

The format of an iBeacon advertisement packet is detailed in Fig. 4. A full iBeacon packet has 30 bytes. From those, more than half belong to the UUID, the unique identifier given to the particular application of the iBeacon. The Major and Minor fields can be used to identify an iBeacon in a set of iBeacons. For example, the minor field can be used to indicate a particular value of temperature, while the major might represent a particular room in a building. Last, the transmitted power field ("Tx Power") is typically used to calibrate the location of the device and to provide accurate positioning information indoors. Regarding the iBeacon prefix, it contains the following fields [20]:

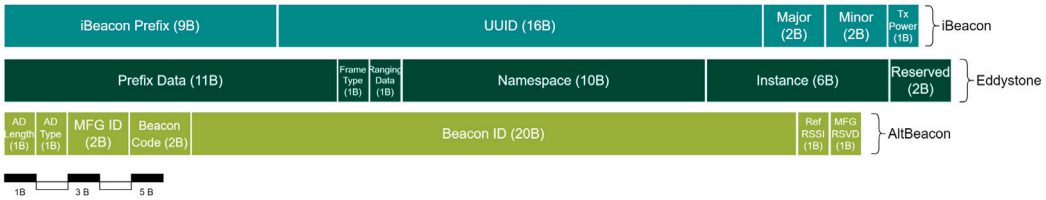

Fig. 3. Packet format comparison between iBeacon (Top), Eddystone (Middle) and AltBeacon (Bottom).

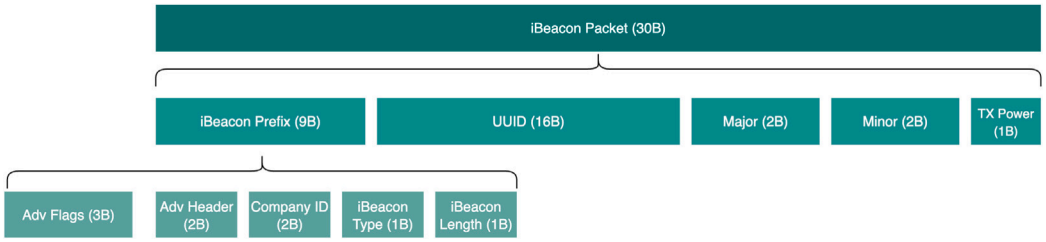

Fig. 4. Detailed iBeacon message formatting.

1. Adv Flags: defines the packet as a BLE General Discoverable packet for broadcasting.
2. Adv Header: establishes that there are 26 more bytes in the packet, corresponding to Manufacturer Specific Data.
3. Company ID: Bluetooth Special Interest Group (SIG) ID, unique identifier for every company that belongs to the Bluetooth standard organization. For iBeacons, this value is always set to "$0x004C$", the identifier for Apple Inc.
4. iBeacon type: secondary ID used by all iBeacons and which must be set to the value "$0x02$".
5. iBeacon length: defines the remaining length of the packet, and is fixed to "$0x15$" for all iBeacons.

Given the structure of an iBeacon packet, it can be seen that only the first 9 bytes of the iBeacon prefix are necessary for a packet to be detected as an iBeacon, while the rest are used for each particular application. This leaves a maximum of 21 bytes to transmit data within one advertising packet. The communication mechanism developed in this work leverages the use of this Bluetooth Beacon packets in order to establish a bidirectional communication channel in a star topology network between the Server and each one of the End Devices. This authentication mechanism is based on the End Device Activation process from LoRaWAN. Besides, the mechanism also relies on the use of SEs as a Root of Trust.

2.3. Secure Elements in IoT

Secure Elements refer to secure cryptographic processors that can safely keep device secrets, such as symmetric or asymmetric keys, as well as perform cryptographic functions, including keys generation, encryption/decryption or random number generation. Some more advanced models, namely the Trusted Platform Module (TPM), can perform other tasks such as platform attestation, where the integrity of the device can be verified. Traditionally, many of these solutions have been geared toward PCs and Server platforms, while IoT is more focused on low power microcontrollers. However, in recent years, new SEs have been designed with the IoT market in mind. These solutions are targeted at low power consumption and small footprints to match with most of the architectures available nowadays. They can be used to provide a Root of Trust to the architecture, so that nodes can be trusted. They can also provide a drop-in replacement to security measures implemented via software, allowing for the addition of new or more sophisticated security measures, as the main processor can delegate these tasks onto them. Last, by managing keys internally instead of in the flash memory, they cannot be retrieved by an attacker, minimizing the likelihood of node impersonation or cloning.

3. Literature review

In this section, the focus will be on the available solutions that aim at providing security to Bluetooth advertisement packets and Beacons. Our solution is also compared to other alternatives presented in the literature, indicating the benefits and drawbacks detected.

3.1. Security in Bluetooth Beacons

By themselves, BLE Beacons do not incorporate any security measures. This means that data is advertised by the devices in plain format, without encryption, and the identity of devices emitting Beacons cannot be verified. This leaves the door open to multiple vulnerabilities [6,23]:

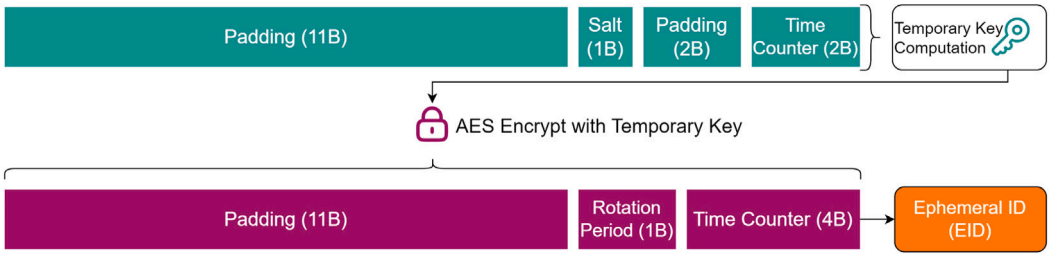

Fig. 5. Eddystone Ephemeral ID generation.

1. Piggybacking: by listening to unsecured Beacons and decoding their data, a competing business can build a database with the same information and use it to compete against a legitimate business. For example, if offers are presented to users by a legitimate business within the range of a Beacon when they are in a specific geographical area, an adversary could offer more competitive deals by leveraging the cloned information.
2. Cloning: since none of the fields are encrypted and the device identity is not verified when advertising Beacons, an attacker can clone a Beacon by copying its information and re-advertising the packet. In cases where Beacons are used for geographically locating users or objects, this can cause confusion and disrupt operations.
3. Hijacking: similarly to cloning, a malicious actor could dump the contents of the Beacon and inject their own malicious code, making detection of attacks harder.
4. Eavesdropping: since Beacons do not implement any encryption techniques, any sensitive information transmitted by them can be intercepted by an eavesdropper. For instance, if Beacons are used to identify users, this information could allow the tracking of a specific person.
5. Silencing: By transmitting cloned Beacons at higher power, the channel can become congested, causing functionality embedded in the Beacons, such as proximity estimation, to become biased.

Some defense strategies and modifications that add security measures to the Beacons have already been proposed to deal with some of these issues [24]:

1. Time-varying ID: where the Beacon UUID is not fixed. This includes the Gimbal Beacons [25], which rotate the ID according to a fixed pattern, or other solutions that use pseudorandom functions to generate the different UUIDs.
2. Anomaly detection: the traces produced by the Beacons are analyzed to detect anomalies in usage patterns, allowing subsequent packets to be ignored. This method aims at reducing the effectiveness of attacks without requiring modifications to the Beacon operation or sacrificing the battery life. However, privacy concerns may arise when the content of the packets needs to be stored and analyzed to identify anomalies [26].
3. Selective Jamming: the premise of this method is that devices not authenticated via an out-of-band mechanism are prevented from transmitting and are selectively jammed. This approach does not require modifications to the Beacons, but it is not considered effective against location-based Beacon service spoofing [27].
4. Eddystone-EID: Eddystone-Ephemeral ID is designed to provide private Eddystone beacons, where the UUID being broadcasted is made ephemeral, changing every set period of time. For the computation of an ephemeral ID, a time counter is used, where each increment must correspond to a 1 s of elapsed time. A rotation period exponent is also needed, i.e., a value between 0 and 15 that corresponds to the power-of-two scale of the time counter. This means that the device will generate a new ephemeral identifier every 2^k s. The EID is generated in two steps: first a temporary key is generated using some cryptographic salt and the top 16 bits of the device's time counter. In the second step, the key is used to compute the final EID, where AES-128 encryption is used to encrypt the rotation period exponent concatenated with the Beacon time counter, as shown in Fig. 5. The problem of this implementation arises from both the need for an out-of-band mechanism and for external elements that have networking access in order to derive these keys, since the external server must be authenticated using asymmetric cryptography, which is usually quite heavy and consuming for embedded devices. Also, there is the need to maintain time counters with small variation on both the Eddystone beacon and the device that provides the key, which might prove challenging and difficult to scale on the Server side. As well, this system only provides a mechanism to generate the ephemeral UUID, which at most would provide users with authentication and privacy since they could not be tracked at an application level. However, the rest of the information being transmitted is still unencrypted and without any integrity guarantees [28]. Eddystone also considers encryption for the Telemetry frame (TLM) using AES-EAX, but not for any other type of packet [29], so the only frames which incorporate some security are Eddystone EID, and encrypted Eddystone TLM. Moreover, Eddystone, the competing formatting for Bluetooth Beacons from Google, has been archived and is no longer an active project maintained by the company [30].
5. Estimote Secure UUID: Estimote Inc. is a private company that has designed and operates a proprietary system to rotate the UUID of iBeacons, to which they hold several patents. While originally providing the Estimote Secure UUID service, the company has since moved on to their own Bluetooth beacon technology, called Estimote Secure Monitoring. From the

available information, the devices need to be provisioned online, by connecting to an Estimote controlled server, which will then provide the necessary cryptographic material. The functioning principles seem to be very similar to the Eddystone EID, where the UUID is rotated periodically according to a cryptographic function to provide privacy and avoid piggybacking. The use of this technology opens the issue of whether to trust an external service with the security provision, as well as possible concerns about privacy and user tracking [31–33].

6. Bidirectional PAwR: Bidirectional Periodic Advertising with Responses (PAwR) is based on the Bluetooth Core Specification version 5.4. It allows for a star topology network to be established, where a bidirectional communication network can be created and devices can answer back to the advertiser. PAwR uses the concept of Periodic Advertising in Bluetooth, where a number of advertisement packets are sent containing advertising data at synchronized intervals. It allows for an unlimited number of devices to connect to one main device that is sharing varying data but at a constant rate and intervals. By dividing the periodic advertising interval in different sub-events, the devices can answer back to the main device, which provides Bidirectional PAwR. Security and encryption for the advertised data is provided with a new feature from Bluetooth 5.4, called Encrypted Advertising Data (EAD) [34]. This, however, requires that the devices had connected and paired once before the exchange of encrypted data, so that they can derive the necessary key material, which might cause privacy and introduce other security issues, as there is a dependency on an out-of-band mechanism. The encryption and integrity is provided using AES CBC-MAC, a similar mechanism to the one used in our implementation [35].
7. BP-MAC: BitWise Precomputable Message Authentication Code achieves high computational speed by pre-computing some of the MAC calculations. For this, once an AES-128 key has been selected, so-called bit tags are calculated for each bit, using the key as a pseudorandom function. The bit tags are calculated by concatenating the bits value "i" and their bit index "$m_n[i]$". In total, there would be $2 \cdot (l-1)$ bit tags to pre-compute, corresponding to the length of the message. Once calculated, they can be stored. Subsequently, a bit mask is calculated to apply it to the bit tags to provide both protection against replay attacks and leakage of information regarding the position of the bits. The masking tag is computed using a monotonic counter as a nonce, "n", and another AES-128 key, and can also be pre-computed. So, whenever a message needs to be sent or is received, the bit tags and masking tag can be retrieved from the previously obtained values, and by performing the XOR operation the MAC is obtained: $tag = AES_{128_{k1}}(i \parallel m_n[i]) \oplus AES-128_{k2}(n)$. Therefore, the only operation that is performed at runtime is the XOR, which reduces significantly the overhead. Note that this algorithm would provide a 128 bit long MAC, which would occupy the length of the UUID in the packet. Its length could be reduced to fit in a single packet reducing part of the UUID, therefore providing the Major and Minor fields with integrity protection [36].
8. Lightweight PKI: an authentication scheme that uses a lightweight PKI can be used to authenticate the BLE devices and to provide privacy and integrity to the messages exchanged [37]. Compact Elliptic Curve Cryptography (ECC) keys as well as custom compact certificates can be used to minimize the problem of the reduced throughput and small size of the packages. By performing an EC Diffie–Hellman (ECDH), an ephemeral key can be derived by both ends to provide secure communication. This scheme was originally designed for general-purpose BLE packets, with a maximum size of 251 bytes. Due to the reduced size of iBeacon packets, at only 21 bytes, the authentication would require additional transmissions and parsing of the certificates to be successful.
9. Rolling Codes: commonly used for key-fobs or garage doors, they could be used to provide authentication and privacy to the data being transmitted, as well as protection against replay attacks [38]. Similarly to Eddystone, it requires some synchronization so that both ends produce the same key, which might prove complicated to scale on the Server side.
10. Lightweight Symmetric Cryptography: authentication and encryption have been proposed to be added using lightweight cryptography algorithms like ChaCha20 [39] as a cipher, and Chaskey [40] as a MAC algorithm [41]. While these algorithms can be run directly in software in really constrained devices, it lacks the protection that could be provided by using a Secure Element (e.g., impersonation, key stealing). Morevoer, the authors do not mention any key derivation algorithm. If the key is statically set and used, this might leave the door open for replay attacks or key reconstruction.
11. LINE Beacon: specification from the LINE Corporation to deploy beacon devices compatible with their services [42]. It incorporates a 7 byte secure message that is generated from concatenating information from the device (timestamp, battery level, and generated MAC) and applying XOR operations three times to it. This approach, much like the Estimote Secure UUID, requires that a 3rd-party server verifies the secure message, which might introduce other vulnerabilities.

Table 1 shows the design goals, security properties, airtime overhead, power budget and main advantages and disadvantages evaluated for some of the previously mentioned remediation techniques in comparison to our Secure iBeacon solution.

As it can be seen, most of these solutions focus on deflecting possible attackers or jammers, rather than on integrating encryption or authentication mechanisms. In addition, while other alternatives to traditional Beacons exists, like Bluetooth Mesh which includes security by using a Diffie–Hellman challenge for key derivation between the nodes [43], they still lack a widespread implementation and adoption [44]. Other custom alternatives like the one proposed by Casanova-Marquez et al. provide private and secure position for users using a custom encryption protocol [11], while others rely on the use of Public Key Infrastructure (PKI) and asymmetric keys like RSA or ECC in order to encrypt the data and verify the message integrity [45,46]. However, these approaches introduce some challenges due to the complexity and cost associated with the need to distribute and maintain the certificates and keys, as well as handling their renewals and revocations, which can hinder scalability and bring on some other security vulnerabilities, especially when the End Devices are distributed and the certificate management cannot be performed over the air. Lastly, in some cases, the memory of the End Device is not protected and it can be easily accessed, so an attacker might be able to modify or read the contents. In this context, this work focused on the use of already established authentication schemes, like the OTAA specified

Table 1
Comparison table for BLE security mechanisms and our Secure iBeacon solution.

Aspect	Eddystone EID	Estimote Secure UUID	Bluetooth PAwR	BP-MAC	Lightweight PKI	Rolling Codes	Lightweight Symmetric Cryptography	LINE Beacon	This work
Design goal	Provide privacy to Beacon devices to reduce tracking	Provide privacy to Beacon devices to reduce tracking	Two way communication over BLE periodic advertisement with security	Fast message authentication with low overhead	Provide authentication for BLE devices	Use of Rolling Codes to protect iBeacons against attacks	Use of ChaCha for crypto operations	Corporate package formatting for applications	Provide Bluetooth Beacons with security, privacy via a lightweight procedure
Security properties	Privacy, protection against spoofing /cloning	Privacy, protection against spoofing/cloning	Encryption, authentication confidentiality. Requires previous Bluetooth connection	Message integrity, authentication	Authenticate the End Node, Privacy, encryption, and integrity for iBeacon packets	Provide authentication and privacy to packets	Provide authentication and privacy to packets	Provide integrity authentication to packets	Authenticate the End Node, Privacy, encryption, and integrity for iBeacon packets
Airtime overhead	No overhead vs. Eddystone Beacons, packets $1B$ bigger than iBeacon, so more airtime needed	No overhead as normal iBeacons are used	No overhead as normal Beacons or extended Beacons are used	Depending on MAC length and how it is parsed onto the packet	Designed for BLE, high overhead for authentication.	No overhead as normal Beacons are used	No overhead as normal Beacons are used	No overhead as normal Beacons are used	No overhead as normal iBeacons are used
Power budget	Increased consumption from AES crypto key derivation and ECC communication with key provider	Overhead from the symmetric encryption and decryption	More consumption from having to advertise and listen in the sort intervals. Overhead from encrypt/ decrypt	Negligible after pre-calculations	Increase in power consumption due to the long authentication process	Negligible increase in power due to crypto operations	Negligible increase in power due to crypto operations	Negligible increase in power due to crypto operations	Increase in power consumption due to the use of a Secure Element
Advantages	Open specification, implementations available.	Easy to manage, SDK available.	Supported by many recent Bluetooth devices	Reduced delay for transmission and reception	Highly compatible with BLE devices	Easy implementation, low consumption	Easy implementation, low consumption	Easy to manage	Fast, lightweight and highly compatible, secured with HSM
Disadvantages	Currently unsupported, need for pseudo synchronization	Use of external server, controlled by private entity	Increased power consumption	Memory consumption, time needed for precalculations	Very consuming authentication due to packet size, complicated PKI management.	Not implemented, need for pseudo-synchronization, vulnerable to replay and interception attacks	Static keys, no PFS, no support for SE due to crypto operations	Use of static keys, managed by private entity	No Perfect Forward Secrecy (PFS)

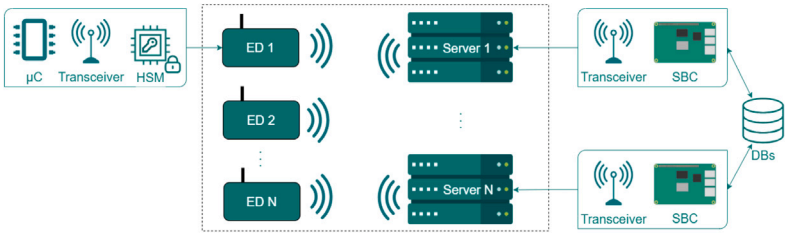

Fig. 6. Communication scheme and devices used in the Secure Beacon system.

in LoRaWAN, to provide security to iBeacons. We propose an over the air End Device activation process to provide privacy and integrity to the user's data. By using a Secure Element, we can provide a Root of Trust to the system, further increasing the security against tampering and impersonation. This developed solution provides certain benefits with respect to previous implementations. In particular, given that every packet is encrypted and that the integrity is guaranteed thanks to the use of MIC and counters, the task of spoofing the beacon packets or replaying them is considerably more complicated for an attacker. Thanks to the integration of the Secure Element, node impersonation is hindered as the symmetric key stored inside cannot easily be stolen by an attacker. Compared to solutions based on PKI, our implementation is more scalable and easier to maintain, while achieving a high level of assurance and security. Thanks to the use of symmetric encryption and not PKI, the power consumption is also reduced. That is, if one end does not have prior knowledge of the Beacon, it would require for the certificate to be sent, which due to the packet size would need many transmissions. This would increase the consumption, as well as require careful considerations to arrange for out-of-order packet arrival to parse the certificate. While there are solutions to these issues, such as only sharing the public key, using certificate pinning or an out-of-band mechanism, implementing these would considerably increase the complexity of the system and reduce scalability.

3.2. Contributions

To address the lack of privacy and security in existing implementations, this work presents a novel authentication mechanism for IoT End Devices in a star topology network. Our solution is based on an already established authentication mechanism to bring security features to iBeacons intended for any general application. In particular, our contributions include:

1. We propose a novel Beacon based authentication mechanism for End Devices that can provide integrity and privacy to devices communicating data. Although the focus in this case is on iBeacon, it can be extended to any of the previously mentioned Beacon format by rearranging the information being sent according to the specific beacon format.
2. We evaluated the proposed mechanism using a SE and compared it to a software implementation to demonstrate its low impact in performance and its benefits by experimentally testing the power consumption and execution time of the proposed solution under different conditions. The scalability and regulatory compliance of the proposed solution is also shown.
3. We define standardized iBeacon advertising packets formats for custom applications, so that data can be exchanged with guaranteed integrity and privacy.
4. We present a real geofencing application where the potential of the proposed solution is demonstrated.

4. Framework and instruments

The different elements that make up the proposed Secure Beacon system are depicted in Fig. 6. As it can be seen, we have considered the traditional BLE beacons star topology, in which the BLE beacons (End Devices) are received by the those BLE Scanners in range. In case an extensive area needs to be covered and one Server is not enough, other servers can be added and connected to the databases. Each of these elements carries out a specific function and is composed by different sub-components.

4.1. End Device

The End Device corresponds to any device with BLE capabilities that transmits the information in the form of BLE beacons. This device must be able to execute both advertising and scanning roles in order to perform the End Device Activation process, as it will be shown in Section 5. Once activated, the device behaves as a typical BLE beacon with respect to the communication. Therefore, an End Device generally consists of a microcontroller and a BLE transceiver. In addition, the End Device is provisioned with a HSM for those tasks related to cryptography, key derivation or random number generation. This part plays a key role in the End Device Activation procedure, since by storing the symmetric key used during the activation procedure in the HSM, the chances of the node being cloned or impersonated are drastically reduced as the key is never exposed or returned to the microcontroller.

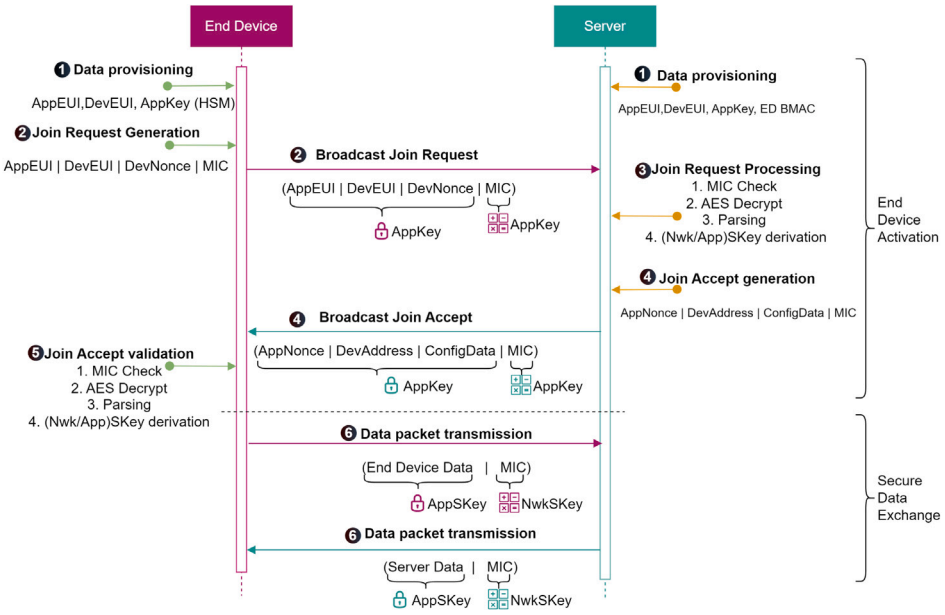

Fig. 7. End device activation flow diagram for the secure beacon.

4.2. Scanner

The scanner is the BLE device in charge of the communication with the End Device during the activation process, as well as of receiving the BLE beacons and send the information to the server. Depending on the application and/or system requirements, the scanner acts as a gateway, transforming the information received over Bluetooth to IP packets that can travel over the Internet infrastructure in order to send the encrypted data to server [47], or it could also communicate with others scanners to reach the server, as done in mesh networks [48].

4.3. Server

The server receives all packets from the End Devices and decodes them. For that, it queries the databases that contain all necessary information related to the End Devices. Depending on the application, both Scanner and Server can be implemented in a Single Board Computer (SBC) as shown in Fig. 6, e.g., a Raspberry Pi, as they provide sufficient functionality and flexibility to implement the necessary services.

5. Implementation details

This section presents the proposed mechanism for the authentication of the End Devices, so-called End Device Activation. Firstly, the procedure is explained in detail, describing the negotiation with the End Device and the different frames exchanged before considering an End Device part of the network. This section also describes the two different BLE beacons formats that have been considered for the communication after the End Device activation. The proposal is validated by means of an experimental implementation, as well as by its threat model.

5.1. End Device Activation

The End Device Activation procedure is based on the OTAA procedure used in LoRaWAN [49], in which the information is encapsulated into the beacon packets, and the security relies on the HSM. Fig. 7 shows the sequence followed for the activation process, which is explained below step-by-step.

1. Data Provisioning: both ends need to be provisioned with some data to detect each other's Beacons and be able to successfully finish the activation process. The data that needs to be provisioned is the following:

   (a) AppEUI: Unique Application Identifier, 64 bits long, used to identify the application.
   (b) DevEUI: Unique Device Identifier, 48 bits long, used to identify each device that connects to the Server. It corresponds to the Bluetooth Device Address of the End Device, which is a fixed global address that must be registered with IEEE [50].

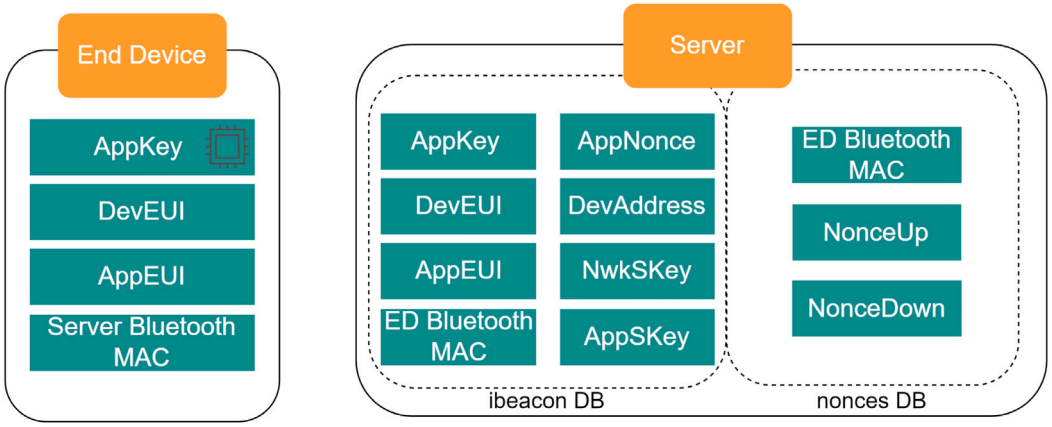

Fig. 8. Data distribution in the End Device (Left) and in the Server (Right).

Fig. 9. Data parsing and operations performed during MIC calculation.

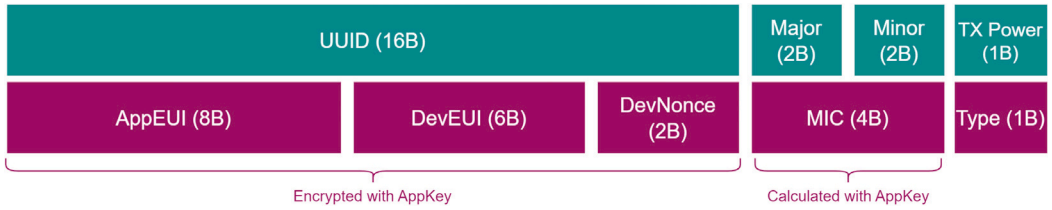

Fig. 10. Mapping of the Join Request message (Bottom) with respect to the iBeacon packet (Top).

(c) AppKey: AES-128 symmetric key that must be kept secret, as it is used in the End Device Activation process and for the derivation of the session keys. It must be stored inside the symmetric key slot included in the HSM, thus making the key much more difficult to be extracted from the End Device by a malicious actor.

(d) Bluetooth Devices Addresses: although this is not a requirement, especially because it could limit the scalability of the networks, the Bluetooth Devices Addresses could also be provided to create a whitelist of devices. This would increase security, as only packets from valid devices would be parsed, as well as reduce the power consumption of the overall system by not processing unrelated packets [27,51].

On the server side, this information is stored in the corresponding database entries. For the End Devices, the AppKey must be securely provisioned and stored in the HSM, while the remaining data (DevEUI, AppEUI and the optional MAC of the Scanner) is stored in flash memory. Given that the information stored in the memory is shared during the activation process, the exposure of both DevEUI and AppEUI does not damage the system's integrity or security. The data provisioning in the Server is flexible, allowing for the information and keys to be generated randomly following the recommendations of the IEEE for Extended Unique Identifiers [52], or provided by the user. The database must store the information associated with each End Device, including the AppKey provisioned in the HSM, its Bluetooth Address as well as the derived session keys (NwkSKey and AppSKey) and the shared cryptographic nonces used to generate such keys. The server also stores a message counter for each device in order to avoid replay attacks [53]. The data distribution in the system can be seen in Fig. 8. Once both ends contain the necessary provisioned information, then they are ready to start the activation process.

2. Join Request Generation: the End Device activation process is triggered by the End Device when it starts advertising the Join Request message. This generally happens when the device is booted up, as the ephemeral session keys are not provisioned yet. For this, the first step is for the End Device to generate a Join Request message by concatenating the AppEUI, the DevEUI and a DevNonce, a cryptographic nonce that is randomly generated using the True Random Number Generator (TRNG) of the HSM at execution time. This concatenated data is then encrypted using the AppKey stored inside of the HSM. Then, the Message Integrity Code (MIC) is calculated so that the Server can check the integrity of the packet and the identity of the

node. This MIC is based on the AES-CMAC standard [54], and it is calculated using the AppKey. As shown in Fig. 9, the data used for the first round of the MIC includes data related to the packet and the End Device as well as some static bytes, as done in LoRaWAN:

(a) confFCnt: this value represents the Confirmed Frame Counter. This field is used to indicate that the server must acknowledge the message by sending back a confirmation downlink. Although these parameter is not used in the implementation shown in this work, we have considered it for future uses.
(b) Dir: this represents the direction in which the message is being sent, with a value of "$0x00$" if the message is sent Uplink, and a value of "$0x01$" for Downlink messages.
(c) DevAddr: it represents the device address that is given by the server during the End Device activation. For the first messages exchanged during the activation, since it is not provided until the Server sends the JoinAccept packet, the DevAddr is set to zero.
(d) FCntUp/Down: this is the frame counter used to keep track of the number of packets sent and received. It provides protection against replay attacks, where the integrity of the message will not be maintained if the counters are modified.
(e) len(payload): last, the size of the data to which the MIC is going to be calculated is appended to the block.

This first block is used as input to the AES-CMAC algorithm. Once the first CMAC is obtained, the second block of data containing the payload data is added to the CMAC calculation. Then, the result of the calculation will be a 16 byte long verification code.
Note that only the four most significant bytes of the MIC are used for the integrity check, as is the case of LoRaWAN and Bluetooth specifications [55]. This is an acceptable compromise between overhead and security, since due to the low packet size and data rate of an iBeacon network, it is unlikely for an attacker to be capable of produce a valid MIC and encrypted packet combination. Then, the Join Request will be 16 bytes long of data, obtained from concatenating the AppEUI, DevEUI and DevNonce, plus the 4 bytes from the MIC, as shown in Fig. 10. The data is organized such that the 16 bytes occupy the UUID data, while the Major and Minor contain the MIC calculated. Last, the Type flag is set to the value of "$0xFF$" during the advertisement to signal it is a Join Request packet to the Server. This Join Request packet is generated and advertised periodically, leaving some free slots to listen for the Server response. This process of emitting-then-listening will be performed for as long as the device does not have the session keys, although a maximum number of retries could also be considered to avoid unnecessary power draw when the server is not reachable.

3. Join Request processing: the Server is continually listening for iBeacons, and when it detects a packet from one of the pre-provisioned Bluetooth Addresses with the flag "$0xFF$" in the Type field, then the Server proceeds to parse the received data, dividing the packet into the data for the activation process and its MIC. The first step is then to check the integrity of the message by recalculating the MIC. With the information that the Server has received, the stored AppKey and the data from the databases, this recalculated MIC can be compared to the MIC that was received. With this, the integrity of the packet can be assessed, as well as the identity of the End Device. Once the packet's integrity has been verified, the packet is decrypted using the AppKey and the data is parsed to separate the AppEUI, DevEUI and the nonce. Both Network Session Key and Application Session Key (NwkSKey and AppSKey) are then derived using the received information and the information stored in the database. For this, first the Server concatenates a particular byte ($0x01$ for the network session key or $0x02$ for the application session key), the AppNonce (a nonce generated at runtime by the Server and stored subsequently in the database), the AppEUI and the DevNonce sent by the End Device in the Join Request message. Finally, the key is derived by simply encrypting the concatenated data using the AppKey, as shown in Fig. 11.
Note that these session keys are never sent to the other end, but are rather stored in the database to code/decode subsequent messages. Regarding the session keys, the NwkSKey is used at the network level to verify the integrity of packets with the MIC. On the other hand, the AppSKey is used to encrypt/decrypt the data at an application level. This allows for data to be received by a Bluetooth scanner and, if necessary, securely and privately relay encrypted data to another Server over the Internet.
4. Join Accept packet generation: after the Join request is processed, the Server can generate the Join Accept packet. For the generation of this Join Accept, the following values are concatenated: the AppNonce, a nonce used for the regeneration of the session keys in the End Device, the DevAddress, used to identify the End Device, and last the ConfigurationData field, used to modify the advertising settings of the End Device. The mapping of this data to the iBeacon packet is shown in Fig. 12.
The two first values of the Join Accept (AppNonce and DevAddress) are used by the End Device to derive the same session keys as in the Server, while the ConfigurationData configures the following settings:

(a) Advertisement interval ($2B$): it indicates how many slots of 0.625 ms (the time window used in Bluetooth advertising) to leave empty between advertisements. The advertisement interval is in the range of 20 ms to 10.24 s [56]. The lower the value, the more messages are sent per second, which also leads to higher power consumption.
(b) Listening interval ($2B$): signals how many slots of 0.625 ms to wait between listening for packets coming from the Server. The higher the value, the less sensitive the device is to receiving data from the other end.
(c) Channels ($1B$): Beacons are transmitted in the Bluetooth channels 37, 38 and 39. This field allows the server to indicate the End Device the channels to use for advertising the beacons.
(d) Tx Power ($1B$): transmission power in dBm. It can be used to adapt the power consumption according to the distance to the Server or to calculate path loss.

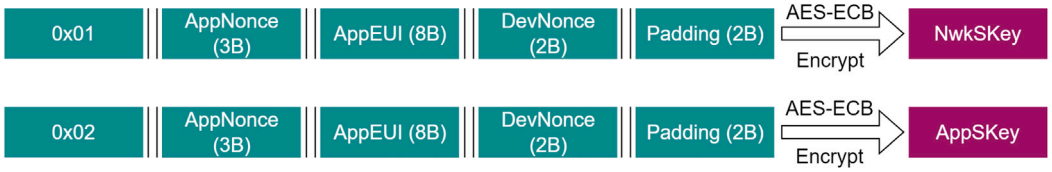

Fig. 11. Scheme followed for the derivation of the session keys.

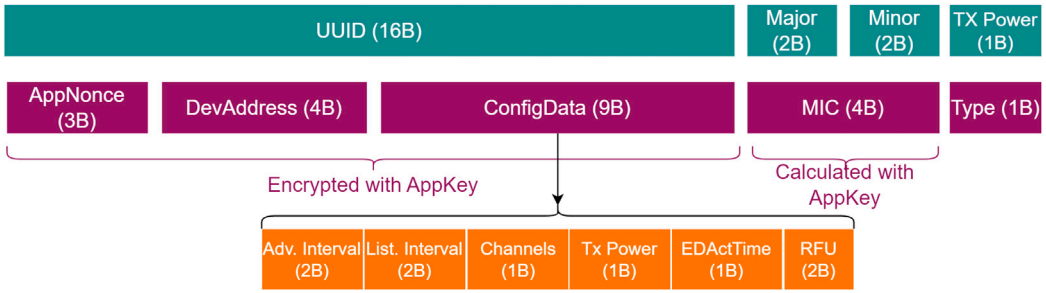

Fig. 12. Mapping of the Join Accept message (Bottom) with respect to the iBeacon packet (Top).

(e) EDActTime ($1B$): this can be used to signal the End Device how frequently it needs to perform the End Device activation, for example every 12 h or every 24 h.
(f) RFU ($2B$): this section has been Reserved for Future Use (RFU) and should be left empty.

All this data, once it is concatenated, will be encrypted using the AppKey and then have the MIC calculated using the procedure in Fig. 9 with the updated corresponding values. Then, the Server advertises this Join Accept packet, where the Type is set to "$0xDD$" to signal to the End Device that it is a Join Accept packet.

5. Join Accept validation: the Server advertises the Beacon containing the JoinAccept message for that particular End Device. While the Server can differentiate the packets coming from each End Device by the Bluetooth Address that is stored inside of the database, the End Device uses the end byte of the packet to determine if it is a Join Accept packet. Note that, even if multiple devices are being activated simultaneously, only the one with a valid MIC will finish the process successfully after assessing the integrity of the message. That is, if an End Device can decrypt the frame correctly and verify the MIC using their AppKey, then that End Device was the addressee of the frame. This process is the same as the one performed by the Server for the validation of the Join Request, so the MIC for the received packet is recalculated using the AppKey. If the MIC matches the received one, the packet has not been modified during transit. Then, the End Device can use the AppKey again to decrypt the packet's content. The received information is used to configure the advertising settings accordingly and to regenerate the session keys using both the AppKey stored in the HSM and the information received from the Server via the Join Accept message. This process is analogous to the one performed by the Server, so both ends should have derived the same keys. Note that these session keys are not stored in the HSM, as they are short lived.
6. Information transmission: finally, after the session keys have been obtained, the End Device and Server can periodically transmit secure BLE Beacons encapsulated into iBeacon frames with up to 16 bytes of data, as depicted in Fig. 13. The process to generate these secure frames starts by encrypting the data, which is produced by first concatenating relevant metadata about the packet, such as the Device Address, the direction of the communication (uplink or downlink), and the corresponding counter value for the message. This formatted data is encrypted using the AppSKey. By performing the XOR operation of this encrypted metadata with the plain data to be sent, the encrypted data is obtained. [57]. From this encrypted data the MIC is calculated using the NwkSKey, as shown in Fig. 9, taking only the 4 most significant bytes. This information is then mapped onto the iBeacon packet as shown in Step 3 to be advertised. In this case, it was chosen to use a Encrypt-then-MAC scheme, i.e., first the plain data is encrypted and then, over the encrypted data, the MIC is calculated. This provides the most ideal security scenario, as it allows for fast check of the integrity of packets (in MAC-then-encrypt, first the decryption must be performed, which consumes time) [36,58]. This alternative also provides integrity for both the plain text and cyphertext. Last, by calculating the MAC after encryption, provided that the output from the encryption is random enough, no information about the structure of the data is provided to anyone sniffing the packets contents [59].

5.2. Secure Beacon example modes

To exemplify the broad spectrum of applications of the developed secure communication mechanism, we propose two schemes for the data transmission at application level (Fig. 14), which could be used depending on the final use case:

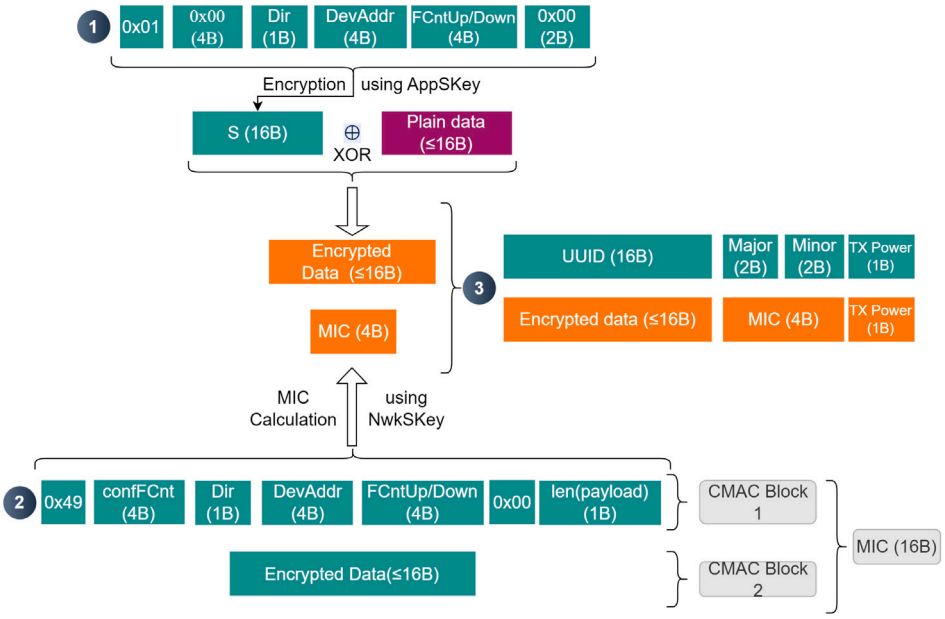

Fig. 13. Generation and mapping of a Secure Data Packet: 1. Encryption of the Data. 2. Calculation of the MIC. 3. Mapping of the data onto the iBeacon Packet.

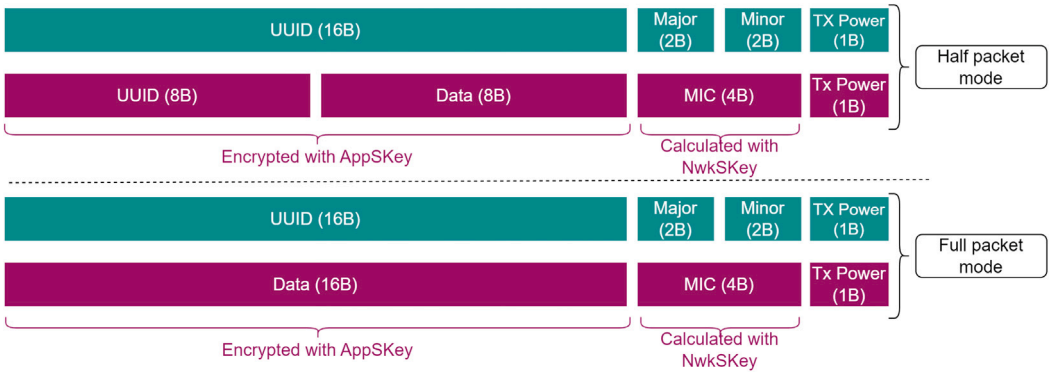

Fig. 14. Mapping of data in the half packet mode (Top) and the full packet mode (Bottom) with respect to the Beacon packet.

1. Secure Beacon half packet mode: in this mode, the 8 most significant bytes (MSB) of the UUID are maintained to signal the particular application, while the 8 least significant bytes (LSB) are used to transmit encoded data. The MIC is, analogously to the Join Request message previously shown, stored in the Major and Minor. In this case, a total of 8 encrypted bytes can be sent. The information regarding "Tx Power" is maintained to keep the compatibility with those applications that requires the positioning of devices/users.
2. Secure Beacon full packet mode: in this mode, the full UUID is replaced to transmit data. This does not hamper the ability of the server to detect who or which End Device is sending the data, as they can be identified using the Bluetooth Address. The MIC is still stored in the Major and Minor sections of the packet. This mode allows for up to 16 bytes of data to be sent simultaneously, and as in the previous mode, the "Tx Power" information is maintained for positioning.

5.3. Technical implementation

Once the fundamentals of the implementation have been provided, the following sections present the components that have been used to validate the proposal, which are schematized in Fig. 15.

5.3.1. Hardware Security Module

In this case, the SE acting as HSM in the End Devices (EDs) is the OPTIGA™ Trust M v3, manufactured by Infineon Technologies AG, which is based on the Common Criteria EAL 6+ certified hardware and counts with PSA Level 3 certification. It supports ECC,

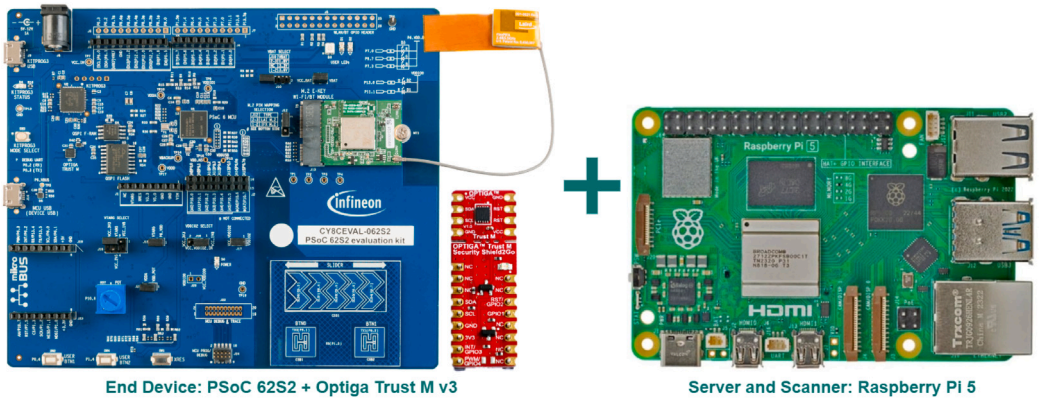

Fig. 15. Hardware used for the implementation.

RSA, and AES keys, as well as hashing functions such as Secure Hash Algorithm (SHA) and Hash-based Message Authentication Code (HMAC). Regarding the storage of keys and device secrets, the OPTIGA™ Trust M can hold up to four certificates and ECC keys, two RSA keys, and includes a dedicated slot reserved for an AES key. It is a perfect candidate for IoT designs due to its low form factor and power consumption [60].

5.3.2. Microcontroller

The Programmable System on Chip (PSoC™) 62, a dual-core microcontroller also from Infineon Technologies AG, was used to implement the functionality of the End Device. In particular, we have used the Evaluation Kit PSoC™ 62S2 model CY8CEVAL-062S2 [61]. In this case, the PSoC™ comes with a Bluetooth transceiver that makes it capable of advertising and listening to BLE frames. In its flash memory, it will contain part of the data used during the activation process, but as explained before, for increased system reliability, the symmetric AES-128 key denominated AppKey will be stored in the OPTIGA™ Trust M. For the End Device, the implementation was carried out in C programming language using FreeRTOS abstraction layer to integrate the functionality, so the porting of the implementation to another microcontroller platform should only depend on adapting the drivers for the Bluetooth transceiver.

5.3.3. Server

The whole functionality of both the Scanner and the Server are integrated using a Raspberry Pi 5, which also contains the database service running. The Scanner and Server functionalities has been implemented in the programming language Go, which is compatible with virtually any Linux running device, provided that the Bluetooth transceiver is compatible with the BlueZ library [62].

5.4. Implementation benefits and drawbacks

This approach presents and combines some of the benefits that can be found in Bluetooth and LoRaWAN. From the latter one, it includes the ability to send data with ensured privacy and integrity, while also authenticating the other end of the communication. However, unlike LoRaWAN, this approach only covers an area of a few hundred meters from the Server. From Bluetooth, it has the benefit of its universality: almost every System-on-Chip (SoC) nowadays has Bluetooth capabilities and can be turned into an End Device. Servers can be either custom made implementations, or simply any laptop, Single Board Computer (SBC) or PC running Linux, given that the Server implementation is compatible with many platforms. Due to the way the approach has been designed, it does not require Bluetooth devices to pair and connect to share data, a process that can suffer from man in the middle attacks. Lastly, unlike in LoRaWAN, Bluetooth allows for major throughput, where End Devices can send much more packets per unit of time, while still maintaining a low power consumption. One common limitation for the discussed implementations in Section 3 and ours is that Perfect Forward Secrecy (PFS) is not provided. PFS ensures confidentiality of all previous communications even if the keys are compromised. This means that it would be possible for an attacker to harvest all available encrypted messages now, and decrypt them in the future when Quantum-Computers are available, in what is known as a "Harvest now, decrypt later" attack. Note that for this attack to be effective, an attacker would need to be able to extract or recreate the symmetric key securely stored inside of the Secure Element. If the key is never leaked, then an attacker could not recompute past session keys and decrypt the contents.

5.5. Security analysis

The use of the End Device activation mechanism developed and described in this paper allows for:

1. Confidentiality: the information will not be accessible by unauthorized parts thanks to the encryption of the data using the AppSKey.

2. Integrity: if the data is altered during transmission, the other end can be aware of the fact thanks to the MIC that is added to every packet, ensuring there is no tampering.
3. Authentication: the identity of the End Device is verified during the End Device activation process when it provides the Join Request encrypted and with the corresponding MIC to the Server. If the MIC validates, and the Tuple of AppKey, DevEUI and AppEUI matches what is stored in the Server, then the device is genuine and should be provided with a Join Accept message to generate the session keys. Due to the use of the HSM, only a genuine device that has control over it would be able to answer the challenge of the End Device activation, guaranteeing that the identity and authentication of devices is strong.
4. Privacy: The End Device's information that is being advertised would be difficult to track at an application level. Due to the use of AES in counter mode, even if the same information is sent in every packet, the cyphertext obtained and advertised will be different each time. In a similar fashion to Eddystone EID, an attacker trying to gather information from the End Device will have a much harder time trying to track them due to the changing UUID.
5. Non-repudiation: in this mechanism, the process that generates the packet ensures that the sender cannot deny having sent it, as the packet contains proof of the sender's identity. However, the same level of non-repudiation is not guaranteed for the receiver. Since the mechanism relies on a best-effort approach, there is no strict guarantee that a packet will be successfully delivered to the intended recipient.

Regarding the attacks that the architecture might suffer, we consider the following adversary model: an attacker that has physical access to an End Device and a node with the capability to act as a Server, that is capable of affecting the Bluetooth spectrum, and that is capable of attacking the Server infrastructure. This malicious adversary would be able to eavesdrop the iBeacon traffic, of jamming, composing and replaying packets, as well as decrypting them if they have access to the key, but they are computationally bounded. If they have access to the End Device or Server, they are capable of performing also side channel attacks. The Server is protected against physical and remote attacks by following best practices: end-to-end encryption using TLS, robust key management in a secure vault or HSM, strong authentication and authorization, use of Firewalls, Secure Boot, and continuous monitoring, among others. In the following sections, a description of the threat model based on the STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege) methodology is presented, with the attack tree presented in Fig. 16 [23,49,63–68]:

5.5.1. *Spoofing*

1. Device cloning: an attacker that is capable of reading the AppKey from the microcontroller's flash memory would be capable of cloning the device and performing the End Device activation by themselves and transmit malicious information. Thanks to the use of the HSM, which never exposes the symmetric key, cloning a device is an unlikely event.
2. MiTM attacks: an attacker which can extract the AppKey from the device and that is capable of extracting the data from the End Device activation using traffic analysis, would be capable of independently obtaining the plain text being sent by the End Device for that session or even forge their own packages. Due to the use of an HSM that stores the AppKey, an attacker obtaining the AppKey is unlikely. However, an attacker with physical access to the device might be capable of dumping the generated session keys and use them until a new End Device activation is performed. If the attacker is capable of obtaining the NwkSKey, they would be capable of introducing their own packets onto the network, while with the AppSKey they would be capable of decrypting any information sent. This could be solved by storing these session keys in a memory protected area so that they cannot be easily extracted, like ARM TrustZone, or by reducing the period that each key is valid for, so that re-keying is necessary more often, rendering the attack less effective.
3. Compromised Server: a malicious device can pretend to be the Server, faking the Join Accept messages so that End Devices mistakenly connect to them instead of to the legitimate Server. This attack is mitigated by the use of encryption and MAC in both the Join Request and Join Accept: a malicious attacker cannot simply generate or replay a Join Accept, as they would require to know the AppKey.

5.5.2. *Tampering*

1. Bit flipping: this attack affects implementations where the Scanner sends the encrypted packet over the internet to another Server. That is, the Scanners are used as Bluetooth gateways that gather the relevant iBeacon packets from their surrounding, verify their integrity and then proceed to relay the encrypted packet content to an Application Server over the internet. If this communication is performed over an insecure channel (e.g., over plain HTTP), an attacker could flip a bit in the AES CTR mode encrypted data and provide a wrong value to the Application Server. This issue can be solved by also relaying the MIC to the Application Server, which can check the integrity again on reception, or by simply using a TLS protected communication between the Scanner and Server to avoid modifications.
2. MiTM attacks: in this case, the attacker is interested in modifying the content of the packets. This type of attack would require knowing the session keys, as well as the counters and unique addresses of each device, which, if stored in a protected memory region, would prove difficult for the attacker to extract. The duration of the attack is also limited, as it would only be valid for the current duration of the session keys.
3. Compromised Server: In this case, the objective of the attacker is to introduce malicious packages to disrupt the functioning of the infrastructure (e.g., malicious Join Accept with wrong configuration for the End Devices). If the Server has good platform security as discussed before (Secure Boot, TPM, TLS used,etc.) these attacks are difficult to realize for an attacker.

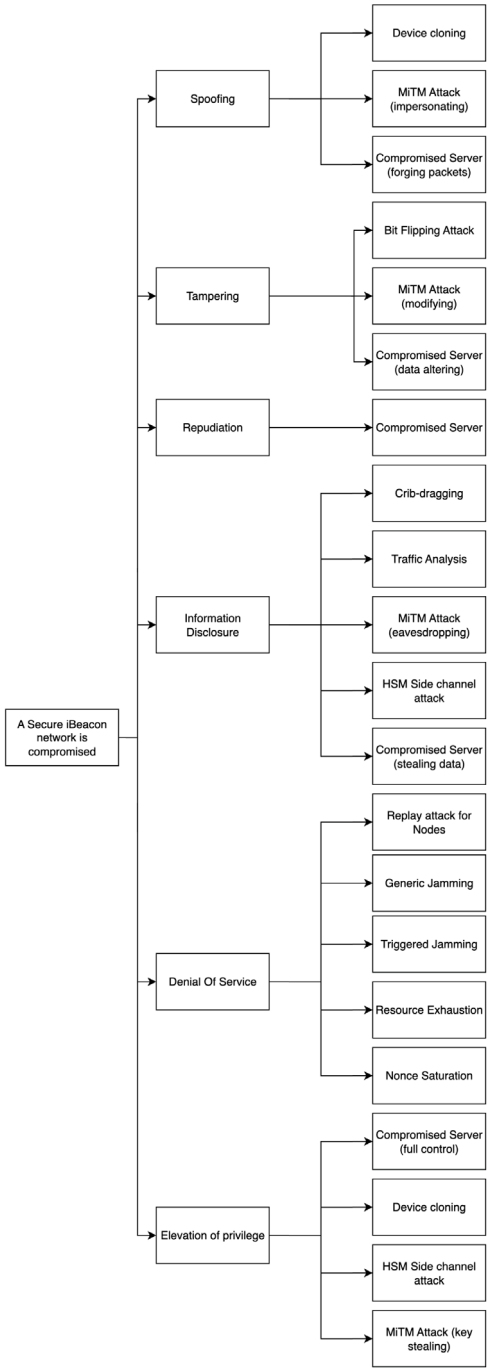

Fig. 16. STRIDE-Mapped Attack Paths Against the presented scheme.

5.5.3. Repudiation

1. Compromised Server: an attacker that controls an End Device might be able to transmit messages and then deny the participation in the exchange. However, due to the use of an HSM that is the only one that contains the AppKey, this cannot happen during the Join Procedure, and the Server can have assurance that if the cryptographic operations verify correctly, the message was produced by the End Device with the HSM. This is also applicable to the session keys, provided they are stored in a secure memory location.

5.5.4. Information Disclosure

1. Crib-dragging: The confidentiality of packets after the End Device activation is provided by using AES-128 CTR mode. As shown in Fig. 13, the packet counter is used as an input for the encryption. Given the use of monotonic counters, if there is a rollover, then the algorithm will produce again the same cyphertext. This would open the door to DoS attacks, as invalid messages that an attacker might have captured become valid again and can be spammed at the Server. Two solutions could be implemented for this: replace the counter value by a nonce, or force reactivation when the counter is reaching its final values. The first solution would introduce too much overhead on embedded devices, as storing all the used nonces would consume flash storage, while the second option has already been considered and is enforced using the Join Accept message, which sends information to the End Device of how many hours the key is valid for and, if the time has elapsed, the End Device activation process needs to be performed again. Do note that, with the 32-bit counters used for the uplink and downlink, around 4 billion messages could be sent per session before the counter would roll back.
2. Traffic Analysis: packets in an iBeacon network can be captured an analyzed in a trivial manner by many Bluetooth enabled devices. Many software tools could be used to analyze the traffic, like Wireshark. Traffic analysis could be used by an attacker to try and de-anonymize users or in tandem with other attacks to detect when to jam the channel to avoid a particular user being correctly served. However, do note that with the encryption used, the confidentiality of the data is guaranteed, and the attacker would not be able to extract the plain data nor relevant metadata.
3. HSM Side channel attack: a microcontroller whose keys are stored in memory could suffer a side channel attack to try to obtain the key. The same scenario could be performed on an HSM to try and extract the key. In this case, due to the chosen HSM being certified by the Common Criteria EAL6+, advance protection against side channel attacks and fault injection have been considered in the design.
4. Compromised Server: in this case, the attacker tries to steal the data from the End Devices after decryption in the Server. If good platform security is exercised, then the attack would not be viable for the attacker.

5.5.5. Denial of Service

1. Replay Attack: these types of attack try to consume resources from the Server so that legitimate users cannot be served properly. It can be realized by constantly replaying advertisement packets to the Server: a captured Join Request packet from an End Device can be advertised to the Server, which would consume resources and air time to answer back to the attacker. These types of attacks have been minimized by two design choices: first, both the JoinAccept and JoinRequest packets are encrypted, and the decrypted information needs to match the DevEUI and AppEUI in the databases. Second, when a DevNonce is used for a particular End Device, it is marked as used. With this, following attempts to perform a Join Request with malicious intentions go unanswered by the Server. End Devices can also be ignored by the Server if they have been advertising Join Request packets at a high rate as a way to limit them. Do also note that the use of the uplink and downlink counters used for data packets makes it so that replay attacks after the End Device activation are more difficult to realize: if the incorrect counter is detected multiple times for different packages in succession, the Server can simply drop the packets and not reply.
2. Jamming attack: an attacker that controls one or several Bluetooth enabled devices could advertise packets on all the advertisement channels, which might cause increased collisions and degraded user experience. The packets sent do not need to contain any useful or compliant information, as their sole purpose is to occupy channel airtime. We can distinguish two types of jamming: generic jamming, which advertises at all possible intervals, to try and degrade as much as possible the experience of the users, which would be easy to detect.Moreover, triggered jamming happens only under certain circumstances, for example, an attacker performing traffic analysis could detect when a particular End Device is advertising (for example, sending the Join Request) given its Bluetooth MAC and jam the channels accordingly. This second type of jamming is more difficult to detect. To reduce such attacks, anomaly detection capabilities could be added to the Scanner, and offending devices should be identified and removed.
3. Resource exhaustion: an attacker that controls several Bluetooth devices could flood the channel forcing the End Devices to spend power on scanning and parsing the unwanted data by performing a Replay attack. To combat this, the End Devices can be limited to only scan and advertise in response to a set of Bluetooth MACs that correspond to the Servers that are known to belong to the Network. An attack that can affect the Servers is the airtime exhaustion, where an attacker performs the replay attacks in an effort to try and make the Server consume as much airtime as possible on them so that it cannot provide service to the legitimate users. Again the solution in this case would be restricting communication to only End Devices whose Bluetooth MAC has been previously provisioned in the Server, as well as not replaying if the DevNonce has already been marked as used.
4. Nonce Saturation: the Server can keep track of the DevNonce used by the End Device, but due to memory limitations the End Device will choose a DevNonce at random. If a DevNonce has already been used, collision could happen and lead to a replay attack or DoS, as an attacker with the captured packets could resend them and they will be validatd correctly. The Server can ignore the Join Request, and wait till the End Device tries to rejoin with a different one, or suspect a replay attack and ignore the device. Both degrade the user experience, making it trivial for an attacker to impersonate an End Device forcing the Server to ignore a legitimate device. To combat this, the use of nonces could be updated to use a monotonic counter for the DevNonce, as it is stated in LoRaWAN 1.1 [69]. This simplifies the management in both ends by not proceeding with the End Device activation if the Nonce is reused, and provides the most efficient use of the 16 bit DevNonce. The session keys used in the system are generated as shown in Fig. 11, where the combination of the AppNonce (24 bits) and the DevNonce

Fig. 17. Testbed used for Secure iBeacon performance measurement.

(16 bits) ensures key freshness, preventing reuse. Even if the DevNonce (which is limited to $2^{16} = 65536$ unique values) wraps around, the AppNonce generated at runtime by the Server and which cannot be reused provides freshness. Do note that, if a key refresh is performed every 12 h, then the DevNonce would provide a unique key for 65536 keys $\cdot \frac{1 \text{ day}}{2 \text{ keys}} \cdot \frac{1 \text{ year}}{365 \text{ days}} \approx 90$ years.

If, regardless, the DevNonce is exhausted or for added security, simply by changing to a new AppEUI (which does not require to modify the AppKey that is stored inside of the HSM, and which can provide challenging on the field) new key values can be obtained.

5.5.6. *Elevation of privilege*

1. Device cloning: an attacker that is capable of reading the AppKey from the microcontroller's flash memory would be capable of cloning the device and performing the End Device activation by themselves and transmit malicious information. Thanks to the use of the HSM, which never exposes the symmetric key, cloning a device is an unlikely event.
2. HSM Side channel attack: due to the chosen HSM being certified by the Common Criteria EAL6+, advance protection against side channel attacks and fault injection have been considered in the design.
3. MiTM attacks: an attacker could target the Server to try and steal the AppKey from the End Nodes. If good platform security is being used, the attack should not be practical to implement.

6. Experimental results

In this section, the experimental results obtained for the technical implementation will be shown. This includes the time taken to perform the activation process, the current consumption, as well as the scalability and compliance with wireless medium utilization.

6.1. *Timing and power consumption*

The feasibility and reliability of the proposed system has been tested through different experiments to obtain relevant metrics, such as current consumption and latency. We used four different End Devices based on the same development kit, as shown in Fig. 17, where one of them was pre-provisioned with the AppKey in the OPTIGA™ Trust M, while the rest used a software crypto library in order to compare the performance of the HSM hardened node against the other ones.

First, Table 2 presents the execution time of the main cryptographic actions. These measurements correspond to the average time and standard deviation of the execution of each function a hundred times, both in software and using the OPTIGA™ Trust M. It can be seen that the OPTIGA™ Trust M takes slightly more time than the software implementation, probably partly due to the serial connection between the microcontroller and the OPTIGA™ Trust M. However, we will show that this slight difference in timing with the software-based implementation does not introduce a signficant delay when performing the activation process with the Server.

A clean measurement of the current consumption of the HSM when performing the cryptographic functions required during the activation process was also conducted. The results are shown in Fig. 18. The time taken and the power consumption was measured during the initialization process (Fig. 18a), during the symmetric encryption and decryption (Fig. 18b and c), and last

Table 2
Comparison of the execution time of the cryptographic functions in the chosen Secure Element and via software.

Function	Time taken by OPTIGA™ Trust M (ms)	Time taken by SW Crypto Lib (ms)
Encrypt	30.1 ± 2.1	11.9 ± 1.4
Decrypt	50 ± 3	12.0 ± 1.5
AES-CMAC	42 ± 3	40.1 ± 1.5

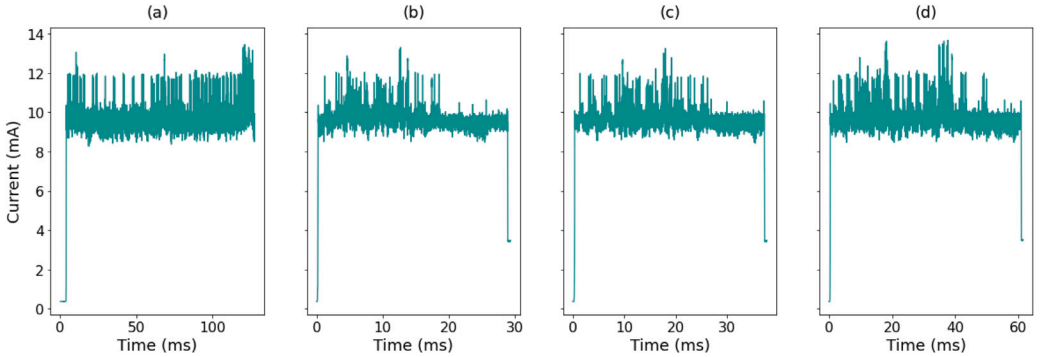

Fig. 18. Current consumption of the OPTIGA™ Trust M during the activation process. (a) Initialization. (b) AES Encrypt. (c) AES Decrypt. (d) AES-CMAC.

Table 3
Time and energy consumption for different cryptographic operations performed by the Optiga Trust M.

Function	Time (ms)	Current (mA)	Voltage (V)	Energy (J)
Encrypt	30.1	9.5	3.3	944 μJ
Decrypt	50	9.5	3.3	1568 μJ
AES-CMAC	42	9.5	3.3	1317 μJ
ECDH Key agreement	60	9.5	3.3	1881 μJ
RSA Encryption	40	9.5	3.3	1254 μJ
RSA Decryption	315	9.5	3.3	9875 μJ

while calculating the MIC using AES-CMAC (Fig. 18d). While active, during the specified times in Table 2, around 9.5 mA are consumed by the HSM. After a successful execution of a function, the HSM falls into sleep mode, with a consumption in this case of around 380 μA. The HSM is in this state until it is required again to perform another cryptographic function. Note that after the successful activation of the End Device, the HSM could be completely disconnected from the power source using a driver circuit, effectively driving its current use to zero.

In Table 3, the energy consumption for the chosen HSM per cryptographic task has been calculated as $E = V \cdot I \cdot t$ using the time measured experimentally, the 3.3 V working voltage, and that approximately 9.5 mA are used while the HSM is active. In addition to the cryptographic functions used in our implementation, we have included in Table 3 the energy consumption of several asymmetric functions that can be performed by the HSM. Typically, asymmetric schemes such as RSA or ECC (e.g., Diffie–Hellman key agreement) are used to exchange partial information and derive a symmetric key. After this derivation, the symmetric key operations consume the time and energy shown in Table 3. As can be seen, asymmetric operations require slightly more energy than the functions used in our proposed mechanism. It is also important to note that establishing and maintaining a Public Key Infrastructure (PKI) for End Devices introduces significant complexity and overhead. This is especially problematic given the low throughput achievable with iBeacons. For these reasons, the proposed End Device activation process offers a more suitable alternative. Considering that the HSM will be turned off after activation, the energy consumed per End Device activation try can be calculated as follows, considering an operating voltage of 3.3 V, an active current consumption of around 9.5 mA, a mean time to execute each command of 40.7 ms (extrapolated from Table 2), and that 10 function calls are done to the HSM during activation:

$$E = P \cdot t = V \cdot I \cdot t = 3.3\ \text{V} \cdot 9.5\ \text{mA} \cdot \frac{1\ \text{A}}{1000\ \text{mA}} \cdot 40.7 \cdot 10^{-3}\ \text{s} \cdot 10 = 12.76\ \text{mJ} \quad (1)$$

Therefore, the increase of consumption due to the use of the HSM is negligible, and the activation process would not particularly reduce the battery life of the End Device with respect to the software version.

The power consumption of the End Devices was also measured under different conditions using the Power Profiler Kit II by Nordic Semiconductor [70]. Note that this instrument allows for obtaining the current consumption that the whole system draws from the power source, so it includes the parasitic consumption from other elements of the development kit, therefore, it is not an

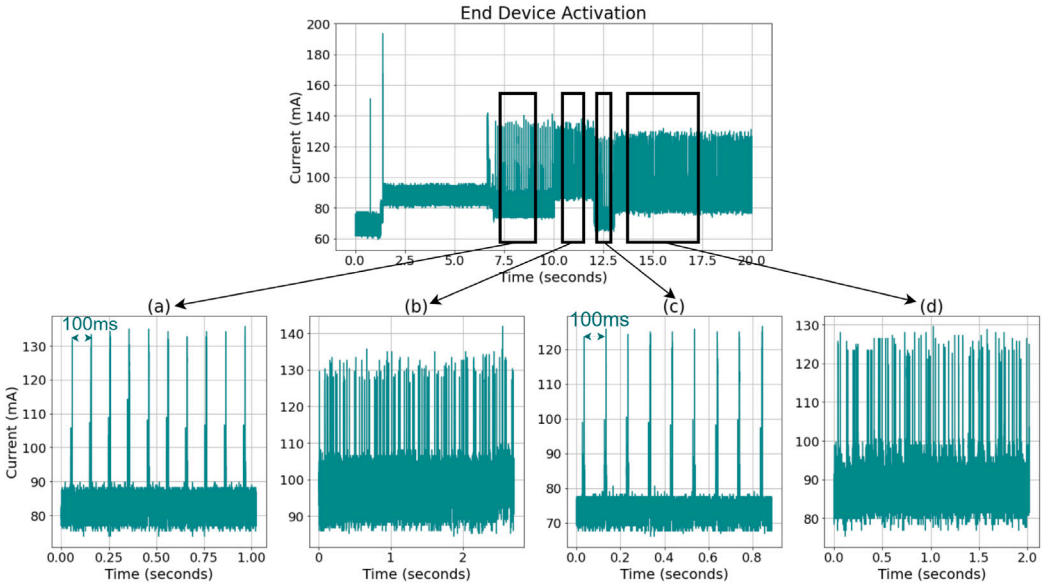

Fig. 19. Current consumption profile of the End Device. (a) Join Request advertising. (b) Scanning mode. (c) Packet advertising. (d) Scanning mode after activation.

Table 4
Average current consumption for the different End Device configurations. A: Advertising, S: Scanning, P: Provisioned.

Power mode	Consumption (mA)	Consumption without MCU baseline (mA)
OPTIGA™ + S	99.30	35.16
OPTIGA™ + A	83.05	18.91
OPTIGA™ + S (P)	90.99	26.85
OPTIGA™ + A (P)	75.08	10.94
SW + S	88.17	24.03
SW + A	74.33	10.19

absolute indicative of the power consumption that might be achieved with a ad-hoc End Device. Then, we measured the current consumption of the End Device during the activation process using both OPTIGA™ Trust M (OPTIGA™ in Table 4) and the software (SW in Table 4) crypto implementations. Table 4 also contains the current consumption of the device when advertising (A) and when it is scanning (S) for both hardware and software security modes. Given that the End Device can put the OPTIGA™ Trust M into sleep mode, or turn it off after the activation process and use the software library with the session keys, the consumption in this state is also provided, which is indicated as provisioned (P) in Table 4. To provide more accurate measurements, the baseline idle current of the MCU on the End Device was measured by running a simple *Hello World* program that prints data over the serial connection. The measured value of 64.14 mA was subtracted from the subsequent power consumption readings.

As seen in Table 4, the use of the OPTIGA™ Trust M increases the consumption around 10 mA with respect to the software implementation. However, also note that after the successful activation of the End Device (P), the OPTIGA™ Trust M enters into low power mode and the current consumption decreases considerably, obtaining a current consumption similar to the implementation based on software. Fig. 19 shows the current consumption profile of the HSM-based implementation at the different stages of the process. In particular, Fig. 19a corresponds to the advertisement of the JoinRequest packet every 100 ms, Fig. 19b is associated with the Scanning mode while waiting for the JoinAccept. The current peaks in that stage are associated with the reception of the surrounding beacons, which may or not correspond with the JoinAccept from the Server. As soon as the JoinAccept is received and all the keys are derived, the End Device puts the OPTIGA™ Trust M into low power mode and start advertising according to the configuration provided in the JoinAccept frame, in this case also at intervals of 100 ms (Fig. 19c). Finally, the End Device could also enter into Scanning mode to receive messages from the Server (although this is not mandatory), as shown in Fig. 19d.

The energy consumed per iBeacon packet emitted was measured for the Bluetooth transceiver used in the system, the Sterling *LWB*95+ [71]. Experimental measurements showed that the on-air time of the transmitted packets is approximately 500 µs, as shown in Fig. 20. The transceiver reaches a peak current consumption of 41.5 mA while advertising, and 20.7 mA when scanning for iBeacon packets. Note that these measurements only reflect the consumption of the transceiver. This means the energy consumed per advertised packet and channel is:

$$E = 3.3\ \text{V} \cdot 41.5\ \text{mA} \cdot 500\ \mu\text{s} = 68.47\ \mu\text{J} \tag{2}$$

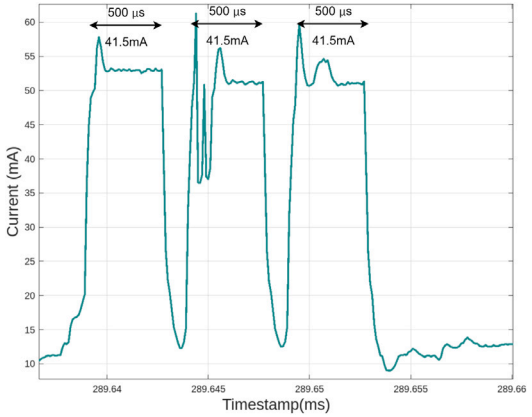

Fig. 20. Current consumption of the End Device during the iBeacon advertisement showing duration and increase in current consumption.

Table 5
Activation time at 1 m distance with up to four concurrent End Devices.

	4 EDs	3 EDs	2 EDs	1 EDs
ED1 (M)	7.7 ± 4.5 s	10.3 ± 3.7 s	11.0 ± 2.2 s	4.22 ± 0.01 s
ED2 (SW)	6.0 ± 0.8 s	5.7 ± 2.4 s	4.6 ± 0.4 s	
ED3 (SW)	4.7 ± 0.5 s	4.5 ± 0.4 s		
ED4 (SW)	5.2 ± 0.7 s			

Table 6
Activation time at 5 m distance with up to four concurrent End Devices.

	4 EDs	3 EDs	2 EDs	1 EDs
ED1 (M)	4.3 ± 0.5 s	5.7 ± 0.5 s	9.3 ± 0.5 s	4.8 ± 0.9 s
ED2 (SW)	4.7 ± 0.9 s	6.7 ± 1.7 s	6.9 ± 2.1 s	
ED3 (SW)	5.7 ± 0.5 s	8.1 ± 0.6 s		
ED4 (SW)	8.5 ± 0.6 s			

And that during the second of scanning it consumes:

$$E = 3.3\ \text{V} \cdot 20.7\ \text{mA} \cdot 1\ \text{s} = 68.31\ \text{mJ} \tag{3}$$

The time required for an End Device to be provisioned depends on the timing configured for both End Device and Server. Increasing the number of JoinRequest and JoinAccept packets also increases the chance of reception. However, note that a single Server cannot resolve all JoinRequest messages at the same time. Therefore, if multiple End Devices are performing the activation simultaneously, they will have to wait for the Server to resolve those JoinRequests that were received earlier. To illustrate this, we have considered an scenario in which up to four different End Devices try to perform the activation process simultaneously. Additionally we have also considered different physical distances to evaluate whether they affect the activation time. For this example, we have considered the following scenario: the End Device transmits the JoinRequest each 100 ms during 3 s and then scans for the JoinAccept for a maximum period of other 3 s. If within that time the JoinAccept is received, the End Device stops scanning, if not, it retries the process. On the Server side, it is continuously scanning. As soon as a JoinRequest frame is received it resolves the request by sending the JoinAccept frame during 6 s. In this case, the Scanner/Server is configured with both Central and Peripheral roles so it can switch back and forth between acting as a central (scanning) and a peripheral (advertising). As soon as a JoinRequest frame is received, the corresponding JoinAccept is sent during 6 s, as the Server can advertise multiple beacons at the same time.

The results obtained for the different distances and different number of nodes can be seen in Table 5, in Table 6, and in Table 7. Times were measured from boot-up, when the system does not have the session keys, until the moment when session keys are derived. For that, the TickCount functionality of FreeRTOS tasks was used. The process was repeated three times to obtain the average value and the standard deviation in all cases.

The average activation time for each case is represented in Fig. 21. As seen, the distance between the Server and the End Devices does not seem to affect the execution time. Also, the timing difference between the OPTIGA™ Trust M and the software implementation, despite the OPTIGA™ Trust M requiring slightly more time, does not suppose a significant impact. In the proposed schema, the activation time of the End Device is in the range of 3–4 s when it is done in the first try. When the number of simultaneous End Device increases, so does the number of retries that some the End Devices requires to perform the activation. As observed, three retries are enough in this scenario to be activated. These values could be improved by using a dedicated Bluetooth

Table 7
Activation time at 10 m distance with up to four concurrent End Devices.

	4 EDs	3 EDs	2 EDs	1 EDs
ED1 (M)	6.3 ± 2.6 s	9.3 ± 0.5 s	11.3 ± 4.5 s	4.8 ± 0.4 s
ED2 (SW)	6.0 ± 1.6 s	5.0 ± 0.8 s	4.5 ± 0.4 s	
ED3 (SW)	5.3 ± 1.3 s	6.1 ± 0.8 s		
ED4 (SW)	4.8 ± 0.4 s			

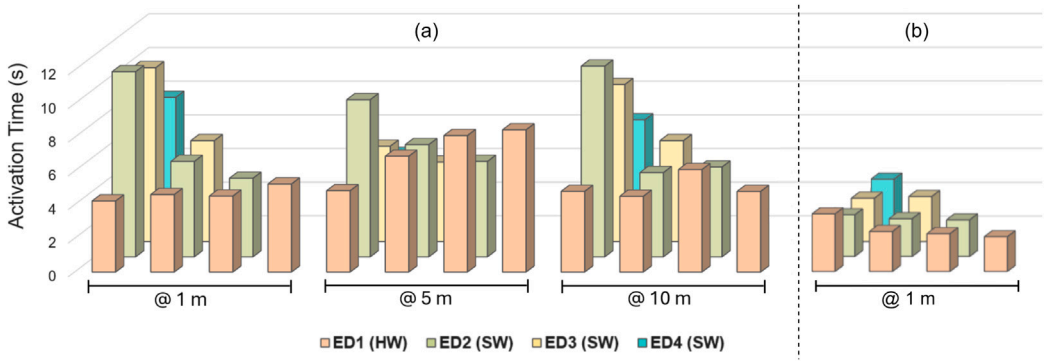

Fig. 21. Average activation time for different of distance and concurrent End Devices. (a) Using one Bluetooth transceiver for scanning and advertising (b) Using a dedicated Bluetooth transceiver in the Server.

Table 8
Activation time at 1 m distance with up to four concurrent End Devices and with a dedicated Bluetooth scanner in the Server.

	4 EDs	3 EDs	2 EDs	1 EDs
ED1 (M)	3.4 ± 1.8 s	2.5 ± 0.5 s	2.6 ± 0.7 s	2.8 ± 1.9 s
ED2 (SW)	2.4 ± 0.7 s	2.2 ± 0.2 s	2.7 ± 0.9 s	
ED3 (SW)	2.2 ± 0.2 s	2.2 ± 0.2 s		
ED4 (SW)	2.1 ± 0.1 s			

Scanner. When using the integrated Bluetooth transceiver of the Raspberry Pi 5, it is only possible to either scan or advertise at any given time. This limitation introduces additional delays during activation, especially when multiple End Devices are involved, due to increased packet loss. To demonstrate this, we used an additional USB Bluetooth transceiver to perform the scanning, giving the Raspberry Pi 5's duplex communication. That configuration allows for a significant reduction of the End Device activation time, as it is shown in Table 8 and Fig. 21. Those measurements were obtained following the same methodology as before at a 1 meter distance.

Apart from the reduction in time needed to perform the End Device activation, we have achieved more efficient usage of the wireless medium, as the advertising interval has been reduced in more than 150% in the End Devices, given the increase of the minimum advertising interval from 100 ms to 250 ms, and in 300% in the Server, as the advertising period has been reduced to 2 s. This shows the potential to further optimize and reduce the End Device activation time. Three solutions could be used to further improve these timings:

1. Optimization of the timing: further experimentation or analytical analysis could help determine the optimal advertising and scanning intervals on both the Server and End Devices to minimize activation time depending on the expected number of End Devices.
2. Multiple Bluetooth Transceiver: using multiple Bluetooth transceivers, each capable of advertising on different channels or operating independently, could provide better access to the medium and help reduce activation time. However, it is also important to evaluate the Bluetooth library limitations for low-level functionalities.
3. Staggered Entry: to avoid an avalanche effect, where multiple End Devices attempt activation simultaneously, leading to a bottleneck in the activation process, it is recommendable to introduce a delay after boot-up. By waiting a randomized or fixed number of seconds before initiating activation, devices can stagger their connection attempts, resulting in reduced packet collisions.

6.2. Scalability and regulatory compliance

To prove the developed End Device activation process and the iBeacon technology that is the base escalates well past the 4 simultaneous users previously presented, we have calculated some metrics obtained statistically regarding collisions and scalability of

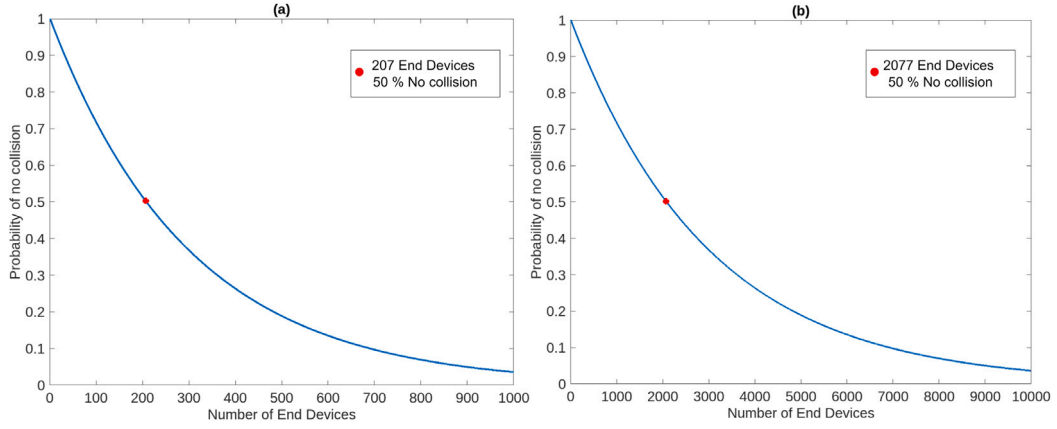

Fig. 22. Probability of no collision at 100 ms advertising interval (a) and 1 s advertising interval (b).

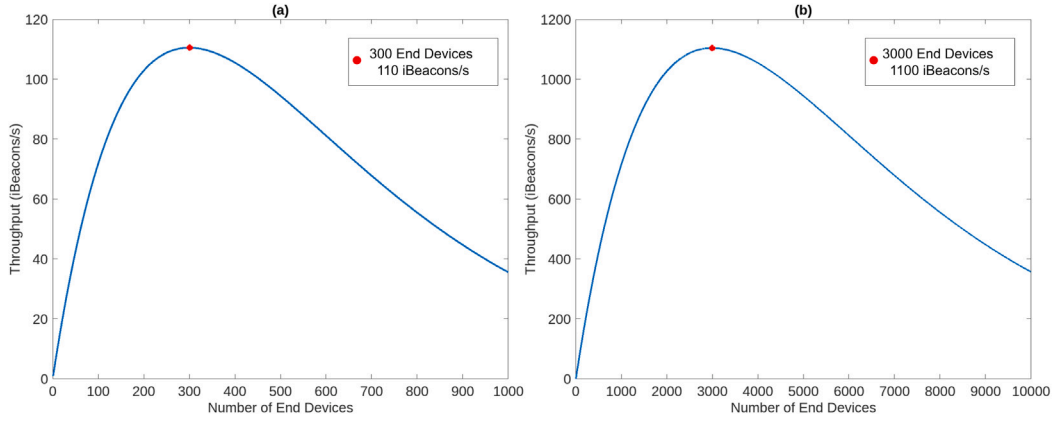

Fig. 23. Throughput at 100 ms advertising interval (a) and 1 s advertising interval (b).

the network. First, the probability of collision of iBeacons was calculated. We assume that a packet is advertised with an advertising interval of "I", and the length on air of the advertising packet is "a". If two packets overlap, then both of them are lost, so there is a window of "$2a$" lost. Since there are three channels, the probability of collision is 1/3 per channel. So the probability of two advertisements overlapping is: $P(overlap) = \frac{2a}{3I}$, and the probability of two packets not overlapping is $P_{no_{overlap}} = (1 - \frac{2a}{3I})$. As each End Device is independent and transmits according to their own schedule, the probability of an End Device not overlapping any other advertisement from other End Devices is the compound probability:

$$P_{no_{overlap}} = (1 - \frac{2a}{3I})^{N-1} \tag{4}$$

where N is the number of End Devices. With an experimentally measured airtime of 500 μs, and the initial advertisement period chosen at 100 ms, with an increasing number of End Devices we obtained the results of Fig. 22, which shows the probability of no collision as the number of End Devices increases.

This means that, with 207 different End Devices, there is around a 50% probability of successfully sending the packet and not suffering from collision. This would mean that, for the particular application, a worst case scenario is that an update is obtained every other 200 ms interval. If the interval is increased to 1 s, then the amount of devices that can send simultaneously increases tenfold to 2077, or again 207 different End Devices could communicate with a probability of no collision of 93%. Last, with 10 s of advertising interval, then approximately 20 700 can communicate with 50% probability of no collision. If the extended advertising channels are also used, which increases from 3 channels to 40 channels used for advertising, this numbers could be improved even further, although these channels might not be fully usable as they carry general Bluetooth purpose data.

The throughput calculated is shown in Fig. 23. At 100 ms of advertising interval, the throughput is maximized at 110 messages that can be sent per second by 300 End Devices. At 1 s of advertising interval the throughput is maximized at around 3000 End devices sending 1100 packets per second.

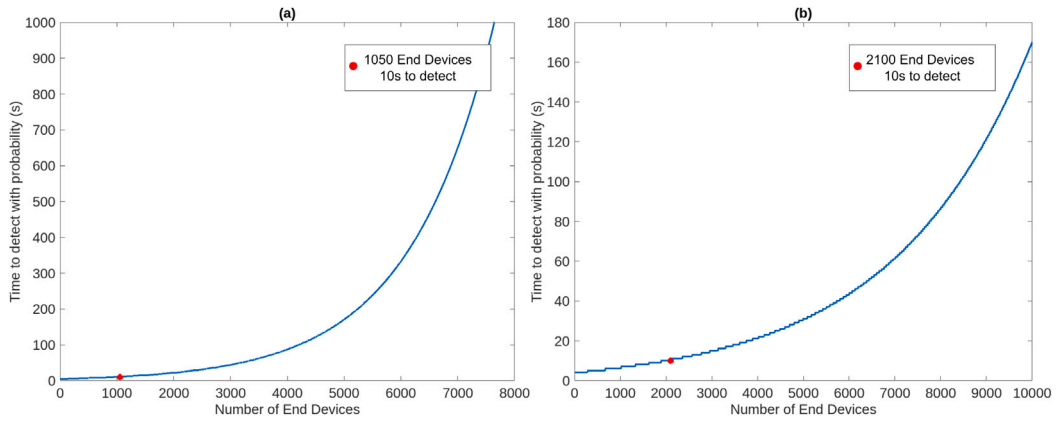

Fig. 24. Time needed to obtain an iBeacon depending on the number of devices with 500 ms advertising interval (a) and 1 s advertising interval (b).

We can also calculate the necessary interval between advertisement to achieve a particular probability of no collision given a number of End Devices as:

$$P_{no_{overlap}} = P_{overlap}^{N-1} \rightarrow P_{overlap} = e^{ln(\frac{P_{no_{overlap}}}{N-1})} \tag{5}$$

If a probability of 75% successful arrival of the advertisement is considered and suppose 1000 simultaneous users, we obtain: $P_{overlap} = 1 - e^{ln(\frac{0.75}{1000-1})} \approx 0.99924$

Substituting the values and solving to obtain the value of "l":

$$1 - \frac{2a}{3l} = 0.99924 \rightarrow \frac{2a}{3(1 - P_{overlap})} = \frac{2 \cdot 500\ \mu s}{3 \cdot (1 - 0.99924)} = 438\ ms \tag{6}$$

So around half a second of advertising period would provide more than a 75% chance of no collision during transmission for 1000 users. The probability that an iBeacon is received within a particular time threshold, depending on the transmission time, can also be obtained. This probability depends on the number of End Devices and the likelihood that packets are missed. Supposing again that 75% of packets are received properly, 500 μs of on air time, 500 ms between advertisements, an increasing number of End Devices and a 99% probability of receiving the packet in the specified time, we obtain Fig. 24, which shows that around 1000 End Devices can transmit every 0.5 s and it is guaranteed at 99% that their message will be obtained by the other end in less than 10 s. If the time between advertisements is increased to every 1 s, then the figure increases to around 2100 End Devices simultaneously emitting [72,73].

Regarding compliance with wireless regulations, ETSI TR 103 665 states that non adaptive devices, like the ones used in BLE Advertising, must not occupy the medium for more than 10% of the time. Note that the scanning time of a device does not enter into effect for this figure. The advertising interval in BLE can be as low as 20 ms or as high as 10.24 s, with a step increase of 0.625 ms. Generally for all devices, the advertising period is set in a bracket between two values, and the device adds some randomness in the time between advertisement to reduce collisions. That is, if the advertising interval is between 500 ms and 1 s, the advertisement can occur in any 0.625 ms window of time between the values. While originally a 100 ms advertising interval was considered, note that this advertising period is rather low, and could be increased, always in accordance to the application at hand. While for our implementation we have exclusively used the Primary Advertising Channel (Channel 37, 38 and 39), with the current Bluetooth 5.0 specification, the extended advertising mode allows for the transmission of advertising packets in all the 40 physical channels from Bluetooth. This means that, instead of having to comply with the hard limit of a 10% utilization per channel for the original 3 channels, it would be 10% utilization for the 40 Bluetooth channels [74]. With the experimentally measured 500 μs airtime of the packets, and the advertising interval at 100 ms, the utilization of the wireless medium can be calculated. The 10% utilization rate is established for a 1 s observation period according to EN 300 328 [75]. This means a maximum of 100 ms of advertising per second and per channel for each and every device. If collisions are not considered, this means that at the experimentally obtained airtime for each iBeacon packet, one device could transmit:

$$\frac{100\ ms}{500 \frac{\mu s}{message}} = 200 \frac{messages}{s} \tag{7}$$

So a utilization of 200 messages per second and per channel is allowed for the End Devices. Considering the three channels in Bluetooth Advertising, it increases to 600 messages per second, or over all Bluetooth channels up to 8000 messages per second. With the 100 ms advertising period previously established, it means that $500 \frac{\mu s}{message} \cdot 10$ messages = 5 ms of airtime are being consumed every second, which is 20 times below the threshold. While the medium utilization will not be a problem for the End Devices, it

could potentially be an issue for the Servers, as they do need to advertise constantly, especially if the environment is crowded with End Devices. In order to reduce this problematic, the advertising interval of the Server would need to be modified in accordance to the application, or the Server could be converted into a Bluetooth Gateway, with multiple Bluetooth devices that can transmit and listen simultaneously. Our approach only requires the exchange of 2 packets, namely the Join Request and Join Accept, for a total of 42 bytes transmitted. This significantly reduces the necessary information to transmit in alternative PKI-based approaches. For instance, the Diffie Hellman key exchange approach proposed by Gupta et al. [37] for the authentication of connection-oriented BLE devices requires a minimum transmission of 254 bytes. Therefore, in the case of BLE Beacons limited at 21 addressable bytes, this approach would imply the need to send a minimum of $\frac{254\,\text{bytes}}{21\,\frac{\text{bytes}}{\text{packets}}} \approx 13\ \text{packets}$ for the authentication, i.e., a minimum of 650% more packets with respect to our implementation. This might pose additional problems, such as the need to parse and retransmit segmented data, or the risk of exceeding airtime restrictions, especially on the server side. In addition, the use of PKI also introduces other challenges, including the distribution of certificates to field devices and the provision of Certificate Revocation Lists (CRLs).

7. Example of use

To demonstrate the feasibility of the proposed mechanism, we developed an example of use in which the secure BLE beacons are used for the rapid, secure and privacy-preserving detection of individuals or object in movement. For that, in addition to the previously described mechanism, each End Device is provided with a series of geometrical points to uniquely identify them. The implementation proposed here is based on the method for controlling the entry and exit of objects in monitored enclosures proposed by Lopez-Ramos et al. [76], which is more secure and robust than using UUID, major or minor filters even when the AppKey is compromised [77].

7.1. Mathematical background

For the application, let us set:

1. The dimension of the affine space over a body K on which we work, n.
2. The dimension of the affine varieties corresponding to the user, m, where $m \leq n$

The dimension of the affine varieties defining the End Node will have dimension k, with $k < m$. The information related to an End Node will be given by the corresponding coefficients of the $n - m$ equations that determine it. If the End Device enters or leaves a controlled area, the Server will verify with the obtained information from their corresponding beacon frames whether it had permission to enter or leave the premises. If multiple messages are received (or not received, if the beacon are sent as of keep-alive messages), then the alarm condition is raised in order to provide a warning of the unauthorized movement of the End Device.

For that, the End Devices are provisioned with n parametric points with dimension k (λ_{jl} with $j = 1, \dots, k$ and $l = 1, \dots, n$). These parametric points are sent in every advertising packet. Therefore, to verify that if an End Device is authorized to transit in the domain of a Server, the Server first calculates the coordinates associated using the parametric equations that determine the variety:

$$x_1 = \rho_1 + \lambda_{11} v_{11} + \lambda_{21} v_{21} + \cdots + \lambda_{k1} v_{k1} \tag{8}$$

$$x_2 = \rho_2 + \lambda_{21} v_{12} + \lambda_{22} v_{22} + \cdots + \lambda_{k2} v_{k2} \tag{9}$$

$$x_n = \rho_n + \lambda_{1n} v_{1n} + \lambda_{2n} v_{2n} + \cdots + \lambda_{kn} v_{kn} \tag{10}$$

where $\rho_1, \rho_2, \dots, \rho_n$ are the affine points acting as the origin of the region's variety and $v_1, v_2, \dots, v_n$ is the set of vectors defining the direction of the region's variety. The parametric points allow for obtaining the coordinates $x_1, x_2, \dots, x_n$. Next, these coordinate are substituted into each of the m general equations that determine the affine variety:

$$a_{11} x_1 + a_{12} x_2 + \cdots + a_{1n} x_n = b_{1'} \tag{11}$$

$$a_{21} x_1 + a_{22} x_2 + \cdots + a_{1n} x_n = b_{2'} \tag{12}$$

$$a_{n-m1} x_1 + a_{n-m2} x_2 + \cdots + a_{n-mn} x_n = b_{n-m'} \tag{13}$$

If the $n - m$ equalities are verified for all points, it means the End Device is transiting through an authorized area, and no action is performed, authorizing the passage. If any of the $n - m$ equalities does not verify, the alarm signal can be activated, given the End Device is transiting through a non authorized area [76].

Therefore, considering the half packet mode previously described where up to 8 bytes of information can be encoded in a frame, those bytes could be used to store the parametric points identifying the user (λ), as shown in Fig. 25. After the End Device activation process, each node can start advertising the particular UUID of 8 bytes, and the parametric points that were provisioned for the application. The Server will check whether the received points fulfill the different equalities in order to detect non-authorized End Devices. Using this information plus the positioning of the device, calculated thanks to the RSSI and the transmitted power

Fig. 25. Mapping of the data in the Beacon packet for the geofencing application (Bottom) with respect to the iBeacon packet (Top).

information advertised with every Beacon packet, it is possible to provide a secure, fast and precise way to provide access control or user location services.

The process would be as follows: when the Server received the packet, first it checks the integrity using the MIC, to then proceed decrypting the content. If this process is successful, the Server then parses the UUID, and if it corresponds to the asset/person tracking application UUID, the Server extracts the parametric points. These points are then introduced in the set of parametric equations. From these parametric equations, n coordinates are obtained, $x_1, x_2, \dots, x_n$. If these coordinates fulfill the equality when substituted in the general equation or equations, then the user has permission to be in that area. If, on the other hand, the point does not fulfill the condition, then permission is denied, and a particular action should be triggered, e.g., alerting the security personnel.

An example of the mechanism is presented now here. For it, let us assume the following parameters: $k = 2, n = 3$ and $m = 2$. This means that there will be three parametric points, two parametric equations for the affine variety, $n - m = 1$ general equations and three coordinates, x_1, x_2, x_3. For instance, the parametric equations could be:

$$x_1 = 1 + \lambda_{11} - \lambda_{21} \tag{14}$$

$$x_2 = 2 + \lambda_{12} - 2\lambda_{22} \tag{15}$$

$$x_3 = 3 + \lambda_{13} - 3\lambda_{23} \tag{16}$$

while the general equation would be:

$$x_1 + x_2 - 2x_3 = 0 \tag{17}$$

From the parametric equations, three points are generated and pre-provisioned for each user on the End Device, for example $(\lambda_{11}, \lambda_{21}) = (1, 1)$, $(\lambda_{12}, \lambda_{22}) = (2, 1)$ and $(\lambda_{13}, \lambda_{23}) = (3, 1)$. This information is periodically advertised in a secure and private manner using the Secure iBeacons. When the packet reaches the Server, it proceeds by substituting the values in the parametric equations to obtain the corresponding x_1, x_2, x_3 points:

$$x_1 = 1 + 1 - 1 = 1, x_2 = 2 + 1 - 2 = 1, x_3 = 3 + 1 - 3 = 1, \tag{18}$$

$$x_1 = 1 + 2 - 1 = 2, x_2 = 2 + 2 - 2 = 2, x_3 = 3 + 2 - 3 = 2, \tag{19}$$

$$x_1 = 1 + 3 - 1 = 3, x_2 = 2 + 3 - 2 = 3, x_3 = 3 + 3 - 3 = 3, \tag{20}$$

Therefore, the points obtained are then $(1, 1, 1), (2, 2, 2)$ and $(3, 3, 3)$. After that, the server verifies these points into the general equation:

$$x_1 + x_2 - 2x_3 = 0 \quad with \quad (1, 1, 1) \rightarrow 1 + 1 - 2 \cdot 1 = 0\checkmark \tag{21}$$

$$x_1 + x_2 - 2x_3 = 0 \quad with \quad (2, 2, 2) \rightarrow 2 + 2 - 2 \cdot 2 = 0\checkmark \tag{22}$$

$$x_1 + x_2 - 2x_3 = 0 \quad with \quad (3, 3, 3) \rightarrow 3 + 3 - 2 \cdot 3 = 0\checkmark \tag{23}$$

In this case, the End Device fulfills the condition and therefore no alarm condition is triggered. In the case of another End Device with the assigned parametric points $(1, 1), (2, 1), (3, 2)$, the general equation result would be:

$$1 + 1 - 2 \cdot 1 = 0\checkmark \tag{24}$$

$$2 + 2 - 2 \cdot 2 = 0\checkmark \tag{25}$$

$$2 + 1 - 2 \cdot 0 \neq 0\times \tag{26}$$

Therefore, since one of the points does not all fulfill the general equation, the alarm signal should be triggered for this End Device. A visual representation of this mechanism can be seen in Fig. 26, where all previous coordinates are represented together with the plane that represents the general equation.

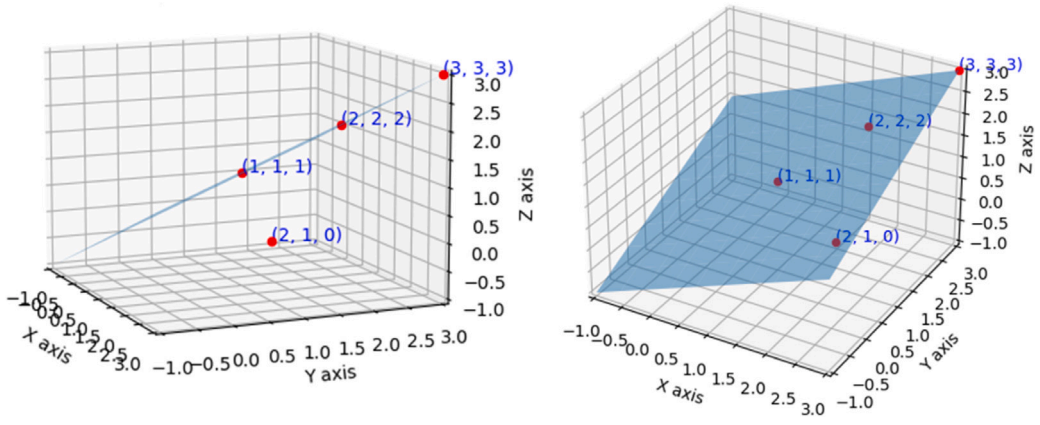

Fig. 26. Graphical representation of the general equation plane and the coordinates obtained from the ED points.

The use of this method presented here in tandem with the End Device activation process provides a more robust system than ones based on rolling UUIDs, like the Eddystone EID. These types of solutions do not provide authentication or encryption, only providing application level privacy to the users by rotating the UUID, while the use of the derived session keys provides the system with privacy, integrity and authentication. On an application level, the user can be provided with a higher level of privacy and anonymization using the affine variety method, as they can be made so that no user identifiable information is being stored, i.e., the End Device can be provided with just the needed parametric points, and no association needs to be made between them and user identifiable data. Last, there is less overhead and complexity with the affine variety solution compared to the Eddystone EID implementation, as shown before in Table 1. On this mathematical basis, an example of application based on the geofencing of restricted areas for the detection of mobile End Devices is presented in the next section.

7.2. Use case: Geofencing

In this example application, the system is intended to be set up in a large gathering, for example a convention center, an amusement park or a concert, where there are private areas for certain users. All the users are given an End Device, which could be a bracelet containing the microcontroller, the secure element, a rechargeable battery, and a RGB LED. Ideally, these bracelets are reusable and reprogrammable after each event. When the user presents their credentials to enter the venue, they get a bracelet with a set of k points assigned depending on the type of ticket. These bracelets advertise periodically the coordinates and also count with periodic scanning windows for receiving messages from the server. In order to provide privacy to the user, the coordinates are not linked to the information of the user (they could even be randomly generated at that moment). In a situation like the one shown in Fig. 27, where there are four different regions, the Scanner/Server of each region will count with a different general equation to check the different coordinates transmitted by each End Device. In case that a Scanner/Server detects an unauthorized End Device, the Server sends Secure iBeacon packets indicating that the bracelet's LED must turn red. Note that this message can be common to all End Devices, since if the Server uses the session keys associated to a particular End Device, only that End Device assigned will be able to decrypt and verify the frame. Note that in this case we could also have additional conditions, such as:

1. Distance: Due to Bluetooth's nature, it is expected that the different regions will slightly overlap each other. Using the RSSI of the received packet plus the information from the transmitted power, an accurate measurement of the distance from the server to the End device can be obtained. If the distance is less than a particular value, then the certainty that the user might have trespassed an area increases.
2. Time: The bracelet sends beacons periodically after the End Device activation, where the advertising time can be configured depending on battery constraints and the level of accuracy the application needs. Considering a message is sent every 10 s, 6 messages will be transmitted every minute. Therefore, we could also adjust the maximum number of messages received in a certain period of time to consider that the user is in a restricted area.

In Fig. 28, the the geofencing application for both the Server and two End Devices is shown. In this case, ED1 does contain points that fulfill the equation, while the ED2's points do not, so this latter one would be considered a trespasser. On the Server side, when it receives a frame from one of the End Devices, first it will check its integrity with the MIC (Fig. 28a). If the integrity check for the frame is successful, the Server will then start parsing the information. When the Server detects the UUID used for the Geofencing application (Fig. 28b), then the rest of the information contained in the packet is parsed to obtain the points. If the End Device is sending valid points (Fig. 28c), i.e., if they fulfill the general equation chosen, then no action is performed. If, on the other hand, the points do not fulfill the general equation (Fig. 28d), the server will address a packet to them so that the bracelet's light can be turned on, and the intruder can be detected. On the End Device side, after a successful End Device Activation process (Fig. 28e),

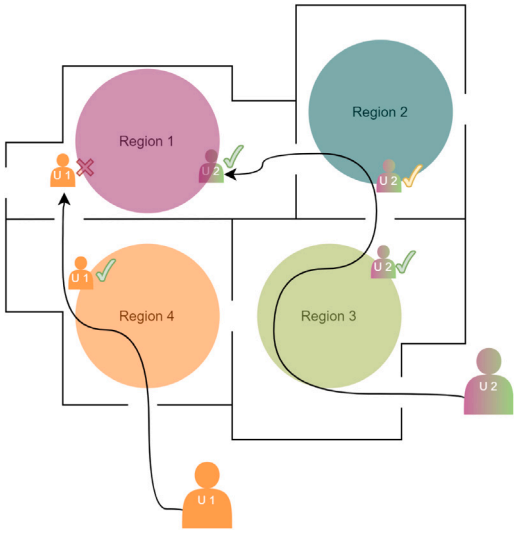

Fig. 27. Diagram example for the geofencing application.

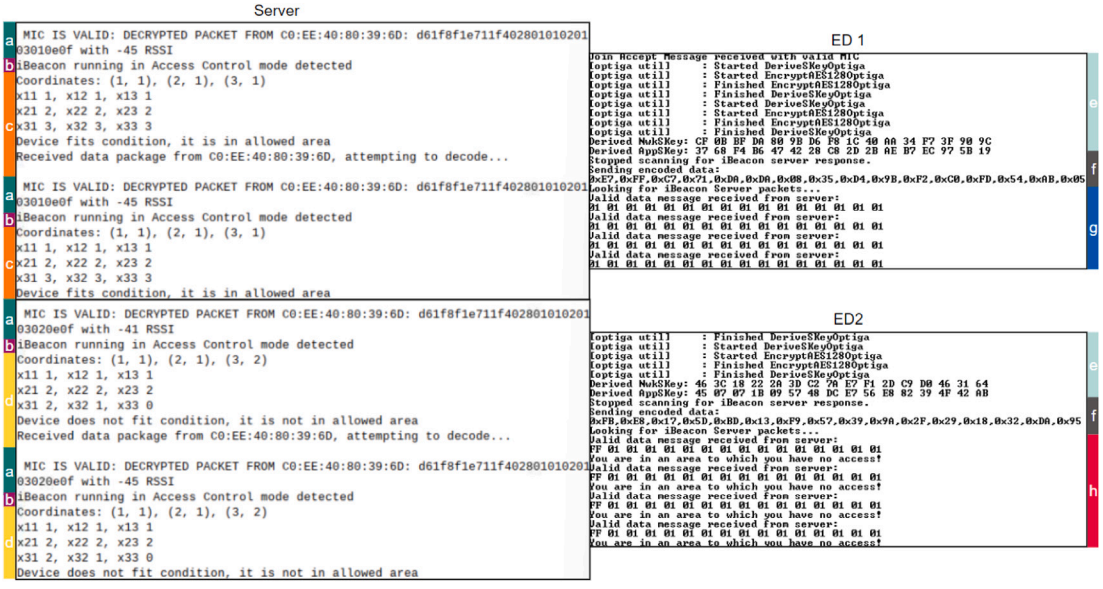

Fig. 28. Geofencing application: (a) Valid packet is received by Server | (b) UUID for the Geofencing application is detected | (c) Device is in a valid area | (d) Device is not in a valid area | (e) ED finishes the activation process | (f) ED starts advertising the encrypted coordinates | (g) ED receives data to not change the LED color | (h) ED received data to change the LED color to indicate intrusion. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the ED will start advertising the encrypted information containing the points, leaving room to listen for the Server's feedback. In this case, the ED1 receives a packet from the Server when it is in a valid region (Fig. 28f), where the ED interprets the data as a signal to keep the LEDs off. Conversely, if the ED is not in a valid region, then the ED receives a specific packet from the Server (Fig. 28g) to turn on the LED, thus identifying the intruder.

Due to the fact that continuous location data is obtained with the system, an implementation of the Geofencing system presented would need to comply with General Data Protection Regulation (GDPR). Thanks to the End Device activation process, integrity and privacy of users is ensured using the derived session keys as only the corresponding Servers would be able to access the transmitted information and verify the End Device's location. Similar to Eddystone EID, unique UUIDs will be generated for each packet, preventing easy tracking and privacy violations of users. Since the system does not require any information related to the user (such as health statistics or movement), the parametric points can be pre-calculated and embedded onto the device when needed. For example, when the users present their ticket, they could be granted a series of parametric points. If the databases do not store

any information regarding the users, like an association between parametric points and Name, then anonymization can be achieved. In case the organism in charge of the Geofencing system decides to perform tracking of the users, then they would need to perform the relevant steps to ensure compliance with the GDPR, like notifying users of the right to withdrawn their consent to be tracked or what personal identifiers they would capture about them, among others [78].

8. Conclusions

In this paper, we present a novel approach to enhance the privacy and security of Bluetooth Beacon packets using Hardware Security Modules. By examining packet formatting and applying robust authentication mechanisms, we designed and implemented an over-the-air scheme that authenticates End Nodes, ensuring the privacy and integrity of each transmitted Beacon frame. The addition of HSMs to the End Devices establishes a Root of Trust, protecting against common attacks like cloning and impersonation, and thereby strengthening the network overall security and stability. To demonstrate this, we have provided a threat model based on the STRIDE methodology, showing how the scheme implemented prevents and mitigates attacks to the architecture. The proposed mechanism has been experimentally tested with multiple PSoC^{TM} 62S2 with the Optiga Trust M as a secure element for the End Device, and a Raspberry Pi 5+ with a TPM as the Server. This setup has been used to analyze key performance metrics of the proposed solution, such as activation time, power consumption, collisions, throughput and scalability. Our results show that these security measures have minimal impact on power consumption and timing compared to traditional Beacon packets, highlighting the practicality of our approach. Finally, we present an example of use, demonstrating the mechanism's utility in providing secure, privacy-preserving indoor location in geofencing applications.

CRediT authorship contribution statement

Miguel Mesa-Simón: Writing – original draft, Visualization, Validation, Software, Investigation, Formal analysis, Data curation. **Antonio Escobar-Molero:** Writing – review & editing, Validation, Supervision, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization. **Luis Parrilla:** Writing – review & editing, Validation, Methodology. **Diego P. Morales:** Writing – review & editing, Validation, Supervision, Resources, Project administration, Funding acquisition, Conceptualization. **José Antonio Álvarez-Bermejo:** Writing – review & editing, Validation, Supervision, Project administration, Methodology, Funding acquisition, Formal analysis. **Francisco J. Romero:** Writing – original draft, Validation, Supervision, Software, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Miguel Mesa-Simon reports financial support, administrative support, equipment, drugs, or supplies, and travel were provided by Infineon Technologies AG. Antonio Escobar-Molero reports financial support was provided by Infineon Technologies AG. Miguel Mesa-Simon and Antonio Escobar-Molero are employed by Infineon Technologies AG If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by Infineon Technologies AG. Additional funding was provided by the Chips JU Project LoLiPoP-IoT (Long Life Power Platforms for Internet of Things) under grant agreement No. 101112286, by the Chips JU Project ECS4DRES (Electronic Components and Systems for flexible, coordinated and resilient Distributed Renewable Energy Systems) under grant agreement No. 101139790, including the top-up funding by Germany, Italy, Slovakia, Spain and The Netherlands, as well as by the Junta de Andalucía – Consejería de Universidad, Investigación e Innovación through the project ProyExcel_00268. This work has also been supported by grant PID2022-140934OB-I00 funded by MCIN/AEI/10.13039/50110001103/ and by "ERDF/EU". Funding for open access charge: Universidad de Granada / CBUA.

Data availability

No data was used for the research described in the article.

References

[1] S. Chen, H. Xu, D. Liu, B. Hu, H. Wang, A vision of IoT: Applications, challenges, and opportunities with China perspective, IEEE Internet Things J. 1 (4) (2014) 349–359, http://dx.doi.org/10.1109/jiot.2014.2337336.

[2] Cisco, Cisco annual internet report (2018–2023) white paper, 2018, URL https://www.cisco.com/c/en/us/solutions/executive-perspectives/annual-internet-report/index.html.

[3] N.Y. Philip, J.J. Rodrigues, H. Wang, S.J. Fong, J. Chen, Internet of things for in-home health monitoring systems: Current advances, challenges and future directions, IEEE J. Sel. Areas Commun. 39 (2) (2021) 300–310.
[4] W. Tao, L. Zhao, G. Wang, R. Liang, Review of the internet of things communication technologies in smart agriculture and challenges, Comput. Electron. Agric. 189 (2021) 106352.
[5] K.E. Jeon, J. She, P. Soonsawad, P.C. Ng, Ble beacons for internet of things applications: Survey, challenges, and opportunities, IEEE Internet Things J. 5 (2) (2018) 811–828.
[6] P. Spachos, K. Plataniotis, BLE beacons in the smart city: Applications, challenges, and research opportunities, IEEE Internet Things Mag. 3 (1) (2020) 14–18.
[7] Rachit, S. Bhatt, P.R. Ragiri, Security trends in internet of things: a survey, SN Appl. Sci. 3 (1) (2021) http://dx.doi.org/10.1007/s42452-021-04156-9.
[8] J.P.S. Sundaram, W. Du, Z. Zhao, A survey on LoRa networking: Research problems, current solutions, and open issues, IEEE Commun. Surv. Tutor. 22 (1) (2019) 371–388.
[9] M. Noseda, L. Zimmerli, T. Schläpfer, A. Rüst, Performance analysis of secure elements for IoT, IoT 3 (1) (2021) 1–28.
[10] R. Verma, S. Gautam, N.S. Bal, S. Kumar, N. Saeed, IoT-enabled energy-efficient and long-range solution for remote patient monitoring using bluetooth low energy 5. x, IEEE J. Radio Freq. Identif. (2025).
[11] R. Casanova-Marqués, J. Torres-Sospedra, J. Hajny, M. Gould, Maximizing privacy and security of collaborative indoor positioning using zero-knowledge proofs, Internet Things 22 (2023) 100801, http://dx.doi.org/10.1016/j.iot.2023.100801.
[12] J. Padgette, K. Scarfone, L. Chen, Guide to bluetooth security, NIST Spec. Publ. 800 (121) (2017) 657–696.
[13] C. Gupta, G. Varshney, An improved authentication scheme for BLE devices with no I/O capabilities, Comput. Commun. 200 (2023) 42–53.
[14] C. Milarokostas, D. Tsolkas, N. Passas, L. Merakos, A comprehensive study on LPWANs with a focus on the potential of LoRa/LoRaWAN systems, IEEE Commun. Surv. Tutor. 25 (1) (2022) 825–867.
[15] LoRa Alliance, LoRaWAN 1.0.3 specification, 2020, URL https://lora-alliance.org/wp-content/uploads/2020/11/lorawan1.0.3.pdf.
[16] Y.H. Ho, H.C. Chan, Decentralized adaptive indoor positioning protocol using bluetooth low energy, Comput. Commun. 159 (2020) 231–244.
[17] C. Huang, H. Liu, W. Wang, J. Li, A compact and cost-effective BLE beacon with multiprotocol and dynamic content advertising for IoT application, IEEE Internet Things J. 7 (3) (2019) 2309–2320.
[18] V. Toral, Y. Houeix, D. Gerardo, I. Blasco-Pascual, A. Rivadeneyra, F.J. Romero, Graphene-enabled wearable for remote ECG and body temperature monitoring, IEEE J. Flex. Electron. (2024).
[19] C. Cesarini, F. Mazzenga, R. Giuliano, G.C. Cardarilli, L. Di Nunzio, F. Fallucchi, R. Fazzolari, M. re, A. Vizzarri, Indoor localization system based on bluetooth low energy for museum applications, Electronics 9 (2020) http://dx.doi.org/10.3390/electronics9061055.
[20] A. Inc., iBeacon Specification, https://developer.apple.com/ibeacon/.
[21] Google, Eddystone Protocol Specification, https://github.com/google/eddystone/blob/master/protocol-specification.md.
[22] AltBeacon, AltBeacon Specification, https://github.com/AltBeacon/spec.
[23] A.C. Chan, R.M. Chung, Security and privacy of wireless beacon systems, 2021, arXiv preprint arXiv:2107.05868.
[24] V.S. Gulhane, S.D. Padiya, Eddystone-UID frame with data confidentiality and integrity for secured data broadcasting by BLE beacons, Res. Sq. (2022).
[25] Infillion, BLE beacons, 2023, URL https://infillion.com/software/beacons/.
[26] J. Wu, Y. Nan, V. Kumar, M. Payer, D. Xu, {BlueShield}: Detecting spoofing attacks in bluetooth low energy networks, in: 23rd International Symposium on Research in Attacks, Intrusions and Defenses, RAID 2020, 2020, pp. 397–411.
[27] S. Bräuer, A. Zubow, S. Zehl, M. Roshandel, S. Mashhadi-Sohi, On practical selective jamming of bluetooth low energy advertising, in: 2016 IEEE Conference on Standards for Communications and Networking, CSCN, IEEE, 2016, pp. 1–6.
[28] Google, Eddystone EID Computation, https://github.com/google/eddystone/blob/master/eddystone-eid/eid-computation.md.
[29] Google, Encrypted TLM Frame Specification, https://github.com/google/eddystone/blob/master/eddystone-tlm/tlm-encrypted.md.
[30] Google Developers, Discontinuing support for android, 2018, https://android-developers.googleblog.com/2018/10/discontinuing-support-for-android.html.
[31] T. Krokosz, J. Rykowski, Verification of IoT devices by means of a shared secret, in: L. Borzemski, J. Swikatek, Z. Wilimowska (Eds.), Information Systems Architecture and Technology: Proceedings of 40th Anniversary International Conference on Information Systems Architecture and Technology – ISAT 2019, Springer International Publishing, Cham, 2020, pp. 175–186.
[32] Secure UUID: I don't understand how it works, https://forums.estimote.com/t/secure-uuid-i-dont-understand-how-it-works/3769.
[33] Estimote Developer Portal, https://developer.estimote.com.
[34] Bluetooth core specification version 5.4, 2023, https://www.bluetooth.com/wp-content/uploads/2023/02/2301_5.4_Tech_Overview_FINAL.pdf.
[35] Novelbits, Periodic Advertising with Responses (PAwR), https://novelbits.io/periodic-advertising-with-responses-pawr/.
[36] E. Wagner, M. Serror, K. Wehrle, M. Henze, BP-MAC: fast authentication for short messages, in: Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks, 2022, pp. 201–206.
[37] C. Gupta, G. Varshney, An improved authentication scheme for BLE devices with no I/O capabilities, Comput. Commun. 200 (2023) 42–53, http://dx.doi.org/10.1016/j.comcom.2023.01.001, URL https://www.sciencedirect.com/science/article/pii/S0140366423000014.
[38] C.-J. Huang, C.-J. Chi, W.-T. Hung, Hybrid-AI-based ibeacon indoor positioning cybersecurity: Attacks and defenses, Sensors 23 (4) (2023) http://dx.doi.org/10.3390/s23042159, URL https://www.mdpi.com/1424-8220/23/4/2159.
[39] Y. Nir, A. Langley, ChaCha20 and Poly1305 for IETF Protocols, 2015, http://dx.doi.org/10.17487/RFC7539, URL https://www.rfc-editor.org/info/rfc7539. RFC 7539.
[40] N. Mouha, B. Mennink, A.V. Herrewege, D. Watanabe, B. Preneel, I. Verbauwhede, Chaskey: a lightweight MAC algorithm for microcontrollers, in: Proceedings of the NIST Lightweight Cryptography Workshop 2015, 2015, URL https://csrc.nist.gov/csrc/media/events/lightweight-cryptography-workshop-2015/documents/papers/session1-mouha-paper.pdf. Session 1 workshop paper.
[41] C. Hernández-Goya, R. Aguasca-Colomo, C. Caballero-Gil, BLE-based secure tracking system proposal, Wirel. Netw. 30 (6) (2023) 5759–5770, http://dx.doi.org/10.1007/s11276-023-03347-z.
[42] LINE Corporation, https://developers.line.biz/en/docs/messaging-api/beacon-device-spec/, URL https://developers.line.biz/en/docs/messaging-api/beacon-device-spec/.
[43] A. Adomnicai, J.J.A. Fournier, L. Masson, Hardware security threats against bluetooth mesh networks, in: 2018 IEEE Conference on Communications and Network Security, CNS, 2018, pp. 1–9, http://dx.doi.org/10.1109/CNS.2018.8433184.
[44] J. Yang, C. Poellabauer, P. Mitra, C. Neubecker, Beyond beaconing: Emerging applications and challenges of BLE, Ad Hoc Netw. 97 (2020) 102015.
[45] P. Golle, M. Jakobsson, A. Juels, P. Syverson, Universal re-encryption for mixnets, in: Cryptographers' Track at the RSA Conference, Springer, 2004, pp. 163–178.
[46] G. Ateniese, J. Camenisch, B. de Medeiros, Untraceable RFID tags via insubvertible encryption, in: Proceedings of the 12th ACM Conference on Computer and Communications Security, CCS '05, Association for Computing Machinery, New York, NY, USA, 2005, pp. 92–101, http://dx.doi.org/10.1145/1102120.1102134.
[47] T. Morita, K. Taki, M. Fujimoto, H. Suwa, Y. Arakawa, K. Yasumoto, BLE beacon-based activity monitoring system toward automatic generation of daily report, in: 2018 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops, IEEE, 2018, pp. 788–793.
[48] I. Natgunanathan, N. Fernando, S.W. Loke, C. Weerasuriya, Bluetooth low energy mesh: Applications, considerations and current state-of-the-art, Sensors 23 (4) (2023) 1826.

[49] H. Noura, T. Hatoum, O. Salman, J.-P. Yaacoub, A. Chehab, LoRaWAN security survey: Issues, threats and possible mitigation techniques, Internet Things 12 (2020) 100303.
[50] J. Du, F. Xu, C. Zhang, Z. Zhang, X. Liu, P. Ren, W. Diao, S. Guo, K. Zhang, Identifying the ble misconfigurations of iot devices through companion mobile apps, in: 2022 19th Annual IEEE International Conference on Sensing, Communication, and Networking, SECON, IEEE, 2022, pp. 343–351.
[51] S. Memon, M.M. Memon, F.K. Shaikh, S. Laghari, Smart indoor positioning using BLE technology, in: 2017 4th IEEE International Conference on Engineering Technologies and Applied Sciences, ICETAS, IEEE, 2017, pp. 1–5.
[52] Guidelines for 48-Bit Global Identifier (EUI-48), IEEE Standards Association, 2022, Archived from the original on July 15, 20F22; Retrieved July 15, 2022 (in English).
[53] F. Farha, H. Chen, Mitigating replay attacks with ZigBee solutions, Netw. Secur. 2018 (1) (2018) 13–19.
[54] J. Song, R. Poovendran, J. Lee, T. Iwata, The AES-CMAC Algorithm, Tech. Rep., RFC 4493, 2006.
[55] Bluetooth core specification version 5.3, 2023, https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=521059&_gl=1*1ndgmmg*_gcl_au*ODgzOTA5NjEzLjE3NDU1NzA3MDY.
[56] K. Townsend, Getting Started with Bluetooth Low Energy: Tools and Techniques for Low-Power Networking, O'Reilly Media, 2014.
[57] IEEE Standard for Local and Metropolitan Area Networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs), IEEE Std 802.15.4-2011 (Revision IEEE Std 802.15.4-2006), 2011, pp. 1–314, http://dx.doi.org/10.1109/IEEESTD.2011.6012487.
[58] P. Gutmann, RFC 7366: Encrypt-then-MAC for transport layer security (TLS) and datagram transport layer security (DTLS), 2014.
[59] M. Bellare, C. Namprempre, Authenticated encryption: Relations among notions and analysis of the generic composition paradigm, in: International Conference on the Theory and Application of Cryptology and Information Security, Springer, 2000, pp. 531–545.
[60] I.T. AG, Optiga™ Trust M SLS32AIA, Infineon Technologies AG, Rev. 3.70.
[61] I.T. AG, CY8CEVAL-062S2 PSoC™ 62S2 Evaluation Kit Guide, Infineon Technologies AG.
[62] BlueZ Project, Bluez: Official linux bluetooth protocol stack, 2023, URL https://www.bluez.org/.
[63] X. Yang, E. Karampatzakis, C. Doerr, F. Kuipers, Security vulnerabilities in LoRaWAN, in: 2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation, IoTDI, IEEE, 2018, pp. 129–140.
[64] P. Beltrán-García, P.J. Escamilla-Ambrosio, E. Aguirre-Anaya, A. Rodríguez-Mota, Availability vulnerabilities evaluation to LoRaWAN, in: M.F. Mata-Rivera, R. Zagal-Flores, C. Barria-Huidobro (Eds.), Telematics and Computing, Springer International Publishing, Cham, 2020, pp. 333–351.
[65] X. Yang, LoRaWAN: Vulnerability analysis and practical expoitation, 2017.
[66] C. Kolias, L. Copi, F. Zhang, A. Stavrou, Breaking BLE beacons for fun but mostly profit, in: Proceedings of the 10th European Workshop on Systems Security, EuroSec '17, Association for Computing Machinery, New York, NY, USA, 2017, http://dx.doi.org/10.1145/3065913.3065923.
[67] F. Hessel, L. Almon, M. Hollick, LoRaWAN security: an evolvable survey on vulnerabilities, attacks and their systematic mitigation, ACM Trans. Sens. Netw. 18 (4) (2023) 1–55.
[68] T. Simões, T. Cruz, B. Sousa, P. Simões, A security-conscious primer on LoRa and LoRaWAN technologies, Eur. Conf. Cyber Warf. Secur. 24 (2025) 638–646, http://dx.doi.org/10.34190/eccws.24.1.3575.
[69] LoRa Alliance, LoRaWAN 1.1 specification, 2020, URL https://resources.lora-alliance.org/technical-specifications/lorawan-specification-v1-1.
[70] N.S. ASA, Power Profiler Kit II Documentation, Nordic Semiconductor ASA.
[71] Ezurio, Sterling LWB95+ Datasheet, https://www.ezurio.com/documentation/datasheet-sterling-lwb5-0.
[72] Ruuvi, Bluetooth Beacon Maximum Density, https://ruuvi.com/bluetooth-beacon-maximum-density/?srsltid=AfmBOoqZ7-BMQdzID5OM0GZQdmtYKqZsxYUBut3u4PjefcIFm7pHHqJz.
[73] How to deal with Broadcasting Collision, https://devzone.nordicsemi.com/f/nordic-q-a/13685/how-to-deal-with-broadcasting-collision.
[74] European Telecommunications Standards Institute (ETSI), System Reference Document (SRdoc); Data Transmission Systems using Wide Band technologies in the 2,4 GHz Band, Tech. Rep. ETSI TR 103 665 V1.1.1, ETSI, 2021, URL https://www.etsi.org/deliver/etsi_tr/103600_103699/103665/01.01.01_60/tr_103665v010101p.pdf. Version 1.1.1.
[75] European Telecommunications Standards Institute (ETSI), Wideband Transmission Systems; Data Transmission Equipment Operating in the 2,4 GHz Band; Harmonised Standard for Access to Radio Spectrum, Tech. Rep. ETSI EN 300 328 V2.2.2, ETSI, 2019, URL https://www.etsi.org/deliver/etsi_en/300300_300399/300328/02.02.02_60/en_300328v020202p.pdf. Version 2.2.2.
[76] J.A.L. Ramos, J.A.Á. Bermejo, M.A.C. Cervantes, Dispositivo, sistema y método para el control de la entrada y la salida de objetos en recintos vigilados, 2012.
[77] S.M. Sharhan, S. Zickau, Indoor mapping for location-based policy tooling using bluetooth low energy beacons, in: 2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications, WiMob, IEEE, 2015, pp. 28–36.
[78] P. Regulation, Regulation (EU) 2016/679 of the European parliament and of the council, Regul. (Eu) 679 (2016) 2016.