Extended version: Tamarin-based Analysis of Bluetooth Uncovers Two Practical Pairing Confusion Attacks

Tristan Claverie, Gildas Avoine, Stéphanie Delaune, José Lopes Esteves

▶ **To cite this version:**

Tristan Claverie, Gildas Avoine, Stéphanie Delaune, José Lopes Esteves. Extended version: Tamarin-based Analysis of Bluetooth Uncovers Two Practical Pairing Confusion Attacks. 2023. hal-04079883

HAL Id: hal-04079883
https://hal.science/hal-04079883

Preprint submitted on 24 Apr 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Tamarin-based Analysis of Bluetooth Uncovers Two Practical Pairing Confusion Attacks*

Tristan Claverie
ANSSI, IRISA, INSA de Rennes
Paris, France
tristan.claverie@ssi.gouv.fr

Gildas Avoine
IRISA, INSA de Rennes
Rennes, France
gildas.avoine@irisa.fr

Stéphanie Delaune
Université de Rennes, CNRS, IRISA
Rennes, France
stephanie.delaune@irisa.fr

José Lopes Esteves
ANSSI
Paris, France
jose.lopes-esteves@ssi.gouv.fr

ABSTRACT

This paper provides a Tamarin-based formal analysis of all key-agreement protocols available in Bluetooth technologies, i.e., Bluetooth Classic, Bluetooth Low Energy, and Bluetooth Mesh. The automated analysis finds several unreported attacks, including two attacks (reported by Bluetooth SIG as CVEs) that exploit the confusion of pairing modes, i.e., when a communicating party uses the secure pairing mode while the other one uses the legacy pairing mode. They have been validated in practice using off-the-shelf implementations for the genuine communicating parties, and a custom BR/EDR machine-in-the-middle framework for the attacker.

CCS CONCEPTS

• **Security and privacy** → **Formal security models**; **Mobile and wireless security**; *Security protocols.*

KEYWORDS

Protocol security, Formal models, Bluetooth, Attacks

1 INTRODUCTION

Bluetooth technologies are more and more deployed in the world as ways to transmit data over-the-air. In 2021, 4.7 billion Bluetooth devices were shipped according to the Bluetooth Special Interest Group (SIG) [23]. There are actually three distinct Bluetooth technologies: Bluetooth Classic (BR/EDR), Bluetooth Low Energy (BLE), and Bluetooth Mesh (BM). While the details differ significantly, all of them allow to secure communications, providing confidentiality, integrity, and authentication.

Many flaws have been discovered over the years in Bluetooth standards, including vulnerabilities in the protocols that are discovered regularly. All those flaws are not equivalent, some of them being related to the use of improper cryptographic primitives [29–31], others are purely protocol-level flaws [2, 3, 14, 35, 36], and a few ones rely on incorrect implementations of cryptographic primitives [7, 17, 33]. The behaviour of Bluetooth stacks was also studied, especially on mobile platforms [4, 38, 39], revealing so vulnerabilities in implementations.

Bluetooth communication security mostly relies on the key agreement step, where two devices exchange a cryptographic key. Many different protocols and sub-protocols can be used to perform this step in Bluetooth, which makes the security analysis highly complex. It is worth noting that analyses of vulnerabilities usually focus on a subset of protocols: whether or not these vulnerabilities also impact other Bluetooth protocols remain so unresponded.

The *pairing confusion* introduced in [35] is an attack that exploits the interaction of two key-agreement protocols in Bluetooth. It consists of a scenario where an entity uses Protocol A while the other communicating entity uses Protocol B, such that they are not aware of this protocol mismatch. Usually, such a mismatched interaction ends with a failure. However, for some protocol pairs, the attacker can exploit messages sent in Protocol A to break the security properties of Protocol B, and conversely.

Formal protocol verification is the process of abstracting a protocol to prove that the considered security properties hold. Tamarin Prover [27] and ProVerif [8] are state-of-the-art tools that automatically perform this formal protocol verification. They have been used for verifying complex protocols such as TLS 1.3 [6, 16] and 5G-AKA [15]. When their analyses complete, they grant either a formal proof that the considered security property hold, or an attack.

Automated formal verification tools can be used to study protocol confusion, in particular pairing confusion, which is done in [36] using ProVerif. This paper does not provide a systematic analysis, though, and only considers perfect cryptographic primitives. Although it is common in the litterature to consider perfect cryptographic primitives, this is far from reflecting the ground truth.

Contributions. In this paper, comprehensive Tamarin models of all Bluetooth key-agreement protocols are detailed. Those models are enhanced with representations of cryptographic imperfections that affect Bluetooth. In particular, they are used to systematically analyse pairing confusions in Bluetooth key agreements. Tamarin so automatically identifies previously published attacks and identifies five new attacks, including four novel cases of protocol confusion. We highlight that the Bluetooth SIG assigned two CVEs for two of those attacks that defeat currently known mitigations against pairing confusions. To explore the practicality of these attacks, a BLE and a BR/EDR Machine-in-the-Middle (MitM) are implemented on the respective pairing methods of those technologies. Two additional attacks defeat proposed patches of BM Provisioning. To the best of our knowledge, this is the first practical MitM implementation on the BR/EDR pairing.

*This work received funding from the France 2030 program managed by the French National Research Agency under grant agreement No. ANR-22-PECY-0006.

Outline. Section 2 provides an introduction to Bluetooth key-agreement protocols and their known flaws. Section 3 introduces formal verification with Tamarin and Section 4 details the modelling choices made for this study. The results are detailed and compared to previous works in Section 5. New attacks are presented in Section 6 and their implementation in Section 7. Section 8 concludes this paper.

2 BACKGROUND

In this section, we introduce the three distinct Bluetooth technologies: Bluetooth Basic Rate / Enhanced Data Rate (BR/EDR), Bluetooth Low Energy (BLE), and Bluetooth Mesh (BM).

2.1 BR/EDR and BLE

Bluetooth Basic Rate / Enhanced Data Rate (BR/EDR) and Bluetooth Low Energy (BLE) were standardised in 1999 and 2010 respectively [10]. BR/EDR is routinely used in audio devices (*e.g.*, earbuds, speakers) while BLE is commonly used in other smart devices (*e.g.*, watches). They have a similar security architecture.

2.1.1 Security properties. In BR/EDR and BLE, the specification defines confidentiality, integrity and authenticity of the communication. Confidentiality and integrity are granted with the use of symmetric keys to protect the communication. Those symmetric keys are generated through a key agreement step. Authenticity is an optional property that depends on whether the key agreement used is authenticated or not.

2.1.2 Key Agreement. In BR/EDR and BLE, the key agreement step is called *Pairing* and is performed between devices respectively called *Initiator* and *Responder*. To uniquely identify each protocol, two concepts are introduced. The term *Pairing mode* refers to the type of Pairing, it can be Legacy or Secure. The term *Pairing method* refers to the protocol name as standardized in the specification.

Table 1: BR/EDR and BLE Pairing protocols

	BR/EDR		BLE	
Pairing Mode	Legacy	Secure	Legacy	Secure
Pairing Method	PIN Pairing	JustWorks Passkey Entry Numeric Comparison Out of Band	JustWorks Passkey Entry Out of Band	JustWorks Passkey Entry Numeric Comparison Out of Band

Table 1 lists the Pairing protocols standardised. The Secure Pairing mode of both technologies contains 4 distinct methods that may be run. The differences between methods lie in the messages required to complete them and input/output capabilities of devices. This mode contains the method JustWorks which is not authenticated, Passkey Entry and Numeric Comparison which are authenticated, and the method Out of Band which may be authenticated. In terms of messages exchanged, the Secure Pairing protocols are identical in BR/EDR and BLE, e.g., Passkey Entry involves the same user interaction and messages exchanged whether in BR/EDR or in BLE, though the cryptographic primitives used are not the same.

The Legacy Pairing mode, on the other hand, is drastically different depending on the technology. In BR/EDR, there is a single Legacy protocol called PIN Pairing. In BLE, there are three distinct protocols, depending on the input/output capabilities of pairing devices: JustWorks, Passkey Entry, and Out of Band. Because methods names have been reused between BLE Legacy Pairing and BLE Secure Pairing modes, this paper uses the mode and the method to identify a specific protocol (e.g., Legacy JustWorks, Secure Out of Band, etc.).

In total, in the latest version of the specification[1], there are five distinct Pairing protocols in BR/EDR and seven in BLE, not counting the variations in user interaction inside a given protocol. A device may use all of them or only a subset of them depending on its configuration. For the sake of conciseness, only the Legacy PIN Pairing protocol in BR/EDR and Legacy Passkey Entry protocol in BLE are detailed below.

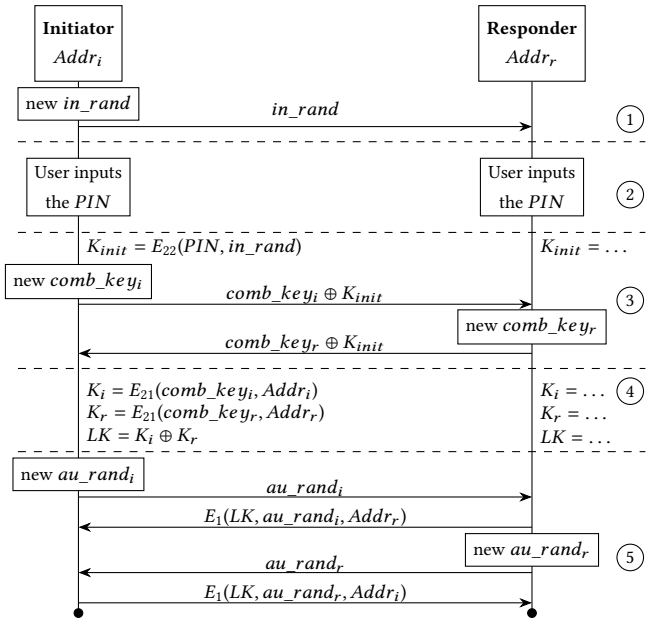

Figure 1: BR/EDR Legacy PIN Pairing and Mutual Legacy Authentication

The protocol Legacy PIN Pairing for BR/EDR is depicted in Fig. 1. Functions E_1, E_{21}, and E_{22} are defined in the specification [10, Vol 2, Part H, §6]. The key agreement starts when the Initiator sends a nonce in_rand to the Responder (1). The user has to exchange a numeric code between devices, called the PIN (2). This PIN is used alongside in_rand and the Initiator address to derive K_{init}. K_{init} is used to mask two nonces $comb\_key_i$ and $comb\_key_r$ (3) which are used to derive the *Link Key* (LK) (4). According to the specification, the Pairing process is over once LK is created, but a mutual authentication procedure has to follow (5).

BLE Legacy Passkey Entry is depicted in Fig. 2. Functions c_1 and s_1 are defined in the specification [10, Vol 3, Part H, §2.2]. The protocol starts with a Feature Exchange step (1), which is used to provide information about input-output capabilities, key size to be negotiated, etc. In Legacy Passkey Entry, the user has to exchange a numeric code between the devices (2). Typically, one device displays a code that the user enters in the other one. This

[1] At the time of writing, Bluetooth Core specification v5.3

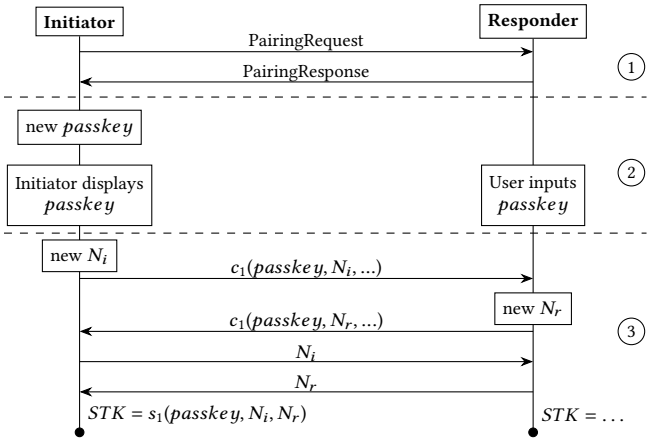

Figure 2: BLE Legacy Passkey Entry

code is used as a symmetric key in a commitment scheme (3). This step is used to authenticate the capabilities and respective addresses of the devices. Finally, they use the nonces exchanged in step (3) to derive a *Short-Term Key* (*STK*) that is then used to encrypt the communication.

2.2 Bluetooth Mesh

Bluetooth Mesh, standardised in 2017 [9], is a networking protocol that creates a Mesh network out of BM devices. BM is dedicated to smart home networks, with applications such as connected lighting, door locks, etc. There are three main communication types in a Mesh network: devices can exchange network-level data, they can exchange application-level data, and they can be each configured by a specific device named "configuration center".

2.2.1 Security properties. The specification defines confidentiality, integrity and authenticity of each communication type. The Network Key (NetKey) is common to all devices in the network, it is used to protect network-level communication. Application Key (AppKey) is common to the set of devices belonging to the same application, it is used to protect application-level communication. There may be several applications in a network, hence several Application Keys. The Device Key (DevKey) is used to protect the communication between a device and the configuration center. The configuration center uses it to perform privileged operations on devices (*e.g.*, to install an AppKey, to rotate NetKey, etc.)

The Network Key and the Device Key are respectively provisioned and generated through a key agreement step. Application Key are sent afterwards, encrypted with the Device Key. Authenticity of the communication depends on the initial key agreement, whether it is authenticated or not.

2.2.2 Key agreement. The key agreement procedure in BM is used to provide each device with the necessary secrets to communicate on this network. It is called *Provisioning* procedure, it runs between a Device and the Provisioner.

There are variants of Provisioning, which depend on how the key exchange is performed (in-band or out-of-band) and how authentication data are exchanged. The possibilities for authentication are No OOB, Input OOB, Output OOB, and Static OOB; where No OOB means no authentication at all. Those two parameters are combined, hence there are eight Provisioning protocols.

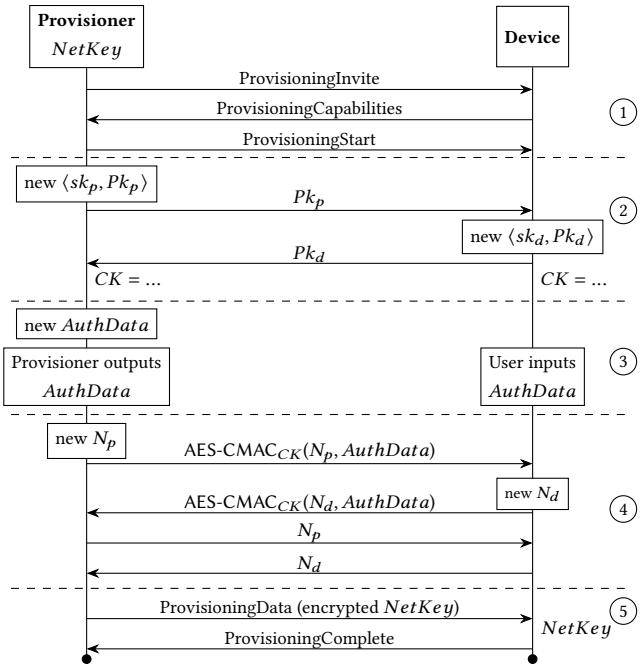

Figure 3: BM Provisioning - In-band key exchange/Input OOB

Fig. 3 depicts a Provisioning variant involving in-band key exchange and Input OOB. First, both devices perform a Feature Exchange step to initiate Provisioning (1). Then, both devices complete an ECDH key exchange and derive a Confirmation Key (CK) (2). The user has to exchange *AuthData* between both devices (3). For example, in Input OOB mode, the Provisioner outputs data (e.g., a numeric code) and the user inputs it in the Device. A commitment protocol is run to authenticate the peers, their respective capabilities, and addresses (4). A new session key is derived from the Confirmation Key and the nonces exchanged in the commitment protocol. Finally, this session key is used to encrypt $NetKey$ (5) and the Provisioning ends.

2.3 Related Work

Bluetooth technologies have been subject to lots of attacks over the years, a survey of those affecting BLE can be found in [18]. Some studies have focused on the security of the reconnection step: BIAS [2] considers the authentication protocol during reconnection in BR/EDR, KNOB [5] the key size reduction in BR/EDR, and BLESA [37] the reconnection in BLE.

There are also passive attacks on Bluetooth technologies. In BR/EDR, Legacy Pairing is vulnerable to offline key recovery from a capture of exchanged messages [31]. Legacy Pairing in BLE has the same flaw although the details differ [30]. In a Secure Pairing protocol, Lindell showed the possibility to retrieve passively an authentication secret [24], which applies to BLE and BR/EDR.

Rosa [29] proposed an active attack on Legacy Pairing in BLE that relies on a flawed cryptographic primitive. Researchers studied

the use of ECDH in the Pairing protocols [7, 17], found flaws in the authentication of public keys and discussed possible attacks. Key size reduction is also studied in BLE [3], which proved to be vulnerable to some extent.

BlueMirror [14] proposed an extensive study of reflection attacks in Bluetooth technologies and showed their applicability to all of them. It also showed cryptographic problems in Bluetooth Mesh Provisioning, breaking its authentication protocol. On Bluetooth Mesh, a reflection attack is independently detailed in [14] and [36]. The security of the Friendship concept in BM is studied in [1].

In [35], the authors define the concept of *Pairing Confusion*, where the attacker forces two devices to use two different Pairing protocols. In their attack, an attacker forces device A to complete Secure Passkey Entry while device B completes Secure Numeric Comparison. They show that in this setup, implementations do not allow the user to distinguish between both protocols. As a result, the attacker can complete them and retrieve the encryption key derived by each device.

Bluetooth was also studied from a formal perspective. Some studies performed manual proofs of some parts of Bluetooth, in various contexts. In [25], a proof of Secure Numeric Comparison is done. A formal analysis of Secure Passkey Entry is proposed in [34]. The security of the reconnection step in BR/EDR and BLE is studied in [20]. Formal studies using automated tools are also detailed in [12], [13], [28], [17], [22], and [36]. They are discussed in depth in Section 5.2.

3 FORMAL VERIFICATION WITH TAMARIN

An introduction to Tamarin is provided in this section.

3.1 Modelling protocols

Tamarin is a tool dedicated to the proof of cryptographic protocols. It represents the messages exchanged and computations as algebraic terms. From a protocol specification and a number of properties expressed in Tamarin's input language, it is able to verify that the protocol matches the stated properties.

At its core, Tamarin is based on multiset rewriting. This means a protocol is represented using a series of multiset rewriting rules. A rule essentially dictates the labelled transition from one set of facts to another.

rule RespSendPublicKey:
[RespDoECDH(*idR*, *idI*), Fr(~*s*), In(*pkI*)]
−[LabRespEndECDH(*idR*, *idI*, 'g'^~*s*, *pkI*, *pkI*^~*s*)]→
[RespEndECDH(*idR*, *idI*, 'g'^~*s*, *pkI*, *pkI*^~*s*), Out('g'^~*s*)]

Example 1: Tamarin rewriting rule

A Tamarin rule is composed of four elements, namely its name, the set of facts that are input to the rule, the set of labels that are produced by the rule, and the set of facts that are output by the rule. In Example 1, if there exists a fact RespDoECDH(*idR*, *idI*) and there is an input message *pkI* in Tamarin's state, it is possible to apply this rewriting rule. This consumes the facts RespDoECDH(...), the fact RespEndECDH(...) is then added to the state. The label LabRespEndECDH(...) is generated by the application of this rule. Out(...) is a special fact and represents the emission of a message over a public channel. In(...) is also a special fact that denotes the reception of a message and Fr(...) represents the generation of a random (fresh) value. The notation~*s* denotes a unique and unguessable random number, the operation ^ represents the exponentiation, and 'g' represents a public constant. In this rule, *idI*, *idR*, and *pkI* are variables, hence can be terms of any form or type.

Tamarin analyses protocols in the so-called Dolev-Yao model [19] where the attacker has full control over the communication channel: it is able to receive, intercept, modify, and forge messages. Tamarin automatically generates rules for the attacker, which enables it to perform common operations, like splitting and concatenating messages, etc. The attacker's knowledge is updated with each message sent on the public channel, hence with each Out(...) produced. Similarly, each message known to the attacker can be sent over the public channel, hence received in any In(...) fact.

In order to represent cryptographic operations, Tamarin enables to define function symbols and their relations through equations. It comes with existing symbols such as xor, symmetric encryption (and decryption), Diffie-Hellman, etc. The set of equations that relate functions together is called an *equational theory*.

3.2 Modelling security properties

The combined use of rewriting rules, functions, and equations is sufficient to create models of cryptographic protocols. To gain insight and knowledge about those protocols, Tamarin allows encoding logical properties that it then tries to verify. These properties are called *lemmas* and are expressed using labels that are produced by rewriting rules.

lemma InitiatorKeySecrecy:
"∀ *id*, *ioCap*, *stk* #i.
InitFinishedPairing(*id*, *ioCap*, *stk*) @#i ⟹ ∄ #j . K(*stk*) @#j"

Example 2: Tamarin lemma

Intuitively, Example 2 expresses that if an Initiator ends the Pairing with a certain key *stk* at time #i, the attacker is unable to retrieve it at any point in time. The lemmas are expressed as logical formulas and the attacker knowledge is represented with fact K. The formulas are expressed naturally as logical formulas, using quantifiers and negations, which allow describing security properties about a protocol. The example formula matches a simple weak secrecy claim about a protocol, namely that the attacker must be unable to retrieve the key *stk*. The lemmas can make use of all the labels defined, but cannot include any fact that is used in the description of the protocol.

3.3 Restricting studied executions

In some protocols, not all executions are valid for a given configuration. For example, some protocols use actions that can be executed only one time (*e.g.*, generation of a master key). In those cases, it is necessary to prevent Tamarin from considering some executions: for one, the model will be closer to the protocol and second, it will improve the computation time to do the proof.

restriction UniqueRestr:
"∀ #i #j. Unique() @#i ∧ Unique() @#j ⟹ #i = #j"

Example 3: Tamarin restriction

This can be modelled with *restrictions*, which also use labels. Restrictions are lemmas that forbid Tamarin to study some executions. In Example 3, if a rule were producing a Unique() label, the restriction would prevent it from being used twice. Restrictions are also logical formulas, so more elaborated formulas could be used.

3.4 Proving protocols with Tamarin

When provided with a lemma, Tamarin tries to prove it is true in all cases or provide an execution trace that contradicts the lemma. This execution trace illustrates the different rules that are applied and the actions the attacker took to contradict the property. From an attack trace, it is possible to manually identify the messages and computations an attacker does to invalidate the property studied. To gain insight into the protocol, the attack can then be analysed to identify its root cause (*e.g.*, lack of integrity of a specific parameter), as well as possible patches.

Another possibility is that Tamarin may not finish the proof within the allocated resources (time, memory). When Tamarin does not finish, it is possible to use an interactive mode and to prove the property manually by guiding Tamarin about the states to explore. Because Tamarin is, at its core, a prover, it does not yield all counterexamples of a lemma for a model. This means that when knowingly studying a flawed protocol, Tamarin is not able to enumerate all the attacks on this protocol.

When modelling complex protocols, it is common that Tamarin takes several days to complete or runs out of resources due to an explosion of states to consider. Restrictions can be used to prevent state explosion, but they have to be carefully created so the model remains correct.

Furthermore, by default Tamarin considers cryptographic primitives to be perfect. However, some primitives have known weaknesses and some protocols use primitives in an incorrect way. Representing cryptographic imperfections requires an extra modelling step so Tamarin can include them in the model.

4 TAMARIN MODELS

This section details the choices made to model Bluetooth key agreements, cryptographic imperfections and patches.

4.1 Bluetooth key agreement protocols

When modelling key agreements in Bluetooth, one needs to tackle the diversity of protocols. In order to model them accurately, one needs to model the user interaction required to complete each of them. In the specification, a single protocol may have several user interaction variations, depending on the input/output capabilities of both devices. For example, in BLE Legacy Passkey Entry, a device may have an input, an output or both. Whether the device outputs or waits for a numeric code depends on the other device's input-output capabilities. To address this variation, Legacy Passkey Entry is modelled as three sub-protocols to represent the different user interaction required. This also applies to other Pairing protocols, increasing so the number of protocols that are represented. In total, there are 13 BLE protocols, 11 BR/EDR protocols, and 8 BM protocol that are modelled to consider all the identified variations.

Pairing Confusion is part of the set of identified vulnerabilities and a systematic study of possible confusions in Bluetooth key agreements is performed. Thus, the interaction of all possible protocols with all possible protocols is studied, with one model per study. This improves the precision and completeness of the analysis, at the cost of a quadratic number of cases to consider. In BLE the interaction of all 13 modelled protocols with all of them is studied, which makes 169 (13×13) cases. Similarly, there are 121 (11×11) cases studied in BR/EDR and 64 (8×8) in BM. Although the term *Pairing confusion* is used across this paper for conciseness, it is noted that in the case of Bluetooth Mesh what is actually studied is *Provisioning confusion*.

It is noted that in practice, the choice of the protocol to use between two legitimate devices is done in the very first step, which is the Feature Exchange. An active attacker has the ability to modify the features sent by each device, hence has the ability to force the protocol to use on each side of the connection. Therefore, studying each pair of protocol makes sense from a Bluetooth's point of view, as this is an accurate representation of an attacker's capabilities.

4.2 Security properties

Confidentiality and integrity of communications come from a shared symmetric key. Those properties are modelled with a key leakage lemma. If the attacker has complete knowledge of the key used by one device after the key agreement, then this property is false. For authentication, the specific property modelled is non-injective agreement. This property is false if an attacker can reach the end of a protocol while impersonating another device.

Some attacks end with an attacker retrieving the symmetric key used by one device and not the other, or simply impersonating a device. In those cases, a user could notice that the key agreement did not complete legitimately because the two legitimate devices cannot communicate together. However, other attacks allow an attacker to compromise the symmetric keys and authentication of both devices at the same time. With those attacks, the user would not notice that its communication are being eavesdropped and could be impersonated, which represents a critical problem for the protocol. To represent these, another property is introduced, called *Compromise Resistance* (CR).

```
lemma WeakSecretInit:
"∀ idI idR ltk #k1 .
  InitFinishedSecPairing(idI, idR, ltk)@#k1 ⟹
    ∄ #k2 . K(ltk)@#k2"

lemma CR:
"∀ idI idR ltk1 ltk2 #j1 #j2 .
        InitFinishedSecPairing(idI, idR, ltk1)@#j1 ∧
        RespFinishedSecPairing(idI, idR, ltk2)@#j2 ⟹
        ∄ #k1 #k2 . K(ltk1)@#k1 ∧ K(ltk2)@#k2"
```

Example 4: Tamarin lemmas in BR/EDR

In Example 4, two security lemmas are displayed. The lemma WeakSecretInit represents the secrecy of the key derived by the Initiator and the lemma CR represents compromise resistance of the protocol. In BLE and BR/EDR, there are five security properties of interest per interaction studied: secrecy of Initiator/Responder key, authenticity of Initiator/Responder and compromise resistance.

In BM, there are nine security properties of interest per interaction studied: secrecy of NetKey/AppKey/DevKey for Device/Provisioner, authenticity of Device/Provisioner and compromise resistance. For each interaction, a functional property is added, it represents whether the interaction can complete successfully in presence of an attacker.

4.3 Modular models

As noted in Paragraph 4.1, studying Pairing confusion requires to study the interaction of all pairs of protocols. This vulnerability also arises from a confusion of the user about its required action, such that the user cannot distinguish two distinct protocols. On the model side, this is represented with a module of rules dedicated to user interactions, used by all sub-protocols. User interactions are modelled through the use of a private channel implemented with Tamarin facts. It is considered that the user acts as defined in the specification and will input/output/confirm data when needed. This represents the confusion from the user's side, who may be unknowingly accepting the key agreement between two distinct protocols if they have matching user interactions.

Bluetooth protocols are not entirely disjoint and share several common parts. For example, each Secure Pairing protocol whether in BLE or BR/EDR starts with the same Feature Exchange and ECDH Key Exchange. In the proposed models, there is a single set of rules that represents those steps, which is used by each appropriate protocol. Another example would be the key derivation steps, which are common to several protocols depending on the technology. In the same spirit, the Tamarin rules corresponding to those steps are common to several protocols and not duplicated.

This approach helps considering the model as a set of modules and not as a simple set of rules. Typically, there is the authentication module, the ECDH module, the key generation module, etc. Each module has a kind of "interface" in the form of one or several facts that are used as input facts or output facts.

To study each pair of protocols independently, it is needed to force Tamarin to consider only the rewriting rules used for a defined protocol and not the others. This is implemented using the Tamarin pre-processor, through Tamarin macros. Macros consist in adding #ifdef and #endif in the model. Before processing the model, the pre-processor of Tamarin writes the block between macros in the studied file only if a command-line flag is provided. Each module of rules is thus surrounded by a macro and is processed only if explicitly stated. Because each case is built as a suite of modules, this approach is used to prevent Tamarin from considering the rest of the model.

4.4 Representing cryptographic imperfections

By default, Tamarin assumes that cryptography is perfect, but Bluetooth is known to be vulnerable to several attacks which rely on cryptographic flaws in the specification. This paragraph details the Tamarin model of cryptographic imperfections, which allow Tamarin to identify attacks based on those vulnerabilities.

4.4.1 Brute-force of low-entropy secrets. Some Bluetooth key agreements use low-entropy secrets, which can be brute-forced by an attacker. Depending on the technology and key agreement, this kind of vulnerability has various shapes, but can be found in each technology [14, 24, 30, 31].

In Tamarin, the names used to represent nonces/passwords are perfect and unguessable by default: if there is a generated value *secret* and the attacker has access to h (*secret*), without further rule the attacker is unable to retrieve the value of *secret*. While this assumption is reasonable for some protocols (e.g., if the secret value is 128-bit long), Bluetooth uses several low-entropy secrets that can be brute-forced in a practical time. To model this capability, special Oracle rules are created to output the targeted secret when the attacker has provided enough information.

```
rule Oracle_f4:
    let val = f4(pk1, pk2, n, s) in
    [ LowEntropyf4(pk1, pk2, n, s), In(pk1), In(pk2), In(n), In(val) ]
  -[AttackerRecoveredPasskey(s) ]->
    [ Out(s) ]
```

Example 5: Oracle rule in Tamarin

The implementation of the passkey recovery [24] from BLE Secure Passkey Entry protocol is done with the rule depicted in Example 5. The function f4 is defined in the specification and is common to several Pairing methods. Only the methods that use a low-entropy secret generate the fact LowEntropyf4 ($pk1, pk2, n, s$) that allows to enter this rule. The attacker also needs to prove knowledge of all the elements to the Oracle by sending them on the public channel. When used, this rule outputs the secret, which becomes available to the attacker. The use of an explicit oracle rule makes it appear in Tamarin's execution traces, therefore one may follow easily the type and number of oracles called in a specific attack.

4.4.2 Malleable Commitment. This issue is present in BLE Legacy Pairing [29] and in BM Provisioning [14]. While both instances of commitment functions in Bluetooth have different cryptographic details, they are conceptually very similar. Both affected commitment procedures use four messages. Those are displayed in step ③ of Figure 2 for BLE and in step ④ of Figure 3 for BM: both devices exchange a commitment value computed from a key, a nonce, an authentication secret, and additional data. Device A sends the first commitment, followed by B. Then both devices exchange their nonces: Device A sends its nonce, and then B replies with its own.

The vulnerabilities rely on the attacker posing as device B. After receiving A's commitment, the attacker needs to send an arbitrary value for A to send its nonce. From A's nonce and commitment, the attacker is able to recover an authentication secret. Then, the attacker crafts a nonce from the sent commitment and recovered authentication secret.

```
functions:
    aes_cmac/2, // Representation of cmac
    get_b1/3, // Used to retrieve first block
equations:
    get_b1(aes_cmac(k, <b1, b2>), k, b2) = b1,
    aes_cmac(k, <get_b1(c, k, b2), b2>) = c,
```

Example 6: Representing malleability in Tamarin

In this paper, the choice is made to implement *malleable commitments* with a pure equational theory. In Example 6, one can see the implementation of this problem that is done for BM. In particular, it is necessary to define an equation to craft a nonce, represented here with get_b1. Then, one has to explicitly state that a confirmation that is used in this way is equal to a proper aes_cmac term. With this type of representation, Tamarin is indeed able to find this class of attacks on the studied protocols.

This type of cryptographic problem is very dependant on the underlying cryptographic specification, and those equations are not suitable for all protocols. In Tamarin, it is impossible to state that this equation holds only if $b1$ and $b2$ have a specific size, in this case the block size of the underlying block cipher: 16 bytes. As a result, those equations give the attacker more power than it has in practice and are not a generic representation of this kind of problem. In the results, it is verified that these equations are applied correctly by the attacker and not in unrealistic cases.

4.4.3 Small subgroup attack on ECDH implementation. In Bluetooth, incorrect ECDH implementations have led to some attacks on implementations [7, 17]. This attack is a type of small subgroup attack that affects BR/EDR and BLE when the validity of received public keys is not verified. The representation of this type of attacks and more generally of incorrect implementations of Diffie-Hellman with Tamarin is extensively discussed in [17]. The authors provide a model of Secure Numeric Comparison with their representation.

In all Bluetooth technologies, the elliptic curves used are P-192 or/and P-256, which are defined over a field of prime order. Therefore, the representation of ECDH provided in their model can be adapted to all Bluetooth technologies. Basically, each public key is represented as a group identifier, the neutral element of the group and the group element. When deriving a Diffie-Hellman key, if the attacker has managed to send an invalid element with respect to the correct group, the key is considered leaked to the attacker. This is representative of elliptic curve cryptography on the groups used in Bluetooth, because an appropriate modification of a public key yields a Diffie-Hellman secret that is on a group of low order (as low as 2). In that case, the secret becomes easily retrievable using brute-force. This representation is adopted in all models of this article.

4.5 Using the models

Using the approach outlined in Section 4.1, studying the interaction of all possible pairs of protocols for all technologies requires studying 354 (169 + 121 + 64) distinct cases, each case containing several properties to analyse. This forms the baseline of the models presented in this paper.

Our first attempt was a version of the model that did rely on the modularization but without using macros. On such a version of the BLE model, proving the simplest lemma required several hours of CPU time but with the current model, 5.74 seconds suffice to study all lemmas for the same interaction. Therefore, macros not only help to specify an interaction to study, but also help to obtain results in practical time by avoiding Tamarin to load all the rules and to compute their refined sources.

Moreover, to gain more insight into the strengths and weaknesses of each protocol, one may want to study the effects of specific imperfections. Similarly, to study the effects of a patch, one may want to study the impact if only one of the two devices is patched. For example, in [17] the authors analyse the outcome of having one device with a patched ECDH implementation and another with a flawed one. The proposed models support this type of configuration through macros. For example, it is possible to study all the mentioned protocols while preventing the attacker to brute-force low-entropy secrets using specific macros. Likewise, it is possible to study all the relevant protocols where one device has a patched version of ECDH using another macro. Overall, using different sets of macros enable to analyse different configurations of the baseline models. The macros that can be used and their effects are detailed in Appendix A.

Table 2: Sizes of the Tamarin models

Model	# rules	# restrictions	# macros	# lemmas	# lines
BR/EDR	117	13	165	605	~11000
BLE	110	12	219	845	~14300
BM	57	8	100	576	~6600

In total, there is one model per technology, containing all subprotocols identified for this technology. Their respective size is detailed in Table 2. Although the models are large, the analysis of all lemmas of all protocols is efficient. The configuration analysed in this paper completed in less than 77 hours of CPU time.

5 RESULTS AND COMPARISON

The results of the study are provided in this section before being compared to the literature.

5.1 Achieved results

This section details the results for BR/EDR and BLE. Results obtained for BM, including a new attack, are detailed in Appendix B. We also slightly modify our model to analyse patches proposed in [36] for BM and uncover several attacks that were overlooked in their study.

The configuration of the models is that devices have a patched ECDH implementation but are vulnerable to all other imperfections. This configuration matches an up-to-date specification. The analysis of BR/EDR and BLE with regards to unpatched ECDH implementations is discussed in Paragraph 5.2.1. The results displayed consider only functional interactions where both devices can reach the end of their protocol.

There are 5 security properties studied for BR/EDR and BLE so a total of 1450 ($121 \times 5 + 169 \times 5$) lemmas. There are 9 security properties studied for BM so a total of 576 (64×9) lemmas studied. For the configuration considered, Tamarin identifies 659 attack traces. All attack traces are manually analysed to identify which result they are related to. Annotated result tables match each attack trace to the underlying weakness used, some of which are new.

5.1.1 BR/EDR Pairing. Properties AuthI and AuthR represent the authentication of the Initiator and Responder respectively. Properties SecI and SecR represent the secrecy of the key derived by the

Initiator and the Responder respectively. Compromise resistance is also studied.

Table 3 presents the results of functional interactions in BR/EDR. Several attacks are related to cryptographic issues. In cells identified with **A1**, a reflection attack is identified where the attacker is able to retrieve the encryption key. This attack relies on a specific property of the xor operator which is built-in in Tamarin, it is presented in [14]. The attacks **A2** and **A5** rely on the brute-force of low-entropy secrets. A variant of the former is described in [31] while the latter is described in [14]. Tamarin finds attack traces that extend these results to compromise the authenticity and secrecy of the protocol. This analysis provides a more accurate view of the impact of this vulnerability with regards to Pairing security.

Several confusion attacks are identified. The original attack [35], identified with **A4**, describes a confusion between Secure Passkey Entry and Secure Numeric Comparison. Tamarin identifies two novel confusion attacks for distinct pairs of protocols. The first one occurs between Legacy PIN Pairing and Secure Numeric Comparison (**A6**). The second one occurs between Legacy PIN Pairing and Secure Passkey Entry (**A7**). Those are discussed in Section 6.

5.1.2 BLE Pairing. The studied properties for BLE are the same as for BR/EDR.

Table 4 presents the results on BLE. Because the Secure Pairing mode contains the same protocols in BR/EDR and BLE, the results of the analysis of Secure Pairing methods interacting with each other (cases Sec*Sec* from Table 3) are identical hence they are not included in this table. As in BR/EDR, JustWorks protocol is vulnerable to MitM attacks by design (**A3**).

Several attacks are related to cryptographic issues. There is a reflection attack against the Initiator in Legacy Pairing (**A8**) which invalidates the AuthI property, it is described in [14]. Attack **A9** relies on the brute-force of low entropy secrets and is presented in [30]. Attack **A5** is similar between Tables 3 and 4. In BLE, it occurs in interactions of type SecPE/LegPE and is an extension of the results presented in [14].

The original confusion attack is correctly identified as for BR/EDR, although the line isn't displayed in Table 4 because it is redundant with lines Sec*Sec* of Table 3. Two novel confusion attacks are identified by Tamarin. A confusion between Legacy Passkey Entry and Secure Numeric Comparison (**A11**) is found. The analysis also demonstrates a possible confusion between Legacy Passkey Entry and Secure Passkey Entry (**A12**). Those are discussed in Section 6.

5.1.3 BM Provisioning. There are 9 security properties studied for BM. We consider 2 authentication properties, and 6 secrecy properties as there are 3 distinct keys (NetKey, AppKey, and DevKey), and secrecy is analysed from the point of view of both entities (Provisioner and Device). Compromise resistance is also studied. Results obtained for BM, including a new attack, are detailed in Appendix B. This new attack allows a desynchronization between the Device and the Provisioner, where the Provisioner successfully completes the protocol while the legitimate Device is prevented from joining the network.

In addition, our model is also used to analyse the patches proposed by [36] to overcome the reflection attack affecting Provisioning protocols. To analyse them in our model, only two specific rules

Table 3: BR/EDR - results of the analysis

Name	AuthI	AuthR	SecI	SecR	CR	CPU Time
LegPINiLegPINi	A1	A2	A1	A2	A2	2204.32s
LegPINiLegPINio	A1	A2	A1	A2	A2	2771.19s
LegPINiLegPINo	A1	A2	A1	A2	A2	2560.88s
LegPINiSecNC	A1	A6	A1	A6	A6	735.61s
LegPINiSecPEi	A1	A7	A1	A7	A7	546.90s
LegPINiSecPEio	A1	A7	A1	A7	A7	646.76s
LegPINiSecPEo	A1	A7	A1	A7	A7	647.84s
LegPINioLegPINi	A1	A2	A1	A2	A2	2211.90s
LegPINioLegPINio	A1	A2	A1	A2	A2	2325.81s
LegPINioLegPINo	A1	A2	A1	A2	A2	2412.69s
LegPINioSecNC	A1	A6	A1	A6	A6	724.13s
LegPINioSecPEi	A1	A7	A1	A7	A7	595.35s
LegPINioSecPEio	A1	A7	A1	A7	A7	632.32s
LegPINioSecPEo	A1	A7	A1	A7	A7	671.47s
LegPINoLegPINi	A1	A2	A1	A2	A2	2464.72s
LegPINoLegPINio	A1	A2	A1	A2	A2	2332.12s
LegPINoSecPEi	A1	A7	A1	A7	A7	658.38s
LegPINoSecPEio	A1	A7	A1	A7	A7	637.28s
SecJWSecJW	A3	A3	A3	A3	A3	145.38s
SecNCLegPINi	A6	A6	A6	A6	A6	490.50s
SecNCLegPINio	A6	A6	A6	A6	A6	497.64s
SecNCSecNC	✓	✓	✓	✓	✓	129.63s
SecNCSecPEi	A4	A4	A4	A4	A4	245.42s
SecNCSecPEio	A4	A4	A4	A4	A4	254.56s
SecOOBiSecOOBo	✓	✓	✓	✓	✓	127.79s
SecOOBioSecOOBio	✓	✓	✓	✓	✓	147.53s
SecOOBoSecOOBi	✓	✓	✓	✓	✓	122.87s
SecPEiLegPINi	A5	A5	A7	A5	A7	3849.83s
SecPEiLegPINio	A5	A5	A7	A5	A7	4139.86s
SecPEiLegPINo	A5	A5	A7	A5	A7	3193.64s
SecPEiSecNC	A5	A4	A4	A4	A4	603.87s
SecPEiSecPEi	A5	A5	✓	A5	✓	8355.98s
SecPEiSecPEio	A5	A5	✓	A5	✓	8859.41s
SecPEiSecPEo	A5	A5	✓	A5	✓	9403.78s
SecPEioLegPINi	A5	A5	A7	A5	A7	3035.24s
SecPEioLegPINio	A5	A5	A7	A5	A7	3131.78s
SecPEioLegPINo	A5	A5	A7	A5	A7	3173.94s
SecPEioSecNC	A5	A4	A4	A4	A4	626.34s
SecPEioSecPEi	A5	A5	✓	A5	✓	8231.80s
SecPEioSecPEio	A5	A5	✓	A5	✓	8011.33s
SecPEioSecPEo	A5	A5	✓	A5	✓	8087.07s
SecPEoLegPINi	A5	A5	A7	A5	A7	3086.12s
SecPEoLegPINio	A5	A5	A7	A5	A7	3433.14s
SecPEoSecPEi	A5	A5	✓	A5	✓	7707.09s
SecPEoSecPEio	A5	A5	✓	A5	✓	8058.43s

A1: Reflection attack on Legacy PIN Pairing, CVE-2020-26555 [14]
A2: Brute-force PIN from exchange [31]
A3: JustWorks is not authenticated
A4: Pairing Method confusion, CVE-2020-10134 [35]
A5: Reflection attack on Secure Passkey Entry, CVE-2020-26558 [14]
A6: (new) Extension to Pairing Method confusion
A7: (new) Pairing Mode Confusion

Table 4: BLE - results of the analysis

Name	AuthI	AuthR	SecI	SecR	CR	CPU Time
LegJWLegJW	A3	A3	A3	A3	A3	5.79s
LegJWSecJW	A3	A3	A3	A3	A3	37.87s
LegOOBLegJW	A8	A3	✓	A3	✓	9.30s
LegOOBLegOOB	A8	✓	✓	✓	✓	12.84s
LegOOBSecJW	A8	A3	✓	A3	✓	14.49s
LegPEiLegPEi	A8	A9	A10	A9	A10	70.70s
LegPEiLegPEio	A8	A9	A10	A9	A10	80.45s
LegPEiLegPEo	A8	A9	A10	A9	A10	81.58s
LegPEiSecNC	A10	A11	A10	A11	A11	132.66s
LegPEiSecPEi	A10	A12	A10	A12	A12	166.26s
LegPEiSecPEio	A10	A12	A10	A12	A12	179.27s
LegPEiSecPEo	A10	A12	A10	A12	A12	175.63s
LegPEioLegPEi	A8	A9	A10	A9	A10	75.88s
LegPEioLegPEio	A8	A9	A10	A9	A10	80.70s
LegPEioLegPEo	A8	A9	A10	A9	A9	92.69s
LegPEioSecNC	A10	A11	A10	A11	A11	137.03s
LegPEioSecPEi	A10	A12	A10	A12	A12	181.66s
LegPEioSecPEio	A10	A12	A10	A12	A12	190.45s
LegPEioSecPEo	A10	A12	A10	A12	A12	196.86s
LegPEoLegPEi	A8	A9	A10	A9	A10	85.66s
LegPEoLegPEio	A8	A9	A10	A9	A10	82.15s
LegPEoSecPEi	A10	A12	A10	A12	A12	188.58s
LegPEoSecPEio	A10	A12	A10	A12	A12	192.86s
SecJWLegJW	A3	A3	A3	A3	A3	43.53s
SecNCLegPEi	A11	A11	A11	A11	A11	123.56s
SecNCLegPEio	A11	A11	A11	A11	A11	122.80s
SecPEiLegPEi	A5	A5	✓	A5	✓	613.19s
SecPEiLegPEio	A5	A5	✓	A5	✓	632.12s
SecPEiLegPEo	A5	A5	✓	A5	✓	653.31s
SecPEioLegPEi	A5	A5	✓	A5	✓	638.30s
SecPEioLegPEio	A5	A5	✓	A5	✓	658.42s
SecPEioLegPEo	A5	A5	✓	A5	✓	662.97s
SecPEoLegPEi	A5	A5	✓	A5	✓	647.77s
SecPEoLegPEio	A5	A5	✓	A5	✓	647.99s

A3: JustWorks is not authenticated
A5: Reflection attack on Secure Passkey Entry, CVE-2020-26558 [14]
A8: Reflection attack in Legacy Pairing [14]
A9: Passkey can be brute-forced in Legacy Passkey Entry [30]
A10: Impersonation in Legacy Passkey Entry [29]
A11: (new) Extension to Pairing Method confusion
A12: (new) Pairing Mode Confusion

need modification. The existing representation of cryptographic imperfections directly applies to their proposed patch, without further effort. The analysis of the patch confirms that the reflection attack is prevented, but other existing attacks remain possible due to two cryptographic imperfections (retrieval of authentication secrets and malleable commitment). The effect of those attacks is that all studied security properties are invalidated: the proposed protocols do not grant key secrecy, authenticity, nor compromise resistance. The flaws in their patch are detailed in Appendix B.2. The ProVerif analysis conducted by [36] missed these attacks as cryptography was assumed to be perfect.

5.2 Comparison with existing models

There are few published formal symbolic analyses of the Bluetooth protocol involving automated tools. For completeness, it is noted that [13] performed a ProVerif [8] analysis of Numeric Comparison but did not identify any weakness. In [12] the authors demonstrated that injective key-agreement does not hold in Numeric Comparison. A study of misbinding attacks is performed in [28] using ProVerif. All those studies focus on various definitions of authentication for one or two Pairing protocols, while the present paper considers all Bluetooth key agreements. The relevance of our model and results are discussed with respect to more accurate models of Bluetooth key agreements in [17], [36], and [22].

5.2.1 Model of the ECDH key exchange. In [17], the authors modify Tamarin to study the security of the Secure Numeric Comparison protocol with regards to small subgroup attacks on the Diffie-Hellman key exchange. This study is an extension of [7] which identified the initial problem with ECDH in Bluetooth Pairing.

In the present study, the analysis of BR/EDR and BLE is also done considering two, one or none of the devices patched. Combined with other problems, this allows identifying more possible attack scenarios where some attacks are combined. It is verified that the patches work with BR/EDR and BLE and for all Secure Pairing methods instead of just one. The results for those configurations are not displayed in this paper.

5.2.2 Analysis of BR/EDR, BLE and BM in ProVerif. In [36], the authors study the key agreements and reconnection step in the three Bluetooth technologies, BR/EDR, BLE, and BM. The first difference is therefore the inclusion of the reconnection step, which they verify in their study and we do not. In their study, they intertwine two different elements. The first is Cross-Transport Key Derivation which is a design choice of Bluetooth to create BR/EDR keys with a BLE Pairing and conversely. The second is the ability in BLE to refuse the establishment of an encrypted connection. In both cases, studying formally this reconnection step requires to make hypotheses on implementations behaviour (e.g., how some error messages are handled by implementations), which they did in [36] and [37]. Because we choose not to perform such hypotheses, the reconnection step is out of scope of the present article.

In terms of protocol analysed, [36] focused on the Secure Pairing protocols for BR/EDR and BLE, omitting all the Legacy protocols. As a result, they did not study the interaction between Legacy protocols and Secure protocols. By contrast, our model contains all standardised protocols, yielding more comprehensive results. Whereas our model is enriched with cryptographic imperfections, the ProVerif model proposed in [36] is not, and as a result, there are several attacks that are missed on Secure Passkey Entry and on Mesh Provisioning. This leads the authors to the erroneous conclusion that Secure Passkey Entry is correctly authenticated. This also means that only the reflection attack is found on Bluetooth Mesh, but more impactful attacks breaking secrecy and authenticity of both the Provisioner and the Device are not identified.

Lastly, in the model proposed in [36], the attacker is unable to act during the Feature Exchange step. It means that both devices would pair using unique addresses and input-output capabilities. This assumption is wrong in the context of an active attacker on

Bluetooth as those two elements can be spoofed. The effect is that a reflection attack on the Secure Passkey Entry protocol is missed [14] (**A5** in Tables 3 and 4), which could be found even under the perfect cryptography assumption.

5.2.3 Analysis of Secure Passkey Entry in Tamarin. In [22], the authors analyse Secure Passkey Entry in Tamarin. Among the attacks they identified, there are Pairing Confusion [35] and the reflection attack on this protocol [14] that we also retrieved in our analysis. The other attacks they identified rely on the hypothesis that the attacker gains the passkey in other ways, due to implementation problems (*e.g.*, bad randomness). In our model we choose *not* to make any assumptions about implementations, hence do not pick up those attacks.

It is noted that the modelling choice for passkey recovery is different from ours. They consider that the passkey is sent in an encrypted form, that can be decrypted later in the protocol. In our model, we use a MAC function[2] as stated in the specification, and an oracle rule is used to model the fact that the secret can be recovered by brute-forcing.

Furthermore, their study tackles only one Pairing protocol, while ours encompasses all Bluetooth key agreements and considers more cryptographic imperfections.

6 NEW ATTACKS

The model developed in this paper identified several new attacks on Bluetooth key agreements. On BM, an attack on the standardised protocol and two attacks on patches from the literature are discussed along BM results in Appendix B. Four new instances of Pairing confusion are identified on BR/EDR and BLE, they are discussed in this section.

6.1 Overview

All identified Pairing confusion break all security properties studied, including compromise resistance. The different confusions identified by the models are:

- Original: Secure Passkey Entry / Secure Numeric Comparison (BR/EDR, BLE) [35]
- Attack A: Legacy PIN Pairing / Secure Passkey Entry (BR/EDR)
- Attack B: Legacy Passkey Entry / Secure Passkey Entry (BLE)
- Attack C: Legacy PIN Pairing / Secure Numeric Comparison (BR/EDR)
- Attack D: Legacy Passkey Entry / Secure Numeric Comparison (BLE)

The original attack is a pairing confusion regarding the method, whereas the new ones are pairing confusion regarding the mode. More importantly, the original attack, as well as attacks C and D can be mitigated by improving the display of expected user actions. In Numeric Comparison, the expected action is for the user to confirm that two numeric codes are equal, while for Passkey Entry the expected action is that the user inputs a numeric code displayed by one device on the other. Some implementations do not have a correct display of expected user actions, which leads to the possible confusion: users input the confirmation code into another device [35].

[2]HMAC-SHA256 in BR/EDR and AES-CMAC in BLE

By contrast, attacks A and B bypass this mitigation because all involved protocol have identical user actions. This section describes attacks A and B as they have the most impact. Each has been attributed one CVE by the Bluetooth SIG.

Both attacks share a similar setup, but rely on different cryptographic weaknesses. The attacker forces one device to use a Legacy protocol which has the same user interaction as Secure Passkey Entry. The attacker uses a cryptographic issue to complete the Legacy protocol, retrieving the encryption key and the passkey/PIN used. Then, the attacker uses the gained knowledge of the passkey to complete the Secure Passkey Entry protocol.

6.2 Attack A: Pairing Mode Confusion in BR/EDR

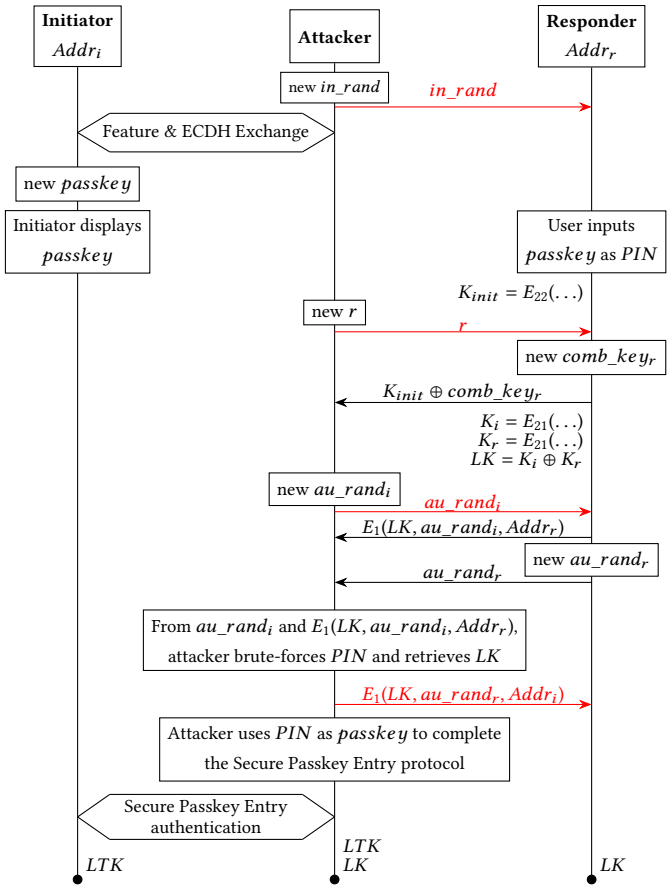

Figure 4: Pairing Mode Confusion in BR/EDR (Attack 1)

The attack is depicted in Figure 4. The attacker forces the Initiator to use the Secure Passkey Entry protocol and the Responder to use the PIN Pairing protocol. To do so, the attacker sends the first message of the PIN Pairing protocol to the Responder which forces it to use this protocol. Then, upon connection of the Initiator, the attacker announces support for Secure Pairing in its features. By modifying its input-output capabilities, the attacker forces a valid

user interaction between PIN Pairing and Secure Passkey Entry, for example the Initiator may display a numeric code (the passkey) and the Responder asks the user to input a numeric code (the PIN).

The PIN can be recovered from the values exchanged in the PIN Pairing protocol and the authentication protocol which serves as key confirmation [31]. Because the PIN is the passkey in the Secure Passkey Entry protocol, the attacker completes the key agreement with the Initiator. In the end, the attacker has successfully completed Pairing with both devices and shares a different encryption key with each of them.

6.3 Attack B: Pairing Mode Confusion in BLE

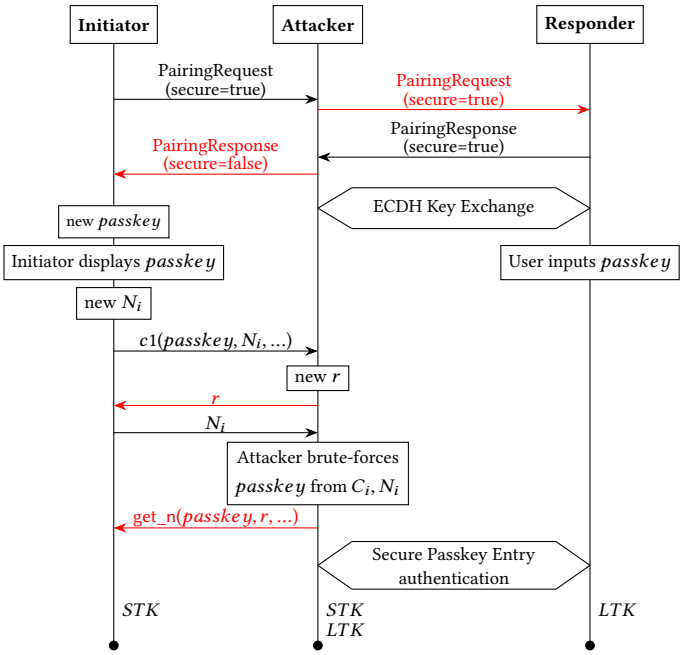

Figure 5: Pairing Mode Confusion in BLE (Attack 2)

The attack is depicted in Figure 5. Function $c1$ is defined in the specification, function get_n computes a correct nonce given a confirmation value: $c1(passkey, \text{get\_n}(passkey, c, data), data) = c$. This results in the malleability of the commitment function in Legacy Passkey Entry protocol, as found by Rosa [29].

The attacker can force the Initiator to use the Legacy Passkey Entry protocol and the Responder to use the Secure Passkey Entry protocol by modifying the input-output capabilities and the *Secure* flag during Feature Exchange. The attacker then completes the protocol on the Legacy side, which makes use of the ability to brute-force the passkey and of the malleability of the commitment in Legacy Pairing. This enables the attacker to recover the passkey, thus to have a legitimate Secure Passkey Entry interaction with the Responder. In the end, the attacker has completed Pairing with both devices while sharing a different encryption key with each of them.

7 PRACTICAL IMPLEMENTATION

To assess their applicability, the attacks have been tested on off-the-shelf devices. This section details those implementations.

7.1 Machine-in-the-Middle attacks

In BR/EDR and BLE, the specification defines a complete protocol stack, from the physical layer to the application layer. It also defines the concept of Controller, which is the entity managing the radio state of the device and Host, the entity that creates logical channels and handles application data. The Host and Controller communicate through an Host-Controller Interface (HCI). On standard Bluetooth-enabled devices such as computers and smartphones, the Controller is implemented by the Bluetooth chipset and the Host is implemented by the operating system.

Pairing happens in the intermediate layers of the protocol stack. Both attacks described in Section 6 require the attacker to implement a custom Pairing procedure. Hence, to perform the attack one needs the ability to receive and craft Pairing messages. In terms of MitM, this means it has to be performed at or below the protocol layer responsible for Pairing.

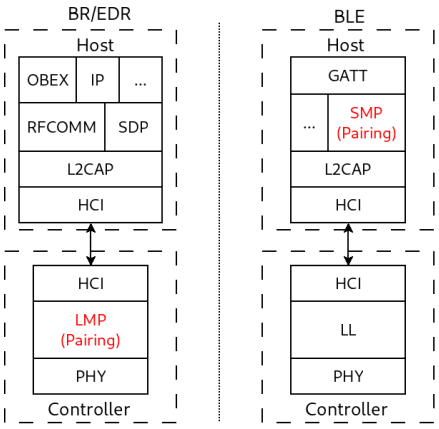

Figure 6: Protocol stacks in BR/EDR and BLE

Figure 6 depicts the protocol stacks of BR/EDR and BLE. The location of Pairing in the protocol stack has major implications on the ability to perform MitM attacks on it. When implemented by the Host in BLE, one may modify the code of the Bluetooth stack to implement those attacks. In BR/EDR, one has to modify the firmware running in a Controller chipset to implement them.

7.2 Pairing Mode Confusion in BR/EDR

In BR/EDR, the Pairing process is implemented by the Controller inside the LMP layer as seen on Figure 6. While messages are completely distinct between Legacy and Secure Pairing, this is the LMP layer which handles all of them.

In order to implement this attack, one needs the ability to act on the LMP layer. To the best of the authors knowledge, only the projects InternalBlue [26] and BrakTooth [21] allow to do so reliably[3]. The official BrakTooth firmware is used without modifications as it already contains the commands to inject LMP messages.

[3]In BrakTooth, LMP message injection is an undocumented feature.

To implement the attacks, the associated driver is modified and enhanced with a custom processing of HCI and LMP messages. The custom driver reimplements the LMP messages necessary to attack the Legacy PIN Pairing protocol. It also implements the HCI messages necessary to control a dongle and complete the Secure Passkey Entry protocol.

To setup the MitM, two BrakTooth dongles are necessary. Two Android devices are used as targets. An adaptation to BR/EDR of the *preconnect* strategy implemented by Mirage in BLE [11] is used. The first dongle creates a connection (i.e., Page) to the target slave device. Another dongle is used to respond to the Inquiry and Page of the target master device. When this setup is complete, custom code can be used to transfer LMP messages from one target to the other.

After the MitM is set up, the Feature Exchange step is controlled such that one dongle accepts Secure Pairing and the other accepts Legacy Pairing. When the legitimate master initiates Pairing, the use of Secure Passkey Entry is enforced while a Legacy PIN Pairing procedure is initiated with the legitimate slave. Doing so, the user is presented with a valid interaction on both sides: the legitimate master displays a code to be input in the slave and the slave waits for a code. The custom PIN Pairing implementation completes the protocol while retrieving the PIN and LK. The PIN is then used by the fake slave to complete the Secure Passkey Entry protocol.

This setup is able to perform MitM on BR/EDR, to complete the Pairing with both sides simultaneously, and to retrieve encryption keys at the end. It was observed that the target slave used keeps responding to Inquiries even after being paged. As a result, to set up the MitM sometimes a few trials were required as the target master did connect with the target slave instead of the fake slave.

To the authors knowledge, this is the first MitM attack implemented in BR/EDR, where two connections are created and synchronised. This is thus the first MitM attack on the Pairing process of BR/EDR to be implemented. This demonstrates that this kind of attacks on BR/EDR is a threat that needs to be protected against.

7.3 Pairing Mode Confusion in BLE

As seen in Figure 6, the Pairing process in BLE is implemented by the Host. More specifically, Pairing is implemented by the Security Manager Protocol (SMP), which is encapsulated inside HCI messages. To implement the attack, the framework Mirage [11] is used. First, Mirage has a built-in support of MitM in BLE through the use of two BLE dongles. Second, Mirage allows to reimplement its own handling of HCI and SMP messages, making it a suitable candidate to perform attacks on the Pairing process.

Two legitimate Android phones are loaded with the *nRF Connect* application. This application allows to scan, connect, and pair to nearby BLE devices. Mirage is used to perform a MitM between those two devices, then custom code allows to complete the attack.

Because Mirage does not support Secure Pairing, the support of Secure Passkey Entry is added to the framework, which includes the definition of relevant SMP messages and cryptographic primitives. Then, the logic of the attack is implemented. First, the legitimate Initiator is forced to perform a Legacy Passkey Entry protocol while the legitimate Responder is forced to perform a Secure Passkey Entry protocol. The user is presented with a valid Passkey Entry interaction and completes it. Then, Rosa's attack [29] is used on the Legacy side to recover the passkey. This is used to complete the Secure Passkey Entry side.

This setup is able to perform the MitM on BLE, complete the Pairing with both devices and retrieve the encryption keys at the end. The implementation of the brute-force is naive and takes a few seconds, yet none of the legitimate device did timeout during Pairing. Overall, this implementation validates the real-world applicability of this attack.

8 CONCLUSION

Bluetooth has a concept of security mode. In BLE and BR/EDR, a mode exists to restrict connections to use only Secure Pairing modes and 128-bit keys. Those modes may be implemented in devices to restrict access to sensitive services. Whether those are implemented and enforced remains an implementation and configuration matter.

The attacks presented in this paper demonstrate that the knowledge of the configuration of one of the two devices is not enough to have complete security guarantees. If one device is configured to use only Secure Pairing but the peer device still allows Legacy Pairing, then the communication between them is not immune to attacks. Also, the user is not able to observe the difference because the proposed mode confusion keeps an identical user interaction as a legitimate exchange.

The original confusion attack relies on similar user interactions that may be confused by users. The statement from the SIG [32] recommends to device manufacturers to make it more obvious which interaction is expected from users, to avoid confusions. They did not modify the underlying protocols, hence no patch is enforced for this problem. The confusions presented in this paper bypass this mitigation because the user interaction is not only similar but identical between both protocols.

The confusion identified in this paper is not done on the Pairing Method, but on the Pairing Mode. When applicable, a possible patch could be to indicate the Pairing Mode used (Legacy or Secure) on the user interface, with specific instructions to not mix them. Still on the user interface, devices supporting Legacy Pairing could indicate that it is an unsecure Pairing mode anyway.

Another possibility would be to restrict the use of Legacy Pairing completely, but this change must be enforced on both devices. The response of Bluetooth SIG to those vulnerabilities is to disable Legacy Pairing on devices, but it will not modify the protocols.

In our opinion, to completely fix those vulnerabilities, changes at the protocol's level are needed. Legacy Pairing could be removed from the specification and the certification of newer devices may include a test to verify it is not implemented. Modifying the protocol to prevent confusion attacks would likely pose compatibility issues, but would adequately solve the problem.

Disclosure process. The attacks were disclosed to the Bluetooth SIG in September, 2021. The SIG accepted Pairing Mode confusion attacks on BLE and BR/EDR and made a public statement on December, 2022. CVE numbers 2022-25836 (resp. 2022-25837) was assigned to the Pairing Mode confusion in BLE (resp. BR/EDR).

REFERENCES

[1] Flor Álvarez, Lars Almon, Ann-Sophie Hahn, and Matthias Hollick. Toxic Friends in Your Network: Breaking the Bluetooth Mesh Friendship Concept. In *Proceedings of the 5th ACM Workshop on Security Standardisation Research Workshop*, SSR'19, page 1–12, New York, NY, USA, 2019. Association for Computing Machinery.

[2] Daniele Antonioli, Nils Ole Tippenhauer, and Kasper Rasmussen. BIAS: bluetooth impersonation attacks. In *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*, pages 549–562. IEEE, 2020.

[3] Daniele Antonioli, Nils Ole Tippenhauer, and Kasper Rasmussen. Key negotiation downgrade attacks on bluetooth and bluetooth low energy. *ACM Trans. Priv. Secur.*, 23(3):14:1–14:28, 2020.

[4] Daniele Antonioli, Nils Ole Tippenhauer, Kasper Rasmussen, and Mathias Payer. Blurtooth: Exploiting cross-transport key derivation in bluetooth classic and bluetooth low energy. In Yuji Suga, Kouichi Sakurai, Xuhua Ding, and Kazue Sako, editors, *ASIA CCS '22: ACM Asia Conference on Computer and Communications Security, Nagasaki, Japan, 30 May 2022 - 3 June 2022*, pages 196–207. ACM, 2022.

[5] Daniele Antonioli, Nils Ole Tippenhauer, and Kasper Bonne Rasmussen. The KNOB is broken: Exploiting low entropy in the encryption key negotiation of bluetooth BR/EDR. In Nadia Heninger and Patrick Traynor, editors, *28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019*, pages 1047–1061. USENIX Association, 2019.

[6] Karthikeyan Bhargavan, Bruno Blanchet, and Nadim Kobeissi. Verified models and reference implementations for the TLS 1.3 standard candidate. In *IEEE Symposium on Security and Privacy (S&P'17)*, pages 483–503, San Jose, CA, May 2017. IEEE. Distinguished paper award.

[7] Eli Biham and Lior Neumann. Breaking the bluetooth pairing - the fixed coordinate invalid curve attack. In Kenneth G. Paterson and Douglas Stebila, editors, *Selected Areas in Cryptography - SAC 2019 - 26th International Conference, Waterloo, ON, Canada, August 12-16, 2019, Revised Selected Papers*, volume 11959 of *Lecture Notes in Computer Science*, pages 250–273. Springer, 2019.

[8] Bruno Blanchet. Automatic verification of security protocols in the symbolic model: The verifier proverif. In Alessandro Aldini, Javier López, and Fabio Martinelli, editors, *Foundations of Security Analysis and Design VII - FOSAD 2012/2013 Tutorial Lectures*, volume 8604 of *Lecture Notes in Computer Science*, pages 54–87. Springer, 2013.

[9] Bluetooth SIG. *Mesh Profile Bluetooth Specification*, 01 2019. v1.0.1.

[10] Bluetooth SIG. *Bluetooth Core Specification*, 07 2021. v5.3.

[11] Romain Cayre, Jonathan Roux, Eric Alata, Vincent Nicomette, and Guillaume Auriol. Mirage : un framework offensif pour l'audit du Bluetooth Low Energy. In *Symposium sur la Sécurité des Technologies de l'Information et de la Communication, SSTIC 2019, Rennes, France*, pages 229–258, June 2019.

[12] Richard Chang and Vitaly Shmatikov. Formal Analysis of Authentication in Bluetooth Device Pairing. In *in "Proceedings of the Joint Workshop on Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis (FCS-ARSPA'07*, pages 45–62, 2007.

[13] Tom Chothia, Ben Smyth, and Chris Staite. Automatically checking commitment protocols in proverif without false attacks. In Riccardo Focardi and Andrew Myers, editors, *Principles of Security and Trust*, pages 137–155, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[14] Tristan Claverie and José Lopes-Esteves. Bluemirror: Reflections on bluetooth pairing and provisioning protocols. In *IEEE Security and Privacy Workshops, SP Workshops 2021, San Francisco, CA, USA, May 27, 2021*, pages 339–351. IEEE, 2021.

[15] Cas Cremers and Martin Dehnel-Wild. Component-Based Formal Analysis of 5G-AKA: Channel Assumptions and Session Confusion. In *NDSS*. The Internet Society, 2019.

[16] Cas Cremers, Marko Horvat, Sam Scott, and Thyla van der Merwe. Automated analysis and verification of tls 1.3: 0-rtt, resumption and delayed authentication. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 470–485, 2016.

[17] Cas Cremers and Dennis Jackson. Prime, order please! revisiting small subgroup and invalid curve attacks on protocols using diffie-hellman. In *32nd IEEE Computer Security Foundations Symposium, CSF 2019, Hoboken, NJ, USA, June 25-28, 2019*, pages 78–93. IEEE, 2019.

[18] Matthias Cäsar, Tobias Pawelke, Jan Steffan, and Gabriel Terhorst. A survey on Bluetooth Low Energy security and privacy. *Computer Networks*, 205:108712, March 2022.

[19] Danny Dolev and Andrew Chi-Chih Yao. On the security of public key protocols. *IEEE Trans. Inf. Theory*, 29(2):198–207, 1983.

[20] Marc Fischlin and Olga Sanina. Cryptographic analysis of the bluetooth secure connection protocol suite. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part II*, volume 13091 of *Lecture Notes in Computer Science*, pages 696–725. Springer, 2021.

[21] Matheus E. Garbelini, Sudipta Chattopadhyayi, Vaibhav Bedi, Sumei Sun, and Ernest Kurniawan. BrakTooth: Causing Havoc on Bluetooth Link Manager. Technical report, Singapore University of Technology and Design, 2021.

[22] Mohit Kumar Jangid, Yue Zhang, and Zhiqiang Lin. Extrapolating Formal Analysis to Uncover Attacks in Bluetooth Passkey Entry Pairing. 2023.

[23] Marcel Jason. New Wireless Trends and Forecasts for the Next 5 Years. https://www.bluetooth.com/blog/new-trends-and-forecasts-for-the-next-5-years/.

[24] Andrew Y Lindell. Attacks on the Pairing Protocol of Bluetooth v2.1. https://www.blackhat.com/presentations/bh-usa-08/Lindell/BH_US_08_Lindell_Bluetooth_2.1_New_Vulnerabilities.pdf, June 2008. BlackHat USA.

[25] Andrew Y. Lindell. Comparison-Based Key Exchange and the Security of the Numeric Comparison Mode in Bluetooth v2.1. In Marc Fischlin, editor, *Topics in Cryptology – CT-RSA 2009*, volume 5473, pages 66–83. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[26] Dennis Mantz, Jiska Classen, Matthias Schulz, and Matthias Hollick. InternalBlue - Bluetooth Binary Patching and Experimentation Framework. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '19, page 79–90, New York, NY, USA, 2019. Association for Computing Machinery.

[27] Simon Meier, Benedikt Schmidt, Cas Cremers, and David A. Basin. The TAMARIN prover for the symbolic analysis of security protocols. In Natasha Sharygina and Helmut Veith, editors, *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, volume 8044 of *Lecture Notes in Computer Science*, pages 696–701. Springer, 2013.

[28] Aleksi Peltonen, Mohit Sethi, and Tuomas Aura. Formal verification of misbinding attacks on secure device pairing and bootstrapping. *Journal of Information Security and Applications*, 51:102461, April 2020.

[29] Tomás Rosa. Bypassing passkey authentication in bluetooth low energy. *IACR Cryptol. ePrint Arch.*, page 309, 2013.

[30] Mike Ryan. Bluetooth: With Low Energy Comes Low Security. In *7th USENIX Workshop on Offensive Technologies (WOOT 13)*, Washington, D.C., August 2013. USENIX Association.

[31] Yaniv Shaked and Avishai Wool. Cracking the bluetooth PIN. In Kang G. Shin, David Kotz, and Brian D. Noble, editors, *Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services, MobiSys 2005, Seattle, Washington, USA, June 6-8, 2005*, pages 39–50. ACM, 2005.

[32] Bluetooth SIG. Bluetooth SIG Statement Regarding the Method-Confusion Pairing Vulnerability. https://www.bluetooth.com/learn-about-bluetooth/key-attributes/bluetooth-security/method-vulnerability/.

[33] Jörn Tillmanns, Jiska Classen, Felix Rohrbach, and Matthias Hollick. Firmware insider: Bluetooth randomness is mostly random. In Yuval Yarom and Sarah Zennou, editors, *14th USENIX Workshop on Offensive Technologies, WOOT 2020, August 11, 2020*. USENIX Association, 2020.

[34] Michael Troncoso and Britta Hale. The Bluetooth CYBORG: Analysis of the Full Human-Machine Passkey Entry AKE Protocol. In *28th Annual Network and Distributed System Security Symposium, NDSS 2021, virtually, February 21-25, 2021*. The Internet Society, 2021.

[35] Maximilian von Tschirschnitz, Ludwig Peuckert, Fabian Franzen, and Jens Grossklags. Method confusion attack on bluetooth pairing. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*, pages 1332–1347. IEEE, 2021.

[36] Jiangliang Wu, Ruoyu Wu, Dongyan Xu, Dave (Jing) Tian, and Antonio Bianchi. Formal model-driven discovery of bluetooth protocol design vulnerabilities. In *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, 23-26 May 2022*, pages 879–897, Los Alamitos, CA, USA, may 2022. IEEE Computer Society.

[37] Jianliang Wu, Yuhong Nan, Vireshwar Kumar, Dave (Jing) Tian, Antonio Bianchi, Mathias Payer, and Dongyan Xu. BLESA: spoofing attacks against reconnections in bluetooth low energy. In Yuval Yarom and Sarah Zennou, editors, *14th USENIX Workshop on Offensive Technologies, WOOT 2020, August 11, 2020*. USENIX Association, 2020.

[38] Fenghao Xu, Wenrui Diao, Zhou Li, Jiongyi Chen, and Kehuan Zhang. Badbluetooth: Breaking android security mechanisms via malicious bluetooth peripherals. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society, 2019.

[39] Yue Zhang, Jian Weng, Rajib Dey, Yier Jin, Zhiqiang Lin, and Xinwen Fu. On the (in)security of bluetooth low energy one-way secure connections only mode. *CoRR*, abs/1908.10497, 2019.

A TAMARIN CONFIGURATIONS

Tamarin macros are heavily used in all models. Overall, using different sets of macros enable to analyse different configurations of the baseline models. We describe here the macros that can be used to configure each model and their effects.

A.1 Tamarin macros in a nutshell

This paragraph uses two examples to illustrate how Tamarin macros work.

```
#ifdef Macro1
rule Rule1: [ ... ] –[ ... ]→ [ ... ]
#endif

rule Rule2: [ ... ] –[ ... ]→ [ ... ]
```

Example 7: Basic use of Tamarin macro

In Example 7, if Tamarin is started with the flag -DMacro1 the rule Rule1 is used, else it is discarded. In all cases the rule Rule2 is part of the analysis.

```
#ifdef NoOracle
#else
rule OracleRule: [ ... ] –[ ... ]→ [ ... ]
#endif
```

Example 8: Implementing ifndef with Tamarin macro

Example 8 shows a simple implementation of an equivalent of #ifndef in Tamarin. In the example, the rule OracleRule is used in every case, except if the macro NoOracle is used on the command line.

A.2 BR/EDR model

In BR/EDR, the following configuration macros are defined:

- NoLowEntropyLegacy: Disables the ability for an attacker to brute-force the PIN used in Legacy PIN Pairing [31];
- NoLowEntropySecure: Disables the ability for an attacker to brute-force the passkey used in Secure Passkey Entry [24];
- InitECDHUnpatched: Represents the fact that the Initiator does not verify the validity of the Responder's public key [17];
- RespECDHUnpatched: Represents the fact that the Responder does not verify the validity of the Initiator's public key [17];

A.3 BLE model

In BLE, the following configuration macros are defined:

- NoLowEntropyLegacy: Disables the ability for an attacker to brute-force the passkey used in Legacy Passkey Entry [30];
- NoLowEntropySecure: Disables the ability for an attacker to brute-force the passkey used in Secure Passkey Entry [24];
- NoMalleableC1: Disables the malleability of the $c1$ commitment function in BLE Legacy Pairing [29];
- InitECDHUnpatched: Represents the fact that the Initiator does not verify the validity of the Responder's public key [17];
- RespECDHUnpatched: Represents the fact that the Responder does not verify the validity of the Initiator's public key [17].

A.4 BM model

In BM, the following configuration macros are defined:

- NoLowEntropyAuthValue: Disables the ability for an attacker to brute-force AuthData used in the Provisioning protocol [14].
- NoMalleableCMAC: Disables the malleability of the *CMAC* commitment function in Provisioning [14]
- ProvECDHUnpatched: Represents the fact that the Provisioner does not verify the validity of the Device's public key [17];
- DevECDHUnpatched: Represents the fact that the Device does not verify the validity of the Provisioner's public key [17];
- PatchProvisioning1: Represents the first patch proposed in [36]: the Provisioner should not accept a reflected confirmation value;
- PatchProvisioning2: Represents the second patch proposed in [36]: the Device computes the commitment value using an inversion of parameters.

B MESH PROVISIONING

The complete Provisioning protocol is shown in Figure 3.

B.1 Results on Provisioning

Mesh Provisioning was analysed with regards to patched ECDH implementations. All other cryptographic imperfections are included in the model. Properties AuthP and AuthD refer respectively to the authentication of the provisioner and of the device. Properties SecPN, SecPA, and SecPD refer to the secrecy of NetKey, AppKey, and DevKey of the provisioner. Properties SecDN, SecDA, and SecDD refer to the secrecy of NetKey, AppKey, and DevKey of the device. Compromise resistance is also studied for Mesh Provisioning.

Table 5 presents the results obtained on BM. The aforementioned 9 properties defined are studied for each pair of the 64 protocols, that is 576 (64×9) lemmas are analysed using Tamarin. For brevity, only the results that lead to functional interactions are displayed. For example, when NoOOB is used by a device an attacker can always conduct an active attack as it is not authenticated (**A13**).

Tamarin correctly identifies a reflection attack (**A14**) on Mesh Provisioning, which is published in [14] and [36]. It also finds attacks which rely on brute-forcing a low-entropy AuthData (**A15**). Finally, Tamarin combines existing results into complete attacks. For example, **A17** uses reflection to complete the key agreement with the Provisioner and brute-forces AuthData [14] to complete with key agreement with the Device. Other combinations of results are identified (**A18** and **A19**), relying on the complete retrieval of AuthData or the malleability of the commitment function. Those results were published in [14]. It is noted that although **A17** and **A18** are combination of existing vulnerabilities of the protocol, the ability to combine them is currently unaccounted for.

Another problem highlighted by this work is the lack of key confirmation in Provisioning. In cells containing **A16**, the attacker can prevent the transmission of NetKey from the Provisioner to the Device and still complete the protocol with the Provisioner. Therefore, even if the provisioner completes the protocol, the device

Table 5: Mesh - results of the analysis

Name	AuthP	AuthD	SecPN	SecPA	SecPD	SecDN	SecDA	SecDD	CR	CPU Time
EiOOBiEiOOBi	A16	A15	A14	A14	A14	A15	A15	A15	A17	12191.64s
EiOOBnoEiOOBno	A14	A13	A14	A13	A14	A13	A13	A13	A13	554.07s
EiOOBoEiOOBo	A16	A15	A14	A14	A14	A15	A15	A15	A17	12140.49s
EiOOBsEiOOBno	A14	A13	A14	A19	A14	A13	A13	A13	A14	568.60s
EiOOBsEiOOBs	A14	A19	A14	A19	A14	A19	A19	A19	A18	4771.21s
EoOOBiEoOOBi	A16	✓	✓	✓	✓	✓	✓	✓	✓	6244.17s
EoOOBnoEiOOBno	A14	A13	✓	✓	✓	A13	A13	A13	✓	570.58s
EoOOBnoEoOOBno	A14	✓	✓	✓	✓	✓	✓	✓	✓	590.67s
EoOOBoEoOOBo	A16	✓	✓	✓	✓	✓	✓	✓	✓	5982.47s
EoOOBsEiOOBno	A14	A13	✓	✓	✓	A13	A13	A13	✓	671.94s
EoOOBsEoOOBs	A14	✓	✓	✓	✓	✓	✓	✓	✓	2337.05s

A13: OOBno is not authenticated
A14: Reflection attack on Provisioning, CVE-2020-26560 [14, 36]
A15: AuthData may be brute-forced, CVE-2020-26557 [14]
A16: (new) Lack of key confirmation in Provisioning
A17: (combination) Reflection and AuthData brute-force
A18: (combination) Reflection and AuthData retrieval
A19: (combination) AuthData retrieval and malleable commitment

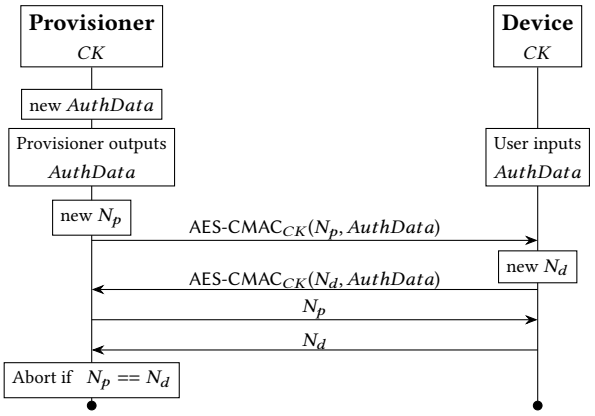

Figure 7: Patch 1 proposed against the Provisioning reflection attack

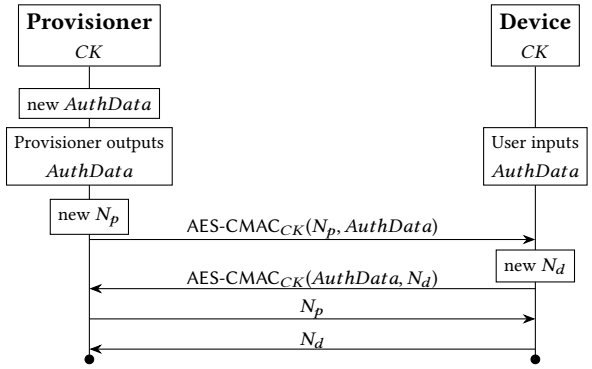

Figure 8: Patch 2 proposed against the Provisioning reflection attack

may not have joined the Mesh network. However, the attacker does not gain any information nor any secret doing so. This demonstrates a new flaw in the protocol, leading to a possible impersonation of a device towards the provisioner.

B.2 Analysis of Provisioning patches

Bluetooth Mesh Provisioning was found vulnerable to a reflection attack in [14, 36]. In [36], the authors propose two patches for this attack by modifying the commitment protocol of the Provisioning. The first patch depicted in Figure 7 consists in a verification that the confirmation and nonce values were not reflected. The second patch, in Figure 8 modifies the computation of the confirmation value sent by the device: the order of arguments in the function are reversed.

The patches are included in the Tamarin model of Bluetooth Mesh. Tamarin identifies possible attacks in both cases due to commitment malleability, which also applies in this case. Those attacks contradict all security properties studied, including compromise resistance. The identified attack on Patch 1 is shown in Figure 9. This exact attack affects the original Provisioning protocol and is already detailed in [14].

The initial analysis of the patches in [36] is performed assuming perfect primitives, hence those attacks were not picked up by the initial analysis. The core problem is that the commitment function used is AES-CMAC, where up to one plaintext block can be retrieved. In the Provisioning protocol, the confirmation is computed

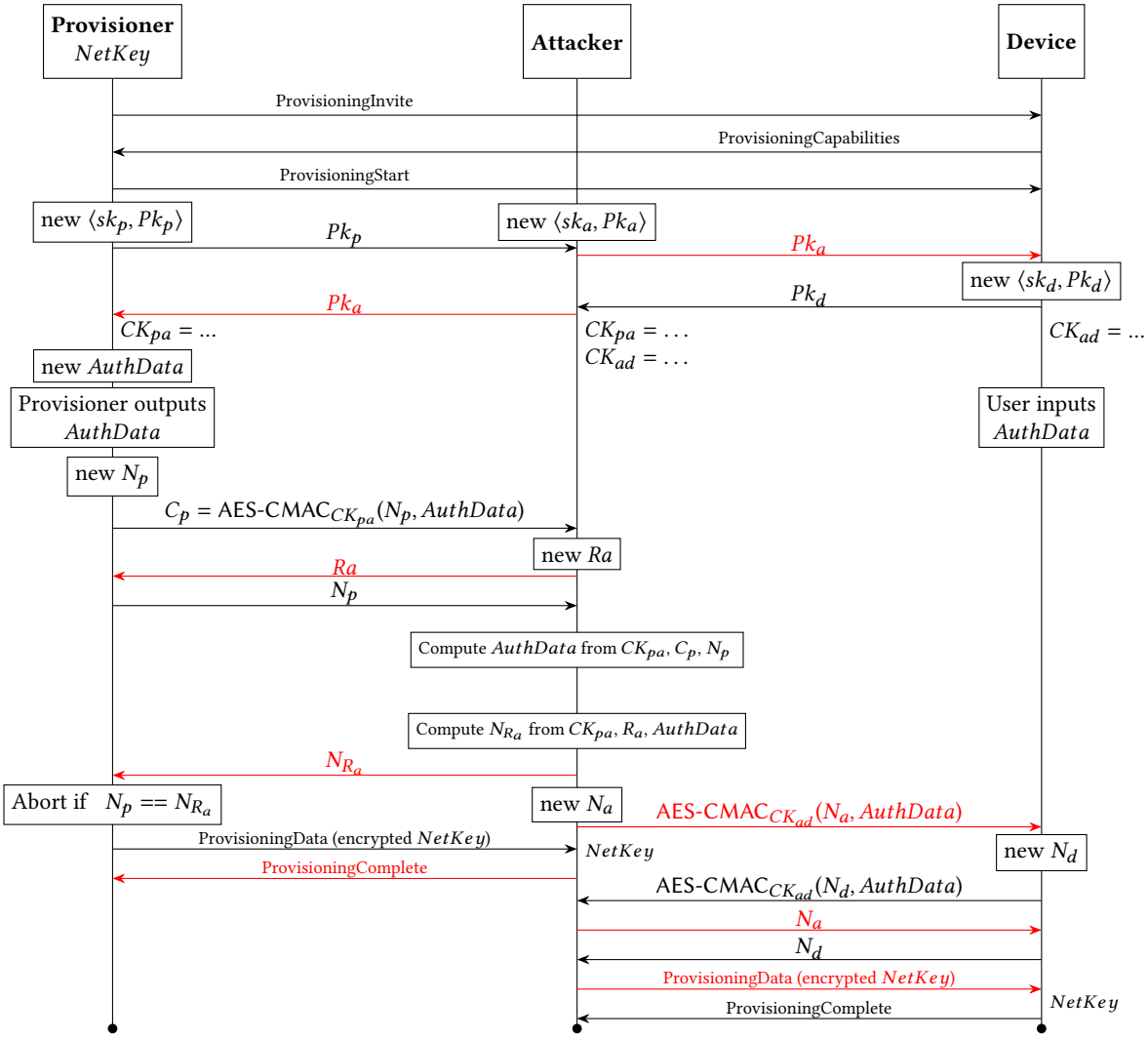

Figure 9: MitM attack against Provisioning patch 1

on two blocks of plaintext: the nonce and *AuthData* are both 16-bytes long. Therefore, to retrieve *AuthData* the attacker retrieves the second block of the computation of the confirmation value. Retrieving the first block allows the attacker to compute a correct nonce for the sent confirmation value, thus successfully completing the protocol.

The attack on patch 2 is almost identical as the one displayed in Figure 9. The only difference is that *AuthData* and the nonce have been swapped in the computation. Therefore to get a matching nonce, the attacker needs to recover the second block of the AES-CMAC instead of the first block for Patch 1. Apart from that, the attack is the same.