Diagnosis: VulneraBLE

Analysis of Bluetooth LE Security and Vulnerabilities in Medical Devices

Wissenschaftliche Arbeit zur Erlangung des Grades
Master of Science
an der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München.

Betreut von	Prof. Dr.-Ing. Georg Sigl
	Lehrstuhl für Sicherheit in der Informationstechnik
	Dr. Stefan Schiffner
	Yonas Leguesse
	European Union Agency for Network and Information Security
Eingereicht von	Matthias Pritschet
Eingereicht am	München, den 9.10.2017

Abstract

Bluetooth Low Energy (BLE) is an energy efficient wireless data communication standard which enables an ever growing number of "smart devices" to easily communicate with host devices like smartphones or personal computers. Because of its popularity it is also being used in a growing number of medical devices. While this can be a benefit for the user, a wireless transmission of health related data is always a risk, especially if the communication is insufficiently secured.

In this thesis we are going to explain the functionality of BLE communication with a focus on the different security mechanisms and their evolution throughout the different revisions of the specification, as well as the respective known vulnerabilities.

In the second step we obtained different BLE-enabled medical devices and analyzed which of them were susceptible to the known attacks and if they employed any security measures. This was done by a black box approach because no prior knowledge about the devices was available and all of the attacks were purely non-invasive. The analyzed devices were pulse oximeters, blood glucose meters, body thermometers and fitness trackers.

We further tried to analyze the Pseudo Random Number Generators (PRNGs) of a Bluetooth-USB-Dongle and of a blood glucose meter and propose a method of performing a Denial-Of-Service Attack by building a simple BLE jamming device with cheap off-the-shelf components.

We found that many of the devices do use neither encryption nor even simple authentication measures, which allowed for relatively simple Man-In-The-Middle and Passive Eavesdropping Attacks, and propose several countermeasures which could have been used to prevent all or most of the attacks.

We have shown that many BLE-enabled medical devices are using insecure communication which poses a high risk to health and privacy. These attacks are not unique to BLE, but are common to many kinds of wireless communication.

Contents

1 Introduction

1.1 Evolution of Bluetooth and Bluetooth Low Energy

The development of the Bluetooth standard started in 1994 when the Swedish telecommunications company Ericsson was looking for a wireless alternative to replace the cables used for serial communication between electronic devices [75]. As some other companies understood the potential of this idea and that a common wireless standard would be beneficial for everyone as it would enable universal interoperability, in 1998, the companies Ericsson, Nokia, Intel, Toshiba, and IBM founded the Bluetooth Special Interest Group (SIG) in order to develop the standards and also to handle non-technical matters like licensing and marketing.

Version 1 of the Bluetooth specification was released in 1999. The next major improvement was in 2004 when Version 2 introduced the optional "Enhanced Data Rate" (EDR) transmission mode which, in theory, increased the data rate of the old "Basic Rate" (BR) transmission mode from 1 Mbit/s to up to 3 Mbit/s. In 2009, an optional "High Speed" (HS) configuration was added to the specification. It contained several technologies such as an enhanced retransmission mode to improve reliable data transmission and an unreliable streaming mode without retransmission or flow control. The High Speed mode also specifies the possibility to hand the data transmission over to a faster alternative physical interface ("Alternative MAC/PHY", "AMP") like 802.11, which is usually used for WIFI.

Parallel to this development of the Bluetooth BR/EDR standard, that is sometimes also referred to as "Bluetooth Classic", the Wibree Alliance started working on a different wireless communication standard in 2001 [74].
Designed to address the problems of wireless technologies at that time, the design goal was to provide a low-cost and low-power wireless data transmission standard [45]. The Wibree standard was released in 2006 and merged into the Bluetooth standard as "Bluetooth Low Energy" in 2009, just before Version 4.0 of the Bluetooth specification was released in June 2010 [10]. One of the key features of Bluetooth LE is the asymmetric design that allows Peripherals to be designed in a very energy- and cost-efficient manner with low requirements regarding processing power and memory[44].

While Bluetooth Classic and Bluetooth LE do have some commonalities on different layers, they function in very different ways and are not compatible to each other. However, as of

today, most Bluetooth controllers that can be found in computers or smartphones are capable of supporting both.
The Bluetooth SIG has also been promoting the term "Bluetooth Smart" to describe Bluetooth LE capable devices, but decided to stop using the term in 2016 [8].
Since 2010, three new versions of the Bluetooth specification have been released, each of them bringing major improvements to Bluetooth LE.

Version 4.1 [11] was introduced in December 2013. It improved the wireless co-existence with LTE, added a bulk data transfer mode, and allowed a device to operate simultaneously in multiple roles, that is, both as Central and Peripheral [89]. It also included a revision of the LE Privacy functionality, now called "LE Privacy 1.1", which simplified the handling of resolvable private addresses and rednering the reconnection address obsolete.

Version 4.2 [12] was released in December 2014. It increased the range and data throughput by increasing the maximum transmission power and the maximum data packet length. This version also brought two major security improvements: *LE Secure Connections* and *LE Privacy 1.2*.
LE Secure Connections finally adds more secure algorithms and pairing schemes which are based on Elliptic Curve Cryptography (ECC). This was necessary because the old pairing mechanisms (now referred to as *LE Legacy*) are suffering some severe security flaws. LE Privacy 1.2 moved the mechanism for resolving private addresses from the Host into the Link Layer. It is therefore also reffered to as "Link Layer Privacy"[14]. This allows to use the whitelist features (automatic reconnection, packet filtering) even when a peer device is using resolvable private addresses.

Version 5 (*Bluetooth 5*) was introduced in December 2016 and did not introduce any new security improvements [92, 9]. However it added two new major features on the physical layer. An optional second modulation scheme allows to transmit data with twice the symbol rate as before (2 Msym/s), and a special coding scheme referred to as *Coded PHY* or *LE Long Range*, significantly extends the wireless range while reducing the theoretical data rate from 1 Mbit/s down to 500 or even 125 kbit/s. While these are no security relevant features, any measure that extends the range of a BLE connection, such as the coded PHY, also extends the physical area in which an adversary can perform an attack.

1.2 BLE as Emerging Technology in Healthcare

One of the main tasks of the Bluetooth SIG is to promote the use of Bluetooth in as many devices as possible [13]. This also includes medical and fitness devices. For this reason, the Bluetooth SIG contains a dedicated Medical Devices Working Group and even a Medical Devices Expert Group.

According to [7, 76], BLE is foreseen to lead the market share in wireless medical and fitness devices. The Bluetooth SIG even claims that "Bluetooth is the ideal wireless standard for hospital or home"[16].

To further promote this development and to facilitate interoperability between devices, dedicated bluetooth profiles have been standardized. They specify how certain medical data can be acquired and exchanged and contain, among others, dedicated profiles for blood pressure, body temperature, blood glucose, heart rate and blood oxygen saturation [15].

1.3 Attacks and Vulnerabilities in Medical Devices

Medical devices are very interesting targets, because of the severe implications an attack can have. Especially the possibility that such an attack could be conducted remotely, by an attacker that might be miles away.
Often, the main motivation for a criminal is of financial nature. Recently there has been a rise in incidents with so-called "ransom ware". These are malicious programs that encrypt the data on a victim's personal computer, promising to decrypt the data only after the victim has payed a certain amount of money to the attackers.

A ransom ware attack could also be performed on medical devices such as insulin pumps which then would cease to provide the essential insulin until the patient pays the demanded ransom. While this particular kind of attack might still sound far fetched, researchers have been able to successfully attack medical devices such as insulin pumps [83, 78, 79, 60], pacemakers [40, 28] and automatic external defibrillators [41].

Furthermore, there is a growing number of reports on dangerous vulnerabilities being found in medical devices that are used by tens or even hundreds of thousands of patients.
Recently, several vulnerabilities in an implantable cardiac pacemaker have been disclosed, that allowed an attacker to wirelessly issue unauthorized commands to the pacemaker or deliberately drain the batteries [47].
A different vulnerability has been published in January 2017 that allowed an attacker to manipulate the communication between an implantable cardiac pacemaker and an associated stationary home monitoring device [46, 98]. According to [68], this attack can cause "Cardiac Devices to malfunction – including by apparently pacing at a potentially dangerous rate".
A third exemplaric incident was the disclosure of several vulnerabilities in an insulin pump in October 2016 [50]. These vulnerabilities would allow an attacker to remotely dispense high doses of insulin, which would then cause the patient to have a dangerous hypoglycemic reaction also known as diabetic shock or insulin shock.

1.4 Security Vulnerabilities in Bluetooth BR/EDR

The Bluetooth Classic protocol has a long history and many functionalities have been added throughout the years. Because of the complexity of the Bluetooth Classic protocol stack and several bad design choices in the specification, Bluetooth is providing a large attack surface. This led to numerous attacks being published. They can be classified in two groups: Attacks on the protocol itself and attacks on flawed implementations of the protocol stack.

Examples and summaries of different attacks on Bluetooth Classic can be sen in [48, 39, 57, 95, 36, 37, 61, 30, 38, 69, 4, 18, 65].

1.5 Motivation and Objectives

As we have explained in the previous sections, BLE is a technology with many advantages and it is quickly rising in popularity among consumers and developers. At the same time, as medical devices are becoming more and more "smart" and interconnected, many of them are already implementing BLE to communicate with other medical devices or devices like smartphones and personal computers. As we have seen, medical devices - especially those with wireless interfaces - can be exploited in different ways and can cause serious harm. We will therefore provide an overview of BLE security vunlerabilities known to date and analyse different BLE-enabled medical devices regarding the security of their wireless interfaces in order to get a general idea about the current state of security.

In the first half this work, we provide an overview on the internal functions of BLE, with a special focus on the security mechanisms. We give some theoretical background on IT security concepts and cryprographic algorithms in chapter 2 and explain how they are implemented and used in BLE. We also explain some general attacks on wireless data transmission protocols to which we will refer to during the subsequent chapters. Chapters 3 and 4 will then summarize the most important building blocks, functions and features of BLE. We consider this to be necessary as the BLE protocol is still something rather new and exotic compared to other, more widely used and understood wireless standards such as WIFI or GSM. After the introduction of the basic principles, we will explain the state of the art regarding vulnerabilities and practical attacks on BLE. In Chapter 7 we then perform our analysis of different medical or health-related devices and their vulnerabilities. Some further experiments that were part of our research are presented in chapter 8. As the time frame of this work was limited, we were not able to practically implement some attacks. These are explained in chapter 6 and provide a starting point for further research. In the last chapter we give a résumé on our findings and present some mitigation techniques. We furthermore try to find reasons to explain the negligence of device manufacturers regarding the security of their devices, and present some ideas on how to improve the current situation.

2 Theoretical Background

2.1 Key Concepts of Information Security

Depending on their application, information systems have different requirements regarding certain aspects of information security. It is necessary to establish some kind of taxonomy and provide definitions and terminology for the key concepts.

This is necessary to further understand which security requirements certain cryptographic methods can provide, and which properties of information systems are compromised by certain vulnerabilities and attacks.

Confidentiality

The term Confidentiality basically means keeping data secret from unauthorized parties. Data which is being stored or transmitted does often contain sensitive information which should be kept secret and should only be disclosed to a limited set of parties. Guaranteeing data confidentiality implies ensuring that this information can't be read by an unauthorized party even if the adversary has physical access to the data storage or the transmission medium.

Integrity

Data Integrity describes the assurance and maintenance of accuracy and consistency of data. This refers to both stored data, as well as to data being transmitted or processed. In general, there are two ways data can change unintentionally. On the one hand here is accidental data corruption, which usually is caused by natural phenomena such as electro-magnetic interference or ionizing radiation but also by technical or human failure, like mistreating a data storage medium, can cause accidental data corruption. On the other hand, there can be malicious attemps of adversaries which intentionally try to alter data in order to obtain a benefit or cause damage.

Availability

Availability describes the requirement of a service or a device to be at the user's disposal. This is not a binary property but also includes the provision of a certain assured quality of service. If a service or a device is not available when it expected to be or delivers a insufficient quality of service, it can cause serious physical or

economical harm, depending on the type of service or device.

Non-Repudiation

Non-Repudiation refers to the requirement of securely and reliably associating a certain action to a single unique party, irrespective of whether or not this party was authorized to commit that action.
These actions in question can be manyfold and do not necessarily have to be of malicious nature.

Authentication

Authentication means the verification of a statement or a piece of information that is claimed true by a second party.
This can mean, for example, that the other party provides some sort of evidence that proves its supposed identity. Because data is often exchanged through an insecure channel, an authentication scheme provides certainity over the identity of the other party and that incoming information does actually originate from that other party.
Message authentication often also implies message integrity because a method that provides protection against malicious manipulation of data, usually also protects against accidential manipulation.

Authorization

Authorization is the act of granting access rights to certain resources or services.
It implies that the identity of the remote party can be verified through authentication.

2.2 Classification of Attacks on Wireless Communication

Attacks on IT Systems can be classified in several categories which depend on the capabilities and the incentives of the adversary. An attacker may want to cause damage to a system, extract confidential information or manipulate data.
There are different kinds of attackers with different incentives, different levels of knowledge and different levels of access to a system [91].
This work concentrates on attackers whose only property is physical proximity to the user. The attacker model on which we base our research is an adversary who has no physical access to the devices and has no knowledge about the internal states and procedures of the user's devices. The attacker's equiment soley consists of a sufficiently powerful Software Defined Radio (SDR) and a personal computer with off-the-shelf, consumer grade hardware.

2.2.1 Passive Eavesdropping

Whenever two parties are communicating over an insecure channel, a third, unauthorized party may get access to the channel and listen to the presumably confidential communication. Because the adversary does not attempt to modify any data in transit but remains transparent and passively listens to the communication, this attack is called *passive eavesdropping*.

Most channels that are being used for electronic communication today can be considered insecure, unless special measures are taken to protect them. Internet traffic, for example, is often passing through many nodes around the world, making it impossible for a communicating party to fully control the whole data path and secure it against access from malicious parties. Another example are most wireless data transmission technologies. Data is transmitted through the air, often omnidirectionally and penetrating windows and walls. This allows an attacker with the appropriate sensitive detection equipment to easily listen to the communication. Even optical fiber cables can be wiretapped by relatively simple means.

In order to prevent such attacks and thus to provide confidentiality over an insecure channel, the data can be encrypted by the transmitting party before being sent through the channel

Different cryptopgraphic schemes exist for this purpose. They all have in common that the key used for decryption must not known to the eavesdropper. This also implies some problems regarding the key exchange, because the key should not be transmitted over the insecure channel for apparent reasons. However, solutions adressing the key exchange exist and will be further discussed in section 2.3.2.

2.2.2 Active Eavesdropping / Man-In-The-Middle

In certain cases, it is possible for an attacker to position itself in between two communicating parties. This enables an attacker not only to access but also to modify each data packet on its way from one legitimate party to the other and even inject own messages. This attack class is therefore called *active eavesdropping* or *Man-in-the-middle (MITM)*.

In addition to the problems associated with a passive eavesdropping attack, this attack does not only break confidentiality, but also authenticity and data integrity. The receiving party cannot guarantee that the party at the other end of the connection is actually the party it thinks it is and it cannot be sure that the data it receives hasn't been tampered with.
MITM attacks are usually rather complex to set up, especially if an attacker does not want to be detected.
MITM attacks on wired connections require physical access to the cable and the technical means to open up the connection and insert a device that allows to modify the data without introducing too much latency or unwanted data corruption. This process can be very complex,

because of today's high data transfer rates.

Performing MITM attacks on wireless connections requires a different approach, as it is rather complex to keep the victim's devices from communicating over the air. As wireless communications often have a certain topology consisting of one base station and several client devices, an attacker can often succeed by setting up a rogue base station to which the victim's device eventually connects to. This is a very popular approach in WIFI Networks, often referred to as *Evil Twin Attack* [5].

While encryption can be used to provide confidentiality, authentication and message integrity require other cryptographic methods like Message Authentication Codes (MACs) or Message Integrity Checks (MICs).

Relay Attack

A relay attack is a derivative of a MITM attack but does not necessarily require the attacker to manipulate or inject packets. Instead, this kind of attack is used when two parties are too far away to communicate. The adversary has two transcievers which are close to each of the victim's devices. The adversary then creates an additional link between the transcievers to relay the wireless signals and enable the victim's devices to communicate [96, 59].

2.2.3 Downgrade Attacks

As IT systems evolve, cryptographic standards and protocols often begin to show weaknesses and become replaced by more advanced and more secure ones. But often these new methods are only supported by modern systems. Old devices may often not receive a necessary firmware upgrade, may not have the processing power for the new methods or may simply not have the capability of performing a firmware upgrade. This makes it necessary for the protocols to include the possibility of falling back to an older version of the protocol in order to still allow modern devices to communicate with devices that do not support the latest and most secure protocols.

This ability to allow, for example, a secure data transmission protocol to fall back to older security mechanisms with known weaknesses, allows an attacker to perform a downgrade attack by forcing this fallback even if both devices could actually support a secure version of the protocol. This fallback can be achieved by different means, depending on the actual protocol. In some cases, it is enough if an attacker disturbs the communication while in other cases the attacker has to inject certain packets or even mount a MITM attack first.

Downgrade Attacks are usually worthless on their own, but they enable an attacker to

break the weaker encryption and thus perform other types of attacks like passive or active eavesdropping.

2.2.4 Location Tracking

A malicious party does not always have the goal to obtain confidential information that is transmitted or stored electronically. Often the physical location can reveal a lot about a person's habits and behaviour. Collecting this data has become increasingly easy, as more people are carrying devices with them that broadcast wireless signals which allow others to uniquely identify them [90, 27].

This might at first glance seem as a neglectable issue, but collecting this data is not only a severe infringement of privacy, but can also be used to prepare more serious crimes.

2.2.5 Denial-Of-Service Attack

Instead of trying to access or manipulate data, Denial-of-Service (DoS)-Attacks attack the "Availability" requirement of IT systems. The main goal is to prevent the devices from fulfilling their intended function [101, 84, 103]. Famous examples are DoS Attacks on webservers which can be used by attackers to demand ransom from the operators in order to stop the attack.

Power Drain Attack

Most of the BLE-enabled devices are powered by a finite source of energy, usually batteries. This is one of the main reasons why BLE was designed to be very energy efficient. However, data transmission is still a very costly process in terms of energy consumption and to limit data transmission as much as possible is a popular way to extend the battery life.
If an attacker manages to force a battery-powered device to transmit data continuously instead of very sparingly, for example by repeatedly requesting the transmission of the same big chunk of data from the device, this can cause the battery to become depleted much quicker than intended by the manufacturer and expected by the user. Another variant of this kind of attack is often referred to as "Sleep Deprivation Attack" [62], in which an attacker drains the battery by keeping the attacked device active and preventing it to enter a power saving sleep mode.
Additionally, if there is a battery charge monitoring system integrated, depending on how the algorithm is designed, it might not even be able to report the quick discharge to the user.

Jamming

Jamming describes the act of deliberately generating electromagnetic noise mostly within a certain freqency range in order to interfere with and disturb or block the legitimate wireless communication of the victim's devices and prevent them from communicating [80, 53]. Jamming is thus operating on the physical layer of a wireless protocol.

Protocol Layer Denial-of-Service

Instead of blocking the physical channel, as in the case of jamming, an attacker can also provoke a DoS through actions in higher protocol layers. A device can be part of a wired or wireless network and simply create damage by misusing theoretically valid operations.

Two famous examples of DoS attacks in IP-based networks are the *Ping Flood* and the *SYN Flood* [3].
In a Ping Flood Attack, an attacker continuously sends ICMP Echo Request Packets (Ping Packets) to a server without waiting for the server's answer. The server tries to respond every request and thus both incoming and outgoing bandwidth of the server are being consumed. This can lead as far as to completely prevent or at least significantly slow down every other communication to and from the server.
In a SYN Flood Attack, an attacker continuously sends SYN Packets, which are normally used to initiate the establishment of a TCP connection, to a server. As the server is allocating resources for every TCP-connection that it thinks is about to be established, it will eventually run out of memory.

2.2.6 Replay Attacks

In a replay attack, an adversary first records a piece of communication between two parties by passively eavesdropping on a channel and then injects certain parts of the message back in the channel at a later point in time.
This attack is particularly successful in simple systems where data transmission is purely unidirectional, like wireless garage door openers, for example.

Today, most protocols include protection mechanisms that prevent an attacker from simply reusing old packets. An example is the BLE data signature function described in section 4.7.

2.3 Cryptography

To protect sensitive and confidential information and to prevent said attacks, people rely on cryprography. It is also playing an important role in BLE. In this section we will introduce the cryptographic schemes and algorithms which are used by BLE.

2.3.1 Symmetric and Asymmetric Crypography

Cryptographic schemes can be divided in two basic classes: Symmetric schemes and asymmetric schemes.

Symmetric schemes base on a single shared secret key that is known only to the legitimate participants of a communicating group. Until 1976, all cryptography was exclusively based on symmetric algorithms.
Symmetric schemes suffer two main problems regarding the key management. The first problem adresses the distribution of the keys over an insecure channel. This issue will be dealt with in section 2.3.2.
The second problem concerns the number of keys that are required when in a group of n parties each party wants to securely communicate with each other party. Because every possible pair of parties requires a dedicated key, every party has to store $n-1$ keys and there a total of $n \cdot \frac{n-1}{2}$ keys present in the group. As the number of keys is growing quadratic with the number of users, securely managing the keys can quickly become a very complex task.

2.3.2 Key Exchange

Before two devices can exchange encrypted and authenticated data, they first have to exchange the necessary keys. Key exchange is a very critical part of every communication protocol, as a compromised key can cause great potential damage.

A shared key can be established in two ways: Through a *key transport protocol*, in which one party securely transfers a key to the other party, or through a *key agreement protocol*, in which two parties generate a public-private key pair and exchange their public keys to calculate a common shared key.

The main problem with any key exchange is that the channel is usually insecure. This means that any key exchange protocol has to take into account that data that is sent over the channel can alway be accessed and modified by an attacker.
In the case of a key transport protocol this causes two problems: Firstly, a key in transit

can be eavesdropped upon. The adversary is then able to decrypt any future encrypted communication and can also inject messages that may be encrypted and/or signed with the key.
Secondly, the key in transit may be modified by a MITM attack. Every traffic between the parties is then required to pass through the attacker, who is then capable to intercept, decrypt, modify, sign and encrypt data on its way from one party to another.
A key transport protocol does therefore always require a second, secure channel to assist with the key transport and provide confidentiality and authenticity.

A key agreement protocol however, does not mind a passive eavesdropper being present during the exchange of the public keys, because an attacker can not calculate the final common shared key without the knowledge of at least one private key. However, a MITM attacker may substitute every public key in transit with the attacker's own public key. This would then again allow the attacker to intercept and modify any data between the parties.
A key agreement protocol does therefore always require a possibility to verify the authenticity of a public key. This problem can be solved by using a *Public Key Infrastructure* (PKI) based on trusted *Certificate Authorities* (CA) that sign public keys and guarantee that a public key belongs to a certain entity. A different solution is the *Web of Trust*, which is a decentralized trust model in which every party can sign the binding between another party and its public key. As more and more parties get their public keys signed and in turn sign other party's public keys, at some point every public key can be authenticated through a so called *Chain of Trust*.

2.3.3 Encryption

Encryption assures confidentiality of data. It maps a plaintext message m to a ciphertext message c using an encryption key K, and allows a recovery of m from c only with knowledge of the decryption key.

In the case of a symmetric algorithm, the encryption key and decryption key are identical. Symmetric encryption algorithms can be classified in two types: *Block Ciphers* and *Stream Ciphers*. While stream ciphers work with a stream of single bits as input and output, block ciphers work on fixed-sized blocks of data. A typical block cipher is the AES algorithm explained in 2.4.1.

If an asymmetric algorithm is used, two different keys are used: a public key for encryption and a private key for decryption. While asymmetric cryptography can theoretically be used to encrypt a message of arbitrary length, it is computationally very expensive especially for large message sizes. This is why in practice so called hybrid encryption schemes are being used. They encrypt a message using a symmetric cipher and a random key and then encrypt the key with an ansymmetric scheme.

2.3.4 Data Signing

Cryptographic signatures assure integrity and authenticity of data.
A signature scheme takes the message and a secret signing key as input and generates a MAC that is usually transmitted or stored together with the actual message. The integrity and authenticity of the message can then be verified in two ways.

If a symmetric algrithm is used, the message can simply be verified by generating a second MAC using the message and the same signing key. If the second MAC is identical to the MAC generated during the signing process, the verification was successful.

If an asymmetric algorithm is used, the public key of the signing party is required to verify the signature. As asymmetric cryptography is computationally very expensive, asymmetric signature schemes usually work on hashes of the message. That means, that both the signature generation and the signature verification do not operate on the message itself, but on a hash value of the message. The message itself can therefore be of arbitrary length. This simplifies the algorithm design, because a hash value is always of a fixed length for a given hash function, but also introduces a potential vulnerability, because the signature is valid for *any* message that results in the same hash value. It is therefore important to use only cryptographically secure hash functions.

A successful signature verification provides authentication and data integrity, if the signing key is not compromised. It assures that the message was signed only by the owner(s) of the signing key and that the message has not been subject to tampering or data corruption.

Examples for symmetric signature schemes are AES-CMAC or AES-CBC, examples for asymmetric signature schemes are RSA-FDH or DSA [93].

2.3.5 Commitment Functions

Commitment functions are cryptographic primitives that allow a party to commit to a certain message without disclosing it, but with the possibility to reveal it later [17]. A special kind of commitment functions which allow commiting only to a single bit, are called *bit commitment functions* [70].

A commitment scheme consists of two phases.
In the first phase, the commit phase, a party Alice commits to a message m and chooses a random opener value r.
Alice then calculates the commitment value c and sends it to the other party, Bob.
In the second phase, the reveal phase, Alice sends the opener value r to Bob.
Bob can then confirm that Alice commited to the message m earlier.

Commitment functions have two distinct properties.
The *hiding property* describes the inability of the receiver to learn anything about the message m from the commitment value c.
The *binding property* describes the inability of the sender to change the message m after the commitment has been made.

To get a more descriptive idea of this rather abstract concept, one can imagine a lockable box in which Alice puts the letter with the message she commits to. Alice then locks the box with a key and gives the box to Bob.
In an ideal case, there is no way for Bob to learn anything about the enclosed letter (hiding property) and there is also no way for Alice to change the content of the letter (binding property). As soon as Alice decides to reveal the message, she simply hands the key to Bob who can then confirm her former commitment.

A violation of the hiding property could be illustrated with Bob being able to peek inside the box and thus gaining more or less information about the message, either just its length, or even some of the content.
A violation of the binding property, on the other hand, could be illustrated with a manipulated box with a false bottom that can open two different compartments depending on which one of two different keys is inserted. This would allow Alice to change her mind even after the commit phase and simply hand the one key to Bob that matches her desired message.

Commitment schemes can be implemented in several ways using existing cryptographic primitives such as hash functions or PRNGs.
It is noteworthy to mention that both perfect binding and perfect hiding can not be achieved simultaneously.
If the commitment value c can only be opened in one specific way, this already can allow a Bob to learn something about the message, for example by brute-forcing the opener values, and thus violating the hiding property. If this should be prevented, a commitment value should be able to be opened in various ways to reveal, ideally, every possible message so Bob does not gain any information about the message to which Alice commited.
This, however, would in turn violate the binding property, because it allows Alice to change her mind after the commitment phase and simply calculate a different opener value that would reveal a different message instead of the message she initially commited herself to.

2.3.6 Randomness

A lot of cryptographic systems and protocols rely on randomness.
Random numbers are used for key generation and for many key exchange and challenge-response protocols.
If a Random Number Generator (RNG) fails to deliver data with a high enough degree of entropy, the cryptographic primitives relying hereupon are rendered vulnerable.

There are essentially two different types of RNGs available: True Random Number Generators (TRNGs) and Pseudo Random Number Generators (PRNGs).
TRNGs usually require dedicated hardware to generate randomness out of a truly random entropy source such as thermal noise. However, this special hardware is often complex and expensive and output rates are often low compared to PRNGs.

PRNGs are taking an initial seed value to deterministically calculate a stream of output data which should look like true randomness to a computationally limited observer.
Because PRNGs have a finite internal state, their output will repeat after a certain, usually very long sequence.

Many PRNGs come with a reseeding function which allows to inject more external entropy, for example from a slow TRNG, in the process. This functionality prevents the output from repeating and also helps in recovering from a compromised internal state. It should be noted that proper seeding is important because a predictable seed usually implies a predictable output.

PRNGs sometimes come as *cryptogryphically secure* (CS-)PRNGs or as DRNGs/DRBGs (Deterministic Random Number Generator/Deterministic Random Bit Generator). These names should emphasize their resistance to cryptanalysis and thus suitability for use in cryptography or point out their deterministic nature respectively.

According to [93], PRNGs should meet four requirements. These are however not generally applicable and may change depending on the specific use case and the possible attacker models which can vary greatly from one case to another.

- Output indistinguishability: It should be hard to tell the output of the PRNG apart from a TRNG as long as initial seed or current internal state are uknown.

- Forward security: It should be impossible for an attacker to recover the previous outputs of the PRNG even if the internal state is known.

- Resistance to state-extension attacks: In some cases an attacker has the possibility to compromise the internal state of the PRNG. Even in this case, it should be hard for the attacker to learn something about the future outputs.

- Compromise of reseeding data should not lead to generator compromise, for example when the attacker knows the reseeding input or if it lacks sufficient entropy.

While the concept of PRNGs is not new and many algorithms can be found in literature or in cryptographic programming libraries, broken implementations have caused, and are still causing a lot of problems. Some examples that should be mentioned in that regard are the Debian OpenSSL disaster (CVE-2008-0166), the Android PRNG bug (CVE-2013-7372) and

the faulty PRNG in Libgcrypt / GnuPG (CVE-2016-6313).

Regulatory agencies often require PRNGs which are being used for cryptographic purposes to pass statistical test suites. However, flawed PRNGs are often hard to detect with these test suites because CS-PRNGs often make use of cryptographic building blocks like hash functions, which mask information about their input and thus may hide internal weaknesses [54].

2.4 Algorithms

Cryptographic primitives are always based on algorithms. The following sections are going to shed some light on the cryptographic functions employed in BLE, as well as their particular advantages, limitations and weaknesses.

2.4.1 AES

The Advanced Encryption Standard [24, 73, 71] is a subset of the Rijndael cipher which was developed by Joan Daemen and Vincent Rijmen in 1998 [25]. It is a block cipher with a block size of 128 bits, and key lengths of 128, 192 and 256 bits, referred to as AES-128, AES-192 and AES-256 respectively. It is a substitution-permutation network and is composed of four different functions which can be performed fast in software and hardware. AES uses several rounds, depending on the key length. A key schedule mechanism generates a seperate key for each round.
According to [93], AES-128 is considered safe for near term use but for long term use AES-256 is recommended.

2.4.2 Operation Modes of Block Ciphers

AES, like all block ciphers, can be used in many different ways, so called operation modes. Depending on the operation mode, a block cipher can be used to securely encrypt any arbitrary amount of data or to generate a MAC [31].

Different operation modes provide different levels of security and performance. Especially the possibility to parallelize block cipher operations during encryption and/or decryption can greatly increase the data throughput.

This section will provide a brief overview of the operation modes used in BLE for encryption and authentication.

CBC

The cipher block chaining mode (CBC) is one of the most commonly used block cipher modes. Figure 2.1 shows a schematic diagram of a CBC-encryption with a block cipher E and a key K.
Before encryption, first plaintext block m_1 is XORed with a chosen initialization vector IV. Every further plaintext block is XORed with the previous ciphertext block before getting encrypted. Because the length of the plaintext message may no be a multiple of the cipher block size, padding has to be added to the last block, which may induce security flaws in certain cases [99].

According to [93], CBC should not be used on its own anymore, because it has shown to be vulnerable against certain attacks when the initialization vector IV is predictable or non-random. BLE does not make use of the CBC mode, but it is mentioned here because it serves as the basis for CMAC and CBC-MAC.

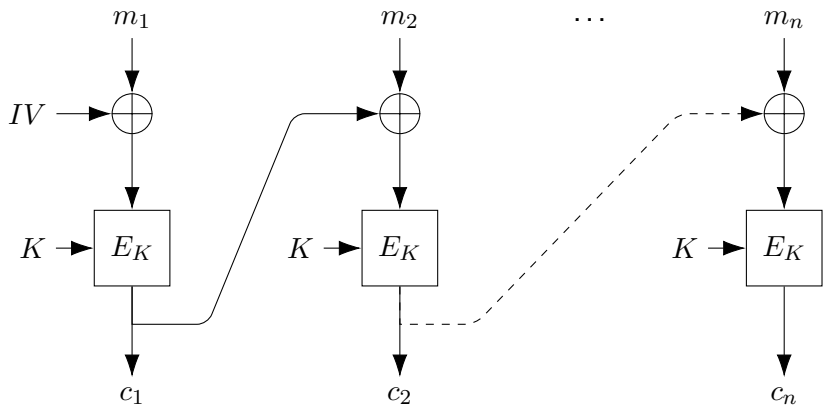


Figure 2.1: Block Cipher in CBC Mode

CBC-MAC

The CBC mode can not only be used to encrypt data, but can also be used to generate a MAC [6]. This requires both parties to have a secret shared key. The MAC is generated by encrypting the message m using a block cipher in CBC-mode but only keeping the last output as MAC, as shown in Figure 2.2.
The size of the MAC is the same as the block size of the underlying cipher. The MAC is

then appended to the plaintext message. The recipient can then perform the same CBC encryption of the plaintext message. If the last cipherblock matches the attached MAC, the message was probably not manipulated on its way.

It should be noted that in the case of generating a CBC-MAC, unlike with CBC encryption, the initialization vector must not be a random number that is attached to the message. Instead it should be a constant value that is already known to the recipient. Otherwise an attacker could change the first block m_1 of the message to m_1' and easily calculate a new IV' so that $IV' \oplus m_1' = IV \oplus m_1$, which would then return the same MAC.

According to [93], using soley CBC-MAC for message authentication should not be considered secure unless special modifications are made. These modifications can include encrypting the last block or prepending the message length.

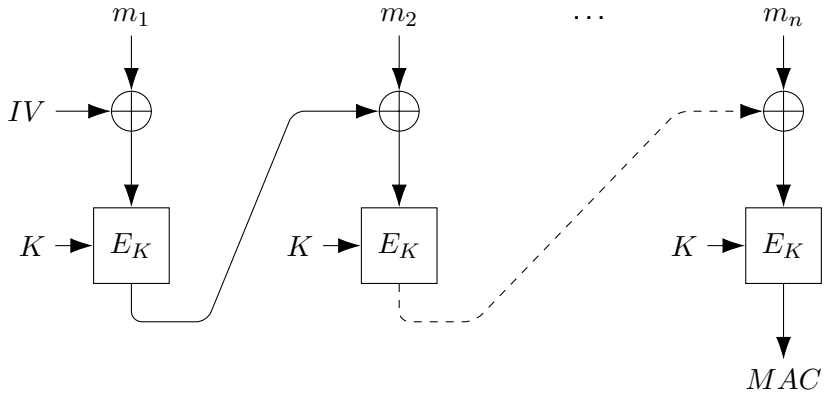


Figure 2.2: Block Cipher in CBC-MAC Mode

CMAC

Due to the security problems of CBC-MAC, the CMAC mode was invented to provide a more secure alternative [33]. It is based on the the CBC-MAC but it includes a modification to the last plaintext block before encryption. If the last plaintext block is exactly one block size in size, the plaintext becomes XORed with a subkey K_1 before being encrypted. If the last plaintext block is smaller, a defined padding pattern will be appended to the plaintext block and it will be XORed with a subkey K_2 before being encrypted. Both subkeys, K_1 and K_2 are derived from the actual signing key.

The size of the resulting value equals the block size of the underlying cipher, which would be 128 bit in the case of AES. For certain usecases, such as MACs, the result may be truncated, leaving only the T_{len} most significant bits as described in [33] without sacrificing too much security.

CTR

Unlike the other previously explained block cipher modes, the counter (CTR) mode does not use the block cipher to encrypt the plaintext, but instead it sucessively encrypts blocks consisting of a initialization vector IV and a counter [43].
The plaintext block is then encrypted by bitwise XORing it with the resulting output of the block cipher as shown in Figure 2.3. This also eliminates any (security) problems associated with padding, because for messages whose size is not a multiple of the block size, the output of the block cipher can be truncated without any security implications.
At the start of an encrypted session, the counter value is set to zero. The IV may be publicly known and thus could be sent to the recipient together with the first message. The IV has to be a nonce and must therefore not be used for another encrypted session. A reuse of the IV renders the scheme vulnerable to certain attacks.

The sizes of the IV and the counter are variable, as long as the total size equals the block size of the underlying block cipher and as long as each of the values does not become too short.
A too small IV space will eventually lead to a IV reuse, while a too small counter value will at some point roll over and also result in a condition similar to a IV reuse.

If, for example, a 128 bit block cipher is used and both IV and counter are 64 bit in size, it allows encrypting up to 256 EiB of data without risking a counter overflow.

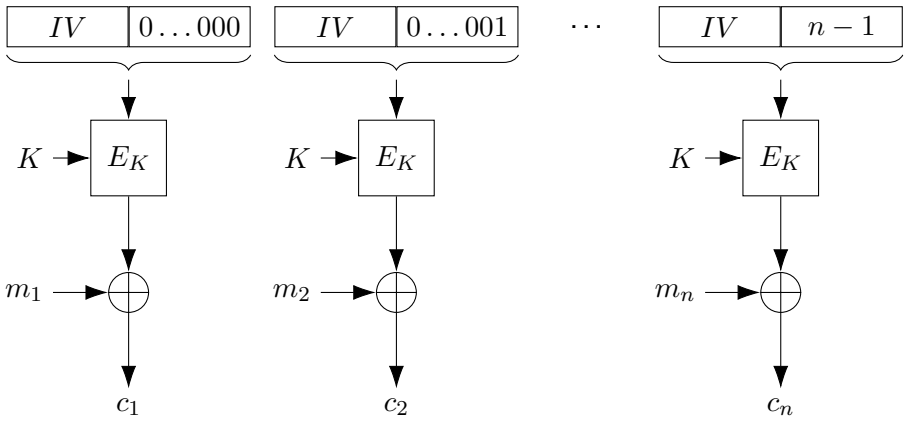


Figure 2.3: Block Cipher in CTR Mode

CCM

The CCM mode (Counter with Cipher Block Chaining Message Authentication Code) is designed to provide authentication and confidentiality [32]. This is achieved by first generating

a MAC using the CBC-MAC algorithm and then encrypting both the MAC and the message using the CTR mode . Even though a proof of security was delivered in [52], the complexity of the specification and the possibility to choose implementation parameters that negatively impact the security of the algorithm, were heavily criticized in [85]. Because of this and because better alternative modes are available, CCM is only recommended for legacy applications [93].

2.4.3 Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) was introduced in 1985 independently by Miller [64] and Koblitz [55].
Before that time, public-key cryptosystems were either based on integer multiplication/factorisation or on the Discrete Logarithm Problem (DLP), which were both easy to calculate in one way, but very hard in the other way.
ECC is based on the Elliptic Curve Discrete Logarithm Problem (ECDLP), which is similar to the DLP, but has a few advantages.

ECC works on a finite field, usually a prime field $GF(p)$. By using the four field operations (addition, subtraction, multiplication, inversion), a group of all points $(x, y),\ (x, y \in GF)$ with two group operations for point addition and point doubling can be defined.These operations are then used to define scalar multiplication.
Now, an elliptic curve and an imaginary point at infinity O are defined over $GF(p)$.It happens that the points on this curve together with O form a cyclic subgroup (which in certain cases contains every point on the curve (Theorem 9.2.1, [77, p. 246]) on which the scalar multiplication can be performed. If the group order of the curve is a prime, every element is a primitive (Theorem 8.2.4, [77, p. 214]) and can be used as a generator, which means that by multiplying the point with any natural number will result in a point on the curve.

The difficulty of the Elliptic Curve Cryptosystem is that scalar multiplication of a point on the curve is simple but the "Division" of two points can be very complex on certain curves E. This means, that to date there is no known method to efficiently, in sub-exponential time, find the natural number a in $aP = A$ ($P, A \in E$) when P and A are given. According to Hasse's theorem (Theorem 9.2.2, [77, p. 247]), the length of the prime p roughly determines the number of elements on the curve. A 192 bit prime thus results in roughly 2^{192} points on the curve.
The Problem is called ECDLP, because the problem is similar to the DLP in multiplicative groups.This implicates that cryptographic systems that rely on the DLP such as the Diffie-Hellman Key Exchange (DHKE) or Digital Signature Algorithm (DSA), can also be used with elliptic curves and benefit from its advantages.

Even though ECC requires a much deeper understanding than RSA or discrete logarithm schemes, it is slowly gaining in popularity. ECC requires considerably shorter keys (256-512

versus 3072-15360 [93]) for same level of security than RSA or DLP.While RSA is still faster than ECC in some applications, this is going to change as key sizes increase. According to [58], for securing a 256 bit AES key, 521 bit ECC can be expected to be 400 times faster than 15 360 bit RSA. And especially in small embedded devices, where memory is a luxury, ECC has a considerable smaller footprint [35].

2.4.4 Diffie-Hellman Key Exchange

In 1976, Diffie and Hellman have found a way to perform a key establishment over an insecure channel [29]. The so called Diffie-Hellman Key Exchange (DHKE) is an asymmetric algorithm in which both parties generate a public and a private key.

Before a key can be established, both parties agree on two integer values p and $\alpha \in \{2, 3, ..., p-2\}$ these are the so called *domain parameters* and do not have to be secret. p is a sufficiently large prime, and α is a primitive element of the multiplicative group $\mathbb{Z}_p^*$ of the prime field $\mathbb{Z}_p$. This means that every element of $\mathbb{Z}_p^*$ can be written as α^i.

Both parties now pick a random number as their private keys (Eq. 2.1) and calculate their respective public keys according to Eq. 2.2. Both parties can now exchange their public keys and calculate their shared key according to Eq. 2.3.

$$k_{priv,A} \in \{1, \ldots, p-1\} \qquad k_{priv,B} \in \{1, \ldots, p-1\} \tag{2.1}$$

$$k_{pub,A} = \alpha^{k_{priv,A}} \bmod (p) \qquad k_{pub,B} = \alpha^{k_{priv,B}} \bmod (p) \tag{2.2}$$

$$k_{AB} = k_{pub,B}^{k_{priv,A}} \qquad k_{AB} = k_{pub,A}^{k_{priv,B}} \tag{2.3}$$

The DHKE works because exponentiation is commutative and that exponentiation in $\mathbb{Z}_p^*$ is basically a one-way function. Because there are no known algorithms that can perform an integer factorization in polynomial time, it is computationally very expensive to calculate a private key even when the public keys and the domain parameters are known.

It should however be noted that the DHKE does not provide any authentication. This means that even though the key exchange can not be compromised by an passive eavesdropper, a MITM can exchange the public keys by its own public key and compromise any subsequent communication. Unless there is a dedicated authentication mechanism in place, DHKE provides no way to prove the authenticity of the public keys.

Elliptic Curve Diffie-Hellman Key Exchange

Cryphographic schemes which rely on the DLP, can also be modified to work on elliptic curves instead [77]. This makes it possible to implement an Elliptic Curve Diffie-Hellman Key Exchange (ECDHKE) which combines a DHKE with the advantages of ECC. It is thus possible, to perform a DHKE while being resource efficient and keeping the size of the transmitted key low.

This makes it an ideal key exchange protocol for low power wireless devices because they often do not have a lot of processing power and transmitting data wirelessly is always a very energy-consuming process.

3 Components of the BLE-Stack

In order to understand the flaws and known vulnerabilities of BLE, it is vital to understand the fundamental principles of the protocol. The following chapters will therefore give a brief introduction to the single components of the BLE communication stack and explain various procedures and properties.

Like any kind of modern digital communication, BLE consists of several layers, each adding more abstraction and functionality to the protocol. Before we discuss how data is actually transferred, we are going to introduce the basic layers and building blocks.

Bluetooth devices generally consist of two parts: The *Controller* and the *Host*. They are connected via the Host Controller Interface (HCI). The *Physical Layer* and the *Link Layer* are implemented in the Controller, all the other blocks are handled by the Host. Figure 3.1 provides a rough overview.

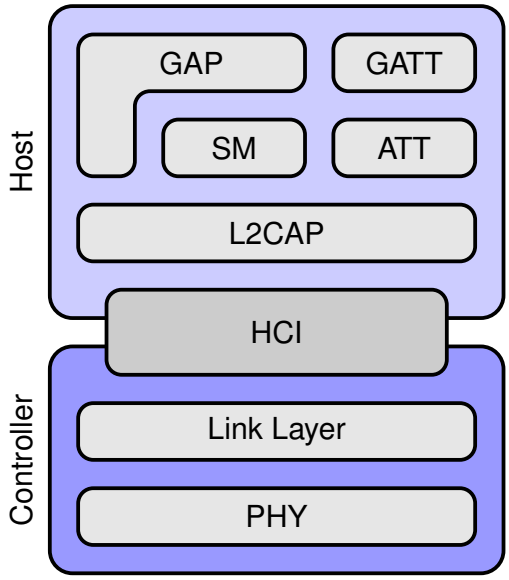


Figure 3.1: Components of the BLE-stack

Unless noted otherwise, the description in chapter 3 and 4 are referring to version 5 of the Bluetooth Core Specification [92].

3.1 Pysical Layer

The Physical Layer, also referred to as PHY or Baseband, is the lowest layer of any electronic data transmission technology. It contains the analog circuitry for interfacing with the transmission medium and performs modulation of digital into analog signals and demodulation of analog signals into digital signals.

In BLE, the Physical Layer operates in the unlicensed 2.4 GHz Industrial, Scientific, Medical (ISM) band, using gaussian frequency shift keying as modulation method. In order to cope with interference and fading, two common problems in radio technology, BLE uses a frequency hopping spread spectrum scheme on 40 channels with center frequencies between 2.402 GHz and 2.480 GHz. Each channel has a bandwidth of 1 MHz. The last three channels (37,38,39) are only used for advertising packets, whereas the remaining 37 channels (0-36) are being used for data packets. The channel numbering is shown in Figure 3.2.

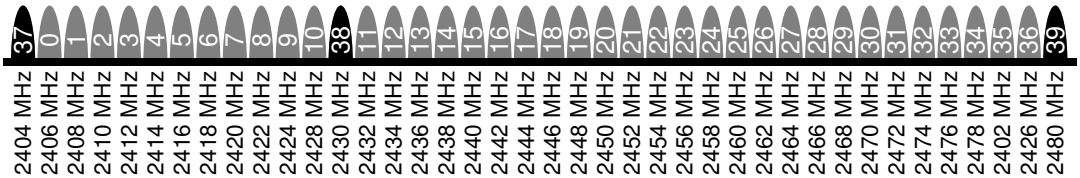


Figure 3.2: BLE channels and corresponding frequencies (from [97])

Because there is usually a lot of different wireless data traffic in the 2,4GHz ISM band, such as WIFI and Bluetooth Classic, BLE uses a technique called *frequency hopping spread spectrum*. This means that the devices synchronously change their channels on a pattern they previously agreed on. This way they can minimize their communication being disturbed by radio interference or even completely avoid crowded channels. The channel selection of the advertising channels is already designed in such a way to minimize interference with WIFI when WIFI-Channels 1, 6 and 11 are used.

The BLE Physical Layer is usually transmitting data with a symbol rate of 1 Msym/s, referred to as "LE 1M PHY". Bluetooth 5 additionally introduced the optional *LE 2M PHY*, which is doubling the symbol rate to 2 Msym/s.

3.2 Link Layer

The Link Layer sits on top of the Physical Layer. Among other things, it takes care of air interface packet framing, bitstream processing, AES encryption and contains a PRNG. Because it has to adhere to the tight timing requirements demanded by the specification to

ensure a seamless communication, it is the only real-time constrained layer of the whole protocol stack and is therefore mainly implemented in hardware.

3.2.1 States

The Link Layer can be seen as a state machine with five possible states as illustrated in Figure 3.3.

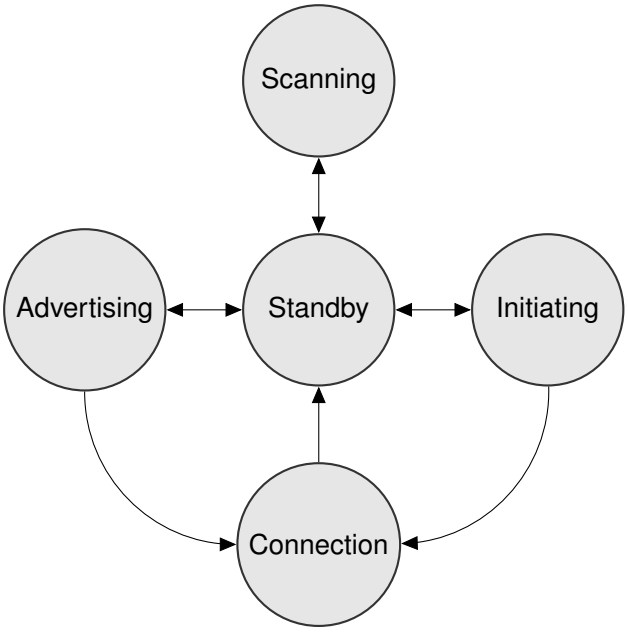


Figure 3.3: Link Layer state machine (from [92])

If the Link Layer is in the Standby State, it does neither transmit nor receive any data. If it is in the Advertising State, it transmits Advertising Packets and may listen to Scan or Connection Requests. If it receives a Scan Request, the Link Layer can transmit a Scan Response Packet which allows the requesting device to inquiry for more information without initiating a connection. Advertising and Scan Response Packets may contain up to 31 B of arbitrary data like device name or service Universal Unique Identifiers (UUIDs).

If the Link Layer is in the Scanning State, it listens for Advertising Packets and may send a Scan Request to an advertising device to inquiry more information.

A device in the Standby State can enter the Initiating State to connect to a device in the Advertising State. It therefore listens to the Advertising Channels and sends a Connect Packet as soon as it was able to receive a Advertising Packet from the respective device. The Link Layer of the initiating device then enters the Connection State in the Master Role, while

the link layer of the targeted advertising Device enters the Connection State in the Slave Role.

In the context of the Generic Access Profile (GAP), Master and Slave devices are also referred to as Central and Peripheral, while in the context of the Security Manager Protocol (SMP), Master and Slave Devices are referred to as Initiator and Responder.

3.2.2 Packet Format

An uncoded Link Layer Packet contains four components: Preamble, Access Address, Protocol Data Unit (PDU) and Cyclic Redundancy Check (CRC).
The Preamble is just a sequence of zeroes and ones to synchronize the clock of the receiving PHY to the clock of the sending PHY. The Preamble is one or two B in size, depending on whether the packet is sent over the LE 1M PHY or over the LE 2M PHY.
The 4 B Access Address is randomly generated by the Link Layer when a connection is established and prevents collisions when more than one device is sending on the same channel. The Access Address of Advertising Packets is always `0x8E89BED6`.
The PDU contains the actual Payload Data and can be between 2 and 257 B in size.
The last 3 B of a Packet contain a 24 bit CRC.

Bluetooth 5 has also introduced new, coded Link Layer Packet formats which enable data transmission over a longer distance by using more that one symbol per bit and thus lowering the data throughput. The coding is achieved by a special mechanism consisting of a *Forward Error Correction Convolutional Encoder* and a pattern mapping scheme. This, however, is beyond the scope of this work.

3.2.3 Bitstream Processing

Before a packet is sent to the PHY for transmission, the Link Layer may encrypt the data. The Link Layer then calculates the CRC and applies data whitening. The latter procedure is used to prevent long successions of ones or zeroes which can cause the clock of the receiving PHY to drift out of sync.
At the same time, the Link Layer also takes care of De-whitening, error-checking and decryption of data coming from the PHY.

3.3 Host Controller Interface (HCI)

The HCI provides a convenient abstraction layer with a cleanly defined interface between the Controller and the Host.
While there are many HCI commands defined in the specification, a lot of additional commands are left vendor specific and may or may not be available on different controllers.
Four different transport layers for the HCI are specified in [92]: USB, UART (with and without flow control lines) and Secure Digital card interface.

There are four different packet types that can be transmitted over HCI.
HCI Command Packets and HCI Event Packets are used for communication between the Host and the Controller. These commands are used for example to manage advertising and scanning procedures, create connections, establish encryption or perform adjustments on the PHY.
The third type are Asynchronous Connection-oriented (ACL) Data Packets, which are used to transfer Logical Link Control and Adaptation Protocol (L2CAP) data to the peer device.

In Bluetooth BR/EDR, the HCI can also transfer Synchronous Connection-Oriented (SCO) data between the Host and a BR/EDR Controller.

3.4 Logical Link Control and Adaptation Protocol (L2CAP)

The L2CAP is the lowest layer of the Host and serves two purposes. Firstly, it manages the multiplexing, packet fragmentation and encapsulation of Data Packets coming from higher layers. These are mainly Attribute Protocol (ATT) and SMP packets, but can also include user defined data.

Secondly, the L2CAP resource manager block is taking care of buffering, relative scheduling and regulating the sequential arrangement of the data fragments that are sent to the baseband via HCI. This is necessary because different L2CAP channels have different priorities and Quality of Service (QoS) requirements and the Controller does only have limited buffering capabilities.

3.5 Generic Access Profile (GAP)

The GAP defines the basic behaviour, functionality and basic requirements of a device. It can be seen as the definition of a dedicated control layer that specifies how devices interact

with each other outside of the actual protocol stack.

The GAP defines two fundamental concepts: Procedures, that are single actions a device can perform and roles which determine how a device behaves in general. Depending on the role of a device, there is an additional set of modes a device can operate in, which further define the device's behaviour.

Furthermore, the GAP works closely together with the Security Manager (SM) and also defines several aspects of security.

3.5.1 Roles

The GAP defines four roles of operation: Broadcaster, Observer, Central and Peripheral. These roles are tightly linked with the states of the Link Layer. A device can operate in different roles and, if the Link Layer supports it, also at the same time.

Roles impose certain restrictions and behavioural requirements and are usually associated with specific device types. Smartphones and computers tend to be Centrals and Observers. Devices with limited functionality or a single specific purpose are usually operating as Peripherals. Sensors or advertising beacons usually work as Broadcasters.

- The **Broadcaster Role** is usually taken by devices that regularly distribute small amounts of data to any listening device in their vincity. Instead of taking the effort to connecting to a device and using connection data packets, devices in the Broadcaster role periodically transmit data unidirectionally through Advertising Packets. In this case, the Link Layer takes the Advertiser role.

- The **Observer Role** is primarily designed for receive-only applications in which an observing device simply wants to collect data, usually enclosed in Advertising Packets, from broadcasting devices without establishing a dedicated connection. Here, the Link Layer takes the Scanner role.

- A device in the **Central Role** is listening for Advertising Packets and can then initiate a connection with a selected Peripheral. Depending on the device's capabilities, a Central can connect to multiple Peripherals and maintain several connections simultaneously. The link layer of Centrals is operating in the master role of the Connected State. Because BLE is an asymmetric protocol, Centrals generally have higher computing requirements.

- A device in the **Peripheral Role** is sending advertising packets and accepts connection requests from Centrals. The Link Layer is in this case operating in the Slave role of the

Connected State. Peripherals usually have low computing requirements and can be designed to be very energy efficient and very cheap in production.

3.5.2 Modes and Procedures

Modes can be seen as subsets of roles which further specify certain aspects of a devices behaviour in regard to its discoverability, connectability and bonding. These modes also impose certain restrictions on which procedures can be performed by a peer device.

Procedures are sets of actions which can be performed to accomplish a certain objective. Procedures are typically coupled with a mode on the peer device and usually consist of Link Layer control sequences and packet exchanges.

The following description of modes and procedures is not complete, but covers the basic elements and should provide a more practical explanation of the rather abstract concept of GAP modes and procedures.

Modes A Peripheral can operate in three Discoverability Modes. These Modes are indicated by two respective bits in the advertising packets: the **LE General Discoverable Mode** flag and the **LE Limited Discoverable Mode** flag. The device can advertise all the time (**general discoverable mode**), or for a limited time of several seconds (**limited discoverable mode**). A device in **non-discoverable mode** may send Advertising Packets, but as neither of the two flags is set, those packets will be ignored by Centrals.

A Peripheral can also operate in three different nodes regarding its connectability. A device in **non-connectable mode** will not respond to connection requests from a Central. In contrast to device in *unidirected connectable mode*, which will repond and establish a connection. A Peripheral in **direct connectable mode** is addressing its Advertising Packets to a certain Central from which it will exclusively accept a Connection Request.

A device in Peripheral or Central Role may be in **bondable** or **non-bondable mode**. This depends on whether or not it allows a peer device to bond, that is to store its keys after a successful pairing procedure.

Procedures A Central can perform two discovery procedures. A **general discoverable procedure** will discover only the Peripherals in general discoverable mode, whereas a **limited discoverable procedure** will discover only the Peripherals in limited discoverable mode.

A Central can also perform several connection establishment procedures. The **direct connection establishment procedure** establishes the connection with a specific Peripheral. The **auto connection establishment procedure** requires the device to keep a list of previously known Peripherals. Once it detects a device from the list, it automatically connects to it. The **general connection establishment procedure** consists of two steps. In the first step, the device scans for advertising devices and hands a list of available devices to the application. In the second step, the application selects one Peripheral from the list and Central establishes a connection. The **selective connection establishment procedure** is similar to the general connection establishment procedure except that it filters the list of devices and forwards only the previously known devices to the application.

By performing the **name discovery procedure**, a Central retrieves the Bluetooth device name of Peripherals. This is a UTF-8 formatted, human readable string that is either part of an Advertising Packet, or can be read through a Generic Attribute Profile (GATT) transaction over an established connection.

The **connection parameter update procedure** allows a device to change the key parameters of a connection, like the connection inteval or the slave latency. A Central can simply perform this procedure and change the parameters, while a Peripheral can request a parameter change which may then be accepted by the Central.

Both Centrals and Peripherals may terminate a connection at any time by performing the **terminate connection procedure**.

3.5.3 Security

The GAP also defines security related modes, properties and procedures by specifying which level of security is required for a certain data exchange and how it can be enforced. For this purpose, the GAP is also strongly linked with the SM.

Security Modes

The GAP defines two security modes, each consisting of different levels.

LE Security Mode 1

- **Level 1** No security (no authentication, no encryption)
- **Level 2** Unauthenticated pairing with encryption
- **Level 3** Authenticated pairing with encryption
- **Level 4** Authenticated LE Secure Connections pairing

LE Security Mode 2

Level 1 Unauthenticated pairing with data signing

Level 2 Authenticated pairing with data signing

These levels describe the current security level a connection operates in. Right after a connection is established, it always operates in Security Mode 1, Level 1. The security level can then be upgraded in various ways, depending on which shared keys are present in both devices.
If both devices have exchanged a Long-Term Key (LTK), they can enable encryption and upgrade to Security Mode 1, Level 2, 3 or 4, depending on if the LTK has been exchanged through an unauthenticated or an authenticated link, or over an authenticated LE Secure Connections pairing procedure. If both devices have exchanged a Connection Signature Resolving Key (CSRK), they can choose to send only signed data PDUs over an unencrypted link and thus upgrade to Security Mode 2, Level 1 or 2, depending on if the CSRK has been exchanged through an unauthenticated or an authenticated link.

A key is defined to be authenticated when it was transferred over an already authenticated link, or exchanged through an authenticated pairing procedure.

As BLE does not offer any procedures to disable encryption of an encrypted connection, it is not possible to fall back to Security Mode 1, Level 1 from Security Mode 1, Level 2-4. It is, however, possible to change the encryption key of an encrypted connection and thus change between the Levels 2, 3 and 4 of Security Mode 1, depending on the security properties of the new LTK.
If a connection or a key, respectively, does not fulfill certain security requirements of a service, the request will be refused and may be tried again after the connection's security Mode has been upgraded.

Security Procedures

The GAP also defines security related procedures.
The **Authentication Procedure** describes the exact procedure on how to handle requests that do not meet the required authentication level for a certain service and how to switch to an authenticated Security Mode.
An **Authorizaton Procedure** has to be performed if a service requires a confirmation by the user in order to continue with the requested procedure. This GAP procedure is only loosely defined by the specification and may be achieved, for example, by promting the user through a Graphical User Interface (GUI) or requesting them to push a button on a device.
Furthermore, the GAP defines a **Data Signing Procedure** and a **Signature Authentication Procedure** which define how PDUs should be signed and how the authenticity of those signatures can be verified. These procedures are only allowed in Security Mode 2 and are explained in detail in section 4.7.

3.6 Security Manager (SM)

The SM contains the implementation of the actual cryptographic protocols.
It provides support for three security relevant procedures: pairing, bonding and encryption re-establishment. Pairing describes a procedure in which a common link encryption key is exchanged and allows the current connection to be encrypted. Because the key is not stored, a pairing procedure has to be performed for every subsequent connection.
This can be prevented by performing a bonding procedure, in which the key is stored in the device's nonvolatile memory and allows to perform an encryption re-establishment of subsequent connections without the need ofperforming a pairing procedure.

In the context of the SM, two roles are defined: the Link Layer master (GAP Central) is referred to as Initiator, the Link Layer slave (GAP Peripheral) is referred to as Responder.
The SM also handles key storage and key management. It has a direct connection to the Controller so it can provide the stored keys during encryption establishment or pairing procedures. It furthermore is responsible for generating and resolving random addresses.

The communication between two SM entities is implemented through the SMP. It is a peer-to-peer protocol which runs over a fixed L2CAP channel and allows the SMs of two devices to directly communicate with each other. Table 3.1 shows a summary of all possible SMP commands

SMP Command	Description
Pairing Request	Sent by Master to initiate pairing
Pairing Response	Sent by Slave to accept pairing initiation
Pairing Confirm	Transport of confirmation values
Pairing Random	Transport of random Values
Pairing Failed	Indicates cancellation of pairing procedure
Encryption Information	Transport of LTK
Master Identification	Transport of EDIV and RAND
Identity Information	Transport of IRK
Identity Address Information	Transport of Identity Address
Signing Information	Transport of CSRK
Security Request	Sent by Slave to request encryption or higher security level
Pairing Public Key	Transport of public key values
Pairing DHKey Check	Transport of the DH-Check values
Pairing Keypress Notification	Informs remote device of user input during passkey entry

Table 3.1: Summary of all commands provided by the SMP

3.7 Attribute Protocol (ATT)

The ATT is a simple, stateless, peer-to-peer client/server protocol. Irrespective of wether a BLE device is currently a master or a slave device on the Link Layer, it can be a ATT server, an ATT client or both.

Most ATT transactions are performed in request-response pairs with a strict sequencing order which is, in the case of both devices acting as both server and client, independent of the directon. As long as a request has not been responded to, no further requests can be sent from the requesting device.

ATT packets are routed through the L2CAP layer and are thus encapsulated in L2CAP packets before being handed to the baseband.

Server and client should agree on a common ATT_MTU Value. This value defines the maximum size of a packet that may be sent between client an server.

3.7.1 Attributes

Attributes are the units in which the data in the a server is organized. Each attribute consists of a value, a handle, an UUID, and a set of permissions.

The attribute handle is a 16 bit value that uniquely identifies an attribute on a server. It is used by the client to reference and access those attributes. Attributes can be grouped by placing a specific attribute at the beginning of a range of other attributes.

The attribute UUID specifies the type of data contained in the attribte value and what the attribute represents. It can also be used instead of a handle to identify attributes in read or write requests. An UUID is a 128 bit value as specified in ISO/IEC 9834.8:2014. In order to reduce the amount of data that has to be stored and transferred, a range of UUID values has been pre-allocated and allows UUID within that range to be represented as 16 bit or 32 bit values. This usually applies to a limited set of often-used, specially assigned or registered purposes. The actual 128-bit UUID values can be obtained by multiplying the 16 bit or 32 bit UUID values with 2^{96} and adding the result to the Bluetooth Base UUID (`00000000-0000-1000-8000-00805F9B34FB`).

The attribute value is the actual data content of the attribute. The size of the value may be either fixed or variable. The specified size limit is 512 B.

Every attribute also has a set of permission values associated with it. These permissions are a combination of four different categories:

- The **Access Permissions** determine if an attribute value is readable, wrietable, both read- and writeable or neither read- nor writeable
- The **Encryption Permissions** determine whether an encrypted link is required to access the attribute
- The **Authentication Permissions** determine whether an authenticated link is required to access the attribute
- The **Authorization Permissions** determine whether the access to attribute requires authorization from a higher layer

These permissions can also be combined. An example would be to allow reading over an encrypted but unauthenticated link and require an encrypted and authenticated link for write transactions.

3.7.2 Operations

The ATT is based on six packet types.
The client can send either Command Packets or Request Packets. Request Packets will be answered by the server with a Response Packet, while Command Packets do not invoke any response from the server.
The server can send either Notification Packets or Indication Packets. Indication Packets will be confirmed by the client with a Confirmation-Packet, while Notification Packets do not invoke any response from the server.

Reading Attribute Values can be read in various ways. Attributes can be referenced by handle or by type(UUID). Depending on the Request, the server may return one or multiple values or just a part of a value. Read transactions always consist of a Request/Response pair.

Writing Attribute values can be written either through a Request/Response pair or through a command. Wtite commands may be signed. Several attribute values or several parts of a value (for example if a value is very large) can be written in a single atomic operation using queued write transactions.

Server Initiated A server may initiate unsolicited data transfers to push data to the client. These can be used for example to inform a client that a value of an attribute has just changed. The server can either send a Notification or an Indication. In the latter case, the client has to confirm the reception of the packet.

Finding Information Because a client may not know which attributes are available on a server, it can issue a "Find Information" transaction or a "Find By Type Value" transaction. The server then returns the mapping of attribute handles with their associated types in the former case and the handle range between an attribute that is referenced by its UUID and its next group delimiter.

Error Handling If a request can not be performed, the server can reply with an Error Response that may give some information about why the transaction failed. The specification lists several possible reasons that can be included in an Error Response. Reasons for Errors can be invalid handles, prohibited read or write operations, insufficient authentication, insufficient authorization, insufficient encryption or insufficient encryption key size.

3.8 Generic Attribute Profile (GATT)

The GATT adds a data abstraction model on top of the ATT.
A GATT profile describes the hierarchy of services, characteristics and attributes which are available in the attribute server. The ATT and the GATT define how data is organized and exchanged between applications. It is the main way of data transfer between BLE devices and, together with the GAP, acts as main interface to a BLE protocol stack.

A GATT server is defined by a GATT profile which the top level of the hierarchy. A GATT profile is composed of several elements that form a hierarchical structure of ATT attributes. A GATT profile consists of one or more services. Each service contains one or more characteristics and may also contain references to other services.

Characteristics contain the actual user data. A **characteristic definition** consists of a **characteristic declaration** and a **characteristic value**.
It may also contain one or more descriptors which add additional metadata to the characteristic. While the characteristic value contains the actual data, the characteristic definition describes the characteristic properties and how it can be accessed. It should be noted that every item (declaration, value, descriptors) is an ATT attribute with a handle, an UUID, a value and a set of permissions.

To enable device interoperability, the Bluetooth SIG has published a list of standardized GATT services and profiles that describe complete use cases. These standards include, for example, services and profiles for heart rate sensors, pulse oximeters, continuous glucose monitoring and blood pressore monitors.

4 BLE Tasks and Procedures

4.1 Advertising and Scanning

A Peripheral whose link layer is in advertising state, can periodically send advertising packets. These advertising packets can contain up to 31 B of data and can be received by another device which is actively listening to packets on the advertising channels, also referred to as scanning.

Advertising packets always contain the adress of the advertising device and may be directly adressed to a certain receiving device. Advertising packets also indicate if an advertising device is able to accept a connection request.

A scanning device can send a scan request packet to a currently advertising device. The advertising device can then reply with a scan response packet which can contain up to another 31 B of data.

The way a device advertises is defined by the GAP and transferred to the Link Layer through the HCI. This configuration includes the content of the advertising and scan response packet, the time interval between the advertising events and the used advertising channels. Usually, advertising packets are sent alternately on all three advertising channels, but may be restricted to only one or two channels.

4.2 Connection Establishment

A connection between two BLE devices provides the possibilitiy for a continuous, bidirectional exchange of data. Fig. 4.1 shows the basic structure of a connection.
The connection intervall serves as timebase for the connection. After every connection intervall, both devices change the radio channel. The master always sends the first packet at the beginning of new connection intervall.

To establish a connection, it requires a device whose Link Layer is currently in advertising state, and a device whose Link Layer is currently in initiating state. After the initiating device

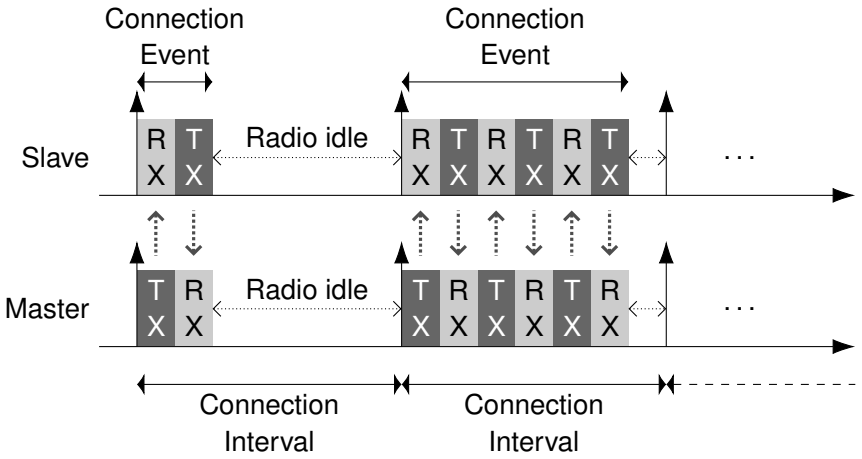


Figure 4.1: Timing pattern of a BLE connection (from [97])

has received an advertising packet from the peer device, it can reply with a Connect packet. The Connect packet contains the device address of both the initiating and the advertising device, as well as several important parameters required for connection establishment. Amongst others, the Connect Packet contains the following data:

Access Address (4 B)
: As described in section 3.2, every packet sent by the Link Layer is prepended with an Access Address to prevent collisions when more than one device happens to send data on the same channel. The Access Address is the same for packets being sent by any of the two connected devices. It is randomly generated by the Link Layer of the initating device and is static as long as the connection exists.

CRCInit (3 B)
: As described in section 3.2, every Link Layer packet contains a 24 bit CRC which is generated by a Linear Feedback Shift Register (LFSR) to check if the packet has been corrupted during transmission. To calculate the CRC for a packet, the LFSR is first initialized with CRCInit and then fed with the packet data. The CRCInit value is randomly generated by the Link Layer of the initating device.

Connection Interval
: The Connection Interval serves as timebase for the connection. The Connection Interval ranges between 7.6 ms and 4 s.

Slave Latency
: The Slave Latency value defines the number of connection events a slave may not listen.

Channel Map
: The Channel Map is essentially a bitfield which indicates which of the 37 data channels may be used for the connection. While usually all 37 channels are used, it may sometimes be advisable to avoid certain channels that might interfere with other wireless communication.

Hop Increment
The Hop Increment is used together with the Channel Map to calculate the channel for the next connection event. The Hop Increment is randomly chosen and has to be in the range of 5 to 16.

After the master has sent the Connect Packet, the connection is considered "created". Because there is no dedicated handshake procedure for a connection establishment, a connection is considered to be "established", once a data packet has been received from the slave.

4.3 Pairing

Two devices can communicate with each other as soon as they have established a connection. However, this connection and the data transferred over it, lacks confidentiality and authenticity. To provide these and other security features, the devices have to exchange one or several keys which should be only known to them and, amongst others, allow to encrypt and authenticate their communication.

This key exchange process is referred to as pairing. If the devices want to re-establish a secure connection at a later point in time without having to perform another pairing procedure, they can decide store the keys which is referred to as bonding.
The Pairing Process is controlled by the SM over the SMP.

There are several different ways of Pairing BLE devices. They provide different degrees of security and protection against certain kinds of attacks. In general, one has to distinguish between "LE Legacy" Pairing and "LE Secure Connections" Pairing.

LE Legacy Pairing was the original pairing method that was specified in Version 4.0 of the Bluetooth Core Specification [10]. It relies on a rather insecure and simple key exchange mechanism and contains other cryptographic flaws.

LE Secure Connections Pairing was introduced in Version 4.2 of the Bluetooth Core Specification [12]. It uses an ECDHKE protocol, which provides a significant increase in security against a passive eavesdropper.

Depending on which pairing method is being used, there are three or four so called *Pairing Association Models* which provide different degrees of authentication for a key exchange.

The *Just Works* Pairing Association Model does not provide any authentication at all. It thus does not provide any protection against MITM attacks, but requires no user interaction.

The *Passkey* Pairing Association Model requires the user to enter a six-digit decimal passkey into one or both devices. In practice, one device generates and displays the passkey and the user is requested to enter it on the other device. However, the specification also allows for a scenario in which the user enters a self-chosen number on both devices.
The *Out-of-Band* Pairing Association Model requires the devices to have the technical means to exchange 128 bit of authentication data over a second, secure channel. This may be, for example, a Near Field Communication (NFC) interface.
The *Numeric Comparison* Pairing Association Model authentication method is only available when LE Secure Connections Pairing is used. It requires the user to compare a 6-digit decimal number that is displayed by both devices and confirm on both devices if they are equal.

The pairing procedure consists of three phases.
In the first phase, two devices exchange their pairing features and agree on a pairing method. In the second phase, the devices either generate a Short Term Key (STK) when LE Legacy Pairing is being used, or the LTK in the case of LE Secure Connections Pairing. The second phase also contains the optional authentication of the key. The third step then includes the exchange of further keys.

4.3.1 Pairing phase 1

To initiate a pairing, Master (Initiator) and Slave (Responder) exchange a Pairing Request and a Pairing Response Packet. Both packets have the same structure and contain information about the features and the requirements of each device. Based on this information, the devices then agree on a suitable pairing and authentication method. Amongst others, the packets contain the following information:

IO Capabilities
Indicates the device's Input and Output capabilities.
Depending on whether a device has a display to show a 6-digit number, a numeric keyboard, and/or just a method to input "yes" and "no", a device falls into one of these categories:
- DisplayOnly
- DisplayYesNo
- KeyboardOnly
- NoInputNoOutput
- KeyboardDisplay

OOB Data Flag
Indicates whether or not the device has the out of band authentication data of the peer device.

Authentication Requirements

- Bonding Requested: If both devices set this flag, they are bonding after successful pairing.
- MITM Protection Requested: This flag indicates that the device requests authenticated encryption.
- Secure Connections Pairing Supported: This flag indicates that the device supports Secure Connections Pairing.

Maximum Encryption Key Size
Both devices exchange their maximum encryption key size to agree on a common value. A device may have an internal minimum encryption key length requirement and may cancel the pairing if the peer device can not support the key size.

Initiator Key Distribution
Indicates which keys the Initiator wants to send or which keys the Responder wants to receive

Responder Key Distribution
Indicates which keys the Initiator wants to receive or which keys the Responder wants to send

Selection of the Pairing Association Model

With the information exchanged in pairing phase 1, the devices agree on a suitable pairing method.

If both devices support LE Secure Connections and have thus set the appropriate flag (SC), an LE Secure Connections Pairing Association Model will be used. Otherwise, if one or both devices have not set the LE Secure Connections flag ($\overline{\text{SC}}$) an LE Legacy pairing association model will be used.

Depending on the LE Secure Connections flag and whether one or both of the devices have set the OOB-Data flag (OOB) or not ($\overline{\text{OOB}}$) the Out of Band (OOB) pairing method will be used. If the OOB pairing method is not being used and none of the devices has set the MITM-flag to request protection against MITM attacks ($\overline{\text{MITM}}$), the Just Works pairing method (JW) will be used.

Figure 4.2 shows the influence of SC-, OOB-, and MITM-Flags on the selection of the pairing association model. It should be noted that only the options in the upper left quadrant will lead to an LE Secure Connections pairing.

In some cases, the pairing method will be determined by the device's IO capabilities ("IO Cap."). Figure 4.3 shows how the respective IO capabilities lead to the Just Works (JW), Passkey (PK) or Numeric Comparison (NC) pairing association model.

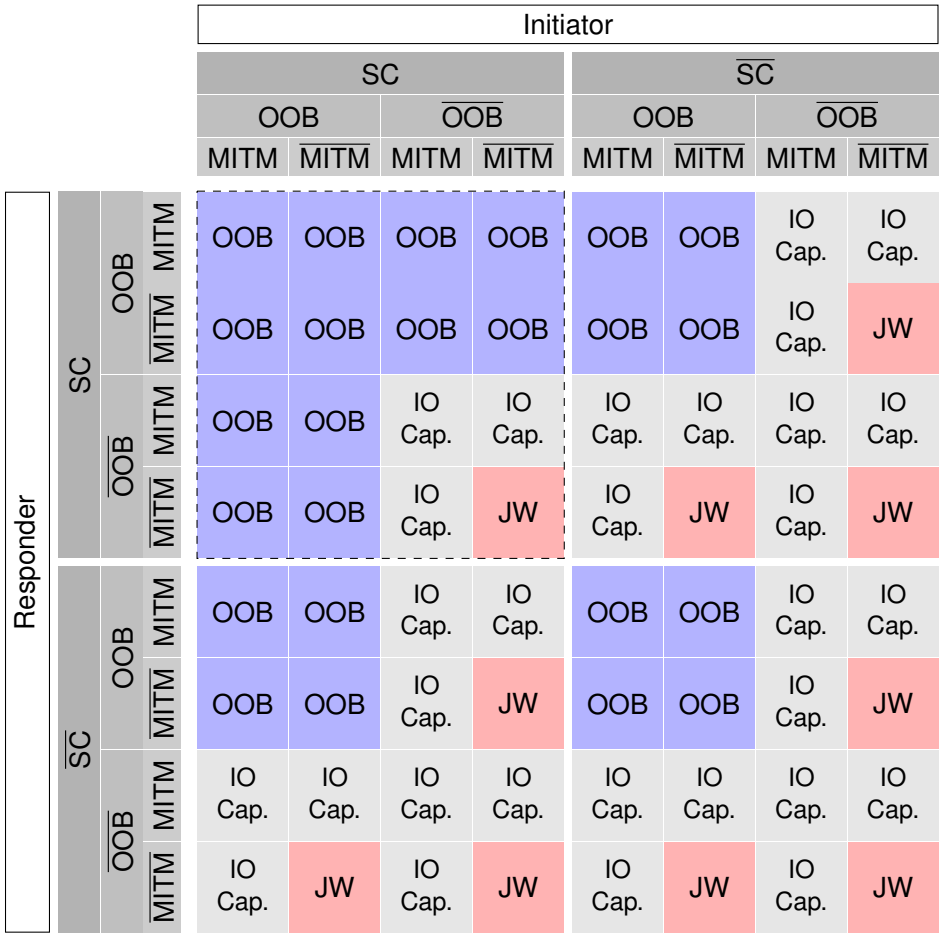


Figure 4.2: Influence of OOB, MITM and SC flags on the Pairing Association Model

4.3.2 Pairing phase 2 - LE Legacy

Both devices start with generating a random value $Mrand$ and $Srand$. They then have to obtain a shared temporary key TK. Depending on the chosen pairing scheme, TK can be obtained in three different ways. If the devices use the Just Works pairing scheme, the key is simply set to zero.
If the devices use the Passkey pairing scheme, one of the two devices randomly generates a number between 0 and 999999 and displays it to the user. The user then enters the number on the other device.
If the devices use the Out-of-Band pairing scheme, one of the two devices generates a random 128 bit value which is transferred to the other device via some other means of communication.

The devices then use a commitment scheme based on Eq. 4.1 to verify that both devices

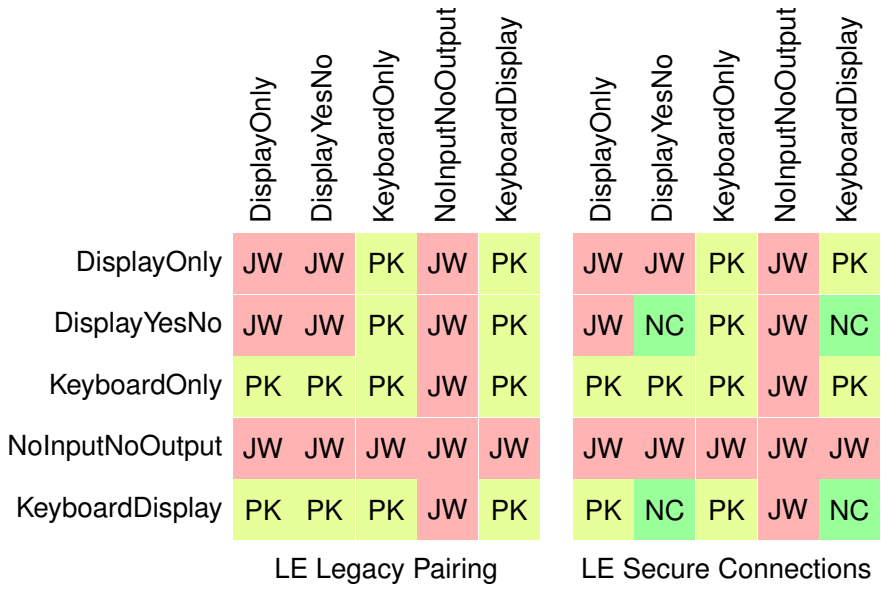


Figure 4.3: Influence of the IO capabilities on the Pairing Association Model

have the same TK. If the devices use the Passkey or the OOB pairing scheme, this step also authenticates the connection and should in theory protect against MITM attacks.

$$c_1(TK, rand, p_1, p_2) = AES_{TK}[AES_{TK}(rand \oplus p_1) \oplus p_2] \tag{4.1}$$

The confirm values are calculated using the respective random value, the TK and some connection related parameters p_1 and p_2 . After the devices have exchanged their confirm values, they successively exchange the corresponding rand values and verify that the other device knew the same TK when it was sending the confirm value.
This commitment scheme, along with its weakness is described in detail in section 6.1.

Now both devices can calculate the STK from TK, $Mrand$ and $Srand$ according to Eq. 4.2 and start encrypting the link by performing the procedure described in section 4.6.

$$STK = s_1(TK, Srand, Mrand) = AES_{TK}((Srand \bmod 2^{64})||(Mrand \bmod 2^{64})) \tag{4.2}$$

Fig. 4.4 shows a summary of the procedure.

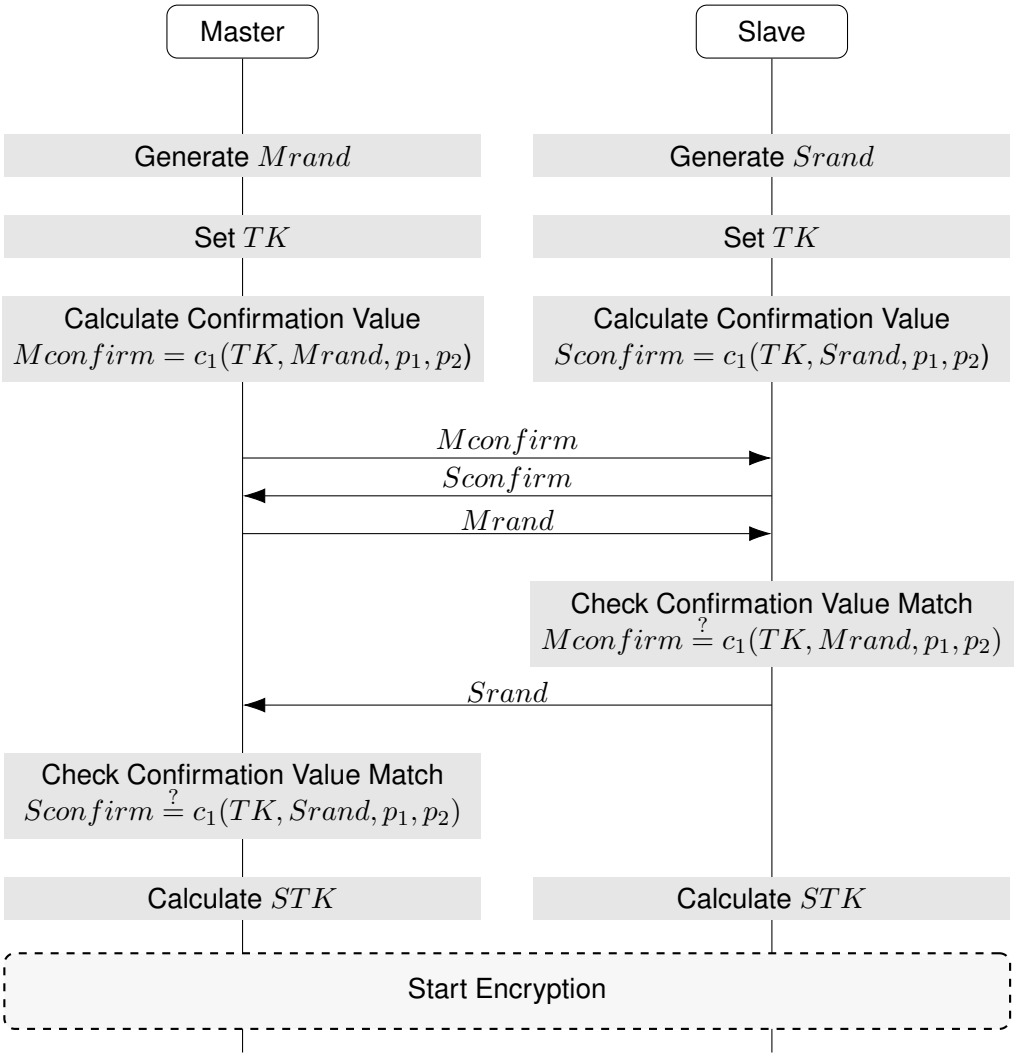


Figure 4.4: LE Legacy Pairing Sequence

4.3.3 Pairing phase 2 - LE Secure Connections

In the case of LE Secure Connections Pairing, the second pairing phase is significantly more complex. Explaining it in full detail would be beyond the scope of this work. We therefore summarize only the procedure of the Just Works, Numeric Comparison and Passkey LE Secure Connections Pairing Association Models to illustrate which security flaws of the LE Legacy Pairing methods have been fixed and which attacks are still possible.

The LE Secure Connections pairing process consists of three steps.

In the first step, both devices simply exchange their public keys.

In the second step, also referred to as "Authentication Stage 1", the authenticity of the public keys is verified. This step makes use of one of the four LE Secure Connections Pairing Association Models.

In the third step, also referred to as "Authentication Stage 2", both devices mutually confirm the successful pairing procedure, generate the actual LTK and encrypt the link.

Authentication Stage 1: Just Works & Numeric Comparison

The procedure for the Just Works and the Numeric Comparison pairing scheme is basically identical, except that in the case of Just Works, the last steps, the generation of the numeric value and the user confirmation, are omitted. Figure 4.5 shows the sequence of events and messages.

Both devices start by generating a random nonce Na and Nb. The nonce prevents replay attacks and thus must be generated freshly for each pairing process. The slave device then calculates a confirmation value Cb by concatenating the values of the x-coordinates of both public keys (PKa_x,PKb_x) with eight bits of zeros and calculating the AES-CMAC value using Nb as key.

The devices then exchange Cb and their respective nonces and the Master verifies the confirmation value. If the two devices are performing a Just Works pairing procedure, they are done with this step and can proceed to authentication stage 2. If they are performing a Numeric Comparison pairing procedure, then each of the devices now calculates an intermediate value (Va,Vb). The last six decimal digits of Va and Vb are then displayed to the user. If the user confirms the equality of the two values by giving the appropriate input on both devices, both devices can proceed with the second stage of the authentication phase.

Authentication Stage 1: Passkey Entry

The beginning of the LE Secure Connections Passkey Entry procedure is similar to the LE Legacy Passkey Entry procedure. Both devices start with an identical random Passkey value. Usually one device randomly generates and displays the Passkey value and the user enters it on the other device. If both devices do not have an appropriate display but only numeric keyboards, the user may also proceed by thinking of a value and entering the same value in both devices.

Unlike with the LE Legacy Passkey Entry procedure, here the passkey value is used only for authentication and is not used for key generation.

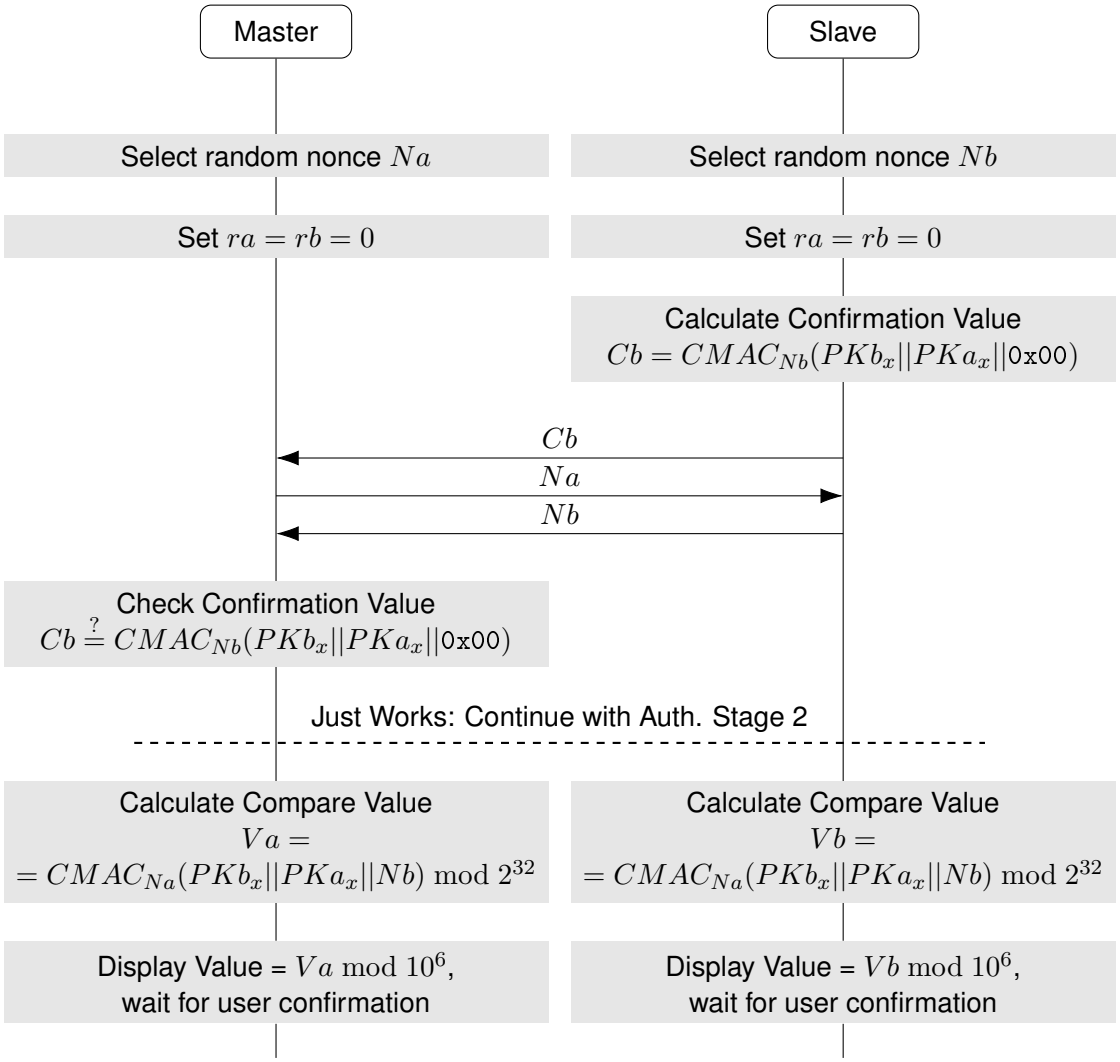


Figure 4.5: LE Secure Connections Just Works / Numeric Comparison authentication sequence

After both devices are set up with the passkey value, they repeatedly perform a mutual commitment procedure for every bit of the passkey. As soon as one commitment check fails on either side, the whole pairing process fails immediately.

This gradual disclosure process prevents an attacker from guessing more that 1 bit at a time. Because the devices generate fresh 128 bit nonces Na_i and Nb_i for every commitment round i, this thwarts any attempt to brute force a bit of the passkey. Furthermore the x-coordinates of both public keys (PKa_x,PKb_x) are included in the commitment value to prevent a MITM attacker from substituting the public keys on both sides with the attacker's own public key.

Figure 4.6 shows the sequence of events and messages.

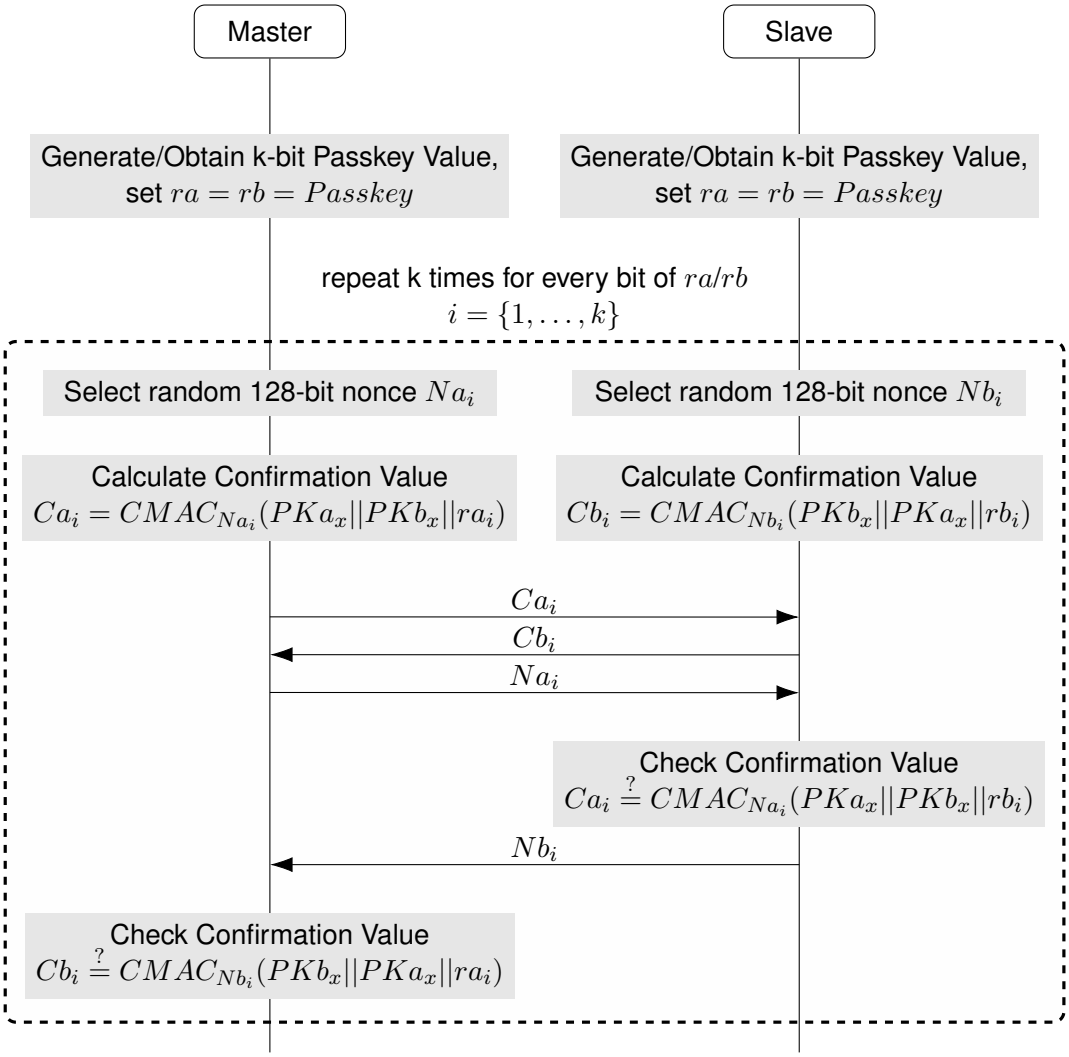


Figure 4.6: LE Secure Connections Passkey authentication sequence

Authentication Stage 2

The second authentication stage is identical in all four protocols. Both devices calculate their common Diffie-Hellman key $DHKey$, and use it to generate the LTK and a second value called $MacKey$ through a rather complex calculation employing two AES-CMAC computations and two given constants $SALT$ and $keyID$. The plaintext input in Eq. 4.3d and 4.3e consists of several concatenated values. A and B are the device addresses, Na and Nb are the nonces used during authentication stage 1. If passkey authentication has

been used, Na_{20} and Nb_{20} will be used.

$$SALT = \texttt{0x6C88 8391 AAF5 A538 6037 0BDB 5A60 83BE} \tag{4.3a}$$
$$keyID = \texttt{0x6274 6C65} \tag{4.3b}$$
$$T(W) = AES - CMAC_{SALT}(W) \tag{4.3c}$$
$$MacKey = AES - CMAC_{T(DHKey)}(\texttt{0x00}||keyID||Na||Nb||A||B||\texttt{0x0100}) \tag{4.3d}$$
$$LTK = AES - CMAC_{T(DHKey)}(\texttt{0x01}||keyID||Na||Nb||A||B||\texttt{0x0100}) \tag{4.3e}$$

Both devices then calculate a so called "Check Value" Ea (master) and Eb (slave). This is again done by concatenating several values and computing the AES-CMAC value as described in Eq. 4.4a and 4.4b. $IOCapA$ and $IOCapB$ contain the device's IO capabilities, OOB Data Flag and Authentication Requirements.

$$Ea = AES - CMAC_{MacKey}(Na||Nb||rb||IOCapA||A||B) \tag{4.4a}$$
$$Eb = AES - CMAC_{MacKey}(Nb||Na||ra||IOCapB||B||A) \tag{4.4b}$$

The devices then exchange Ea and Eb and re-calculate the Eq. 4.4. with their local values. If the results of are identical, the devices have mutually confirmed the successful completion of the key exchange and can start to encrypt the link with the LTK.

4.3.4 Pairing phase 3

Now that the link is encrypted and optionally authenticated, the devices can exchange the keys as indicated in the Key Distribution fields of the Pairing Feature exchange. In both cases, the devices may exchange their CSRK and Identity Resolving Key (IRK).
If case of an LE Legacy pairing, the devices may additionally exchange the LTK (including Encrypted Diversifier (EDIV) and RAND) and use it to encrypt any further communication.

It should be noted, that with LE Legacy pairing in theory both devices may generate and exchange a LTK, because they may decide to change their link layer roles later. This is however rarely used, because most devices are only designed to operate either as Master or Slave.

4.4 Keys and Key Management

4.4.1 Keys and Values

BLE uses three secret keys (LTK, CSRK, IRK) and two auxiliary values (EDIV, RAND) to provide encryption, message integrity and privacy.

LTK
: The 128 bit Long-Term Key (LTK) is being used for encrypting communication in the Link Layer. It is either locally generated by the slave and transferred to the master during LE Legacy Pairing, or established in both the master and the slave through an elliptic curve Diffie-Hellman key agreement protocol during LE Secure Connections Pairing. In LE Legacy Pairing, a LTK may theoretically be shorter than 128 bit, but not shorter than 56 bit. In this case, the empty MSbits have to be set to zero when a 128 bit value is required.

EDIV
: The 16 bit Encrypted Diversifier (EDIV) serves as Master Identification Value. The Master sends the value together with the RAND when it is requesting link encryption so the slave can use it to recover the previously exchanged LTK. A new EDIV is generated every time a new LTK is generated. This value is only used with LE Legacy Pairing.

RAND
: The RAND is a 64 bit value is sent together with the EDIV to enable the slave to recover a previously exchanged LTK. A new RAND is generated every time a new LTK is generated. This value is only used with LE Legacy Pairing.

CSRK
: The 128 bit Connection Signature Resolving Key (CSRK) is used to sign data as described in section 4.7. It is exchanged with a peer device during pairing which can then use it to verify signatures.

IRK
: The 128 bit Identity Resolving Key (IRK) is used to generate and resolve Resolvable Private Addresss (RPAs). It is generated by the device using the LE Privacy feature, and is exchanged with a peer during pairing.

4.4.2 Key Management

The said keys are generated, stored and managed by the Host. The specification proposes two methods of how this can be implemented: Either via a Database Lookup or via a more complex, but memory efficient key hierarchy dcheme.
This functionality is implemented in the SM, but the specification is rather vague regarding the actual implementation of the host.

Key Hierarchy

In this case, the slave device only stores an Encryption Root (ER) and an Identity Root (IR), both 128 bitin size. These values can be assigned, randomly generated by the device during manufacturing or generated by some other method, as long as it is ensured that they contain 128 bitof entropy.
A third value, the Diversifier Hiding Key (DHK), can be generated and stored like ER and IR, but can also be obtained as a part of the key hierarchy through IR.
The Key Hierarchy approach is especially useful for Peripherals with a very limited memory size because they only have to store 2×16 B for ER and IR.

LTK and CSRK
The ER value is used to generate and recover the CSRK and, when LE Legacy pairing is being used, also the LTK.
During pairing, the device generates a 16 bit Diversifier (DIV) and a 64 bit Random value RAND.
To generate the LTK, the DIV is encrypted with the ER.
To generate the CSRK, the value `0x10000` (r) is added to the DIV before encrypting it with ER.

RAND and DHK are then used to encrypt the DIV. This EDIV is then sent to the Central, together with the RAND and the three keys. If the Central wants to reestablish an encrypted connection to the device, it inludes both RAND and EDIV in the Encryption Request Packet. The Peripheral can then use these values to recalculate LTK and CSRK as shown in Figure 4.7

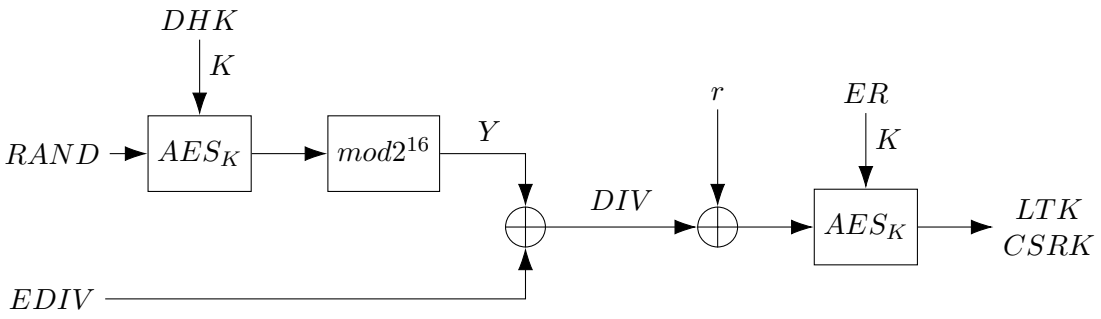


Figure 4.7: LTK/CSRK recovery

IRK and DHK
The IR can be used to generate the IRK and the DHK. Because these values are static, the

specification suggests to generate these values by simply encrypting the numbers 1 (for the IRK) and 3 (for the DHK) through AES using the IR as key.

Database Lookup

Instead of generating keys using a root key, every key may also generated completely randomly, as long as it is ensured to contain 128 bit of entropy. The keys can then be stored in a database and the EDIV is only used as index.

The specification also mentions the alternative method of using the DIV as index. In this case, every time a new set of keys is generated and distributed, an additional random value RAND has to be generated. Together with a DHK, which is also stored in the database, an EDIV value is calculated, like in Figure 4.7.
RAND and EDIV are then distributed with the keys and can later be used to recover the DIV and thus the keys in the database.

Key Management with LE Secure Connections

When two devices agree to pair using LE Secure Connections, the LTK is generated by the Security Manager as shown in 4.3.3. The function takes, the shared Diffie-Hellman key and several other known values as input and generates the LTK as well as the MacKey, which is used during the subsequent authentication process.

When using LE Secure Connections, the specification leaves most of the implementation of the Host. Especially regarding bonding and re-encryption connections with a previously exchanged key.

The abovementioned key generation and key management are only partially applicable when LE Secure Connections pairing is used because in this case, EDIV and RAND are not transmitted during Pairing. The lack of these values prevents using the key hierarchy scheme explained above to re-establish encryption using a previously exchanged LTK, because the Host has no way of identifying the remote device and thus can't provide the respective LTK to the Link Layer.

4.5 Security Requirements

For BLE, security requirements are defined in four different ways.

The SM (section 3.6) handles the pairing, key generation and key exchange. It uses the SMP to communicate with the SM of the peer device and can request certain security properties for the pairing procedure.

The GAP (section 3.5) defines and enforces certain security properties. It is closely linked to the SM and the GATT server.

The ATT defines a set of permissions to control how attributes may be read and written and which security properties are required.

In addition to the Attribute Permissions defined by the ATT, each GATT Characteristic Value has another set of properties which determine how the value may be accessed. These "*Characteristic Properties* can be used to restrict the access to a value and allow it to be accessed only by certain operations.

4.6 Data Encryption

The data which is being exchanged between two BLE devices can be encrypted. The encryption and decryption of PDUs is handled in the Link Layer. To enable data encryption in the Link Layer, the devices have to have performed a successful pairing. That is, they have either agreed on a common LTK using the ECDHKE (LE Secure Connections), or the master has successfully obtained the slave's LTK (LE Legacy).

To start an encrypted session, both master and slave have to agree on an *Initialization Vector* IV and a *Session Key Diversifier* SKD. Both are composed of two parts, a master part (SKD_m and IV_m) and a slave part ($SKDs$ and IV_s).

Each of these values is randomly generated by the respective Link Layer during the set up if the link encryption. The 128 bit SKD is then calculated by concatenating the 64 bit SKD_m and the 64 bit SKD_s and the 64 bit IV is calculated by concatenating the 32 bit IV_m and the 32 bit IV_s.

The respective Hosts supply the required LTK to the Link Layer, enabling them to calculate the *Session Key* SK which is unique for each encrypted session.

$$IV = IV_m||IV_sSKD = SKD_m||SKD_sSK = AES_{LTK}(SKD) \tag{4.5}$$

All subsequent PDUs are then encrypted using AES-CCM with SK as the encryption (and decryption) key.

The 13 B CCM nonce is constructed from the 64 bit IV, a single direction bit (1 for packets sent from the master, 0 for packets sent from the slave) and a 39 bit packet counter which is incremented for each newly encrypted data packet.

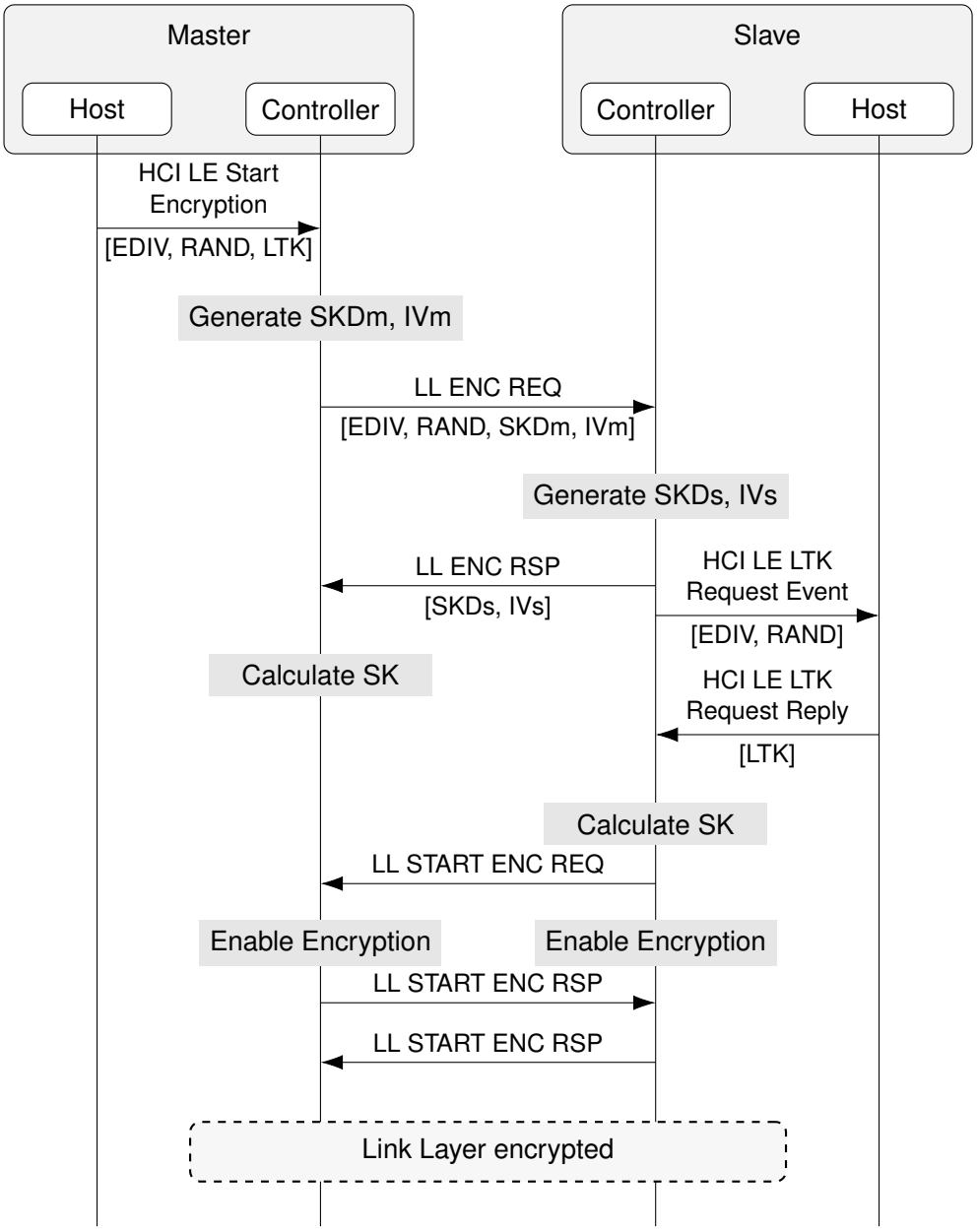


Figure 4.8: Link Layer encryption procedure

4.7 Data Signing

Even though, authenticated encryption with the LTK as required in LE Security mode 1 level 3+4 already offers both data integrity and authenticity, BLE also offers Data Signing without encryption as security feature. This procedure can only be used in LE Security mode 2, which provides two security levels for authenticated and unauthenticated pairing with data signing. However, this security mode does not allow any encryption to be used and does therefore not provide any confidentiality.

According to the specification, "[t]he data signing method is used by services that require fast connection set up and fast data transfer"[92]. While the user may implement an own protocol on top of the L2CAP layer that makes use of the specified data signing procedure, the ATT Signed write command is the only actual usecase for this Data Signing method that is mentioned in the specification.

The Signature is 12 B in size and consists of a 4 B SignCounter Value and a 8 B MAC. The SignCounter is stored locally and increased by one after each generation of a MAC. To sign a PDU, the SignCounter is appended to the PDU and the resulting message is encrypted using the AES-128 block cipher in CMAC mode and the CSRK as key. The lower 64 bit of the 128 bit CMAC output are then truncated, leaving only the 64 most significant bits as the actual MAC. The SignCounter and the MAC are then appended to the PDU, and the packet can be further processed and sent to the peer device.

In the peer device, the message will be disassembled in PDU, SignCounter and MAC. In order prevent replay attacks, the sign counter of a incoming signed message should always be compared to the one of the previous signed message. If the current value was previously used, i.e. is not bigger than the previous value, a replay attack is likely to happen and the packet should be discarded. The peer device should therefore securely store the last verified SignCounter, and should only accept messages whose SignCounter is larger than the one of the previous signed message. After this check, The CMAC of PDU and SignCounter is calculated again using the stored CSRK of the sender. If the calculated MAC matches the MAC contained in the signature, the signature is valid and the PDU can be further processed.

4.8 Privacy

BLE Devices are identified using a 48 bit device address. This address is transmitted in plain text as part of Advertising, Scanning and Connecting Messages. This makes is possible to track devices and thus the presence of the user which poses a threat to privacy [90]. BLE offers different types of device addresses and dedicated mechanism to prevent the users

privacy

A device address may either be a public device address or a random device address. Random device addresses are either static or private. Private random device addresses are either resolvable or non-resolvable.

A **Public Device Address** is hard coded in each device and requires obtaining a MAC address block from the IEEE registration authority.

A **Static Random Device Address** is a randomly generated address. The address stays unchanged while the device is powered, but may be initialized to a new value after a power cycle. This will, however, also invalidate the address stored in peer devices and devices won't be able to reconnect using the old address.

A **Private Non-Resolvabe Device Address** is a randomly generated address which is changed periodically.

A **Resolvable Private Address (RPA)** is generated periodically using a randomly generated 24 bit number PRAND. PRAND is then hashed by AES-encrypting it with the IRK and taking only the lower 24 bit of the output. The RPA is then generated by concatenating PRAND and the hash value to form the the 48 bit address.
A sevice that uses RPAs is also supposed to have an *Identity Address* which can either be a public or a random static device adress. The Identity Address is exchanged (confidentially) together with the IRK during the pairing key exchange procedure.
If a peer device receives an advertising packet from the device using a RPA, it can perform the same hash function on the PRAND part of the RPA successively using all the IRKs it has stored from previous bonding procedures. As soon an IRK results in the same hash value as in the hash part of the RPA, the peer device has successfully resolved the device's identity.

5 Practical Attacks on BLE

This chapter will explain several attacks on BLE for which practical implementations are already existing.

5.1 Passive Eavesdropping

The LE Legacy Pairing methods Just Works and Passkey in 4.0 and 4.1 rely on a deliberately weak key exchange mechanism. If an attacker is able to sniff the pairing sequence, he can easily guess or brute-force all the required information to decrypt the entire key exchange sequence and recover every key, including the LTK.

> "None of the pairing methods provide protection against a passive eavesdropper during the pairing process as predictable or easily established values for TK are used. If the pairing information is distributed without an eavesdropper being present then all the pairing methods provide confidentiality. Note: A future version of this specification will include elliptic curve cryptography and Diffie-Hellman public key exchanges that will provide passive eavesdropper protection."[10]

5.1.1 Explanation of the Attack

Cryptographically secure key exchange mechanisms, such as Diffie-Hellman, require a certain degree of processing power to perform the complex mathematic operations. When the first version of the specification was released in 2010, the Bluetooth SIG decided to use a less secure but also computationally less demanding key exchange algorithm instead, in order to keep both hardware requirements and energy consumption low.
The values required to generate the 128 bit STK, are the Temporary Key (TK), $Mrand$ and $Srand$. The latter two are transmitted in plaintext, and the TK just contains about 20 bit of entropy. This means that if an attacker is able to sniff these packets, it it possible to easily recover the STK and decrypt the consecutive communication.

5.1.2 Practical Realization

Even though, this weakness was already being pointed out in Version 4.0 of the Bluetooth Core Specification [10], a practical attack was first presented by Mike Ryan in 2012 [87]. He also released a software called "CrackLE" [88] which takes a a recorded stream of packets as input, automatically looks for pairing events, extracts the necessary parameters, brute-forces the TK, and finally decrypts the subsequent communication, including the key exchange.

The challenge in this kind of attack is to reliably capture the required packets. A successful brute fore attack on the pairing process requires the Connect Request Packet, both $Mrand$ and $Srand$ and both LL_ENC_REQ and LL_ENC_RSP packets.

To capture the packets, Michael Ossman and Dominic Spill have developed the ubertooth [63, 94], a SDR that is specially designed to capture BLE and Bluetooth Classic traffic. They also provide the corresponding software that facilitates the data acquisition process.

5.1.3 Impact Assessment

The necessary hardware and software tools for performing an passive eavesdropping are available and comparatively cheap. However, it requires quite some time and dedication to perform a successful attack with low-end devices like the ubertooth.

Depending on how much an attacker is willing to spend, sniffing and decrypting LE Legacy communication is quite easy.

Depending on the actual attack scenario and the targeted device, The implications of an attack may differ. As this is a purely passive attack, an attacker may not manipulate any data, so the damage is somewhat limited compared to a MITM attack. Nevertheless, medical devices can transmit critical data which often allows to draw conclusions regarding the health condition of a person. It should be in the user's own interest to keep sensitive data from getting into the hands of an adversary.

5.2 LE Legacy Just Works MITM Attack

If a connection between two devices is not authenticated, it is very easy for an attacker to mount a MITM attack, because the users can not verify that the device they are currently connected/connecting to, is actually the device they intended to connect to.

The BLE specification offers two (LE Legacy) or three (LE Secure Connections) pairing methods that allow the user to authenticate the connection between the devices.

These methods were designed to prevent a MITM attack, because to obtain the data on the authentication channel, the attacker needs physical access to the devices which is usually not the case

However, BLE also offers the Just Works pairing scheme, which does not provide any authentication. It allows the user to connect to any device without noticing.

5.2.1 Explanation of the Attack

The attack is very similar to the "Evil Twin" attack in WIFI networks [5], in which a rogue WIFI access point is set up that is indistinguishable from the victim's actual access point because it uses the same parameters (SSID, etc.). The victim's device can then be forced to connect to the rogue access point from where on the attacker is in a MITM position.

To perform a MITM attack with BLE, the attacker needs to have two programmable BLE devices in the vincinity of the user. This can be as simple as a small computer with two USB Bluetooth dongles. One Bluetooth controller acts as Central and connects to (and optionally pairs with) the user's Peripheral. The attacker now knows the device's advertising data as well as the content of the GATT database. This information can be used to create a malicious Peripheral whose BLE properties are indistinguishable from the real Peripheral. As soon as the user tries to connect to the Peripheral, he will inevitable connect to the malicious Peripheral. The attacker is now in a MITM position and may forward, intercept and manipulate any packets on their way between the user's Central and Peripheral.

5.2.2 Implementations

In 2016, two tools that enable the simple set up of such attack, have been published independently of each other: *BTLEjuice* by Damien Cauquil [26] and *Gattacker* by Sławomir Jasek [49].

They both use the libraries Noble [67] and Bleno [66] to implement a BLE Central which connects to the victim's Peripheral, and a BLE Peripheral which impersonates the victim's Peripheral by copying the Advertising Data and the GATT database. As soon as the victim's Central connects to the fraudulent Peripheral, the attacker has full control over the data being transferred.

Pairing

The tools support MITM attacks on unencrypted connections and LE Legacy Just Works pairing. It should be feasable to perform the same attack even when both devices use perform a "LE Secure Connections" Just Works pairing procedure, as it also does not provide authentication. Unfortunately, the underlying Noble and Bleno libraries do not yet support "LE Secure Connections".
Providing a proof-of-concept would be an interesting possibility for further research.

5.3 Denial-Of-Service

5.3.1 Jamming

Jamming the entire BLE frequency band requires quite some effort. However, if the devices are not yet connected, it's enough to jam the three advertising frequency bands. This can easily be done with three USB Bluetooth dongles and the debug commands, as we show in section 8.3.

5.3.2 Power Drain Attack

In many cases an attacker can easily connect to an advertising device and repeatedly request data from the device under attack. Because wireless data transmission consumes a lot of energy, the battery of the device can be drained quicker than expected and may render the device useless once the battery voltage drops below a certain level.

5.3.3 Connection Blocking

A more sophisticated and difficult to perform attack is the "Connection Blocking" attack. Because BLE Peripherals can usually only connect to one device at a time, an attacker can simply try to connect very quickly to the device and thus prevent the user from connecting to it.
This is similar to setting up a MITM attack, except that there is no requirement for the attacker to set up a fraudulent Peripheral. However, this can be done additionally, to make the attack less obvious to the user.

5.4 Location Tracking / Privacy

Devices can be used to track persons if advertise all the time, even periodic adress changes using a Resolvable RPA or a completely random address don't always help, if there is other, static data part of the Advertisement Packet. Some smartwatches, for example, use a periodically changing address, but at the same time advertise a device name with a more or less unique ID. This ID is part of the human readable device name and should simplify the pairing process for the user when several same devices are in proximity.

Location tracking is a severe privacy risk. However, the attacker has to have dedicated hardware in the respective places. BLE devices can only be tracked in two ways. Either the device itself is advertising and the attacker has deployed receivers, or the attacker has deployed devices which are constantly advertising, so called beacons, and at the same time control over a receiving device which is being carried around by the victim. The latter is the case when users are running a dedicated application on their smartphone, as is the case with Beacon Ad Campaigns.

6 Theoretical Attacks and Weaknesses

In addition to the attacks presented in the previous chapter, there are several other attacks which should in theory be feasable but have, to our knowledge, not been implemented yet.

6.1 Circumventing the LE Legacy Passkey Authentication

The Passkey Authentication Method is designed to authenticate the link and protect against MITM attacks. According to the specification [92] "The passkey Entry method provides protection against active “man-in-the-middle” (MITM) attacks as an active man-in-the-middle will succeed with a probability of 0.000001 on each invocation of the method."

In most cases, a Peripheral will generate the 6-digit Passkey and display it to the user, who then inputs it on a Central such as a smartphone. This passkey serves two functions: It authenticates both devices against each other, and it serves as Temporary Key TK from which the STK is derived at a later stage.

The method is designed to prevent MITM attacks in which an adversary sets up a malicious Central and a malicious Peripheral.

This method of authentication is based on the assumption that an attacker does not know the Passkey that is displayed by the real Peripheral and entered in the Central. The attacker's malicious devices can therefore not pair to their respective real counterparts.

To achieve this mutual authentication and at the same time to prevent prematurely disclosing TK, the authentication scheme is realized through a commitment scheme that is based on the commitment function c_1 shown in Eq. 6.1. The function generates a commitment value C from TK, a random value $rand$ and two publicly known values p_1 and p_2 which are composed of the two device addresses and the content of the pairing request and response packets.

$$C = c_1(TK, rand, p_1, p_2) = AES_{TK}[AES_{TK}(rand \oplus p_1) \oplus p_2] \tag{6.1}$$

The exact sequence of the procedure is as follows. TK_S is the Passkey that is shown on the Peripheral, TK_M is the passkey the user enters on the Central. In a typical scenario, the values are identical.

1. The master calculates a commitment value $Mconfirm$ consisting of the user-entered Passkey TK_M, a secret random value $Mrand$ and several public, connection specific values (device addresses, pairing request command) and sends it to the slave.
2. The slave calculates a commitment value $Sconfirm$ consisting of its generated Passkey TK_S, a secret random value $Srand$ and several public, connection specific values (device addresses, pairing request command) and sends it to the master.
3. The master sends the value $Mrand$ to the slave. The slave can now try to confirm the master's knowledge of the Passkey by calculating $Mconfirm_S$ using $Mrand$ and TK_S and verifying if $Mconfirm \stackrel{?}{=} Mconfirm_S$. If the two values are not equal, the slave aborts the pairing process.
4. The slave sends the value $Srand$ to the master. The master can now try to confirm the slaves's knowledge of the Passkey by calculating $Mconfirm_M$ using $Srand$ and TK_M and verifying if $Sconfirm \stackrel{?}{=} Sconfirm_M$. If the two values are not equal, the master aborts the pairing process.

This way, it is impossible for an attacker to gain knowledge about the Passkey before committing to it due to the hiding property of the commitment scheme: It is impossible to recover TK from any of the confirmation values C.

However, Tomáš Rosa showed in [86] that the commitment function c_1 lacks the binding property. This allows an attacker who knows TK to calculate an opener value $rand$ (Eq. 6.2) that will fulfill the commitment scheme regardless of the value previously committed to.

$$rand = AES^{-1}_{TK}[AES^{-1}_{TK}(C) \oplus p_2] \oplus p_1 \tag{6.2}$$

Because of the fixed order of the commitment scheme, an attacker can only implement a fraudulent slave device. As the commitment scheme does not directly disclose the TK, the attacker has to perform a brute-force search to recover the TK from $Mconfirm$ and $Mrand$. As there are only 10^6 different values possible for TK, which corresponds to about 20 bit of entropy, this could be achieved within a few seconds, especially on modern CPUs with hardware accelerated AES.

By exploiting this flaw, a MITM attack on LE Legacy pairing with Passkey Authentication could be implemented as follows.

1. An Attacker sets up a fraudulent Central $M_{Attacker}$ and connects to the victim's Peripheral S_{Victim}. The attacker hereby gains knowledge about the properties and characteristics of S_{Victim}.

2. The attacker sets up a fraudulent Peripheral $S_{Attacker}$ by using the properties and characteristics of the real Peripheral.
3. The victim's Central M_{Victim} connects to $S_{Attacker}$ and initiates a pairing process. M_{Victim} requests the user to input the Passkey.
4. Simultaneously, $M_{Attacker}$ initiates a pairing process with S_{Victim}, which causes S_{Victim} to generate and display the Passkey.
5. The user inputs the Passkey displayed by S_{Victim} into M_{Victim}.
6. $S_{Attacker}$ can now successfully pair with M_{Victim}. The attacker learns TK_M in this process.
    a) M_{Victim} calculates $Mconfirm$ and sends it $S_{Attacker}$
    b) $S_{Attacker}$ chooses a random TK_S, calculates $Sconfirm$ and sends it to M_Victim
    c) M_{Victim} sends $Mrand$ to $S_{Attacker}$. The attacker can now recover TK_M through a brute-force attack using $Mconfirm$ and $Mrand$, and can then calculate a new $Srand$ from $Sconfirm$ and TK_M using Eq. 6.2.
    d) $S_{Attacker}$ then sends the new $Srand$ to M_Victim, which will confirm that $Sconfirm = c_1(TK_M, Srand, p_1, p_2)$ and thus assume that $S_{Attacker}$ has generated the TK and that it is the legitimate peer device. $S_{Attacker}$ and M_{Victim} can now complete the pairing.
7. Given the user has entered the passkey correctly, the attacker now knows TK and can use it to perform a regular pairing procedure between $M_{Attacker}$ and S_{Victim}.

At the time of writing no practical realization of such an attack is known to us.
An implementation, similar to existing tools like mentioned in section 5.2, seems however very feasable. It could be achieved by implementing the necessary functionality into existing BLE frameworks like Noble [67] and Bleno [66]. A proof-of-concept would be an interesting topic for further research.

6.2 LE Secure Connections Just Works MITM Attack

A MITM attack is possible during an LE Legacy Just Works pairing procedure because the pairing scheme does not provide any method to authenticate the connection. Even though, the LE Secure Connections pairing methods offer an increased security level due to the usage of strong cryptographic algorithms, it still provides an unauthenticated Just Works pairing scheme.

An attacker could therefore perform a MITM attack very similar to the one explained in section 5.2, except that the attacker would have to perform two LE Secure Connections pairing procedures. It should in theory be enough to add LE Secure Connections support to Noble and Bleno. This would then also allow the reuse of existing frameworks such as BTLEjuice or Gattacker.

6.3 LE Secure Connections Authenticated Pairing Downgrade Attack

At the time of writing, there are no known weaknesses regarding the LE Secure Connections pairing schemes Numeric Comparison and Passkey. The ECDHKE provides immunity against passive eavesdropping, and their authentication procedures prevent any attempts to inject different public keys in order to perform a MITM attack.

It is, however, very likely that devices which support LE Secure Connections, are backward compatible and are going to use LE Legacy pairing with pre-4.2 devices.

While the BLE specification in theory provides the possibility to pair only using LE Secure Connections pairing schemes, this would also mean that a device manufacturer would loose a big part of the target group, namely those with older smartphones, for example.

In this case, an attacker could set up a MITM attack and force both devices to use LE Legacy Pairing. This can easily be achieved by setting the Secure Connections flag in the Pairing Request or Response Packet to 0. Because it can be assumed that both devices require authenticated pairing, an LE Legacy Passkey pairing procedure will take place on each side of the MITM. The passkey authentication can then be bypassed using the procedure described in section 6.1.

6.4 Static Passkey

In theory, the transition to LE Secure Connections Pairing should provide more security. But because the Pairing schemes are very similar to the "Secure Simple Pairing" (SSP) methods that were introduced in Bluetooth Classic 2.1 in 2007, it is possible to come back to attacks on SSP which have already been published some time ago.
One of these attacks was published by [4] and requires a static Passkey. This can be either printed in the device's instruction manual, on the device's case or it can be displayed electronically but may be hard coded and not randomly re-generated at every new pairing attempt.
The LE Secure Connections Passkey Confirmation procedure consists of a alternating bitwise commitment protocol between the two parties. This commitment step is repeated 20 times to mutually confirm the equality of the 20 bit of the 6-digit Passkey without disclosing more than one bit at a time. Once a bit does not match on both sides, the procedure is aborted.
In the case of a static Passkey, this behaviour can be used to recover the (secret) passkey of a device by performing not more than 20 repeated pairing attempts. Because the Passkey is confirmed bit by bit, the attacker can try pairing with a random passkey and can then learn up to which bit his chosen passkey was correct. The attacker can then change that bit of

his Passkey and try to pair again. Because the attacker has a 50% chance of guessing a bit correctly, he will in average need to pair only 10 times to recover the full 20 bit of the Passkey.

It should be noted that a static passkey is allowed by the Bluetooth Classic Specification, but not by the Bluetooth LE Specification. It is therefore questionable if devices that combine a static Passkey with LE Secure Connections Passkey Entry Pairing will obtain the necessary certification by the Bluetooth SIG.

6.5 Out-of-Band Eavesdropping

The mutual authentication of two devices via a second channel is, despite of the additional hardware requirements, a good way to combine authenticated key exchange with a pleasantly simple user experience. However, the choice of the OOB medium is very important, because it can provide an additional attack surface for passive eavesdropping and OOB attacks on the OOB data which can then be used to compromise the whole pairing process. As an example, the Bluetooth specification mentions the use of NFC to exchange the OOB data [72]. Depending on the proximity and the skills of the attacker, this may or may not be a secure channel as several attacks on NFC communication have been published [19, 42, 56, 20, 2].

7 Device Analysis

One of the main objectives of this work is to provide an overview over the use of BLE in medical or medical-related devcies and to assess their security vulnerabilities regarding the BLE interface in certain attack scenarios. In order to obtain this information, we acquired some samples of different medical devices and performed a limited security assessment.

During our market research on BLE-enabled medical devices, we came across two major problems.

Medical Devices are expensive and complicated to obtain without prescription
The major criteria was to find devices whose manipulation would cause serious physical harm to the user. Suitable candidates in this regard were, for example, wireless insulin pumps or wireless spinal cord stimulators. It turned out, however, that these devices are very expensive and require at least a prescription from a physician, what makes them very hard to obtain.

Implantable Medical Devices don't use Bluetooth
While public information about these devices is hard to find, the available publications on implantable devices such as Cardioverter-Defibrillators, Deep Brain Stimulators or implantable drug infusion pumps state that they don't communicate via BLE. Often manufacturers implement their own proprietary protocol and use different radio frequencies for data transmission.

Methodology & Tools For the assessment a total of 11 devices has been acquired, out of which 2, a blood glucose meter and a blood pressure monitor, have unexpectedly turned out to use Bluetooth Classic instead of BLE.
For every of the 9 remaining devices we have analyzed the possibility of performing a passive eavesdropping attack, a MITM attack and analyzed potential privacy risks. Furthermore we proposed countermeasures that could be implemented soley in software, so they could be integrated by the manufacturer without changing the current hardware. Table 7.1 shows the hardware and software which has been used to perform the analysis.

Attacker Model The analysis is based on two rather simple attacker models.

Device	Description
Ubertooth One	Software-defined Radio
Adafruit BLE Sniffer	Bluetooth sniffing
Cypress CySmart CY5677	BLE 4.2 USB Dongle
Cypress CY8CKIT-042-BLE-A	BLE 4.2 Evaluation Board
CSR8510 USB Dongle	Bluetooth 4.0 USB Dongle
Digilent Inc. Analog Discovery II	multi-function instrument
Raspberry Pi 2	Single Board Computer
Google Nexus 6	Smartphone

(a) Hardware used for the analysis

Program	Description
Wireshark 1.12	Network Protocol Analyzer
CrackLE	BLE Decryption Tool
BTLEjuice	MITM-Framework
CySmart 1.2	General BLE Debugging
PSoC Creator 4.0	Integrated Development Environment
ubertooth-btle	BLE traffic sniffing
ubertooth-specan	Spectrum Analysis
hcitool, hciconfig, hcidump	General linux tools
ent	Statistical Test Suite

(b) Software used for the analysis

Table 7.1: Tools used for the analysis

The main attacker model consists of an adversary that is trying to harm the user in some way. Either through data theft, data manipulation, or simply through preventing the victim from using a device.
This attacker has no physical access to neither the victim nor the victim's devices. The attacker also has no access to the internal processes, memory content and data structures present in any of the victim's devices. He has, however, control over a SDR and a computer with at least two Bluetooth adpters (USB Bluetooth dongles.) Both the SDR and the computer are located in vicinity of the victim, that is within the Bluetooth range of the victim's devices.

In some cases, a second attacker model can be applied. In this model, the user can be regarded as the adversary that is trying to manipulate data to gain personal benefits. We consider the user to have the same hardware equipment as the attacker described above, that is a SDR and a computer with at least two Bluetooth adapters.
He has no access to the internals of his devices, but is actively acting in a way to facilitate the attacks. This can include to deliberately un-pair and re-pair two devices when necessary, and keep all the devices in close proximity.

Threat Model The particular threat model depends very much on the attacker model and the specific medical device being targeted.

The main attacker model is allows for a number of both passive and active attacks.
Passive attacks do not involve any data manipulation, but aim at extracting any kind of user data which can, among other things, give information about the user's health condition or physical location.
Active attacks imply active intervention in the communication between the devices. Consequences of such attacks range from simply preventing the victim from using his device, to actively manipulating data on which the victim eventually bases his decisions on. In certain cases, this can lead to medical conditions remaining unnoticed or treated in a wrong way, both which may cause serious physical harm to the victim.

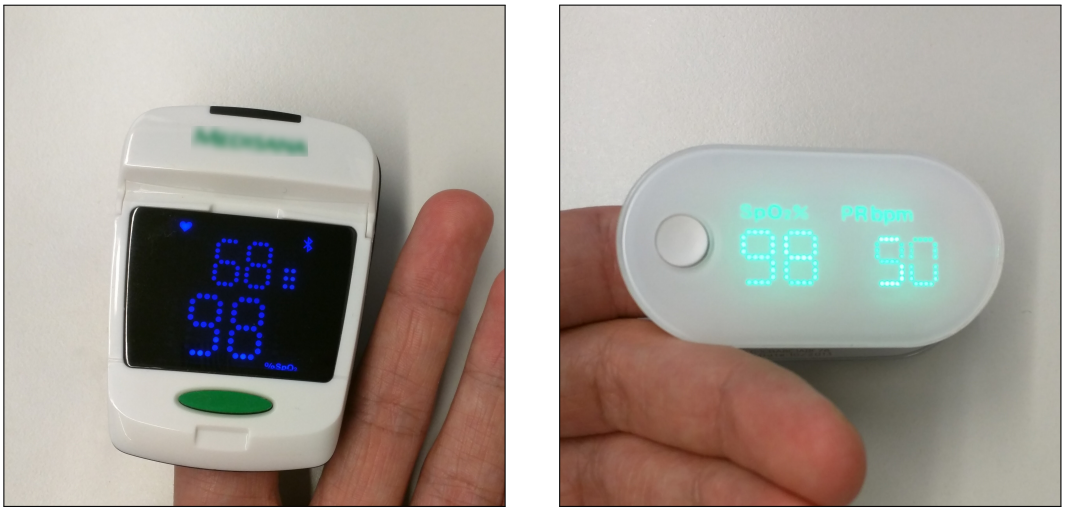
In the case of the secondary attacker model, the threat model involves a third party, that is willing to grant the user certain benefits if the user complies to certain predefined rules.
As an example, a insurance company could be willing to offer reduced insurance rates if the user agrees to regularly perform a set of physical exercises, which will be monitored through a fitness tracker. By manipulating the data on the way from the fitness tracker to the smartphone, where the data is being collected and sent to the insurance company, the user could pretend to perform the exercises and thus technically betray the insurance company.

7.1 Pulse Oximeters

A pulse oximeter is a device which measures a person's blood oxygen saturation and heart rate. Some devices even calculate the perfusion index, an indirect measure of peripheral perfusion. pulse oximeters work by shining light of two different wavelengths into the skin and measure the relative absorption by the pulsating, arterial blood. This works because the apsorption spectra of hemoglobin depends on its saturation with oxygen.
The detecting photosensor can either be on the same side of the body as the light source, measuring the reflection of the light, or on the opposite side of the body, measuring the transmission of the light. While reflectance pulse oximetry works on nearly any part of the body, transmittance pulse oximetry thin section of the body such as a finger or earlobe. The two devices we have analyzed, are both designed to be applied to a person's finger and use transmittance pulse oximetry.

7.1.1 Pulse-Oximeter 1

The first pulse oximeter has a push button and a LED display. It is shown in Figure 7.1a. The display consists of indicators for Bluetooth connectivity, a 6-segment bargraph display



(a) Pulse oximeter 1 (b) Pulse oximeter 2

Figure 7.1: The two pulse oximeters in operation

and five 7-segment digits for displaying the heartrate (3 digits) and the oxygen saturation (2 digits).

The device does not support pairing at all and thus all communication is unencrypted. It was thus possible, to passively sniff the communication. The Bluetooth connectivity of the device is only active for a short time. The device connects to a smartphone during the measurement, and transmits its data using a single GATT Notification. The device then immediately switches off. We were therefore not able to perform a MITM attack though this should in theory be feasible without any problems.
The GATT Notification contained only 6 B of data. An example can be seen below:
`55:aa:03:62:47:ac`
The packet contains three constant bytes (`55:aa:03`), the blood oxygen saturation (`62`), the heart rate (`47`) and a checksum (`ac`) which can be calculated by adding up the first five bytes and then discarding everything except for the least significant byte.

7.1.2 Pulse Oximeter 2

The second pulse oximeter has a push button and a LED display. It is shown in Figure 7.1b. The display consists of indicators for Bluetooth connectivity, battery status, a 6-segment bargraph display and a total of five 7-segment digits for displaying the oxygen saturation (2 digits) and the heartrate (3 digits).

As can be seen in Figure 7.2a, the corresponding smartphone application not only shows the

Listing 7.1: Content of the Notification Packets

```
a0 11 f0 f2 ac a7 62 41 05 d9 04 c1 90 c1 01 c9 04 b3 04 51
a0 11 f0 f4 ac a7 62 41 05 d9 04 c1 90 5c 04 4c 04 46 04 07
a0 11 f0 f6 ac a7 62 41 01 d9 04 c1 90 d8 03 7f 03 25 03 90
a0 11 f0 f8 ac a7 62 41 01 d9 04 c1 90 7e 02 37 02 f2 01 b9
a0 11 f0 fa ac a7 62 41 00 d9 04 c1 90 5a 01 03 01 ab 00 18
a0 11 f0 fc ac a7 62 41 01 d9 04 c1 90 00 00 00 00 00 00 11
a0 11 f0 fe ac a7 62 41 00 d9 04 c1 90 00 00 00 00 00 00 12
a0 11 f0 00 ac a7 62 40 03 8c 04 3e 92 9d 02 0c 03 fe 02 f6
a0 11 f0 02 ac a7 62 40 05 8c 04 3e 92 35 04 08 04 e9 03 7d
a0 11 f0 04 ac a7 62 40 04 8c 04 3e 92 de 03 b6 03 6e 03 58
a0 11 f0 06 ac a7 62 40 00 8c 04 3e 92 b1 02 53 02 fd 01 51
a0 11 f0 08 ac a7 62 40 00 8c 04 3e 92 70 01 33 01 f5 00 e7
a0 11 f0 0a ac a7 62 40 00 8c 04 3e 92 4c 00 00 00 00 00 9b
```

heartrate, oxygen saturation and the average perfusion index as the numeric values, but it also shows a real-time graph which apparently shows the momentary perfusion.

The device does neither require nor support pairing. It simply connects and transfers the data in plain text. It was thus possible to passively sniff the communication, as well as performing a MITM attack.

After some initial setup, the device periodically sends a GATT Notification to the smartphone, each containing 20 B of data. An excerpt can be seen in listing 7.1, where one line corresponds to one Notification Packet. The device sends one notification every 100 ms to 300 ms.

The proprietary, undocumented data format initially caused some problems regarding our MITM attack. However, we were able to decode the format and craft packets that passed the devices' internal checks.
One packet consists of a consecutive number (red), the blood oxygen saturation (violet), the heart rate (blue), some data that seems to be used for the real-time graph (orange) and a checksum (green). The checksum can be calculated by simply summing up the bytes 3 to 19, and then discarding everything except for the least significant byte.
This allowed us to modify and inject any arbitrary value. The result can be seen in Figure 7.2b.

(a) Normal Operation (b) Manipulated data due to MITM-Attack

Figure 7.2: Smartphone application

Privacy Concerns

In addition to the nonexistent security measures, we also discovered a severe privacy risk. As long as the battery is not empty, the device advertises all the time, even when it is not in use. There is no way to prevent this, which is problematic in several ways.

- The device advertises with its full name "Pulse Oximeter". If a person carries the device for example in a purse during a job interview, it can easily be detected and might generate suspicion about the interviewee's health state.
- The device is capable of conducting measurements without a smartphone and can internally store a number of measurements. These values can be downloaded by any smartphone capable of running the appropriate, freely available application.
- The device has a static address, this means that its advertising packets can be easily used to track the location of a person.

7.2 Blood Glucose Meter

During the last decades, small, portable and cheap electronic blood glucose measurement Devices, have revolutionized the life of millions of people suffering diabetes worldwide. But even though medical research is progressing every day, with 415 million people suffering diabetes worldwide in 2015, and 642 million people estimated to be suffering Diabetes by 2040, Diabetes is still on the rise [102, 100].
With the exception of people suffering only light forms of diabetes, most diabetics use a blood glucose meter to measure their current blood glucose levels. They then use this information to decide on how much insulin they have to inject themselves or how to adjust their insulin pumps. Even people that are using a Continuous Glucose Monitoring (CGM) System as described in section 7.3, have to use a blood glucose meter for calibration purposes.

Many modern blood glucose meter feature a wireless interface. This is being used to transmit the acquired data to a smartphone, insulin pump or CGM receiver. Looking at the growing number of diabetics and the dangerous consequences of hyper- and hypoglycemia, it quickly becomes clear that by manipulating these values severe harm can be done.

The blood glucose meter we analyzed had a monochrome graphical Liquid Crystal Display (LCD) and four buttons as shown in Fig. 7.3.
The device uses LE Legacy Pairing with Passkey authentication to pair with a smartphone. While bypassing the Passkey authentication and performing a MITM attack seemed to be feasable in theory as explained in section 6.1, it would have required a significant amount of time and effort to implement.

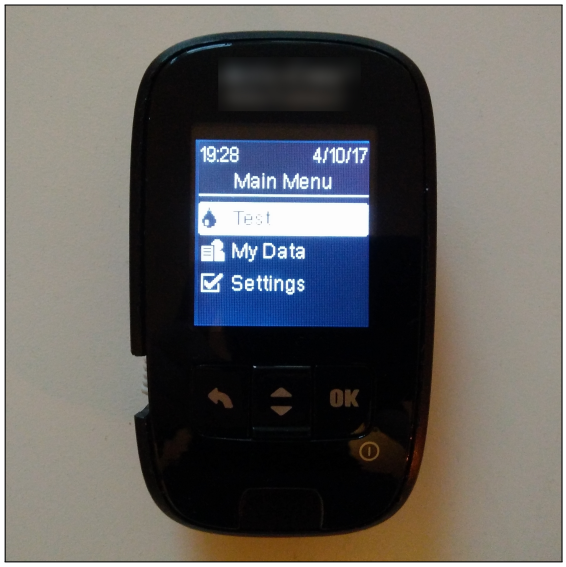


Figure 7.3: The blood glucose meter

Instead, we used an ubertooth SDR and the ubertooth-btle programm to capture the data between the device and a smartphone. This turned out to be far more difficult than expected, because of the hardware limitations of the ubertooth and because many other BLE devices were transmitting at the same time, making it even harder to capture the correct advertising and connection packets. We were able to silence most of our other devices by putting them in an aluminium diecast enclosure and storing the enclosure in a nearby fridge.

After this step, we had to repeatedly unpair and re-pair the device with the smartphone until we finally observed a connection establishment packet which allowed us to follow the connection.

Unfortunately, most of the time we were not able to capture all the packets that are required in order to decrypt the communication as described in section 5.1. It required several attempts until we were finally able to record a complete connection, pairing and data exchange sequence between the device and the corresponding smartphone application. After successfully decrypting the communication with crackle, we then obtained the keys and the communication in plaintext.

7.3 Continuous Glucose Monitoring System

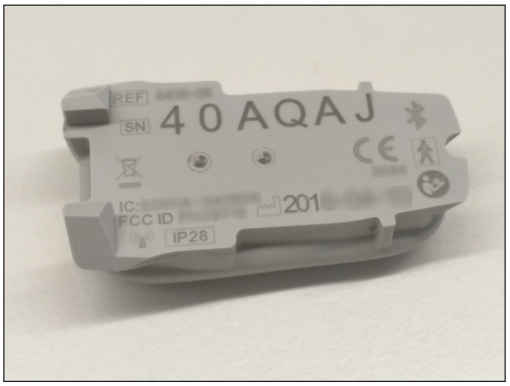
Continuous Glucose Monitoring (CGM) Systems allow the user to continuously monitor the glucose levels in the interstitial fluid. They usually consist of a sensor, a transmitter and a receiving unit.
While there are several different vendors and systems available on the market, which all work in a similar way, the following desciption refers to the single model we were able to analyze.

The sensor is a plastic carrier with a thin needle that is composed of two electrodes covered with a chemical compound. The needle is inserted under the skin and the plastic carrier is attached to the surrounding skin with sticky tape.

The actual data acquisition happens inside the transmitter which snaps onto the sensors plastic carrier and connects to the sensor electrodes with two spring contacts.
The transmitter containins the printed circuit board including batteries and is completely molded in plastic with the exception of two metal contacts.
While the sensor is usually being replaced and discarded after some days, the transmitter has a battery life span of several months.
Figure 7.4 shows top and bottom side of the transmitter.

The receiving unit can be either a smartphone, a dedicated hardware receiver, or integrated in a insulin pump. While the latter option would theoretically allow for a completely automated

(a) Top side of the transmitter



(b) Bottom side of the transmitter

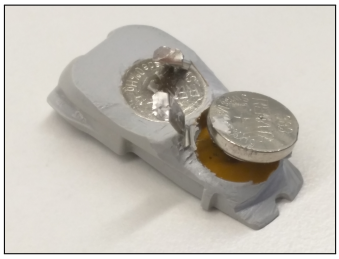
Figure 7.4: The CGM Transmitter

closed-loop system ("Artificial Pancreas"), these systems are only slowly entering the market and still have some limitations.
As of now, all CGM-systems on the market require the user to perform at least two calibration measurements per day with a common blood glucose meter.
It should be noted here, that as soon as insulin pumps become fully independend of human interaction and rely soley on the information they get from said glucose sensors, this also implies that the information being transmitted wirelessly should be especially protected against malicious attempts of manipulation.

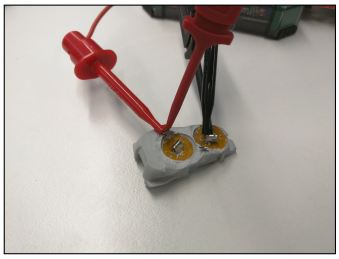
Because both CGM-sensors and CGM-transmitters are very expensive, we decided to acquire an used transmitter and remove the empty batteries as shown in Figure 7.5a. We were then able to power the device externally (Figure 7.5b) and built a small test rig (Figure 7.5c) that additionally allowed us to monitor the power consumption through a shunt resistor.
Unfortunately, the corresponding smartphone application refused to accept the refurbished transmitter.

We also attempted to create a replacement circuit to simulate the sensor electrodes, but we were not able to obtain a new or even used sensor, which made it impossible to study its electrical characteristics.

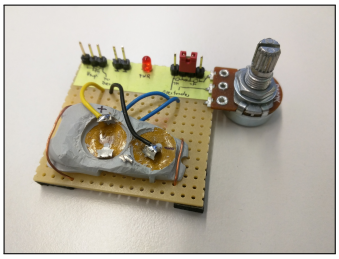
By observing the advertising behaviour of the transmitter and its power consumption, we found out that it was only advertising for about 7 s once every 5 min. Afterwards, the device enters a deep sleep state in which it consumed only 6 µA. Even connecting to the device did not have any effect and did not extend this time window. Figure 7.6a shows the acquired traces that are corresponding to the transmitter's power consumption in a 10 min window. Figure 7.6b shows a close-up on one of the bursts. Because we used a 100 Ω shunt resistor and the oscilloscope's voltage range was set to 100 mV per division, the current consumption in the figures is 1 mA per division.



(a) Transmitter with partially removed batteries



(b) Externally powered transmitter



(c) Test rig

Figure 7.5: Different stages of transmitter reanimation

Because of the very short active time window, we were not able to perform any attacks. Even connecting and reading the characteristics of the GATT database was only partially possible, because our tools were too slow.

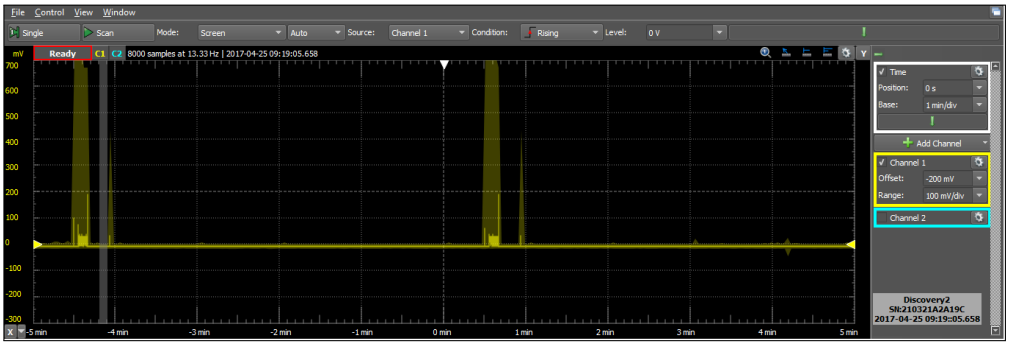
What caught our attention, however, was that the corresponding smartphone application requires the user to enter a six digit alphanumeric code which is printed on the back of the transmitter. This code could be used for application layer authentication or encryption measures. While it could be a coincidence, we found that two of the six digits were included in the advertising packet, which essentially reduces the entropy of the code from $\sim$31 bit down to $\sim$21 bit. In an attack scenario, this would significantly facilitate a brute-force attack on the code.

7.4 Thermometer

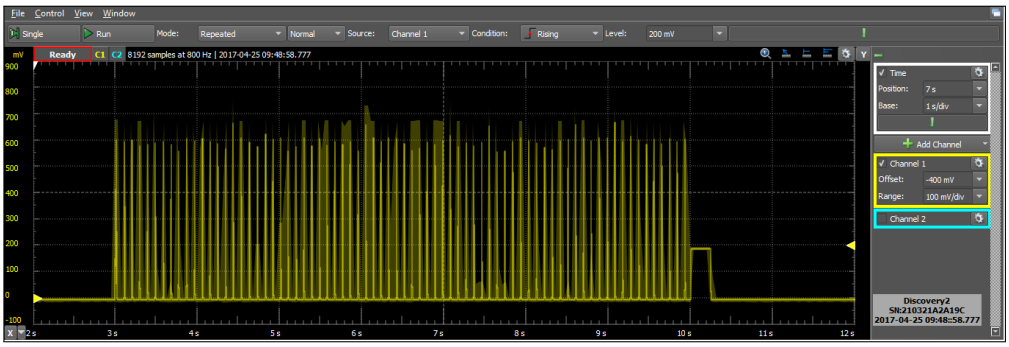
The probably most common electrical medical device which can be found in many households, is the clinical thermometer. It's being used to measure the body temerature to diagnose hypo- and hyperthermia and helps to make decisions on how to proceed with the treatment.

The thermometer we analyzed, has a rather unique appearance. It consists of a small, flat plastic housing, measuring about 52 mm$\times$32 mm$\times$7 mm with an exposed metallic temperature sensor, measuring about 5 mm in diameter. According to its instruction manual, the device is supposed to be stuck to the skin just below the armpit with the sensor being in contact with the skin. For this purpose, several disposable patches of medical grade adhesive tape are shipped with the device.

The fact that this device does not have any own capabilities to display the measured temperature makes this device very interesting for MITM attacks, because the user has to rely on the readings provided by the smartphone app. If these readings are being manipulated, a potentially harmful condition can stay undetected.



(a) 10 min window length



(b) 10 s window length

Figure 7.6: Tracing the transmitter's power consumption

The device at hand seemed to be sold under at least three different brands. We found two different applications in the Google Play Store that seemed compatible with the device. Both had rather bad ratings which have proven not to be without reason, because we have not been able to take the device into operation. Even without performing any attacks and by using two different smartphones with two very different Android versions, both apps failed to detect the device most of the time, and failed to display any data in those rare cases they managed to connect.

The device does not include any ASCII-String in neither its advertising nor in its scan response packet. There is thus no human readable data available to identify the device. Instead the smartphone application relies on the device serial number to identify the device. This serial number is printed on a label on the inside of the battery compartment and is also broadcasted in the advertising packet.

The device did not use any pairing and the connection was thus unencrypted. A MITM attack with BTLEjuice was partially achieved. While few GATT transactions could be observed, most of them contained only the Name of the OEM. After about two seconds, no more communication was taking place. This leads to the conclusion that if the device and the smartphone Application were working as intended, a MITM attack including data manipulation

would have been possible.

Though there is the possibility of the communication being secured against such attacks with application layer encryption or data signing, this is rather unlikely. Especially the fact that the serial number is transmitted in plaintext in the advertising packet, renders it useless for any cryptographic purposes.

(a) Front side with sensor (b) Open battery compartment and cover

Figure 7.7: The temperature monitor

7.5 Heart Rate Monitor

Chest strap heart rate monitors have been on the market for more than 25 years. While the first models were communicating over proprietary, vendor specific RF protocols, modern devices are using standards such as ANT or BLE to communicate with smartphones or smart watches [81].

We have examined two similar looking models which are shown in Figure 7.8. Both lack any input or output possibilities except for two metal buttons which connect to electrodes integrated in a textile chest strap. Both devices did not support pairing and are thus sending data completely unencrypted.
They were thus susceptible to both passive sniffing as well as to a MITM-attack, which allowed us to inject any arbitrary value. The results of this attack can be seen in Figures 7.9a and 7.9b.

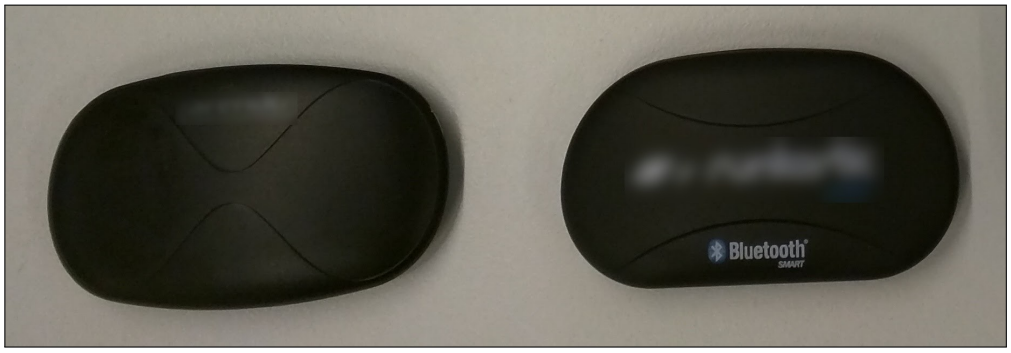


Figure 7.8: The two heart rate monitors

7.6 Fitness Trackers

Fitness trackers have become more and more popular throughout the last years, especially as they are getting more affordable. They enable the user to record their movements throughout the day and allow to keep track of specific activities like walking, running or cycling. Most fitness trackers also offer heart rate measurements and give feedback about the user's sleep quality when worn during the night. Some devices even feature a display and one or several buttons. This allows them to function as watch, display notifications from the connected smartphone and even allows them to remote control particular functions of the smartphone.

Fig. 7.10 shows the two fitness trackers we have analyzed.

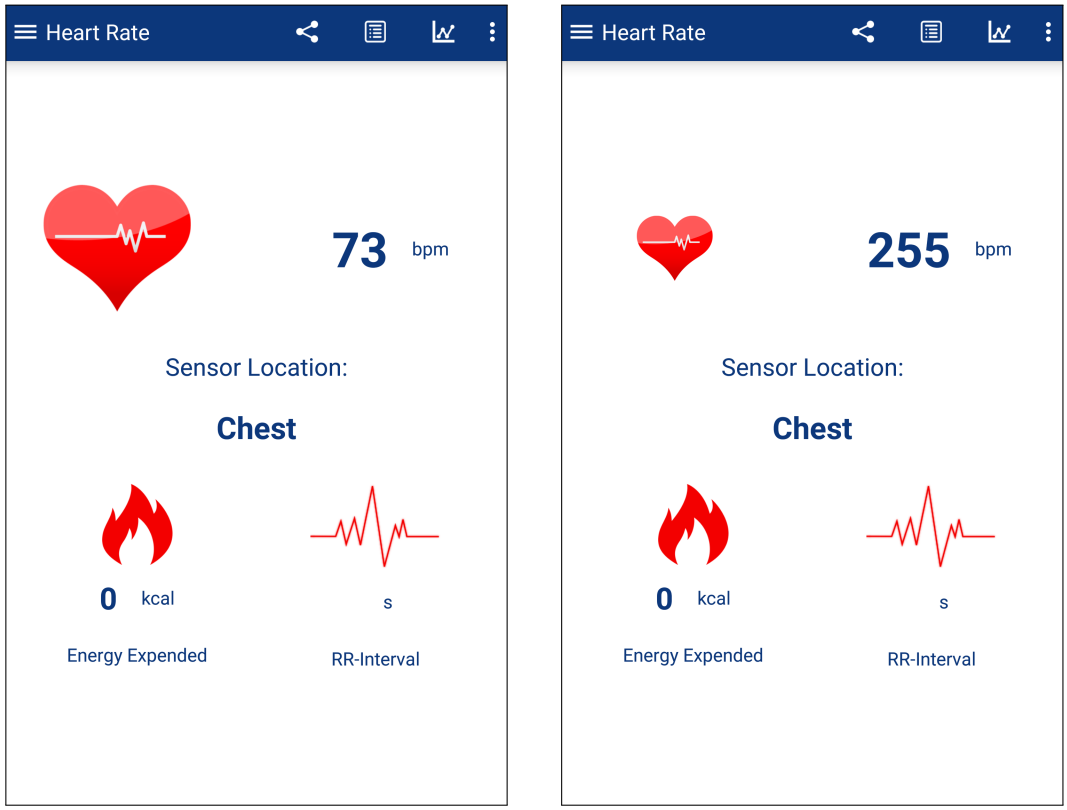
7.6.1 Fitness Tracker 1

The first device we were investigating features three white LEDs and a touch sensitive surface which is able to detect simple touches, but no swipes or similar gestures. It furthermore has a vibrating motor, a motion sensor and an optical heartrate rensor on the back, facing the skin.

The device does not use any pairing, but the application simply connects to the device and requires the user to tap the device to finish connecting. This is at least some security feature that makes it harder for the attacker to force a new connection.

We have successfully performed a passive eavesdropping attack during a heart rate measurement procedure initiated by the smartphone application. The device did not use any encryption and the data was simply sent in plaintext.

We were however not able to perform a MITM attack. This was probably due to the fact that



(a) During normal operation

(b) After injecting an arbitrary value

Figure 7.9: Screenshots of a smartphone application displaying the data from the heart rate monitor

the device was expecting some initialization data right after the connection establishment. Because our MITM attack requires several seconds to connect to the fitness tracker, instantiate the cloned device and then connect the smartphone application to the MITM proxy, we were not able to meet those tight timing requirements.

7.6.2 Fitness Tracker 2

The second fitness tracker we analyzed, features a monochrome OLED-Display, and, much like the previous device, a touch sensitive button, a vibrating motor, a motion sensor and an optical heartrate sensor on the back.

Because the device does not use any pairing, we were able to perform a MITM attack.

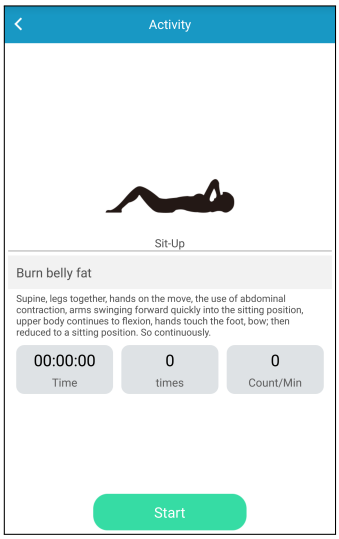
The device offers to track certain activities like walking or cycling and certain physical exer-

(a) Fitness tracker 1

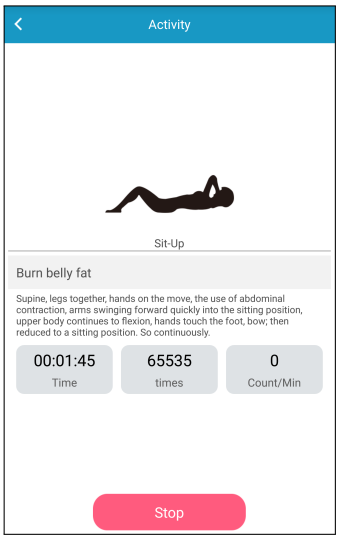
(b) Fitness tracker 2

Figure 7.10: Images of the two assessed fitness trackers

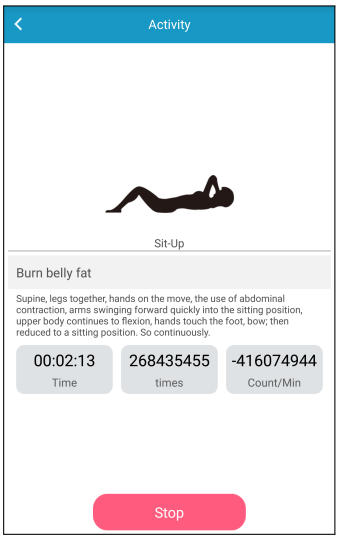
cises like sit-ups and jumping jacks. We were able to send a manipulated GATT notification to the connected smartphone application that allowed us to pretend having done any arbitrary number of exercise units. Fig. 7.11a shows the initial screen of the smartphone application before a "sit-up" exercise is started. In Fig. 7.11b we simply sent a manipulated notification to the smartphone. Fig. 7.11c shows that the smartphone application is not checking the incoming data for plausibility. This could indicate that the application might be vulnerable to some more sophisticated attacks that could be provoked, for example, through buffer overflows.



(a) Application set-up for counting sit-ups



(b) Application after injecting a manipulated data



(c) Application showing missing data sanity checks

Figure 7.11: Smartphone application in sit-up counting mode

The device is further capable of remotely controlling certain features of the smartphone

when the corresponding smartphone application is running in the background. This includes for example taking a picture and controlling the music playback. These functions do not necessarily pose a security threat, but could lead to dangerous situations, for example when a starting music playback is distracting the user in a dangerous traffic situation.

7.7 Summary

We have found that from nine devices, only one device uses authenticated pairing. Six devices do not use any pairing at all and transfer data completely unencrypted. However, we were only able to successfully perform a MITM attack on four of those devices.

We were not able to test the two remaining devices and are thus not able to make a proper statement. From what we could see, however, they might only use weak application layer encryption at most.

Table 7.2 summarizes which devices used encryption and authentication and on which devices we were able to successfully perform a passive eavesdropping of MITM attack.

	Enc.	Auth.	P. Eavesdropping	MITM
Blood Glucose Meter	Y	Y	Y	N
CGM Transmitter	?	?	?	?
Thermometer	?	?	?	?
Heart Rate Monitor 1	N	N	Y	Y
Heart Rate Monitor 2	N	N	Y	Y
Pulse Oximeter 1	N	N	Y	N
Pulse Oximeter 2	N	N	Y	Y
Fitness Tracker 1	N	N	Y	N
Fitness Tracker 2	N	N	Y	Y

Table 7.2: Summary of the assessment

8 Further Experimentation

In this chapter we present several other experiments which we have conducted during our research. They might serve as starting points for future investigations.

8.1 Presence Tracking

During our research, we repeatedly received advertising packets containing always the same device name, but always a changing device address. Further investigation lead to the conclusion that these packets are broadcasted by a colleague's fitness tracking watch.
The watch apparently used a periodically changing RPA to prevent tracking and provide some level of privacy.

However, we were able to bypass this privacy protection mechanism because of two implementation flaws: (1) The device only changes its address, but not the content of the advertising packets, which, amongst others, included the static device name. Because we knew that there is only one person among our colleagues who is wearing such a device, the periodic change of the device address did not impact our experiment.

Even if this would not have been the case, (2) we could have been able to track the device because of the predictable timing behaviour of the address change. By looking at our logs, we were able to spot advertising packets from a new device address reliably every 15 minutes. Because at the same time we did not receive any further advertising packets from the old address, we were able to keep track of the device addresses for some time.

After 54 h of logging incoming advertising data in our vincinity, we have obtained a result as shown in Fig. 8.1. The presence times of the person are clearly visible.

The received signal strength indicator (RSSI), however, is not very meaningful in this case. While in theory it would correlate with the proximity of the person to our receiver, the data is not very reliable because obstacles such as furniture and people were constantly moving throughout the office, causing random attentuations and reflexions to the radio signal. It would, however be possible to distribute several receivers throughout a building and use triangulation methods to precisely track the location of a person.

This experiment serves as a good example to show the limitations of the LE Privacy feature.

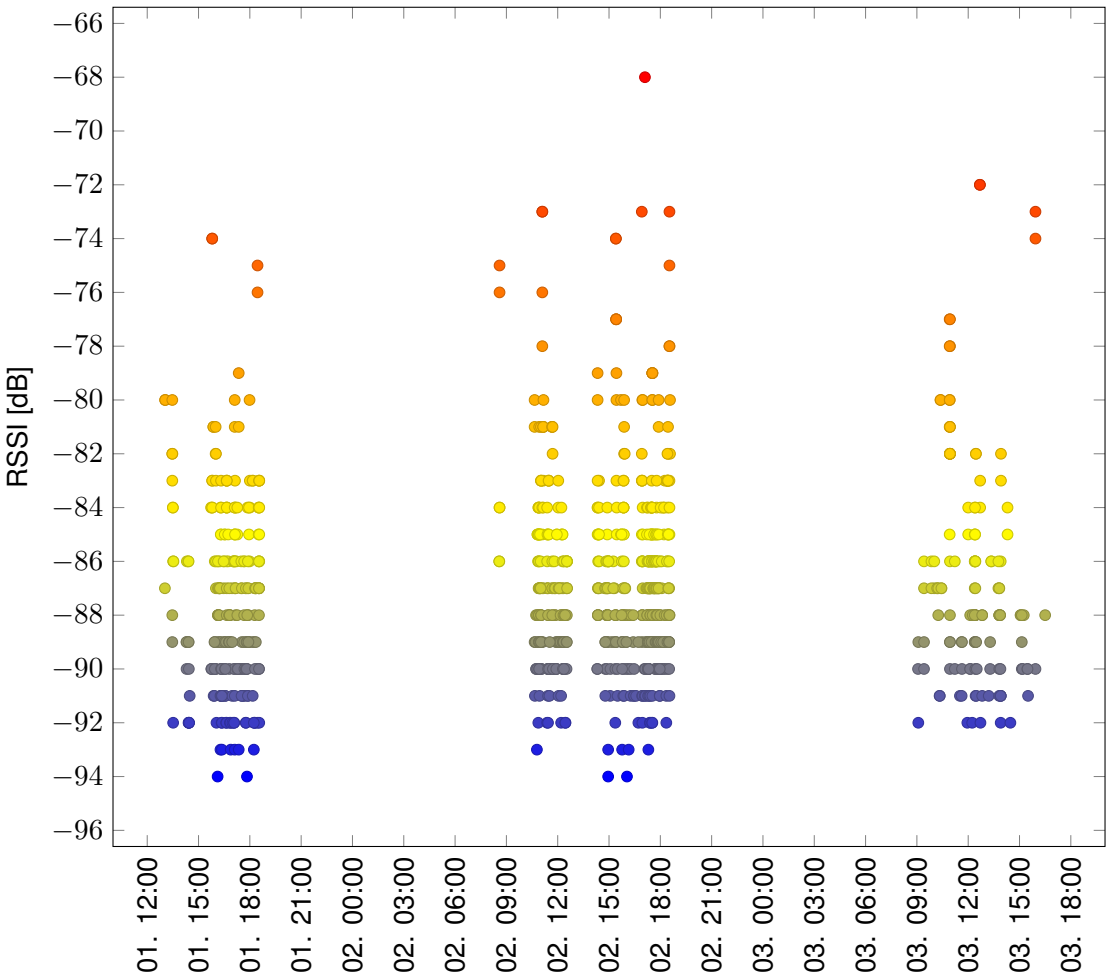


Figure 8.1: Signal strength of advertisement packets from a unique fitness tracking watch during 54 h timeframe

8.2 Pseudo Random Number Generator

As per specification, BLE controllers are required to contain a PRNG which is capable of providing pseudo-random numbers which are being used for many security relevant functions such as the generation of keys, Passkeys and nonces. If the PRNG is compromised in such a way that an attacker can guess its future output, the security functions that use the generated numbers can not be considered secure anymore.

While a proper cryptographic analysis would have gone too far, we had a quick look at the pseudo-random numbers generated by three devices:

- an off-the-shelf Bluetooth 4.0 USB dongle
- the PRNG of a microcontroller with integrated BLE interface
- the passkey values generated by a device using LE Legacy pairing with Passkey authentication.

8.2.1 Off-the-shelf Dongles

Since the first version of BLE, V4.0, the specification requires the implementation of a HCI_LE_Rand-Command which enables the Host to request 8 B of pseudo-random data from the Controller's internal PRNG .

We wrote a small Python script which repeatedly requested 8 B of pseudo-random data via the HCI of a USB-Bluetooth 4.0 dongle. The script itself calls the `hcitool` program to send the HCI-command to the Controller. It then formats the data and writes it to a file for later inspection.

The experiment was split in two parts.
In the first part of the experiment (A), the dongle was left plugged in a PC and pseudo-random data was successively being requested and stored.

In the second part of the experiment (B), the 5 V line of the USB connection was cut open and the dongle's power was supplied by an external, computer controlled power supply. Because the power supply of the dongle could now be controlled through the PC, it allowed us to simulate repeated plugging the dongle in and out of the USB port.

We ran two data acquisition cycles.
In the first one (B1), we disconnected the power only for 3 s, which was the minimum time for the operating system to handle a USB disconnection and a subsequent reconnection. In the second cycle (B2), we disconnected the power for 10 s. The goal was to find evidence of a weak or even hard-coded seed of the PRNG.

To provide a better comparability, we also analyzed a reduced subset of the datasets (A) and (B1). We used the programm Ent [51] to analyze the statistical properties of our data. The result can be seen in Table 8.1.

	A	B.1	B.2	A	B.1
Size of dataset (Bytes)	$\sim 1.11 \times 10^9$	1.14×10^7	7.98×10^6	7.98×10^6	7.98×10^6
Entropy (Bits per Byte)	8.000000	7.999981	7.999980	7.999975	7.999975
Arithmetic mean value	127.4982	127.4715	127.4843	127.5322	127.4669
Serial Correlation Coeff.	-0.000051	0.000297	0.000294	0.000735	-0.000124

Table 8.1: Summary of the assessment

8.2.2 CyBLE Evaluation Board

As an example of how a real BLE Peripheral could be developed, we used a Cypress CyBLE Development board [22, 21] and tried to obtain more information about the Deterministic Random Number Generator (DRNG).

The datasheet of the BLE component (v3.30) [23] lists two different functions:
`CyBle_GenerateRandomNumber`, which generates an 8 B random number and
`CyBle_SetSeedForRandomGenerator`, which seeds the DRNG with a 32 bit value.

We wrote a short programm that outputs the result of the `CyBle_GenerateRandomNumber` directly after the reset at initialization over a UART interface.

As it is to be expected for a DRNG, the output is the same after every reset. Unfortunately, there is no documentation available about the actual implementation of the DRNG. After we added a software delay between the reset and the generation of a random number, we could observe that after some time the DRNG seems to get reseeded internally.

Even though, the result of this experiment is not very meaningful, it demonstrates the importance of proper seeding. If an attacker is able to reset the device, it would in certain cases be possible to predict the output of the DRNG.

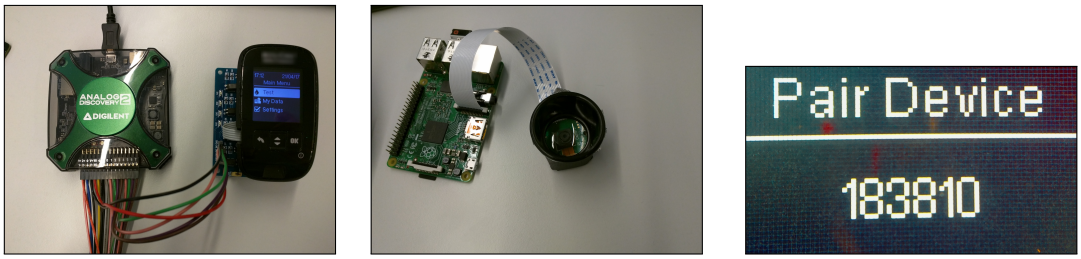
8.2.3 Passkey Generation

The datasheet of the Cypress BLE Component [23] mentions the fix of an error in the Passkey generation.
In an earlier version of the component, accidentally only 16 bit were used to generate the Passkey. The value was therefore never exceeding 65535.

Because the blood glucose meter described in section 7.2 was the only device using Passkey authentication we had at hand, we set up an experiment to automatically generate Passkeys on the device and save them to a file for later inspection.

To automate the pairing request on the device, the electrical connections of the four buttons were connected to four relays which in turn were controlled by a computer (Fig. 8.2a). To obtain sharp images from a close distance, we attached a magnifying lens to a camera module (Fig. 8.2b) and attached this assembly to the device to caputure the content of the display. Figure 8.2c shows a raw, unprocessed image.

A Python script running on the Raspberry Pi was automatically controlling the blood glucose meter and requesting it to repeatedly enter and exit the BLE pairing mode. Each time after the device entered the pairing mode, the camera took a picture of the LC-Display. Simple optical character recognition was used to extract numeric value from the image and save it to a file.



(a) user input simulation (b) image acquisition assembly (c) raw camera image

Figure 8.2: Setup for automated passkey acquisition

After leaving the set-up running unattended over a weekend, we found that the battery of the blood glucose meter did run out after having successfully collected 5520 Passkey values, which is not enough to perform a proper analysis or to make a statement about the quality of the device's internal PRNG.
Furthermore we later discovered two flaws in our program: we forgot to save a timestamp together with each number, which would have allowed us to recreate the sequence in which the numbers were generated. And due to a programming error we only counted each number once, essentially ignoring duplicate values.

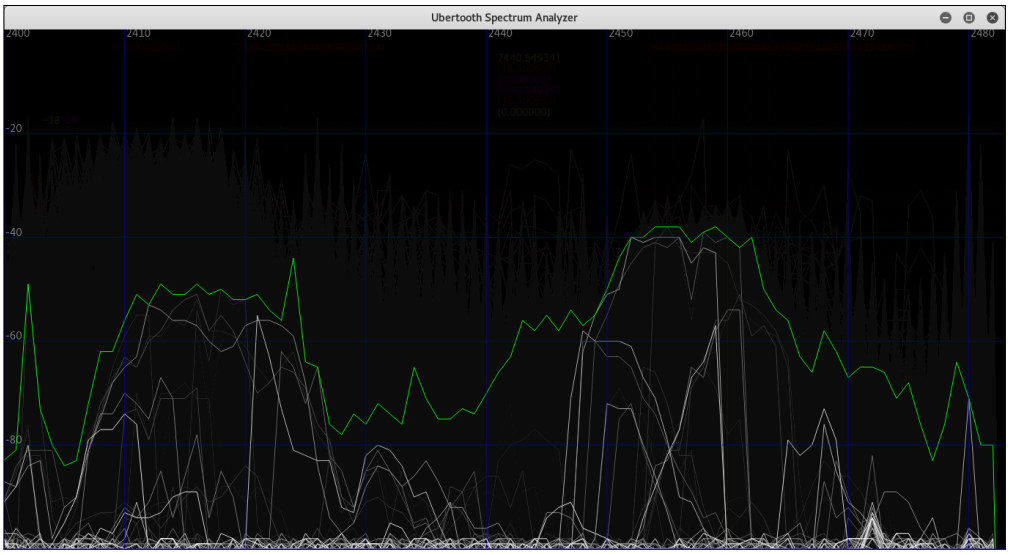
Despite the flaws in our experiment, we were however not able to spot any obvious bias regarding a certain number range or digit.

8.3 Jammer

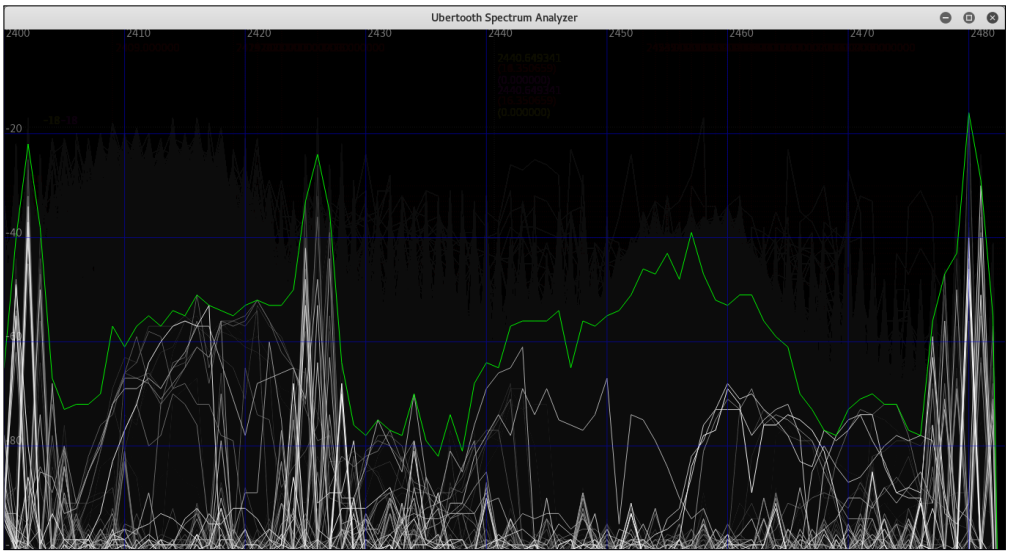
We were able to build a simple jamming device (jammer) from easily obtainable, off-the-shelf components. The device consisted of a Raspberry Pi 2 single board computer and three Bluetooth 4.0 USB dongles.

We wrote a Python script that sends an HCI command to each of the three Bluetooth dongles, requesting them to enter the transmitter test mode. In this mode, each dongle transmits a special test sequence with maximum power on one of the 40 BLE channels. By setting each of the dongles to one of the three advertising channels, we were able to successfully block any communication on the advertising channels in our proximity.
Possible use cases for such a device are not always of malicious nature as described in 2.2.5. Blocking the advertising channels can be used as an external security measure to prevent an adversary from connecting to and attacking a device or to provide privacy by interfering with treasonous advertising packets.
Figure 8.3 shows our jammer and an ubertooth which we used to perform a spectrum analysis. The result of the spectrum analysis without and with activated jammer can be seen in Figure 8.4a and Figure 8.4b.

Figure 8.3: Laptop with ubertooth (left) and jammer (right)



(a) without jammer



(b) with active jammer

Figure 8.4: Frequency spectra

9 Mitigation Methods

9.1 Manufacturers

Whether or not a BLE enabled device is able to transmit data securely, is primarily within the responsibility of the device manufacturer. They are the ones that design a device and decide on hardware and software specifications. However, these design choices are not always easy because they have to take several factors into account.

9.1.1 Pairing

Pairing is a vital security concept of BLE. Even though two devices can communicate as soon as a connection is established, data confidentiality and authenticity can not be provided without pairing. As we have seen in chapter 7, many devices do not use any kind of pairing but instead transmit sensitive data in plain text.

The different BLE pairing association models provide different levels of security and have different requirements regarding the device's user interface. Even if the manufacturing cost is playing an important role and a manufacturer is not willing to implement a user interface just for the sake of security, it is always possible to use the Just Works pairing association model While this might add some complexity to the software compared to no pairing, it does not have any hardware requirements.

9.1.2 Just Works Pairing Association Model

The Just Works pairing association model can be implemented with only little effort on any device and can provide a secure key exchange if no attacker is present during the pairing process. Nevertheless it is susceptible to MITM attacks due to the lack of authentication and should therefore be avoided.

9.1.3 Out-Of-Band Pairing Association Model

The OOB pairing association model provides a very secure way to authenticate two devices and exchange keys securely. However, it requires both devices to support data exchange over a second channel.

While NFC seems to be the most common choice for OOB data exchange, the BLE specification does not further specify on which way the data exchange should be performed. Depending on the use case, many other options are possible, greatly differing in respect to hardware requirements and security.
OOB data could be transmitted **optically**. Most Smartphones and computers have cameras and IR sensors for proximity detection. A Peripheral could transmit data by showing patterns on a display or blinking a single LED.
OOB data could be transmitted **acoustically**. Peripherals could use internal loudspeakers or even simple beepers to generate sound in which the OOB data is modulated. A Central with a microphone, for example a smartphone, could record the sound and extract the data.
OOB data could be transmitted **haptically**. Smartphones and Peripherals like smartwatches or fitness tracker often contain accelerometers and vibrating motors. By creating a momentary mechanical connection netween the devices, like holding them together, they could exchange data by vibrating in certain patterns. These small movements could then be registered by an accelerometer.

These are just examples of how a OOB data exchange could be achieved with mimium hardware requirements.

9.1.4 Application Layer Security

In any case, it is always possible to provide a secure data exchange by applying security measures on application level. As BLE can be used to transmit abritraty data, a manufacturer can simply ignore all security measures provided by the specification and simply implement own methods for data encryption and authentication on top of the BLE stack.

9.1.5 Privacy

As we have shown in section 8.1, a device that is currently advertising may reveal a person's location, which can be used for malicious purposes. The first step to minimize this risk is to limit the time a device is advertising. A user may, for example, have to push a button to put the device into advertising state for a limited time during which a connection could be

established. If no connection establishment or further user interaction takes place during that time, the device could return to a standby state.

As a second, additional or alternative option, the BLE privacy feature should be used. Even though, implementing this feature requires some effort, it can help protecting a user's privacy. It should be noted that not only the device address, but also the content of the advertising packet can be used to track a user. It is therefore also necessary to pay attention to the content of the advertising packet as well.

9.2 Consumers

Consumers do not have many possibilities to increase the wireless security of their BLE-enabled devices.
While it is generally advisable to only use the latest technologies because they usually benefit from the latest security improvements and bug fixes, this is not always an option for consumers.

Regardless of which Bluetooth versions the devices support: If an attack does not take place during pairing, a connection between two devices can be considered safe. This is valid for all pairing association models for both LE Secure Connections and LE Legacy. As soon as both devices have exchanged or established a common LTK, the communication is encrypted with 128 bit AES-CCM, which is considered secure. Users should therefore only pair devices in a secure environment and be very suspicious and cautionous when a previously paired device unexpectedly requests a new pairing procedure.

Devices that do not support pairing and hence do not offer any possibility to encrypt or authenticate data, always pose a risk. Users should only use those devices in secure environments where an attacker is not likely to be present.

Depending on the device, it is in general advisable to disable a device's Bluetooth connectivity or even the whole device when it is not in use. This not only saves energy, but also prevents attackers to connect to a device and perform mailicious activities.

To minimize the risk of an advertising device to reveal a person's location, a user has two possibilities: The device or its Bluetooth connectivity can either be switched off, or the connection between two devices can be kept alive and prevent the slave device entering the advertising state.

10 Summary

10.1 Possible Causes of neglected Security

As we have shown, many BLE-enabled medical devices lack the most basic security measures. They put the user's confidential data at risk and enable potentially dangerous attacks.

To understand how this situation can be resolved, and BLE-enabled devices can be made more secure, it is vital to have a look at the reasons that drive manufacturers to such careless design choices.

10.1.1 Interoperability and Backwards Compatibility

Manufacturers strive to make their device interoperate with as many other devices as possible. What seems to be a good decision for the user, can have an negative impact on security, as backward compatibility to old and possibly insecure protocols enable an attacker to perform, for eample, a downgrade attack. We have analyzed every mobile phone listed by the website [34] as of 21. June 2017 and have evaluated which smartphones and tablets were released between 2010 and mid 2017 and which version of Bluetooth they support. The result of our analysis can be seen in Table 10.1. It can be seen that in 2016, more than one year after its release, less than 20% of the new devices actually support Bluetooth Version 4.2. The table also shows that even in 2017 stil more than half of the newly released devices do not suport Bluetooth 4.2. This pushes device manufacturers to support backwards compatibility and hence provide the basic precondition for downgrade attacks.

Version	2010	2011	2012	2013	2014	2015	2016	2017*
4.0	-	5	116	301	671	520	311	76
4.1	-	-	-	-	9	151	147	31
4.2	-	-	-	-	-	12	101	70
5	-	-	-	-	-	-	-	5

Table 10.1: Number of smartphones/tablets released oer year and the supported Bluetooth Versions. *The data of 2017 includes only the first half of the year

10.1.2 Economical Reasons

One major reason why security is often neglected in BLE-enabled devices is that it often comes with a major price tag in the development and production process.
Except for the Just Works Pairing scheme, every other method has special minimal hardware requirements like sufficient displaying capabilities, buttons or even a NFC-interface for OOB authentication data exchange. Mass production is a very cost sensitive process, and often even small hardware changes can have a huge negative impact on the profit margin. Furthermode, those pairing schemes also have implications on the software side and unlike Just Works, they require user interaction with the device and often a smartphone. This significantly complicates the software development and testing process, delaying the time-to-market and thus causing further financial damage.

10.1.3 User Experience

Pairing schemes like Passkey or Numeric Comparison have a negative impact on the user experience as they complicate the usage and may be a further source for operating errors. This is especially true for medical devices which are more likely to be used by elderly or less tech-savvy people.
One has to bear in mind that security is a rather abstract concept and most consumers do not care if a device is designed to be secure or not.

10.1.4 Semiconductor Industry

Whether a product can make use of the security features introduced with Bluetooth 4.2, is also a question of which hardware components are chosen. Modern Smartphones often use System-on-Chips (SoC), which often have a Bluetooth controller integrated. However, not all of those SoCs support the latest Bluetooth Version.
A good example is the Qualcomm Snapdragon Family [82], which can be found in many mobile devices like smartphones and tablets. The first SoC supporting Bluetooth 4.2, the Snapdragon 626, was released at the end of 2016, about two years after the release of Version 4.2 of the Bluetooth Specification in December 2014. Even newer models, like the Snapdragon 205, which was released in early 2017, only supports Bluetooth 4.1. The Snapdragon 630, 660 and 835 SoCs which became available in 2017Q2, on the other hand, are already supporting Bluetooth 5.

This shows that the internal processes in semiconductor companies seem to follow complicated rules and security seems not to play a very big role. Instead, they try to focus on features

that appear more "concrete" to the user, such as longer range and faster data transmission offered by Bluetooth 5.

10.1.5 No Resources for Developers

During our research we also had the chance to speak with people that are programming smartphone applications. They have experienced that it is very complicated to find the approriate information to help them to fully utilise the security features of BLE in smartphones. Many resources, especially online, only focus on how to make BLE work as quickly and uncomplicated as possible, without proposing any security features or even mentioning the security risks [1].
And indeed, after some research in the Google Android Developer Documentation we were unable to find any documentation even closely related to the BLE security features. This should not be the case. Instead, basic information on the security concepts of BLE and the necessary resources that teach developers how to enable secure communications over BLE should be available very easily.

10.2 Conclusion

As we have seen, despite its complexity, BLE offers many advantages and it is very likely to be seen in more and more applications that can benefit from a energy efficient wireless interface. We have shown that BLE is not inherently insecure. While the BLE specification does provide several security mechanisms, it is in the end up to the manufacturer to properly implement them in a device.

BLE is already a part of many different medical devices ranging from low-end devices like heart rate monitors and thermometers to very complex devices like infusion pumps and spinal cord stimulators. We have shown that morst of the devices we analyzed were susceptible to various attacks because manufacturers did completely neglect device security and hence put the consumer's health and privacy at risk.

We hope that this work serves as an inspiration for future research and improvements in the area of BLE and medical device security and that we raised awereness on the sides of both consumers and manufacturers.

Bibliography

[1] Yasemin Acar et al. “You Get Where You’re Looking For: The Impact of Information Sources on Code Security”. In: 2016 IEEE Symposium on Security and Privacy (SP). IEEE, May 2016, pp. 289–305.

[2] Mohamed Mostafa Abd Allah. “Strengths and Weaknesses of Near Field Communication (NFC) Technology”. In: *Global Journal of Computer Science and Technology* 11.3 (Mar. 2011).

[3] Ross Anderson. *Security Engineering*. Wiley Publishing, Inc., 2008.

[4] Johannes Barnickel, Jian Wang and Ulrike Meyer. “Implementing an Attack on Bluetooth 2.1+ Secure Simple Pairing in Passkey Entry Mode”. In: *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE, June 2012, pp. 17–24.

[5] Kevin Bauer, Harold Gonzales and Damon McCoy. “Mitigating Evil Twin Attacks in 802.11”. In: *2008 IEEE International Performance, Computing and Communications Conference*. IEEE, 2008, pp. 513–516.

[6] Mihir Bellare, Joe Kilian and Phillip Rogaway. “The Security of the Cipher Block Chaining Message Authentication Code”. In: *Journal of Computer and System Sciences* 61.3 (12th Sept. 2001), pp. 362–399.

[7] Bluetooth SIG. *Analysts Forecast Bluetooth Smart To Lead Market Share In Wireless Medical And Fitness Devices*. 2nd Apr. 2013. URL: https://www.bluetooth.com/news/pressreleases/2013/04/02/analysts-forecast-bluetoothsmart-to-lead-market-share-in-wireless-medical-and-fitness-devices- (visited on 04/10/2017).

[8] Bluetooth SIG. *Bluetooth Brand Guide*. Dec. 2016.

[9] Bluetooth SIG. *Bluetooth Core Specification 5.0 FAQ*. 2016.

[10] Bluetooth SIG. *Bluetooth Core Specification v4.0*. 30th June 2010.

[11] Bluetooth SIG. *Bluetooth Core Specification v4.1*. 3rd Dec. 2013.

[12] Bluetooth SIG. *Bluetooth Core Specification v4.2*. 2nd Dec. 2014.

[13] Bluetooth SIG. *Bluetooth SIG Annual Report 2014*. 2015.

[14] Bluetooth SIG. *Bluetooth Technology Protecting Your Privacy*. URL: https://blog.bluetooth.com/bluetooth-technology-protecting-your-privacy (visited on 04/10/2017).

[15] Bluetooth SIG. *GATT Specifications*. URL: https://www.bluetooth.com/specifications/gatt (visited on 04/10/2017).

[16] Bluetooth SIG. *Medical & Health*. URL: https://www.bluetooth.com/what-is-bluetooth-technology/where-to-find-it/medical-health (visited on 04/10/2017).

[17] Gilles Brassard, David Chaum and Claude Crépeau. “Minimum Disclosure Proofs of Knowledge”. In: *Journal of Computer and System Sciences* 37.2 (1988), pp. 156–189.

[18] Elaina Chai, Ben Deardorff and Cathy Wu. *Hacking Bluetooth*. 2012.

[19] N. A. Chattha. “NFC - Vulnerabilities and Defense”. In: *2014 Conference on Information Assurance and Cyber Security (CIACS)*. IEEE, June 2014, pp. 35–38.

[20] Cheng Hao Chen, Iuon Chang Lin and Chou Chen Yang. “NFC Attacks Analysis and Survey”. In: *2014 Eighth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*. IEEE, 2014, pp. 458–462.

[21] Cypress Semiconductor Corporation. *CY8CKIT-143A PSoC4 BLE 256KB Module*. URL: www.cypress.com/CY8CKIT-143A.

[22] Cypress Semiconductor Corporation. *PSoC 4 Pioneer Kit Guide (Rev. D)*. 26th Nov. 2013.

[23] Cypress Semiconductor Corporation. *PSoC Creator Component Datasheet Bluetooth Low Energy (BLE) 3.30*. 19th Dec. 2016.

[24] Joan Daemen and Vincent Rijmen. *AES Proposal: The Rijndael Block Cipher*. 3rd Sept. 1999.

[25] Joan Daemen and Vincent Rijmen. “The Block Cipher Rijndael”. In: *Smart Card Research and Applications: Third International Conference, CARDIS’98*. LNCS 1820. Springer, 1998, pp. 277–284.

[26] Damien Cauquil. *BtleJuice - Bluetooth Smart (LE) Man-in-the-Middle Framework (V1.1.4)*. 17th Jan. 2017. URL: https://github.com/DigitalSecurity/btlejuice.

[27] Siraj Datoo. “How Tracking Customers In-Store Will Soon Be the Norm”. In: *The Guardian. Technology* (10th Jan. 2014).

[28] Tamara Denning et al. “Patients, Pacemakers, and Implantable Defibrillators: Human Values and Security for Wireless Implantable Medical Devices”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2010, pp. 917–926.

[29] Whitfield Diffie and Martin Hellman. “New Directions in Cryptography”. In: *IEEE transactions on Information Theory* 22.6 (1976), pp. 644–654.

[30] John Paul Dunning. “Taming the Blue Beast: A Survey of Bluetooth Based Threats”. In: *IEEE Security & Privacy* 8.2 (2010), pp. 20–27.

[31] Morris Dworkin. *Recommendation for Block Cipher Modes of Operation: Methods and Techniques*. NIST Special Publication 800-38a. National Institute of Standards and Technology, 12th Jan. 2001.

[32] Morris Dworkin. *Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality*. NIST Special Publication 800-38c. National Institute of Standards and Technology, 20th July 2007.

[33] Morris Dworkin. *Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication*. NIST Special Publication 800-38b. National Institute of Standards and Technology, 10th June 2016.

[34] *GSMArena*. URL: https://www.gsmarena.com/ (visited on 04/10/2017).

[35] Nils Gura et al. "Comparing Elliptic Curve Cryptography and RSA on 8-Bit CPUs". In: *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop*. LNCS 3156. Springer, 2004, pp. 119–132.

[36] Keijo MJ Haataja and Konstantin Hypponen. "Man-in-the-Middle Attacks on Bluetooth: A Comparative Analysis, a Novel Attack, and Countermeasures". In: *3rd International Symposium on Communications, Control and Signal Processing (ISCCSP 2008)*. IEEE, 2008, pp. 1096–1102.

[37] Keijo Haataja and Pekka Toivanen. "Practical Man-in-the-Middle Attacks against Bluetooth Secure Simple Pairing". In: *4th International Conference on Wireless Communications, Networking and Mobile Computing*. IEEE, 2008, pp. 1–5.

[38] Keijo Haataja and Pekka Toivanen. "Two Practical Man-in-the-Middle Attacks on Bluetooth Secure Simple Pairing and Countermeasures". In: *IEEE Transactions on Wireless Communications* 9.1 (Jan. 2010), pp. 384–392.

[39] Creighton T. Hager and Scott F. Midkiff. "An Analysis of Bluetooth Security Vulnerabilities". In: *Wireless Communications and Networking, 2003*. Vol. 3. IEEE, 2003, pp. 1825–1831.

[40] Daniel Halperin et al. "Pacemakers and Implantable Cardiac Defibrillators: Software Radio Attacks and Zero-Power Defenses". In: *2008 IEEE Symposium on Security and Privacy*. IEEE, May 2008, pp. 129–142.

[41] Steve Hanna et al. "Take Two Software Updates and See Me in the Morning: The Case for Software Security Evaluations of Medical Devices". In: *HealthSec'11: Proceedings of the 2nd USENIX Conference on Health Security and Privacy*. USENIX Association, 2011.

[42] Ernst Haselsteiner and Klemens Breitfuss. "Security in Near Field Communication (NFC)". In: *Workshop on RFID Security*. 2006, pp. 12–14.

[43] Helger Lipmaa, Phillip Rogaway and David Wagner. *Comments to NIST Concerning AES Modes of Operations: CTR-Mode Encryption*. Sept. 2000.

[44] Robin Heydon. *Bluetooth Low Energy: The Developer's Handbook*. Prentice Hall, 2012.

[45] Mauri Honkanen, Antti Lappetelainen and Kalle Kivekas. "Low End Extension for Bluetooth". In: *Proceedings of the 2004 IEEE Radio and Wireless Conference*. IEEE, 2004, pp. 199–202.

[46] ICS-CERT. *Advisory ICSMA-17-009-01A*. 6th Feb. 2017.

[47] ICS-CERT. *Advisory ICSMA-17-241-01*. 29th Aug. 2017.

[48] Markus Jakobsson and Susanne Wetzel. “Security Weaknesses in Bluetooth”. In: *Topics in Cryptology — CT-RSA 2001*. LNCS 2020. Springer, 2001, pp. 176–191.

[49] Sławomir Jasek. *Gattacking Bluetooth Smart Devices*. 3rd Aug. 2016. URL: http://gattack.io/ (visited on 03/09/2017).

[50] Jay Radcliffe and Tod Beardsley. *R7-2016-07: Multiple Vulnerabilities in Animas OneTouch Ping Insulin Pump*. 4th Oct. 2016. URL: https://blog.rapid7.com/2016/10/04/r7-2016-07-multiple-vulnerabilities-in-animas-onetouch-ping-insulin-pump/ (visited on 04/10/2017).

[51] John Walker. *ENT: A Pseudorandom Number Sequence Test Program*. 28th Jan. 2008. URL: http://www.fourmilab.ch/random/.

[52] Jakob Jonsson. “On the Security of CTR + CBC-MAC”. In: *Selected Areas in Cryptography (SAC 2002)*. LNCS 2595. Springer, 2003, pp. 76–93.

[53] Kanika Grover, Alvin Lim and Qing Yang. “Jamming and Anti-Jamming Techniques in Wireless Networks: A Survey”. In: *International Journal of Ad Hoc and Ubiquitous Computing* 17 (2014).

[54] Vladimir Klebanov and Felix Dörre. “How Do We Know Our PRNGs Work Properly?” 33. Chaos Communication Congress. 29th Dec. 2016.

[55] Neal Koblitz. “Elliptic Curve Cryptosystems”. In: *Mathematics of computation* 48.177 (1987), pp. 203–209.

[56] Henning Kortvedt and S. Mjolsnes. “Eavesdropping near Field Communication”. In: *The Norwegian Information Security Conference (NISK)*. Vol. 27. 2009.

[57] Adam Laurie, Marcel Holtmann and Martin Herfurt. *Hacking Bluetooth Enabled Mobile Phones and beyond - Full Disclosure*. 21st Chaos Communication Congress, 2004.

[58] Kristin Lauter. “The Advantages of Elliptic Curve Cryptography for Wireless Security”. In: *IEEE Wireless communications* 11.1 (2004), pp. 62–67.

[59] Albert Levi et al. “Relay Attacks on Bluetooth Authentication and Solutions”. In: *International Symposium on Computer and Information Sciences (ISCIS 2004)*. LNCS 3280. Springer, 2004, pp. 278–288.

[60] Chunxiao Li, Anand Raghunathan and Niraj K. Jha. “Hijacking an Insulin Pump: Security Attacks and Defenses for a Diabetes Therapy System”. In: *2011 IEEE 13th International Conference on E-Health Networking, Applications and Services*. IEEE, 2011, pp. 150–156.

[61] Andrew Y. Lindell. *Attacks on the Pairing Protocol of Bluetooth v2.1*. Black Hat USA, 2008.

[62] Thomas Martin et al. “Denial-of-Service Attacks on Battery-Powered Mobile Computers”. In: *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications*. IEEE, 2004, pp. 309–318.

[63] Michael Ossmann and Dominic Spill. *Great Scott Gadgets - Ubertooth One*. URL: https://greatscottgadgets.com/ubertoothone/ (visited on 04/10/2017).

[64] Victor S. Miller. “Use of Elliptic Curves in Cryptography”. In: *Advances in Cryptology — CRYPTO ’85 Proceedings*. LNCS 218. Springer Berlin Heidelberg, 1985, pp. 417–426.

[65] Nateq Be-Nazir Ibn Minar and Mohammed Tarique. “Bluetooth Security Threats And Solutions: A Survey”. In: *International Journal of Distributed and Parallel systems* 3.1 (31st Jan. 2012), pp. 127–148.

[66] Sandeep Mistry. *Bleno: A Node.Js Module for Implementing BLE (Bluetooth Low Energy) Peripherals*. 24th Apr. 2017. URL: https://github.com/sandeepmistry/bleno.

[67] Sandeep Mistry. *Noble: A Node.Js BLE (Bluetooth Low Energy) Central Module*. 18th Apr. 2017. URL: https://github.com/sandeepmistry/noble.

[68] Muddy Waters Capital LLC. *MW Is Short St. Jude Medical (STJ:US)*. 25th Aug. 2016. URL: http://www.muddywatersresearch.com/research/stj/mw-is-short-stj/.

[69] Thrinatha R. Mutchukota, Saroj Kumar Panigrahy and Sanjay Kumar Jena. “Man-in-the-Middle Attack and Its Countermeasure in Bluetooth Secure Simple Pairing”. In: *Computer Networks and Intelligent Computing: 5th International Conference on Information Processing (ICIP 2011)*. CCIS 157. Springer, 2011, pp. 367–376.

[70] Moni Naor. “Bit Commitment Using Pseudo-Randomness”. In: *Advances in Cryptology — CRYPTO ’89 Proceedings*. LNCS 435. Springer New York, 1990, pp. 128–136.

[71] James Nechvatal et al. “Report on the Development of the Advanced Encryption Standard”. In: *Journal of Research of the National Institute of Standards and Technology* 106.3 (2001), p. 511.

[72] NFC Forum. *Bluetooth Secure Simple Pairing Using NFC*. 9th Jan. 2014.

[73] NIST. *Federal Inf. Process. Stds. (NIST FIPS) - 197 - Announcing the Advanced Encryption Standard (AES)*. Federal Inf. Process. Stds. (NIST FIPS) 197. 26th Nov. 2001.

[74] Nokia. *Nokia Introduces Wibree Technology as Open Industry Initiative*. 2006. URL: https://www.nokia.com/en_int/news/releases/2006/10/03/nokia-introduces-wibree-technology-as-open-industry-initiative (visited on 04/10/2017).

[75] Nordic Semiconductor. *A Short History of Bluetooth*. 14th July 2014. URL: https://www.nordicsemi.com/eng/News/ULP-Wireless-Update/A-short-history-of-Bluetooth (visited on 04/10/2017).

[76] Orthogonal. *The Growing Significance of Bluetooth BTLE in Healthcare*. 20th Aug. 2015. URL: http://orthogonal.io/medical-softtware/the-growing-significance-of-bluetooth-btle-in-healthcare-html/ (visited on 04/10/2017).

[77] Christof Paar and Jan Pelzl. *Understanding Cryptography*. Springer Berlin Heidelberg, 2010.

[78] Nathanael Paul and David C. Klonoff. “Insulin Pump System Security and Privacy”. In: *USENIX Workshop on Health Security and Privacy*. 2010.

[79] Nathanael Paul, Tadayoshi Kohno and David C. Klonoff. “A Review of the Security of Insulin Pump Infusion Systems”. In: *Journal of diabetes science and technology* 5.6 (2011), pp. 1557–1562.

[80] Konstantinos Pelechrinis, Marios Iliofotou and Srikanth V. Krishnamurthy. “Denial of Service Attacks in Wireless Networks: The Case of Jammers”. In: *IEEE Communications Surveys & Tutorials* 13.2 (2011), pp. 245–257.

[81] Polar Electro Oy. *Innovations*. URL: https://www.polar.com/en/about_polar/who_we_are/innovations (visited on 01/10/2017).

[82] Qualcomm. *Snapdragon Mobile Platforms, Processors, Modems and Chipsets*. 3rd Jan. 2017. URL: https://www.qualcomm.com/products/snapdragon (visited on 05/10/2017).

[83] Jerome Radcliffe. *Hacking Medical Devices for Fun and Insulin: Breaking the Human SCADA System*. Black Hat USA, 2011.

[84] David R. Raymond and Scott F. Midkiff. “Denial-of-Service in Wireless Sensor Networks: Attacks and Defenses”. In: *IEEE Pervasive Computing* 7.1 (2008).

[85] Phillip Rogaway and David Wagner. “A Critique of CCM”. In: *IACR Cryptology ePrint Archive* 2003.70 (2003).

[86] Tomas Rosa. “Bypassing Passkey Authentication in Bluetooth Low Energy.” In: *IACR Cryptology ePrint Archive* 2013.309 (2013).

[87] Mike Ryan. “Bluetooth: With Low Energy Comes Low Security”. In: 7th USENIX Workshop on Offensive Technologies (WOOT 2013). 13th Aug. 2013.

[88] Mike Ryan. *Crackle: Crack and Decrypt BLE Encryption*. 13th Mar. 2017. URL: https://github.com/mikeryan/crackle.

[89] Teemu Savolainen. *Bluetooth 4.0 Update to 4.1 and What It Means for IPv6 over Bluetooth Low-Energy*. IETF-89 Proceedings, 5th Mar. 2014.

[90] Seth Schoen. *Location Tracking: A Pervasive Problem in Modern Technology*. 11th Dec. 2013. URL: https://www.eff.org/de/deeplinks/2013/12/location-tracking-pervasive-problem-modern-technology (visited on 04/10/2017).

[91] Adam Shostack. *Threat Modeling: Designing for Security*. Wiley, 2014.

[92] Bluetooth SIG. *Bluetooth Core Specification v5.0*. 6th Dec. 2016.

[93] Nigel P. Smart et al. *ECRYPT-CSA D5.2 Algorithms, Key Size and Protocols Report*. 17th Oct. 2016.

[94] Dominic Spill. *Bluetooth Sniffing with Ubertooth*. Kiwicon, 2012.

[95] Dominic Spill and Andrea Bittau. “BlueSniff: Eve Meets Alice and Bluetooth”. In: *1st USENIX Workshop on Offensive Technologies (WOOT 2007)* (2007).

[96] Paul Syverson. “A Taxonomy of Replay Attacks”. In: *Proceedings The Computer Security Foundations Workshop VII*. IEEE, 1994, pp. 187–191.

[97] Kevin Townsend et al. *Getting Started with Bluetooth Low Energy: Tools and Techniques for Low-Power Networking*. First Edition. O’Reilly, 2014.

[98] U.S. Food and Drug Administration. *Safety Communications - Cybersecurity Vulnerabilities Identified in St. Jude Medical’s Implantable Cardiac Devices and Merlin@home Transmitter: FDA Safety Communication*. 9th Jan. 2017. URL: https://www.fda.gov/MedicalDevices/Safety/AlertsandNotices/ucm535843.htm (visited on 04/10/2017).

[99] Serge Vaudenay. “Security Flaws Induced by CBC Padding-Applications to SSL, IPSEC, WTLS”. In: *Advances in Cryptology — EUROCRYPT 2002*. LNCS 2332. Springer, 2002, pp. 534–546.

[100] WHO. *Diabetes Fact Sheet*. July 2017. URL: http://www.who.int/mediacentre/factsheets/fs312/en/ (visited on 04/10/2017).

[101] Anthony D. Wood and John A. Stankovic. “Denial of Service in Sensor Networks”. In: *Computer* 35.10 (2002), pp. 54–62.

[102] World Health Organization. *Global Report on Diabetes*. 2016.

[103] Fei Xing and Wenye Wang. “Understanding Dynamic Denial of Service Attacks in Mobile Ad Hoc Networks”. In: *2016 IEEE Military Communications Conference (MILCOM 2006)*. IEEE, 2006.

List of Figures

List of Tables

Acronyms

ACL Asynchronous Connection-oriented 28

ATT Attribute Protocol 28, 34–36, 52, 54

BLE Bluetooth Low Energy 1, 3–5, 10–12, 17, 18, 24, 25, 29, 32, 36, 37, 39, 51, 52, 54, 56–60, 63, 64, 66, 73, 83–87, 89–94, 102

CGM Continuous Glucose Monitoring 72–74

CRC Cyclic Redundancy Check 27, 38

CSRK Connection Signature Resolving Key 32, 48–50, 54

DHK Diversifier Hiding Key 50, 51

DHKE Diffie-Hellman Key Exchange 21–23

DIV Diversifier 50, 51

DLP Discrete Logarithm Problem 21–23

DoS Denial-of-Service 10, 11

DRNG Deterministic Random Number Generator 85

DSA Digital Signature Algorithm 21

ECC Elliptic Curve Cryptography 3, 21–23

ECDHKE Elliptic Curve Diffie-Hellman Key Exchange 23, 39, 52, 64

ECDLP Elliptic Curve Discrete Logarithm Problem 21

EDIV Encrypted Diversifier 49–51

ER Encryption Root 50

GAP Generic Access Profile 27–33, 36, 37, 52

GATT Generic Attribute Profile 31, 36, 52, 58, 59, 69, 70, 75, 80

GUI Graphical User Interface 32

HCI Host Controller Interface 24, 28, 37, 84, 87

IR Identity Root 50, 51

IRK Identity Resolving Key 48–51, 55

ISM Industrial, Scientific, Medical 25

L2CAP Logical Link Control and Adaptation Protocol 28, 33, 34, 54

LCD Liquid Crystal Display 72

LFSR Linear Feedback Shift Register 38

LTK Long-Term Key 32, 40, 45, 47–52, 54, 56, 91

MAC Message Authentication Code 9, 14, 17–19, 21, 54

MIC Message Integrity Check 9

MITM Man-in-the-middle 8, 9, 13, 22, 39, 41, 43, 46, 57–64, 66, 69, 70, 72, 75, 77–79, 81, 89

NFC Near Field Communication 40, 90, 93

OOB Out of Band 41, 43, 65, 90, 93

PDU Protocol Data Unit 27, 32, 52, 54

PRNG Pseudo Random Number Generator 1, 15–17, 25, 83, 84, 86

QoS Quality of Service 28

RNG Random Number Generator 15, 16

RPA Resolvable Private Address 49, 55, 60, 82

SCO Synchronous Connection-Oriented 28

SDR Software Defined Radio 7, 57, 67, 73

SM Security Manager 29, 31, 33, 39, 49, 52

SMP Security Manager Protocol 27, 28, 33, 39, 52, 103

STK Short Term Key 40, 43, 56, 61

TK Temporary Key 56, 57

TRNG True Random Number Generator 16

UUID Universal Unique Identifier 26, 34–36