Extrapolating Formal Analysis to Uncover Attacks in Bluetooth Passkey Entry Pairing

Mohit Kumar Jangid*
The Ohio State University
jangid.6@osu.edu

Yue Zhang*
The Ohio State University
zhang.12047@osu.edu

Zhiqiang Lin
The Ohio State University
zlin@cse.ohio-state.edu

***Abstract*—Bluetooth is a leading wireless communication technology used by billions of Internet of Things (IoT) devices today. Its ubiquity demands systematic security scrutiny. A key ingredient in Bluetooth security is secure pairing, which includes Numeric comparison (NC) and Passkey Entry (PE). However, most prior formal efforts have considered only NC, and PE has not yet been formally studied in depth. In this paper, we propose a detailed formal analysis of the PE protocol. In particular, we present a generic formal model, built using Tamarin, to verify the security of PE by precisely capturing the protocol behaviors and attacker capabilities. Encouragingly, it rediscovers three known attacks (confusion attacks, static passcode attacks, and reflection attacks), and more importantly also uncovers two new attacks (group guessing attacks and ghost attacks) spanning across diverse attack vectors (e.g., static variable reuse, multi-threading, reflection, human error, and compromise device). Finally, after applying fixes to each vulnerability, our model further proves the confidentiality and authentication properties of the PE protocol using an inductive base model.**

I. Introduction

Being a vital short-range wireless communication technology, Bluetooth has been used by numerous devices for various applications (e.g., earbuds, wearables, and sensors) [12]. Unfortunately, the past several years have also witnessed numerous security flaws, from implementation (e.g., leaking UUIDs [61], or misconfigurations [54]) to specification (e.g., BAIS attacks [5] and Method confusion attacks [53]) that have rendered billions of Bluetooth devices vulnerable to intruders [51]. While luckily these flaws have been discovered, most of them were identified with manual efforts. Therefore, rigorous approaches are needed to systematically reason about the security properties and correct construction of Bluetooth protocols.

Formal verification (FV) is a promising technique for verifying the security of protocols (e.g., TLS 1.3 [19], [22], [10], the Noise framework [26], [29], Signal [18], [16], [28], 5G authentication key exchange [9], and WPA2 [21]). Recognizing its potential, various prior attempts (e.g., [20], [14], [37], [7], [35]) have been made to prove and discover attacks in Bluetooth secure authentication pairing protocols (which allows two or multiple devices to negotiate keys). However, most of these work cover only one of the secure authentication pairings, namely Numeric Comparison (NC), and the other competitive pairing method, Passkey Entry (PE), considered as at the same security level as NC, has been under-investigated. For example, the prior work [57] models a simple version of PE and uncovers only two attacks on PE. However, there could be other attacks (our paper ultimately uncover 5 attacks against PE). We believe that this is due to the higher complexity of the PE protocol (which involves a multitude of messages and confirmations exchanges; asymmetric human interactions; and the involvement of a loop in the authentication phases) compared to the NC protocol. Therefore, we hypothesize that complete modeling of the long and sophisticated PE pairings will expose a large attack surface and may uncover unknown attacks of PE.

To this end, we started with analyzing an accurate Bluetooth pairing environment and the threat model for state-of-the-art attacks involving PE pairing. In particular, we selected the method confusion attack [53], an attack against the PE protocol, as the starting point of our modeling. The attack involves a sophisticated threat model, device capabilities, and entity interactions. In the attack, the user attempts to pair up two legitimate devices, and the intruder hijacks the pairing sessions and initiates different pairing methods on each of the pairing devices. That is, one device is tricked into running PE pairing, which requires the user to input a passcode, and another device is tricked into running NC pairing, which requires the user to compare and confirm a passcode. At this point, the user, unable to differentiate the pairing methods, mistakenly enters the passcode for the NC pairing into the device that is running PE pairing, which allows the passcode to be leaked to the intruder to complete the pairing on both sides and then succeeds with a MitM attack.

During this initial investigation, we found that the method confusion attack covers intricate details of PE pairing. For example, it requires accurate value format abstraction of passcode (i.e., the pairing authentication value produced by PE), parallel PE instances, device passcode customization by an intruder, a careful abstraction of human interaction (entering the passcode) and human errors (mistakenly entering the passcode of an incorrect pairing method). Building a model that can capture such attacks in detail should be able to uncover hidden vulnerabilities that lie beneath the complexity of the PE protocol complexity.

Therefore, in this paper, we present in great detail how we have developed such a systematic formal PE model with device access control (possession of a device for users or intruders), human interaction (e.g., viewing, entering or

The first two authors contributed equally to this paper.

Network and Distributed System Security (NDSS) Symposium 2023
27 February - 3 March 2023, San Diego, CA, USA
ISBN 1-891562-83-5
https://dx.doi.org/10.14722/ndss.2023.23119
www.ndss-symposium.org

confirming the passcode), refined threat model (e.g., intruder's device possession and brute-force derivation), and device capabilities (e.g., displaying static or random passcode). During the modeling process, we faced multiple non-trivial challenges including (i) modeling complex PE protocol, and integrating many features into one unified model, (ii) modeling refined intruder capabilities, and (iii) analyzing large protocol traces. To overcome these challenges, we carefully augmented the design of the pairing protocol environment in Tamarin [32], one of the state-of-the-art symbolic protocol verification tools. With iterative refinement, we gradually upgraded the model complexity and captured semantic intruder capability to explore the attack surface. The large traces were optimized and studied piece by piece for comprehension. .

Encouragingly, our model discovers five attacks, including three known attacks: (1) the confusion attack [53] (the root cause of which is a human mistakenly entering the passcode of a different pairing), (2) the reflection attack [15] (because the devices fail to check the identity), and (3) the static passcode attack [50] (because the devices display static passcode), and more importantly two new attacks: (1) group guessing attack (because the devices use random functions that are not thread safe, displaying the same passcode for multiple devices) and (2) ghost attack (because the devices leak the passcode to intruders). We have validated the two new attacks on the real world devices, and the results have proved the effectiveness of our model and confirmed our insights.

Contributions. In short, we make the following contributions in this paper:

- **In-depth Model for PE Pairing.** We propose a detailed formal model for Bluetooth Low Energy secure pairing protocols with refined adversary threat model, device access control, human interactions (as well as human errors) and device capabilities.
- **Uncovered Attacks and Corresponding Fixes.** Our model[1] formally verifies confidentiality and authentication properties, and discovers three known attacks and two new attacks. After applying the corresponding fixes to the attacks with inductive base case of two PE authentication loops, the model verifies the confidentiality and authentication properties.
- **Lessons Learned.** We explain the mechanism behind the PE pairing method, and draw insights through the modeling process and the uncovering of attacks. For example, the protocols with large loops can potentially trade off FV model complexity by minimizing the loops without compromising model results.

II. Background

A. Bluetooth Passkey Entry (PE) Pairing

Bluetooth resorts to its security through pairing protocols, where two or multiple devices (e.g., Bluetooth Mesh [8]) negotiate keys between each other and use the negotiated key to further encrypt and decrypt the communication. While Bluetooth includes multiple versions (e.g., Bluetooth Classic, which is mainly for audio streaming, and Bluetooth Low Energy, which is mainly for IoT devices), their pairing protocols are similar. At a high level, there are four pairing methods, i.e., Just Works (JW), Passkey Entry (PE), Numeric Comparison (NC) and Out of Band (OOB). In this paper, we focus on the PE pairing method, since other pairing protocols are either insecure (e.g., JW), very implementation specific (e.g., OOB [60]), or well-investigated (NC). Figure 1 illustrates the workflow of PE pairing between two devices `A` and `B`.

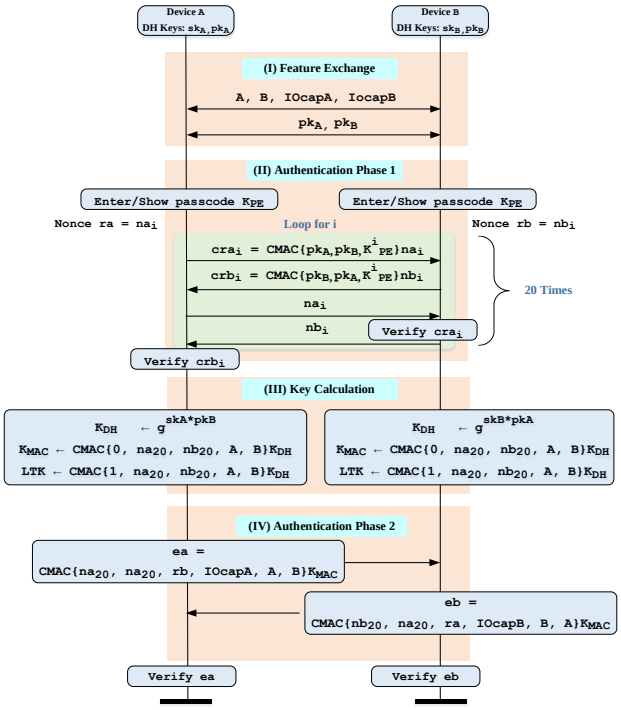

Fig. 1: The protocol flow of PE pairing.

(I) Feature Exchange. Before authentication begins, the two devices exchange their pairing features (i.e., `IOcapA`, and `IocapB`, as illustrated in Figure 1). The pairing features are mainly refer to as the support of a specific input and output capabilities (e.g., presences of screens or keypads), through which the devices can further decide which pairing method should be selected in the following procedures. Then, the devices use the Elliptic Curve Diffie–Hellman (ECDH) key exchange protocol to exchange each other's public key (i.e., pk_A, and pk_B) and produce a DH key for future reference.

(II) Authentication Phase 1. In this step, the two pairing devices select an association method based on the exchanged pairing features. In this paper, we assume that PE is selected as the pairing method. There are two ways to perform PE pairing depending on the input and output capabilities of the two pairing devices. The first case is that one device has a display while the other device has a keypad. In this case, one device displays a six-digits passcode. The user "view" the passcode (i.e., K_{PE}), and physically "enter" the displayed passcode into the other device, and the transferred passcode is used to drive the key in the following steps. The second case is that both devices only have keypads. In this case, the user is required to input the same passcode (i.e., K_{PE}) into the two devices. PE can defend against MitM attacks, because the user is involved and obligated to transfer authenticated values (i.e,

[1]Our Tamarin code is released at Github: https://github.com/OSUSecLab/bluetooth-pairing-formal-verification.

passcode) from one device to the other. After the authentication values are exchanged, the two devices have to make sure the authentication values are the same. To this end, the two devices produce commitments and exchange them 20 times (the reason of why 20 times will be explained in §III). Particularly, during each round, the two devices produce two nonces (denoted as na_i and nb_i respectively, where i is the particular round), and exchange them for future references. Device A calculates a CMACed commitment:

$$cra_i = CMAC\{pk_A, pk_B, K^i_{PE}\}na_i$$

and sends the CMACed commitment to device B. Similarly, device B also calculates:

$$crb_i = CMAC\{pk_A, pk_B, K^i_{PE}\}nb_i$$

and sends it to device A. The two devices both check whether the obtained value is equal to the calculated value. If so, the two devices then initiate the second round. The two devices confirm the authenticated values until all 20 times' CMACed commitment checks are all passed. Then, the authentication values will be used to derive keys.

(III) Key Calculations. After the successful completion of authentication phase 1, based on the generated DH key and the transferred authentication values, the two devices generate a long-term key (LTK):

$$K_{LTK} = CMAC\{0, na_{20}, nb_{20}, A, B\}K_{DH}$$

and a MAC key:

$$K_{MAC} = CMAC\{0, na_{20}, nb_{20}, A, B\}K_{DH}$$

Generally, LTK is saved for the communication encryption.

(IV) Authentication Phase 2. The second authentication phase ensures that the devices have completed the previous steps honestly. This is achieved by exchanging the history of preciously agreed values (CMACed) and verifying the same values on both ends. Particularly, device A performs:

$$ea = CMAC\{nb, na, rb, IOcapA, A, B\}K_{MAC}$$

and sends the value to device B. Similarly, device B performs:

$$eb = CMAC\{nb, na, ra, IOcapB, A, B\}K_{MAC}$$

where rb and ra are two random numbers exchanged in this step. The two devices both check whether the obtained value is equal to the calculated value. If so, the pairing succeeds. In particular, even if the passcode is leaked to an intruder in the authentication phase 1, she cannot bypass phase 2 until she was able to spoof the DH public key of the other pairing device and derive the same DH session key. Finally, the two devices will use K_{LTK} to produce session keys, and the session key will be used to encrypt the rest of the Bluetooth traffic.

B. Tamarin Prover

Overview of Tamarin. Tamarin prover is a symbolic verification tool to model cryptographic protocols. Being such a prover, it takes the modeled protocol to be proved and security properties (which specify the security goal and the security requirements of a protocol) as inputs, and outputs whether or not the modeled protocol satisfies the security properties, and further reasons a protocol from given security properties.

- **(Input-I) Modeled Protocol.** The protocol that needs to be verified is specified in the form of Multi-Set Rewriting (MSR) *Rules*. At a high level, MSR *Rules* allow modeling of protocol agents, model variables, and operations over the variables. All variables in *Rules* are mathematical symbols. Basic operations over the symbols involve: (i) combining existing symbols to derive new composite symbols (i.e., terms); (ii) breaking down terms into its constituent symbols or terms; and (iii) substituting one term into another term (i.e., rewriting). Cryptographic or limited algebraic operations are extensions of these basic operations.
- **(Input-II) Security Properties.** Security properties provide the security requirements of the protocol, which are in the form of first-order logic (FOL) [48]. Typical examples of protocol security properties are confidentiality and authenticity [23], [55], [30].
- **(Output) Satisfiability of Security Properties.** Ultimately, Tamarin determines whether the security properties hold for all executions of the model. Since proving properties for a given model can be undecidable, Tamarin execution does not always terminate and conclude if the security properties hold. Different heuristics are applied in Tamarin to mitigate the undecidability for certain class and patterns of protocols. These classes of protocols cover many real-world protocols, and hence Tamarin can prove properties for many real-world protocols.

Semantics and Execution of Tamarin. In Tamarin, protocol processes are modeled as a sequence of MSR *Rules*, where each *Rule* roughly corresponds to a protocol checkpoint. A *Rule* is made up of three components: *Premise* (which usually defines the inputs of the *Rule*), *Conclusion* (which usually defines the outputs of the *Rule*), and *Action* (which is usually used to label the specific protocol checkpoints that are used to specify protocol properties or behaviors). The sequence of *Rules*, bound together with *Facts*, forms one complete process. The *Fact* is in the format of $\mathsf{F}(t_1, t_2, \cdots, t_n)$, where F is the name of *Fact*, and t_i are the model variables.

The execution of the Tamarin model is driven by the demand of the property logic (aka *Lemma*). For a given property, Tamarin's goal is to find a model execution trace that contradicts the property (failure trace) or to verify that all possible model executions satisfy the property (proof). The detailed failure traces can be observed in the Tamarin interactive GUI [49], [3] rendered with HTML and JavaScript. Additionally, the Tamarin engine allows a proved property to be reused to prove other properties. In particular, the proved property, denoted by *Helper lemma*, acts as a sub-proof to prove other properties. In addition, FOL encoding can also be used to enforce the behavior of the model. This encoding is known as *Restriction axiom*. Tamarin adopts Dolev Yao intruder [24] capabilities by (i) providing all public channels data to the intruder; (ii) allowing the intruder to generate new data and apply cryptographic and model algebraic operations to compose or manipulate known data; (iii) replaying or rerouting the known data to entity receiving endpoints.

III. Security Analysis of PE Pairing

The complete flow of the PE pairing is illustrated in Figure 1. Recall that in order to validate that the user views and enters the same passcode and that the involved devices

are indeed communicating with each other (e.g., not a MitM intruder), the PE pairing process sends the entered passcode value from device B to device A. Since the two devices do not share any pre-shared keys, theoretically it is impossible to securely establish a shared secret without having an existing shared secure channel among the entities [13]. However, in the Bluetooth environment, user interaction with owned devices in the same physical proximity allows partial mitigation of this theoretical limit. In particular, the intruder's ability to derive the shared key can be made probablistically harder. The rest of this section explains the motivations behind the design of PE authentication phases 1 and 2. The cases discussed below assume an alternative design.

(I) Hardening PE Security Using Encrypted Commitment. As shown in case I of Figure 2, assume that one attempts to harden PE security by first exchanging pairing features (❶), then generating random nonces on A and B (❷), and using the nonces to encrypt the passcode (❸) and public DH key (❺). The encrypted chunk, known as the commitment message, is passed on to the other side followed by exchanging the nonces for the verification of the same passcode. In this approach, since both devices commit to the same passcode value before revealing the encryption keys (the nonces), it is impossible for an active MitM intruder to derive the passcode during the commitment phase. In particular, the intruder cannot decrypt the committed passcode before the nonces are exchanged (❹). Nevertheless, this restriction is not completely secure because it allows an active MitM intruder to delay the nonce exchange in one session and collect the passcode from the other session and connect to a victim device on the other side. As such, encrypting the commitment is not enough to protect the PE protocol.

(II) Hardening PE Security Using CMACed Commitment. A second attempt to harden PE security is to CMAC the commitment data instead of encryption in step ❸, while steps ❶, ❷ and ❺ are the same as of the prior case. This attempt is illustrated in case II of Figure 2. The use of CMAC prevents the MitM intruder from obtaining the passcode directly. However, the intruder may still obtain the passcode by enumerating all the possibilities of the passcode (❹). In particular, the intruder works as a relay between A and B, and sends the intruder's own public key to A and B, respectively. As a result, A, which communicates with the intruder, believes that it is communicating with B, and sends the CMACed commitment to the intruder. Once the intruder obtains the commitment, which is calculated from:

$$cra_i = CMAC\{pk_A, pk_i, K_{PE}\}na$$

the intruder can delay the session and use pk_A, pk_i, na to brute force the passcode. Since the passcode is only 6-digits long (e.g., 123456), the intruder can easily obtain such a passcode [34], and successfully pass the check initiated by A. Similarly, since the intruder has already obtained the passcode, the intruder can use the passcode to pass the check initiated by B. Ultimately, the intruder launches the MitM attack. Again, using CMACed commitment can be easily broken by the intruder.

(III) Hardening PE Security Using 20 Loops of Commitments. To eliminate the above weaknesses, the Bluetooth SIG designed the PE protocol to exchange the passcode one bit

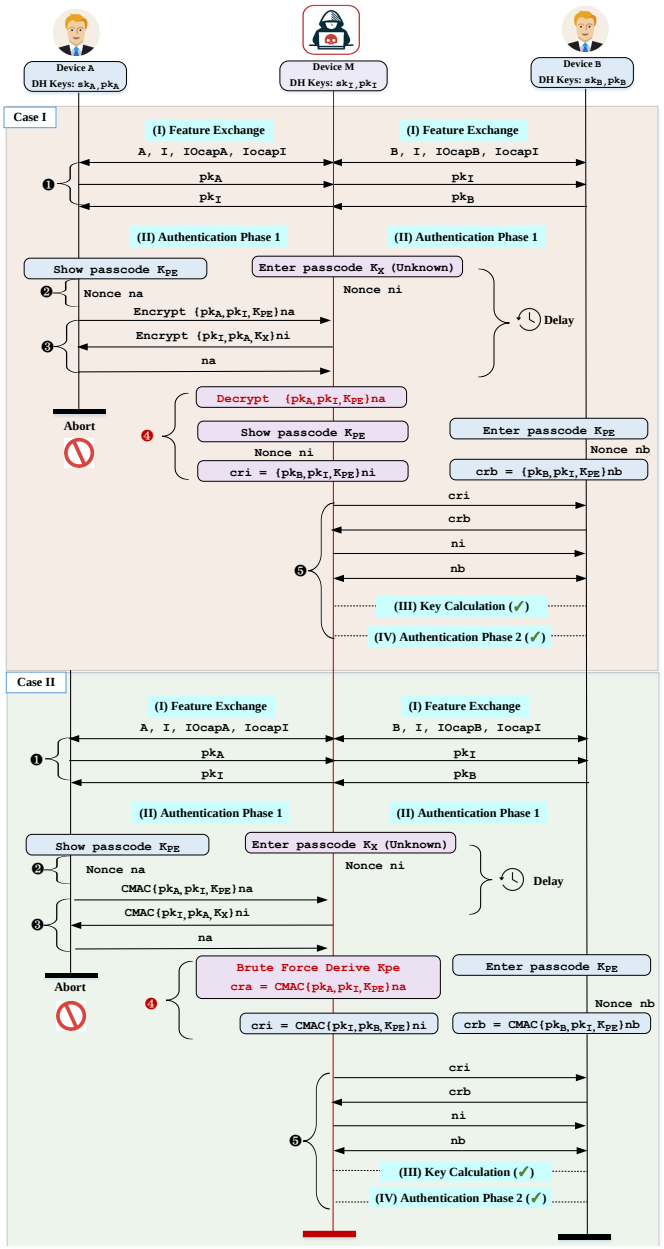

Fig. 2: MitM attacks on the single PE loop implementation.

at a time through commitment messages in 20 consecutive loops. The passcode is a 6-digit value and it is 20 bits (e.g., 999,999=0xF423F) in length as shown in Figure 1. In PE protocol, the intruder can carry out the case II attack for the first loop iteration. She can also learn the first bit of the passcode as it requires trivial brute force for two possible options of a bit. However, she cannot easily advance to the second and further loop interations. To continue in the second loop, she needs to correctly guess the 2nd bit of the passcode before exchanging the nonces. Otherwise, the protocol will not continue. Once the intruder makes any guess mistake, the PE paring aborts and the intruder has to restart the pairing again. Consequently, to crack the whole passcode, the intruder has to guess each bit of PE passcode value 20 times correctly. This renders the

success probability of the intruder to $(1/2)^{20} = 0.000000954$, which is reasonably secure.

IV. Terms, Scope, and Threat Model

A. Terms

Throughout the paper, we will use the following terminologies: (i) *Device*. A Bluetooth enabled equipment, which can engage in Bluetooth protocol communication and can exchange messages, and we use symbols A and B to denote the two Bluetooth devices. (ii) *Central.* A Bluetooth device that acts as a central controller for many surrounding Bluetooth devices. Such a device supports multiple connections at the same time and typically initiates connection requests with surrounding devices. Examples of central devices are smartphones, tablets, and personal computers. (iii) *Peripheral.* A Bluetooth device that typically receives connections from a central device. Such a device usually supports only one connection at a time, but since 4.2, the peripheral devices can accept more than one connection from centrals. Examples of peripheral devices are keyboards, smart locks, and smart lights. (iv) *User*. A legitimate human who owns a Bluetooth enabled device(s). The user becomes a victim when exploited by an intruder. (v) *Intruder*. A malicious human who illegitimately exploits the vulnerabilities of victim's devices. In general, an intruder is also referred to as an attacker or an adversary. (vi) *Entity*. A generic term for Bluetooth protocol participants that could be a user, a device, or an intruder.

B. Scope

Scope of the Pairing Method. We focus on PE pairing by following the Bluetooth specification 5.2 [45] (Vol 3, Part H, Section 2.3.5.6, page 1645). Additionally, we made some customization by allowing the devices to have both static and random passcode. While Bluetooth SIG has explicitly mentioned that a static passcode should not be used since 4.2 [43], they never explained what kinds of attacks can be caused by that, thereby leading the device manufacturers (e.g., TI [52]) still follow the earlier PE protocol (configuring a static passcode for user convenience). As such, we added this feature to our model to further investigate its security impacts. Finally, we also model NC in our paper, since some attacks exploit the vulnerability existing in NC to attack PE (e.g., method confusion attacks [53], which will be explained in §V-A). JW and OOB pairing are not considered; since JW is vulnerable to many attacks [60], [38] and does not involve human interactions, and OOB implementations use application-specific non-Bluetooth channel.

Scope of the Attacks. Not all Bluetooth attacks are of our focus. In particular, attacks that occur after pairing are not considered (e.g. BLESA [56], BadBluetooth [58], MisBonding [33]). In Tamarin, keys are symbolic terms. Therefore length or entropy of the keys are secure and cannot be downgraded (e.g., KNOB attacks [6] and downgrade attacks [60] are out of our focus). The DH cryptography is assumed to be perfect. Thus our model does not capture the Invalid Curve Attack [20]. Finally, hardware and side-channel attacks are beyond the scope of this paper.

Scope of the Model. Our model follows the scope implied by inherent Tamarin assumptions [27]. Specifically, we allow unbounded instances of protocol users, devices, and sessions. Each device can communicate with other devices in parallel when needed. The model is queried for authentication lemmas formulated according to Lowe's hierarchy of authentication [30]. The confidentiality lemmas are formulated as adversaries' ability to derive the protocol secrets (data and keys) in all possible executions of the model.

C. Threat Model

Apart from the Dolev Yao adversary model [24], we assume the following: (i) *Access of Malicious Device.* The intruders can own many devices and can access the messages produced by their own devices in a precise generation time (e.g., they can reuse the NC passcode in PE pairing). (ii) *Access of Victim Devices.* The intruder can also physically access the user devices momentarily (e.g., when the user is away from her office, her keyboard can be briefly accessed by the intruder) or install malware (which does not have root permissions to access the LTK) on the victim devices. These assumptions are widely discussed in various papers [60], [57], [58]. For example, BadBluetooth [58] assumes that the smartphone is hacked by malware that allows the smartphone to pair with the malicious device. Bluetooth downgrade attack [60] assumes that the attacker can physically touch the Bluetooth device to pair with it. In fact, these assumptions are included in a semi-compromised device threat model (e.g., one of the devices is compromised by malware attacks or by physical access of the intruder), which is also a common threat model used in formal Bluetooth verification [57].

V. Model Design

A. Initial Target — Modeling The Method Confusion Attack

As discussed in §III, PE pairing is a complicated procedure that involves multiple rounds of nonces exchanges, 20 loops of commitments exchanges, and asymmetric human interaction. As such, we need to set an initial target attack such that the formal model captures the complete details of the PE protocol. Our intuition is that if our model captures such an attack, it may capture more attacks with similar settings. Fortunately, we noticed that the Method Confusion attack [53] satisfies all these requirements: first, in the attack, the victim executes the PE protocol completely; second, the attack involves human interactions as well as human mistakes (the user mistakenly inputs one device's passcode into another). Thus, we set this attack as the initial target attack for our formal model.

Method confusion attack works against devices running PE and NC (Numeric Comparison) pairing. We explained PE in §II, and now we briefly explain NC. In NC association, two pairing devices both display a six-digits passcode. It is a CMACed value of public keys of two pairing devices with nonces as a salt and also as a CMAC key. In this case, the user has to check whether the displayed passcode is the same to "confirm" that the two devices are communicating with each other and not a third party intruder. Therefore, similar to PE, NC can defend against MitM attacks as well.

However, method confusion attacks can break the guarantees of both NC and PE pairing. Now we provide more details of the attack as shown in Figure 3. There are two devices A and B owned by a user, and two devices AI and BI owned by an intruder. The attack begins with the intruder controlling the PE session with one of the devices (A on the left side) and the NC session on the other device (B on the

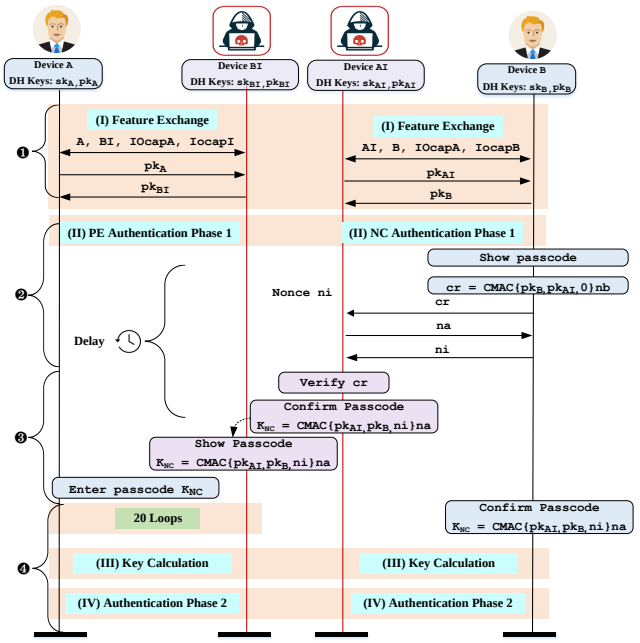

Fig. 3: Illustration of the Method Confusion attack.

right side), exchanging the pairing features with the two victim devices (❶). The intruder waits for the NC session to show the passcode K_{NC} (❷). Then the intruder uses the same passcode, and displays it on BI (at the PE session on the left). At this point, the user observes that B displays passcode K_{NC}, and A asks to enter the passcode (❸). At this checkpoint, note that the device B displays passcode for NC pairing with AI and not A. Given that the user is not aware of this fact (the user gets confused about pairing methods and the pairing steps based on their survey [53]), it is very likely that the user will enter K_{NC} in A. As a result, the intruder is able to pair both victim devices (❹). In particular, A pairs up with BI. Intruder at this point also hits the "Yes" prompt in AI to confirm the passcode K_{NC} with B. Such a pairing gives the user an illusion that A and B are successfully paired with each other. Note that since the intruder has the passcode K_{NC} for both pairing sessions and that she was able to initiate the feature exchange phase with her own DH keys, she can perform all the later steps simply by following the protocol steps.

B. Design Challenges and Solutions

To model the method confusion is non-trivial, and we face at least three challenges. First, our model must have the capabilities to model the complex pairing protocols (e.g., both NC and PE need to be modeled), and the interactions between the entities (e.g., there could be multiple users, each of which owns multiple devices). Second, our model has to capture the intruder capabilities that are not supported by Tamarin by default. For example, to attack PE, the intruder may guess the bit of passcode as discussed in §III, and such capabilities are not defined by Tamarin. Finally, the modeling of such a complex attack may result in large traces or proof non-termination. We now explain these challenges (C), and their solutions (S) in greater details:

C1. Complex Pairing Protocol and Interactions between Entities. As discussed in §III, PE pairing itself is a complicated pairing protocol. First, the user and intruder device should be provided with appropriate access controls (e.g., a user should be able to access and configure the owned devices' settings but should not have access to the other users' devices). Second, the combination of NC and PE flow should execute in parallel within the same model for an attacker to exploit the confusion attack. Lastly, the human passcode interaction should be consistent and uniform between the two pairing methods. Note that in the NC pairing, the passcode is generated at both initiator and the responder side and is confirmed with human interaction. Whereas in the PE, the passcode is displayed only on one side and only entered on the other pairing device or entered the same passcode on the both sides.

S1. We next explain how the three sub-challenges discussed in C1 can be solved. Specifically, to solve the first sub-challenge, our idea is to first establish a relationship network between entities. To be more specific, the relationship network is a hierarchical three-layer network that consists of users, devices, and protocols. The first layer (user layer) instantiates multiple users; the second layer has multiple devices that are owned by each of the users at the first layer (i.e., one user can have one or more devices); the third layer consists of the protocol processes running within the devices of the second layer. Since each entity is specified with a dedicated Tamarin term, the user and intruder devices are provided with appropriate access controls.

Second, for the NC and PE pairing integration, note that these pairings differ only in the authentication phase 1. To combine the two protocols, our design is to generalize the protocol variables before and after the differing authentication phase. Starting from the single initiator and responder processes, the protocol flow is split into PE and NC authentication phase 1 as a possible choice of execution. Additionally, for each pairing method, different choices of flow are possible. For example, in PE pairing, each device could either show the passcode or require the user to enter the passcode. The displayed passcode could be set static or it can be dynamically changed in each session. All these choices are abstracted as protocol branches and can be taken randomly in our model. Finally, as the branch closes for the PE and NC protocols, the variables are merged back into the common flow using dummy variables to generalize specific variables of the individual protocol. In this way, whenever the model executes, it assumes all possible combinations of pairing method and protocol customization.

Finally, for the consistent human interaction abstraction, we proceed as follows. Each human action uses encryption over the passcode, which is further wrapped inside Tamarin *Facts*. For displaying the passcode, each pairing entity releases the passcode encrypted with a user ID (which is uniquely assigned to each user). Similarly, the case where the user inputs the same passcode on both two devices can also be modeled by encrypting the passcode using the same user ID.Wrapping in Tamarin facts avoids replaying of human interaction action by the adversary. The human error is modeled by not binding NC or PE human interactions with their pairing method type. Since the user ID is explicitly kept inaccessible to the intruder using the Tamarin *Fact* properties, it preserves the secure nature of human interaction: uninterruptible, modifiable, or breakable by the intruder. The access control works across the unbounded user-device and intruder-device protocol communication.

C2. Modeling Precise Intruder's Capability. Modeling intruder's capabilities is challenging, as some capabilities are not directly supported by Tamarin. Particularly, the intruder needs to guess the bit of the passcode, but such capability is not included in the original Tamarin. Recall that in authentication phase 1 of PE pairing (Figure 1), there are 20 times iteration, and each of them exchanges the commitment messages produced from passcode. The message is CMACed with AES encryption and sent in each iteration. Modeling the message in literal accuracy — irreversible keyed message digest — in Tamarin would mean that the intruder cannot derive the bit from each iteration. This is because being a symbolic verification tool, Tamarin assumes that all cryptography operations are perfect [3]. However, in reality, there are only two possible values for the passcode bit, and the intruder can easily break the security of the passcode bit, since guessing a correct bit requires only two brute-force attempts.

S2. To model the capabilities of guessing a bit of the passcode, we represent each bit of the passcode using the equational theory: $\boldsymbol{merge}(split1(v), split2(v)) = v$, where $split1(v)$ and $split2(v)$ represent two bits of the generic model variable v (in our case, variable v represents the passcode) [2]. Thereafter, encryption of each bit of the passcode with the nonce is generated in each round. Since in each round, the nonces are exchanged after exchanging the commitment messages, the intruder is able to obtain the exchanged nonce and break the confidentiality of the bit of the passcode, thereby modeling the procedure of guessing the passcode. This method also preserves the time order of the checkpoint when the intruder is able to derive the passcode bits from the recent commitment message exchange among legitimate entities. In other words, the intruder derives the passcode bits only after the nonce exchange and not anytime before.

C3. Heavy Tamarin Model and Large Traces. Integrating multiple pairing features into one unified model is challenging. In particular, our approach combines the long PE protocol together with NC protocol, resulting in various possible protocol branches. Meanwhile, the model also integrates device access control, human interactions, and human mistakes. Consequently, Tamarin execution can run for unreasonably long time before yielding a result. Parsing and debugging such large traces is a challenging task.

S3. We use the following approaches to resolve the heavy Tamarin model and large traces. First, we formulated security lemmas that involve fewer constraints than the standard authentication lemmas and are hence more effective to execute. More details of the symbolic formulation of these lemmas are explained in Appendix §A. Second, the inbuilt DH equational theories introduced a lot of unnecessary variants in the model. Therefore we replace them with the simpler user-defined DH equational theory. Third, we ran the model under limited scope (e.g., a limited number of sessions and user devices). Observing the run time of terminating model lemmas under a bounded scope often project the non-termination root causes to an unbounded case. For example, an attack observed within the bound of 3 parallel sessions is also effective under unbounded scope, but the corresponding lemma may not terminate in the unbounded model. Such a relative observation of the lemma run time in the bounded scope allows easy debugging and finding the root causes of non-termination.

C. Model Design and Implementation

In the Bluetooth environment, the main entities are users and intruders, devices. In the following, we first present how we model complex pairing protocols and interactions, followed by modeling the actions of intruder, and finally we explain how we resolve the heavy traces.

1) **Modeling Pairing Protocols and Interactions:** In the following, we explain how we model the entities and the ownership of devices, protocols processes, and interactions of the users including human mistakes.

Modeling Entities and The Ownership of Devices. The entities in our model are the intruders, the users, and their owned devices. The intruder is already built-in to the Tamarin FV engine. To assign a user to a device and the process that runs on the device, we navigate the respective terms of the entity of the user and device, denoted by *user ID* and *device ID*, through the *Facts* that bind the different *Rules* of the pairing process. Specifically, *Rule* of each protocol process is bound together with *Fact* of the form: $\mathsf{F}_{\mathsf{step}}(U, A, B, var1, var2, ...)$, where the participating devices IDs A and B in the protocol are accessible to the user ID U. That is, the user U, has exclusive access control to interact with the devices A and B. The variables $var1, var2, ...$ are the session variables observed by the protocol process. Throughout the execution of the process, the user ID is kept secret through the Tamarin *Fact* properties [3]. Concurrent execution of processes is a direct result of the Tamarin *Rule* execution criteria.

Modeling Pairing Protocols. We model both the PE and NC protocols based on the described pairing steps introduced in the Bluetooth specification. For simplicity, our model does not include the feature exchange, but allows the devices to freely choose their pairing methods. Additionally, in the PE pairing, the passcode can either be random or static. To set the passcode static in each run, the encrypted passcode is reused in different runs by Tamarin executions. To set it random, an additional restriction axiom over the displayed step is introduced to enforce the passcode to be unique for each run. In the model the displayed passcode *Rule* ShowPasscode receives the passcode from the *Rule* GenPasscode that generates the passcode.

$$\frac{F_{prev}(U, A, B, ...), In(Enc_U(K_{PE}, \text{“}SetPasscode\text{”}))}{F_{next}(U, A, B, ...), Out(Enc_U(K_{PE}, \text{“}atA\text{”}))}\ \textbf{ShowPasscode}(K_{PE})$$

$$\frac{!User(U), Fr(\sim K_{PE})}{Out(Enc_U(\sim K_{PE}, \text{“}SetPasscode\text{”}))}\ \textbf{GenPasscode}(U, \sim K_{PE})$$

Thereafter, the restriction axiom:

$$(\forall\ \textbf{ShowPasscode}(K_{PE})\ @i_1 \wedge \textbf{ShowPasscode}(K_{PE})@i_2 \Rightarrow i_1 = i_2)$$
$$\wedge (\forall\ \textbf{EnterPasscode}(K_{PE})\ @i_1 \wedge \textbf{EnterPasscode}(K_{PE})@i_2 \Rightarrow i_1 = i_2)$$

ensure that all the protocol display and the enter steps of corresponding action in the PE pairing will use a unique random value for the passcode value.

Modeling Human Interactions. We now explain how we model human interactions. Each human action uses encryption over the passcode passed through Tamarin *Facts*. Specifically, at the step of displaying the passcode, each pairing entity releases the passcode encrypted with a user ID. In the model, the user ID represents the user, and it is also used to encrypt all variables that are accessible to the user. In this way, a user can have multiple devices and she can use the user ID to perform various actions over the passcode on the owned devices. Specifically, in the model, the display encryption is in the form $\boldsymbol{SecCh(Enc_U}(K_{PE},\text{“}atA\text{”}))$, where K_{PE} is the passcode and the label; “atA” denotes that it is generated at device A; $SecCh$ is the Tamarin Fact used for secure communication. Obviously, “viewing” a passcode is the reverse process of encryption, where the user decrypts the passcode as $\boldsymbol{Dec_U}(K_{PE},\text{“}atA\text{”})$ using her private user ID. For displaying and entering the passcode in PE pairing, the encryption takes place at the passcode display end and the decryption on the other end. For entering the same passcode on both side, the devices obtain passcode from the GenPasscode rule and thus also allow customizable passcode feature. Finally, for NC pairing, encryption and decryption take place at both ends of the pairing device, effectively modeling the mutual passcode confirmation.

2) **Modeling Precise Intruders' Actions:** We further strengthen the Intruder on the top of insights discussed in **S2**. During the pairing process, the intruder may appear at any stage of the pairing, and the time order of the steps should be strictly specified using action labels. Tamarin Dolev Yao capabilities allow the built-in intruder to impersonate any protocol entity. However, when Tamarin intruder impersonates any model entity (or possesses a device with access control described in C1) her protocol steps are not registered as action label in the model trace. Such precise intruder action labels were required for a clean derivation of Method Confusion attack. Fortunately, the intruder's device possession can be modeled, which involves a dedicated MSR *Rule* in the format:

$$\frac{!User(U)}{Out(U)}\textbf{MakeIntruder}(U)$$

which leaks the user ID U to the public channel. In this way, all the protocol processes running with the leaked user ID acts as protocol initiated and impersonated by the intruder. Also, for each process *Rule* our model explicitly leaks the private variables (*e.g.* nonces) encrypted by user ID to the public channel. This way the Tamarin intruder can access and exploit the impersonated process variables at the instance when the related action is executed in the model.

3) **Resolving Tamarin Model and Large Traces:** As discussed in **S3**, we noticed that the built-in Diffie-Hellman equational theory [3] had a large set of rule variants [31], [39], [40] to satisfy the logarithmic operation formats. The explosion of these variants caused a cascading verification burden on the Tamarin proof process, resulting in nontermination. The replaced simpler DH equational theories can be observed in Tamarin Code 1 in Appendix §B. Currently, there is no automated approach for debugging the Tamarin non-termination in the community. Therefore, manual analysis of the interactive Tamarin proof is a common way resolve non-termination. In this way, we were able to write helper lemmas to terminate the time-consuming lemmas in the unbounded model. To debug and parse very large Tamarin traces, we first symbolized and acronymized the long complex protocol terms, such as DH public keys and encrypted or decrypted terms. Thereafter, we draw pen-and-paper style traces to build a succinct representation of the whole trace. Such a representation helped us to see the attack trace from the bird-eye view and quickly find the root cause. After repeating the process multiple times, we learned which protocol components were crucial to understand the attack (e.g., the user and device identifiers for PE and NC threads and intruder actions) and which components only add to the details of the protocol flow (e.g., actions labels and the respective tagged variables). We utilized the Tamarin Python graph rendering script [1] to eliminate the non-essential component of the traces and generate the manageable traces.

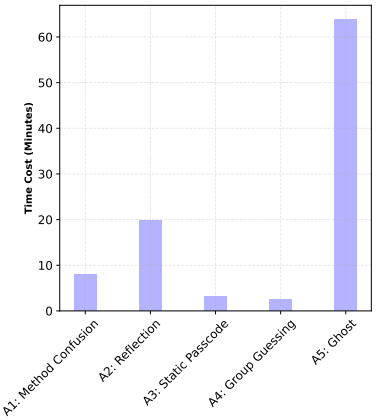

Fig. 4: Time cost for Tamarin to produce the attack traces for the discovering individual attacks in the vulnerable model.

VI. Results

A. Model Setup

Quantitative Assessment of Model Development. Our model consists of about 1,700 lines of code. We defined 26 lemmas, 13 functions/equational theories with 5 branches. Compared to Wu et al.'s [57] formal model, whose PE part is around 600 lines of code with 38 functions/equational theories and 2 branches, our model is complex enough to discover more PE pairing-related attacks. For example, we model multiple loops in authentication phase 1, while Wu et al.'s [57] formal model apparently does not. As discussed in §III, missing those details may lead to the failure of discovering new attacks. In terms of model development timeline, integrating and verifying the NC and PE protocol features took roughly 3 months.

Environment. All the models in our research were run with Tamarin-prover 1.6.1, on Mac OS X - 10.14.6, Intel(R) i5-7360U CPU @ 2.30GHz processor with 8GB RAM.

B. Overall Results

In the vulnerable models, the standard authentication lemmas and the derived lemmas with fewer constraints resulted into the attack. The visual representation of these attacks can be observed in Tamarin interactive GUI page. The trace discovery time for all attacks is presented in Figure 4. As

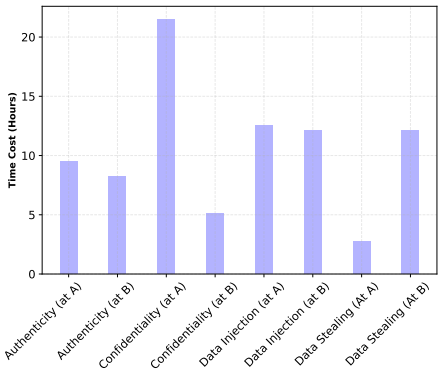

Fig. 5: Time cost for verifying the security lemmas in the patched model. In symbolic formulation lemmas are proved with respect to each protocol entity's perspective. *A* and *B* denote the protocol entities as initiator and responder repsectively.

discussed in §V-B-C3, proving standard properties for the patched model takes a long time. Therefore, we broke down the confidentiality and authenticity lemmas into sub-properties, which consists of fewer constraints and thus are easier to prove. The constraint-reduced lemma encodings for data stealing from user device and data injection at the user device are explained further in Appendix §A. The subproperty lemmas were further granularized based on the hierarchy of authentication [30], based on the point of view of protocol entities, and properties with respect to session variables namely data and key. Finally, after proving constraints heavy lemmas, we reused them to restrict the state space of the model and reduce the time to prove other lemmas. In total, we have reused 13 helper lemmas, some of which are in the form of sub-property lemmas, to complete verification of the patched model. The complete lemma reuse mapping (Figure 10) and a piece of key Tamarin code snippet (Tamarin Code 2, Tamarin Code 1 can be found in Appendix §B), and the time cost for the verified properties of the patched model are presented in Figure 5. It can be observed that all the models in total are finished with approximately 31 hours.

C. Uncovering Known Attacks

Our model successfully captures the targeted Method Confusion attack (A1). Further, as per our hypothesis the model also uncovers two other existing attacks, namely reflection attacks (A2), static passcode attacks (A3). More importantly, it also uncovers two new attacks, namely group guessing attack (A4) and ghost attack (A5). In the following, we describe these attacks in detail.

A1: Confusion Attack. The attack has been confirmed by Bluetooth SIG (CVE-2020-10134) and has been broadly disclosed to its member companies. Its detailed Tamarin trace can be observed in Appendix C and in the provided Tamarin code folder. We will not go into details here as we have already discussed the attacks in §V-A.

A2: Reflection Attack. This attack [15] exploits the PE pairing method. The assumption made here is that the user device does not check whether the DH public key received from the other entity is a reflected copy of the device's own public key. Recall that in the public key exchange step described in §II-A, the two legitimate devices need to exchange their public keys, and then the key is used to perform a challenge response in authentication phases 1 and 2. Overall, the attack executes in three steps. (i) The intruder positions herself as MitM and spoofs as one of the user's devices. (ii) In the challenge response process, the intruder then tricks the user's device into responding its own challenge, and returning the response to the intruder. Consequently, the reflected DH public key and the PE protocol commitment messages successfully pass the checks on the reflected entity. (iii) By exploiting reflection in one of the MitM sessions, the intruder completely bypasses all authentication phases for the other sessions. This attack has two variants: partial impersonation in BIAS attacks [5] (only pass authentication phase 1 without LTK access), and BlueMirror attacks [15] (complete impersonation). Our model captures the complete impersonation and the successful MitM attacks.

Attacks	Pairing Methods	# of Intruder Devices	Root Causes
A1. Method Confusion	PE, NC	2	Human Errors
A2. Reflection	PE	1	Missed Identity Checks
A3. Static Passcode	PE	1	Static Passcode Across Sessions
A4. Group Guessing	PE	21	Static Passcode Across Threads
A5. Ghost	PE	2	Compromised Devices

TABLE I: Summary of Uncovered Attacks

A3: Static Passcode Attack. This attack [50] works against the PE pairing method and assumes that the user devices use the same passcode for all PE pairing sessions. Although Bluetooth SIG has explicitly mentioned that a static passcode should not be used since 4.2 [43], many Bluetooth stack implementations (e.g., TI [52]) still allow setting a static passcode for the pairing. The static passcode attack can be deployed in the following steps. (i) The intruder sniffs the PE pairing traffic and analyzes the exchanged packets. As discussed in §II-A each loop of PE pairing reveals one bit of PE passcode. For each loop, the intruder needs only two guesses to derive the bit. (ii) After obtaining the complete traffic of a legitimate device communication with the PE pairing, the intruder learns the complete passcode. (iii) Later, the intruder initiates the pairing process and performs the MitM attacks, since at this time the same passcode is used, which is already known to the intruders, she can spoof with the user devices.

D. Uncovering New Attacks

A4: Group Guessing Attack. The Bluetooth specification warns developers not to use the static passcode since 4.2 (which means all the Bluetooth devices that followed Bluetooth 4.2 or earlier Bluetooth specifications are all subject to the group guessing attacks). In Version 5.2, Vol 2, Part H page 990, the specification specifies that "*The Passkey should be generated randomly during each pairing procedure and not be reused from a previous procedure.*" However, the specification does not specify how to generate such a passcode. Therefore, there could be confusions that mislead the developers. For example, one attempt to avoid static passcode could be to use a random function to generate the passcode. This solution can work correctly for a single-threaded Bluetooth connections. However, it can fail in the case of a concurrent

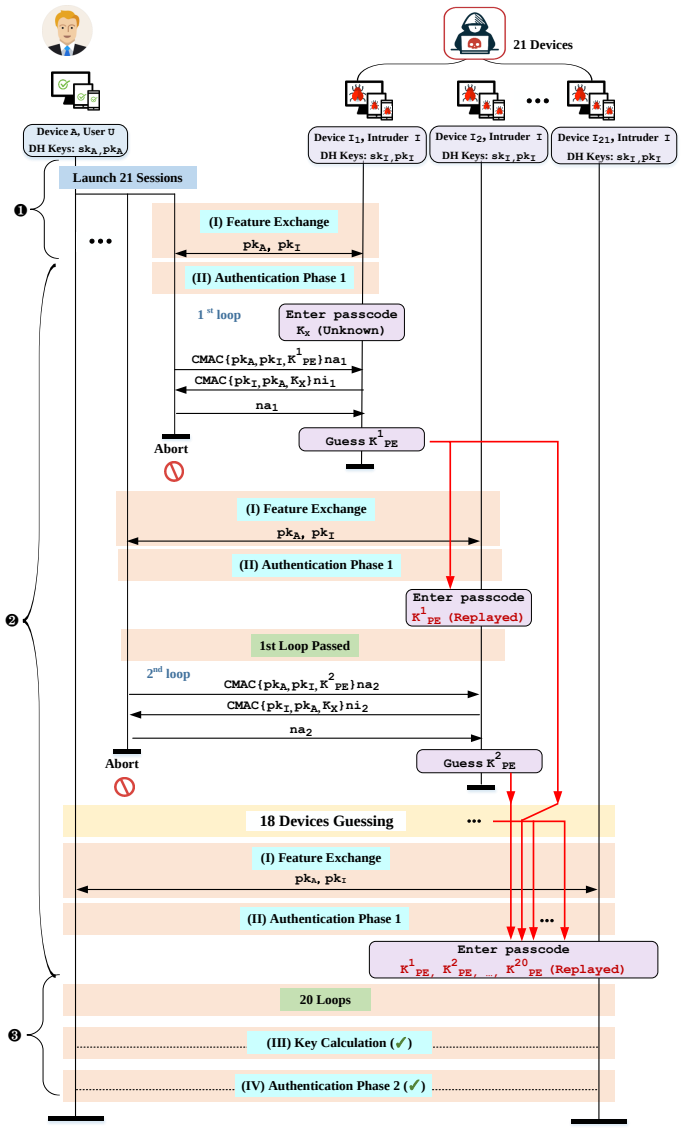

Fig. 6: A complete attack sequence for the Group Guessing attack. Tamarin trace for the same attack shows the attack pattern over two loops.

session from the same device. Particularly, if the random generation function incorrectly provide the same passcode for the concurrent sessions, it would produce the same passcode in all concurrent sessions. For example, the random function used by the devices may produce a passcode for the pairing sessions to consume, and the passcode could be the same (e.g., due to non-thread-safe [17]) until the devices finish pairing. In general, there could be many ways (e.g., the random function may consume the same timestamp as its seed) that can compromise the randomness of passcode generation. All of these cases can be exploited for Group Guessing attack. The specification mentions that the devices should not reuse a passcode from a *previous* procedure, while it does not prevent the multiple connecting devices from using a passcode that will be used in all sessions simultaneously (which has never been used before). In this manner, a Bluetooth application developer trying to comply with the specification could fail to implement a complete solution to avoid static passcode. Our responsible disclosure to Bluetooth SIG also confirms our observation: group guessing attacks could be of concerns, as it is theoretically possible.

As illustrated in Figure 6, the group guessing attack can be executed in the following three steps. (i) The intruder connects multiple peripherals with a single Bluetooth central owned by a user. Since the central is running a stack that uses non-thread-safe random functions, the central will generate the same passcode for each of the simultaneously connecting peripherals (❶). (ii) Each connecting peripheral allows the intruder to guess and learn one bit of the passcode (as explained in §III). A wrong guess at a peripheral will trigger disconnect a central. However, it does not prevent the intruder to replay the learned bit in the other simultaneously connecting peripherals. As shown in the Figure 6, the intruder can continue replaying the learned bits and guess the rest of the bits through other peripherals. Since there are 20 bits in total, in the worst case, the intruder may use up the first 20 peripherals to guess all the passcode correct (❷). (iii) With the learned passcode, the intruder completes the pairing session against the user central on the 21st peripheral connection. Note that the attack assumes the user is away from the central device and she does not notice the passcode display prompt of pairing connection (❸).

PoC and Its Practicality. We have validated such attacks on our own development board, namely, CC2640R2F, which allows us to customize the passcode generation function. In particular, the passcode callback was registered with `GAPBondMgr` when the development board started to enter or display the passcode, and we can modify the way of generating a passcode by overwriting the logic of passcode generation before the passcode is finally fed into `GAPBondMgr_PasscodeRsp` (which displays the passcode to be entered). That is, we explicitly develop a function that is non-thread-safe, which returns the same passcode if two or more devices pair the central at the same time. We then use the other six development boards to simulate the attack procedure. Theoretically, in the worst case, the attack requires 21 device to launch when each device only guesses a bit correct. However, to simplify the procedure, we let the attack devices to simulate a case where each of our device can guess three bits (i.e., 20/6=3) correctly. Ultimately, we validated our attack.

We emphasize here that such an attack discovery was made possible by the unbounded session design of our model Specifically, without allowing unbounded processes for each device, the group guessing attack would not work. Furthermore, the sophisticated requirement of this attack hints at the reasons why such attacks have not been uncovered so far. Although we have not yet found any vulnerable implementation, we believe that multithreading is prevalent in modern IoT devices. Therefore, it is only a matter of time that such attacks will be revealed in the near future. Therefore, cautioning developers about using thread-safe random functions in the Bluetooth specification will help developers build secure applications.

A5: Ghost Attacks. With a successful active MitM attack in wired LAN networks, users find it difficult to detect the presence of the attacker. This is also because the user devices are located in a physically distant location and they are not aware of the responses on the other side of the connection. On the

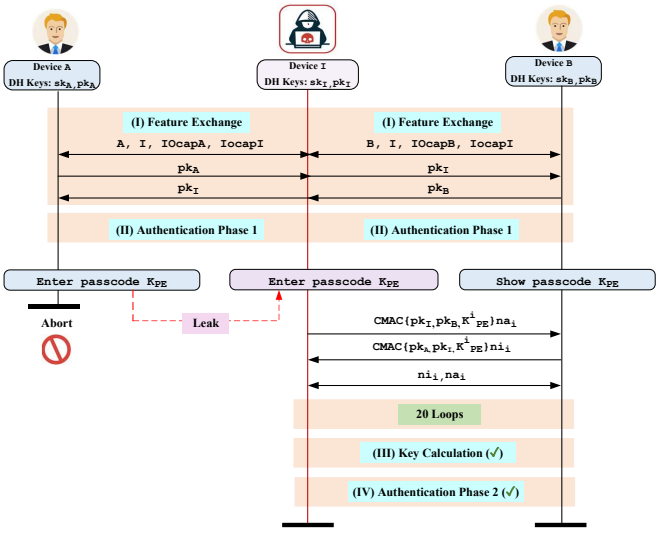

Fig. 7: Tamarin produced authentication failure trace of *Ghost attacks*. The highlighted red protocol actions denote the key intruder's moves to derive the attack.

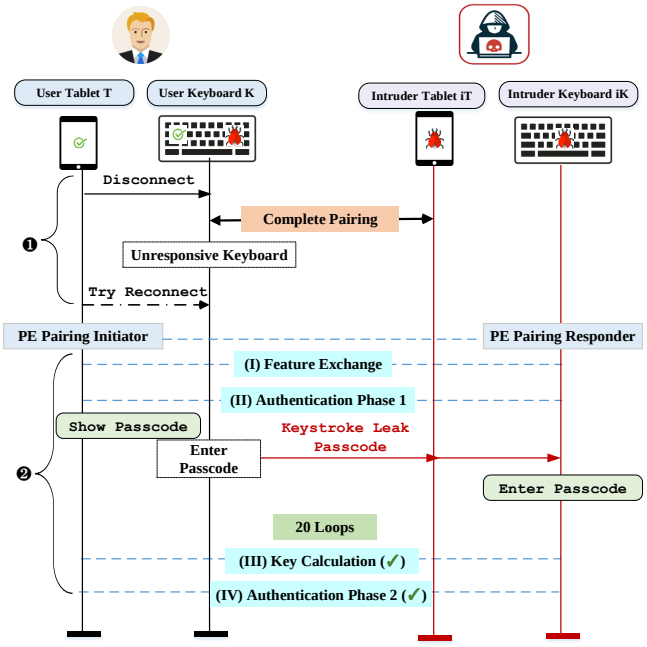

Fig. 8: Ghost attack via compromised peripheral device.

other hand, Bluetooth devices connect to the wireless network. Here, the involvement of the human interaction requires both devices to be present within the purview of a user. Surprisingly, even in this case, an active MitM of the intruder can synchronize two simultaneous double impersonated sessions in such a way that visual responses among the user devices occur in near real-time. The active MitM attack remains "hardly-detectable" for the user. For instance, in the confusion attack, the checkpoint where the PE pairing enters the passcode, the user is not aware that the keystrokes are actually being sent to the intruder. We denote such flaws as *Ghost keystroke*.

To further explore the fundamental weakness, our model allowed the entered PE passcode to leak to intruders. To our surprise, Tamarin's execution produced an authentication failure trace, as illustrated in Figure 7. This trace conveys two interesting clues for a feasible attack. These clues are: (i) both active MitM sessions should use PE paring; (ii) the intruder should be able to replay the passcode from one session to another made possible by the *Ghost Keystroke* flaw. With these two clues, we were able to extrapolate the idea to two attacks. In the explanation of the two new attacks below, the use of the pronouns *he* and *she* are frequently used to denote a victim user and the intruder, respectively.

A5 - (I): Ghost Attacks via Compromised Peripheral. This type of attack works against the PE pairing method and assumes that the user has a compromised peripheral (e.g., keyboard). To be more specific, the attack involves user peripheral K; user central T; intruder peripheral iK; and intruder central iT as shown in Figure 8. The attack can be carried out in the following steps. The intruder disconnects the existing user device connections and manages to complete a pairing with the user peripheral K to obtain the user's keystroke (❶). This can be achieved in many ways. Assume that the peripheral device is a keyboard. When the user is away from his keyboard, the intruder can physically access the user's keyboard for a short time and completes a full pairing connection between the user's keyboard and an intruder central (as assumed in [60]). As such, the keyboard will send all the user's keystrokes to the intruder central. Next, when the user observes an unresponsive keyboard K and tries to reconnect it with his central T, the intruder pairs with the user's central T using a separate PE pairing through iK. Here, the user thinks that the re-pairing occurs among his two legitimate devices T and K. Therefore, he enters the PE passcode displayed on the peripheral device K (❷). Since the peripheral device K is compromised by the intruder, all keystrokes from the peripheral device will be logged at the intruder's central iT. Therefore, the intruder enters the same passcode in iK and completes the full pairing with the user central T. At this point, the intruder can obtain sensitive user data through the established pairing.

A5 - (II): Ghost Attacks via Compromised Central. This type of attack works against the PE pairing method and assumes that the user has a compromised central (e.g., a tablet). It involves user peripheral L; user central T; intruder peripheral iL; and intruder central iT as shown in Figure 9. The attack can be carried out in the following steps: The intruder initiates connections with user's compromised central and peripheral devices in different sessions. When the user tries to pair his corresponding devices T and L, the intruder manages to obtain the displayed content (including the passcode) from the user central T (❶). For example, the user can install a malware on the user's tablet (as assumed in [33]) and can capture screenshots while pairing. As such, once the central T shows the passcode, the intruder obtains it through a separate Internet connection. Thereafter, she displays the same passcode through her central iT to connect to the user peripheral L. At this point, since the user thinks that the pairing occurs between his devices T and L, he enters the displayed passcode into his peripheral L (❷). At this moment, the intruder enters the same passcode in her peripheral iL to connect to the user's central T to complete the full pairing with the two user devices.

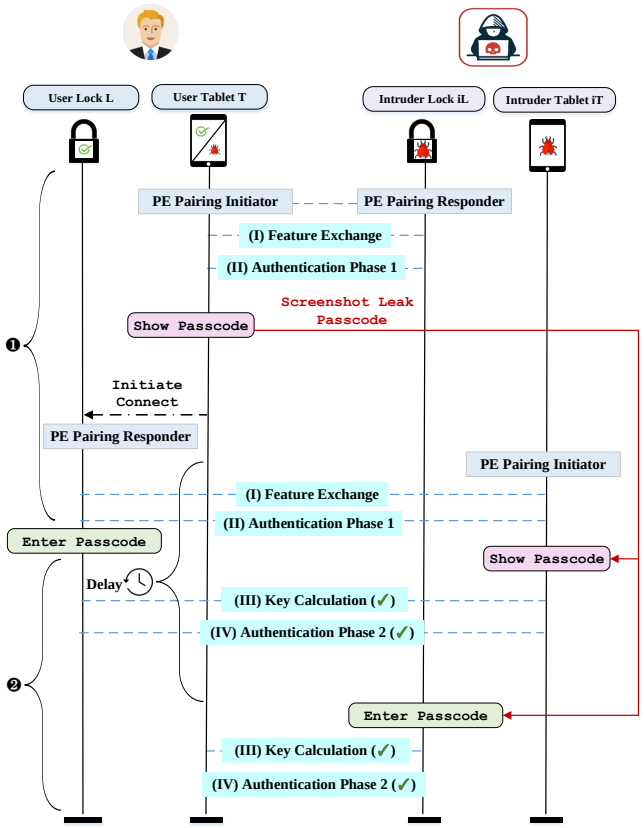

Fig. 9: Ghost attack via compromised central device.

Thereafter, the intruder can exploit the the paired devices connections to obtain the unintended user information.

PoC and Its Practicality. We have validated our two types of ghost attacks on our own devices. For ghost attacks via compromised peripherals, we used two TI CC2640 development boards to simulate the attack procedure. One board works as the central connecting to the victim's keyboard, and the other as the peripheral connecting to the victim tablet, as shown in Figure 8. We first assume that the intruder can physically access the victim keyboard. The keyboard uses the PE pairing protocol and the intruder can choose a passkey for the intruder's peripheral and enter it on the keyboard in a matter of seconds to connect the intruder's central to the victim keyboard. Therefore, the intruder's peripheral and intruder's central can then work as a relay to deploy the MitM attacks. For the ghost attacks via compromised centrals, we developed a Java-based malware that can take screenshots (`robot.createScreenCapture(.)`), and execute it on our own Windows 10 machine. The victim is again, a keyboard. The malware takes a screenshot every second and updates the screenshots to its back-end for reference. Once we capture a screenshot containing any passcode, we then quickly set up two TI CC2640 development boards to pair with the Windows 10 and keyboard correspondingly, which allows the two boards to launch MitM attacks on those two victims.

It can be observed that the practicability of those two attacks assumes the capabilities of compromising one of two pairing devices. This assumption is in fact practical and widely discussed in various Bluetooth attacks, e.g., Missing Bond attacks [33] and BadBluetooth [58] require malware to be installed on centrals to work, and downgrade attacks [60] require the attackers to briefly access the Bluetooth devices.

VII. Countermeasures

This section discusses the root causes and the fixes of all the attacks uncovered in this paper. Specifically, after applying the fixes to each of the five attacks, we derive a new patched model. In this model, we verify the effectiveness of the five fixes within the scope of the model assumptions.

Fix for Confusion Attack. The root cause of the confusion attack [53] is that the user cannot differentiate the UI prompts for the PE and NC passcodes. Our model fixes this error by humans actually confirming the paring method along with the passcode match. In Bluetooth devices, such errors can be mitigated by designing a distinguishable UI that is easy to detect or by incorporating an incompatible passcode value format for PE and NC. The latter case prevents an intruder from reusing the NC passcode to the PE pairing.

Fix for Reflection Attack. The root causes of the reflection attack is that the involved devices do not check if the received DH public key from the other devices is a reflected copy of the devices' own public key. As such, we fix the flaw by explicitly enforcing such check on each of the devices.

Fix for Static Passcode Attack. The root cause of the static passcode attack is that the devices allow static passcode, which can be fixed by enabling the random passcode.

Fix for Group Guessing Attack. The root cause of the group guess attack is that the devices use a random function that is not thread-safe. As such, we fix the model by enforcing the implementation to be thread-safe, where we use a random passcode for each of the pairing devices even if these devices initiated the pairing simultaneously.

Fix for Ghost Attack. The root causes of two ghost attacks are that the device or the user mistakenly leaks their passcode for PE pairing (i.e., the intruder can steal the passcode from an input device or an output device). As such, we fix the attack by disabling the capabilities of leaking passcode for the intruder. For instance, users can utilize the lockable input device, made capable by the manufacturing vendors and the OS support, to prevent brief physical access. Additionally, the screenshot feature can be disabled on the passcode display screen.

VIII. Discussion

Prevalence of PE Pairing. In our paper, we mainly focus on the PE pairing method. Therefore, we would like to understand the prevalence of the PE method to approximate the impact of our work. We believe that the PE pairing method is popular for two reasons. First, according to the Bluetooth SIG's reports [44], [46], nearly all smartphones, tablets, and PCs support Bluetooth LE, and all those BLE-enabled devices support PE. In the last decade 2013-2021, Bluetooth SIG have shipped a total of 18 billion BLE devices. Since all these devices have screens and keypad, they have to support PE pairing. In 2022, the number of global smartphone users is estimated at 6.6 billion [36], the number of PC users is around 2 billion [41], and the number of tablet users is 1.28 billion [4]. There could be up to 9.88 billion devices that

support PE. Second, PE is the only secure PE method for some of the devices, and those devices are very likely to use PE for their security. One type of such devices is keyboard. This is because PE only requires the devices to have a keyboard or a screen, while other secure pairing methods such as NC (which require the devices to have both keyboard and screen) or OOB (which requires the devices to have a channel other than Bluetooth such as NFC) will require extra hardware.

Lessons Learnt. We draw several insights from our modeling and attack-discovery process. First, the extrapolating over the attack traces derived from FV tools can uncover unforeseen attack vectors. Note that FV tools can reason about the possible state-spaces of the modeled protocol. When searching for traces of security property violations, the FV engine explores unbounded protocol interaction with the adversary's actions. Therefore, even for a single property violation, FV can provide multiple distinct traces showing different ways to exploit the same vulnerability. Similarly, different variations of the security lemmas provide more diversity in the attack traces. In this way, a manual investigation of the variety of traces can further throttle the vulnerability discovery. Second, for long iterations of loops in a protocol starting with minimal loops could potentially be a good trade-off to handle model complexity. Recall that each PE authentication loop iteration adds probabilistic hardening of the protocol security. Since Tamarin symbolic logic does not consider probabilistic intruder capabilities, modeling of 2 out of 20 loops in our model did not downgrade the intruder capabilities. Consequently, most of our attack traces revealed an attack pattern over 2 iterations of the PE authentication phase 1 loop that could be extrapolated over the original loop of 20 iterations. Finally, an in-depth understanding of the exact protocol behavior, threat model, and nature of the FV tool is crucial for the precise abstraction of the model. This understanding is crucial to differentiate the literal protocol accuracy and intruder behavior versus the semantic intruder capabilities. Modeling the latter case captures the precise attack surface.

Limitations. We are aware that our model is not perfect and has multiple limitations. First, we focus only on the pairing process and not on other stages such as feature exchange and re-connection, or related features such as ECDH group theory and OS APIs. Therefore, our model cannot detect attacks such as BadBluetooth [58], Misbonding attacks [33] and invalid curve attack [20]. Second, when we model the pairing protocol, we do not consider legacy pairing such as PIN based protocol, which are subject to various attacks. Third, our model does not match the literal accuracy of 20 iterations of PE pairing authentication phase 1. Instead, it only models the first two iterations. While this is a limitation, we believe that it at least establishes our effort as useful because of two reasons. First, the fact that once we fixed all attacks' flaws, the patched model verifies the confidentiality and authentication for two loops, and establishes the base case for the inductive proof towards 20 iterations. Consequently, this partial proof can be extended to establish the complete proof of 20 iterations. Second, the discovery of the attacks from our model with two iterations, with precise attack traces, demonstrates that attack patterns over two iterations are enough to identify the full attack pattern.

Vulnerability Disclosure. We have reported the new attack vectors to Bluetooth SIG. They informed us that the group guessing attack (A4) could be of concerns. In particular, Bluetooth SIG responded that simultaneous connections are supported in a few Bluetooth topologies. In those cases, thread latencies and implementation-specific checks on thread handling could make it difficult for the execution of the group guessing attack. Nevertheless, the attack could be of concern if an implementation is found that does not handle the implementation (e.g., non-thread-safe) carefully. For the ghost attacks (A5), Bluetooth SIG does not consider a compromised device threat model within their scope. However, as discussed in §IV, compromised device assumption has been considered in many previous research work [60], [33], [58], [57], [42]. These work show that compromising a user device can cause serious attacks. Additionally, we emphasize that both of the new attack vectors highlight important concerns for the development of the secure Bluetooth protocol.

Previous Work	PE Pairing Supported	Unbounded Sessions	Human Interactions	Human Errors	Comprised Devices	Identified Attacks	Verification Tools
Chang et al. [14]	✗	✓	✓	✗	✗	–	ProVerif [11]
Arai et al. [7]	✗	✓	✗	✗	✗	–	ProVerif [11]
Ngyyen et al. [35]	✗	✗	✗	✗	✗	–	Stand Spaces [25]
Cremers et al. [20]	✗	✓	✓	✗	✗	–	Tamarin [32]
Sethi et al. [42]	✗	✗	✓	✗	✓	–	ProVerif [11]
Wu et al. [56]	✗	✗	✗	✗	✗	–	ProVerif [11]
Wu et al. [57]	✓	✗ *	✓	✓	✓	A1, A2	ProVerif [11]
Our Model	✓	✓	✓	✓	✓	A1-A5	Tamarin [32]

*Only Data Transmission is unbounded

TABLE II: Comparison of our formal model with previous researches on Bluetooth. The column labels indicate the various features incorporated into the formal model.

IX. Related Work

Most of the previous efforts in formal verification of Bluetooth protocol cover NC pairing. As shown in Table II, Chang et al. [14] model the legacy NC pairing and derive attack based on the fact that the displayed hashed passcodes are not bound to the session identifiers and hence can be exploited and misinterpreted for another concurrent session. Arai et al. [7] model an improved NC protocol proposed by [59] where instead of the user confirmation of the same displayed passcode, the user enters the PIN on each of the devices involved and confirms the PIN using additional commitment message exchange. Ngyyen et al. [35] model and formalized a separately proposed OOB pairing method as an extension to Stand Spaces [25]. Cremers et al. [20] detected cryptographic implementation flaws in which choosing DH parameters from small subgroups or invalid curves allows an easier derivation of DH share keys. It achieves this attack accuracy by detailed modeling of DH group theory in Tamarin. Sethi et al. [42] model the NC pairing based on semi-compromised devices, which allows a device to bind to an unintended (or intruder) device. BLESA paper [56] model the Bluetooth re-connection procedure and demonstrate spoofing attacks caused by missing implementation checks over responses to the authentication capability of other devices. Compared to these research efforts, our model focuses on PE pairing and discovers many attacks in one generic model.

Closest to our work is Wu et al. [57]. In addition to secure pairing PE and NC, it models breath of other pairing protocols

including BC, BLE, and Mesh protocols. In contrast, we model PE pairing in depth. Such depth can be observed by the fact that out of the five uncovered PE attacks in our paper Wu et al [57] uncovered only two PE attacks — the confusion attack and reflection attacks. The rest of the discovered attacks do not pertain to PE pairing. We believe it is due to the missing abstraction of passcode customization and fully unbounded sessions in their formal model. Specifically, Wu et al. [57] uncovered existing: MisBonding [33], Co-located [47], Bad-Bluetooth [58], BLESA [56], Method Confusion [53] and new: Mesh Provisionin [57] and Cross Stack Illegal Acces [57]. Whereas our model uncovers existing: Static Passcode [50], Reflection [15], Method Confusion [53] and new: Group Guessing Attack, and Ghost Attack. Furthermore, from a design perspective, Wu et al. [57] used a modular model design, whereas our model unifies all pairing components such as feature exchange, authentication, and data transmission into one model. Overall, our work can be seen as complementing their work by going in-depth on modeling design.

X. Conclusion

This paper presents a detailed formal analysis of Bluetooth PE pairing. In the process, we rediscover three existing attacks and uncover two new attacks. In addition, we provide a patched model with fixes for all those attacks where the authentication and confidentiality properties of the model are verified. Our model approach indicates that modeling precise protocol behavior, entity interactions, and a refined and accurate intruder's capability could allow discovery of large classes of attacks. We believe that the insights presented through our research process and the uncovered attacks will provide valuable feedback to improve the state-of-the-art verification techniques and new modeling techniques that could be standardized into mainstream formal verification.

Acknowledgements

We would like to thank Cas Cremers, Jannik Dreier and Ralf Sasse for Tamarin related discussion on Google groups and anonymous reviewers for their invaluable comments. This research was supported in part by NSF awards 1834213 and 2112471. Any opinions, findings, conclusions, or recommendations expressed are those of the authors and not necessarily of the NSF.

References

[1] "Improving Tamarin Graph Output," 2015, retrieved September 5, 2022 from https://github.com/tamarin-prover/tamarin-prover/blob/develop/misc/cleandot/README.md.

[2] "Tamarin bit-split Theory discussion," 2021, retrieved October 19, 2022 from https://groups.google.com/g/tamarin-prover/c/6pr-mudKmwY/m/Pw-1PzUjCgAJ.

[3] "Tamarin Manual," 2021, retrieved January 18, 2021 from https://tamarin-prover.github.io/manual/book/001_introduction.html.

[4] amazon, "How many tablets are there in the world?" 2019, https://alexaanswers.amazon.com/question/6gxbVYtvaFGnlB82046wAG Accessed: 2021-06-30.

[5] D. Antonioli, N. O. Tippenhauer, and K. Rasmussen, "Bias: bluetooth impersonation attacks," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 549–562.

[6] D. Antonioli, N. O. Tippenhauer, and K. B. Rasmussen, "The {KNOB} is broken: Exploiting low entropy in the encryption key negotiation of bluetooth br/edr," in *28th {USENIX} Security Symposium ({USENIX} Security 19)*, 2019, pp. 1047–1061.

[7] K. Arai and T. Kaneko, "Formal verification of improved numeric comparison protocol for secure simple paring in bluetooth using proverif," in *Proceedings of the International Conference on Security and Management (SAM)*. The Steering Committee of The World Congress in Computer Science, Computer . . . , 2014, p. 1.

[8] M. Baert, J. Rossey, A. Shahid, and J. Hoebeke, "The bluetooth mesh standard: An overview and experimental evaluation," *Sensors*, vol. 18, no. 8, p. 2409, 2018.

[9] D. Basin, J. Dreier, L. Hirschi, S. Radomirovic, R. Sasse, and V. Stettler, "A formal analysis of 5g authentication," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2018, p. 1383–1396. [Online]. Available: https://doi.org/10.1145/3243734.3243846

[10] K. Bhargavan, B. Blanchet, and N. Kobeissi, "Verified models and reference implementations for the TLS 1.3 standard candidate," in *2017 IEEE Symposium on Security and Privacy*. Los Alamitos, CA, USA: IEEE Computer Society, may 2017, pp. 483–502. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SP.2017.26

[11] B. Blanchet, "Modeling and verifying security protocols with the applied pi calculus and proverif," *Foundations and Trends® in Privacy and Security*, vol. 1, no. 1-2, pp. 1–135, 2016.

[12] S. Bluetooth, "Bluetooth market update," 2021, https://www.bluetooth.com/wp-content/uploads/2021/01/2021-Bluetooth_Market_Update.pdf Accessed: 2021-06-30.

[13] C. Boyd, "Security architectures using formal methods," *IEEE Journal on Selected Areas in Communications*, vol. 11, no. 5, pp. 694–701, 1993.

[14] R. Chang and V. Shmatikov, "Formal analysis of authentication in bluetooth device pairing," *Fcs-arspa07*, vol. 45, 2007.

[15] T. Claverie and J. L. Esteves, "Bluemirror: Reflections on bluetooth pairing and provisioning protocols," in *2021 IEEE Security and Privacy Workshops (SPW)*, 2021, pp. 339–351.

[16] K. Cohn-Gordon, C. Cremers, B. Dowling, L. Garratt, and D. Stebila, "A formal security analysis of the signal messaging protocol," *Journal of Cryptology*, vol. 33, no. 4, pp. 1914–1983, 2020.

[17] CPPReference, "Pseudo-Random Number Generation: rand() CPP Reference," 2021, retrieved Nov 5, 2021 from https://en.cppreference.com/w/c/numeric/random/rand.

[18] C. Cremers, J. Fairoze, B. Kiesl, and A. Naska, "Clone detection in secure messaging: Improving post-compromise security in practice," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '20. Association for Computing Machinery, 2020, p. 1481–1495. [Online]. Available: https://doi.org/10.1145/3372297.3423354

[19] C. Cremers, M. Horvat, J. Hoyland, S. Scott, and T. van der Merwe, "A comprehensive symbolic analysis of TLS 1.3," in *ACM SIGSAC Conference on Computer and Commuincations Security*. ACM, Oct. 2017, pp. 1773–1788.

[20] C. Cremers and D. Jackson, "Prime, order please! revisiting small subgroup and invalid curve attacks on protocols using diffie-hellman," in *2019 IEEE 32nd Computer Security Foundations Symposium (CSF)*. IEEE, 2019, pp. 78–7815.

[21] C. Cremers, B. Kiesl, and N. Medinger, "A formal analysis of IEEE 802.11's wpa2: Countering the kracks caused by cracking the counters," in *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Aug. 2020, pp. 1–17. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/cremers

[22] A. Delignat-Lavaud, C. Fournet, M. Kohlweiss, J. Protzenko, A. Rastogi, N. Swamy, S. Zanella-Beguelin, K. Bhargavan, J. Pan, and J. K. Zinzindohoue, "Implementing and proving the TLS 1.3 record layer," in *2017 IEEE Symposium on Security and Privacy*, 2017, pp. 463–482.

[23] W. Diffie, P. C. Van Oorschot, and M. J. Wiener, "Authentication and authenticated key exchanges," *Designs, Codes and cryptography*, vol. 2, no. 2, pp. 107–125, 1992.

[24] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198–208, 1983.

[25] F. J. T. Fábrega, J. C. Herzog, and J. D. Guttman, "Strand spaces: Why is a security protocol correct?" in *Proceedings. 1998 IEEE Symposium on Security and Privacy (Cat. No. 98CB36186)*. IEEE, 1998, pp. 160–171.

[26] G. Girol, L. Hirschi, R. Sasse, D. Jackson, C. Cremers, and D. Basin, "A spectral analysis of noise: A comprehensive, automated, formal analysis of diffie-hellman protocols," in *USENIX Security Symposium*, 2020.

[27] M. K. Jangid, G. Chen, Y. Zhang, and Z. Lin, "Towards formal verification of state continuity for enclave programs," in *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 573–590. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/jangid

[28] N. Kobeissi, K. Bhargavan, and B. Blanchet, "Automated verification for secure messaging protocols and their implementations: A symbolic and computational approach," in *IEEE European Symposium on Security and Privacy*, 2017, pp. 435–450. [Online]. Available: https://hal.inria.fr/hal-01575923/document

[29] N. Kobeissi, G. Nicolas, and K. Bhargavan, "Noise explorer: Fully automated modeling and verification for arbitrary noise protocols," in *2019 IEEE European Symposium on Security and Privacy*, 2019, pp. 356–370.

[30] G. Lowe, "A hierarchy of authentication specifications," in *Proceedings 10th Computer Security Foundations Workshop*, 1997, pp. 31–43.

[31] S. Meier, "Advancing automated security protocol verification," Ph.D. dissertation, ETH Zurich, 2013.

[32] S. Meier, B. Schmidt, C. Cremers, and D. Basin, "The tamarin prover for the symbolic analysis of security protocols," pp. 696–701, 2013. [Online]. Available: http://tamarin-prover.github.io/

[33] M. Naveed, X.-y. Zhou, S. Demetriou, X. Wang, and C. A. Gunter, "Inside job: Understanding and mitigating the threat of external device mis-binding on android." in *NDSS*, 2014.

[34] C. Neskey, "Are your passwords in the green?" *Benchmark for Bruteforce Speed*, 2022, https://www.hivesystems.io/blog/are-your-passwords-in-the-green Accessed: 2022-09-01.

[35] T. Nguyen and J. Leneutre, "Formal analysis of secure device pairing protocols," in *2014 IEEE 13th International Symposium on Network Computing and Applications*, 2014, pp. 291–295.

[36] oberlo.com, "How many people have smartphones in 2022?" 2022, https://www.oberlo.com/statistics/how-many-people-have-smartphones Accessed: 2021-06-30.

[37] R. C.-W. Phan and P. Mingard, "Analyzing the secure simple pairing in bluetooth v4. 0," *Wireless Personal Communications*, vol. 64, no. 4, pp. 719–737, 2012.

[38] M. Ryan, "Bluetooth: With low energy comes low security," in *7th {USENIX} Workshop on Offensive Technologies ({WOOT} 13)*, 2013.

[39] B. Schmidt, "Formal analysis of key exchange protocols and physical protocols," Ph.D. dissertation, ETH Zurich, 2012.

[40] B. Schmidt, S. Meier, C. Cremers, and D. Basin, "Automated analysis of diffie-hellman protocols and advanced security properties," in *2012 IEEE 25th Computer Security Foundations Symposium*. IEEE, 2012, pp. 78–94.

[41] scmo.com, "How many computers are there in the world?" 2019, https://www.scmo.net/faq/2019/8/9/how-many-compaters-is-there-in-the-world Accessed: 2021-06-30.

[42] M. Sethi, A. Peltonen, and T. Aura, "Misbinding attacks on secure device pairing and bootstrapping," in *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, 2019, pp. 453–464.

[43] B. SIG, "Bluetooth SIG, core specification version 4.2," *Specification of the Bluetooth System*, 2014, https://www.bluetooth.com/specifications/specs/core-specification-4-2/ Accessed: 2022-08-18.

[44] ——, "Bluetooth SIG, bluetooth market update," 2018, https://www.bluetooth.com/bluetooth-resources/2018-bluetooth-market-update/ Accessed: 2022-08-18.

[45] ——, "Bluetooth SIG, core specification version 5.2," *Specification of the Bluetooth System*, 2019, https://www.bluetooth.com/specifications/specs/core-specification-5-2/ Accessed: 2022-08-18.

[46] ——, "Bluetooth SIG, bluetooth market update," 2022, https://www.bluetooth.com/2022-market-update/ Accessed: 2022-08-18.

[47] P. Sivakumaran and J. Blasco, "A study of the feasibility of co-located app attacks against {BLE} and a large-scale analysis of the current application-layer security landscape," in *28th {USENIX} Security Symposium ({USENIX} Security 19)*, 2019, pp. 1–18.

[48] R. M. Smullyan, *First-order logic*. Courier Corporation, 1995.

[49] C. Staub, "A user interface for interactive security protocol design," B.S. thesis, Eidgenössische Technische Hochschule Zürich, Department of Computer Science, 2011.

[50] D.-Z. Sun, Y. Mu, and W. Susilo, "Man-in-the-middle attacks on secure simple pairing in bluetooth standard v5.0 and its countermeasure," *Personal Ubiquitous Comput.*, vol. 22, no. 1, p. 55–67, Feb. 2018. [Online]. Available: https://doi-org.proxy.lib.ohio-state.edu/10.1007/s00779-017-1081-6

[51] TheHackerNews, "New bluetooth vulnerability exposes billions of devices to hackers," 2021, https://thehackernews.com/2020/05/hacking-bluetooth-vulnerability.html.

[52] TI, "Bluetooth programming official guide," 2022, http://dev.ti.com/tirex/content/simplelink_cc2640r2_sdk_1_35_00_33/docs/ble5stack/ble_user_guide/html/ble-stack/gapbondmngr.html.

[53] M. von Tschirschnitz, L. Peuckert, F. Franzen, and J. Grossklags, "Method confusion attack on bluetooth pairing," *Under submission*, 2020.

[54] H. Wen, Z. Lin, and Y. Zhang, "Firmxray: Detecting bluetooth link layer vulnerabilities from bare-metal firmware," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020.

[55] T. Y. Woo and S. S. Lam, "A semantic model for authentication protocols," in *Proceedings 1993 IEEE Computer Society Symposium on Research in Security and Privacy*. IEEE, 1993, pp. 178–194.

[56] J. Wu, Y. Nan, V. Kumar, D. J. Tian, A. Bianchi, M. Payer, and D. Xu, "{BLESA}: Spoofing attacks against reconnections in bluetooth low energy," in *14th {USENIX} Workshop on Offensive Technologies ({WOOT} 20)*, 2020.

[57] J. Wu, R. Wu, D. Xu, D. J. Tian, and A. Bianchi, "Formal model-driven discovery of bluetooth protocol design vulnerabilities," in *IEEE European Symposium on Security and Privacy*, 2022.

[58] F. Xu, W. Diao, Z. Li, J. Chen, and K. Zhang, "Badbluetooth: Breaking android security mechanisms via malicious bluetooth peripherals." in *NDSS*, 2019.

[59] T.-C. Yeh, J.-R. Peng, S.-S. Wang, and J.-P. Hsu, "Securing bluetooth communications." *Int. J. Netw. Secur.*, vol. 14, no. 4, pp. 229–235, 2012.

[60] Y. Zhang, J. Weng, R. Dey, Y. Jin, Z. Lin, and X. Fu, "Breaking secure pairing of bluetooth low energy using downgrade attacks," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020, pp. 37–54.

[61] C. Zuo, H. Wen, Z. Lin, and Y. Zhang, "Automatic fingerprinting of vulnerable ble iot devices with static uuids from mobile apps," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019.

APPENDIX A
MODELING SECURITY PROPERTIES

Confidentiality and authenticity are standard properties of protocols. Confidentiality is defined as the property of the protocol that the protocol secrets are available only to the intended legitimate protocol entities. Whereas, authenticity is defined as the property of a protocol where all the protocol secrets originate and receive only at indented legitimate protocol entities (not from an unauthenticated intruder).

The queries in our model are largely a formulation of Lowe's hierarchy of authentication [30] and other well known literatures [23], [55]. The only difference is that we integrate the user component into the formulation, which is not exactly a protocol principal. Rather, the user owns the principal devices, and the user term also serves for device access control, human interaction, and error part together. In our queries, we tried to generalize the format of Lowe's agreement queries and fit the user component into it while maintaining the integrity of the user actions. Therefore, the following sections explain the mechanics behind extended symbolic formalization for authentication queries.

To build the confidentiality and authenticity lemmas in traditional formalization, two *Action* labels §II-B are designated after the authentication and key exchange (AKE) part of a protocol finishes. The format of the *Action* label is: $\textbf{Checkpoint}_{\mathbf{X}}(id_1, id_2, ..., var_1, var_2, ...)$ where id_i is the ID of i th entity engaged in the protocol as perceived by the entity X and var_i is one of protocol variables that are agreed between the entities of the protocol.

In Bluetooth, the entities are the user with ID U and his devices with IDs A (Intitator) ,B (Responder). The *Action* label $\textbf{Running}_{\mathbf{A}}(...)$ (i.e., sending data) and $\textbf{Commit}_{\mathbf{B}}(...)$ (i.e., receiving data) are tagged with the IDs of the entities participating in the protocol and the session variables. Sessions variables are chosen based on the protocol data that needs protection from the Intruders. To verify the confidentiality and authenticity provided by Bluetooth pairing protocols, we choose $data_A$ and $data_B$, generated at A and B respectively, as the protocol secrets, and key, which is the negotiated key in the pairing process, as the session data that should remain secret for all protocol executions.

Confidentiality. The lemma for confidentiality uses the *Action* label $\mathsf{Commit}_{\mathrm{B}}(...)$ as shown below:

$$\forall\ \textbf{Commit}_{\mathbf{B}}(U, A, B, data, Key)\ @i_B \wedge \neg(\exists\ \textbf{MakeIntruder}(U)@i_I) \Rightarrow \neg(\exists\ \mathbf{K}(data)@i_I)$$

With respect to the legitimate device B the lemma states that all $\mathsf{Commit}_{\mathrm{B}}$ instances occur without leaking data to the intruder. The legitimate entity is indicated by the action label: MakeIntruder (as described in **S1**). Overall, the encoding states that at any time, the legitimate device B, which belongs to a user U, receives encrypted data $data_A$ from the negotiated key key, and the received data are never obtained by the intruder.

This confidentiality of data is observed by the entity B. To complete the confidentiality property for the model, similar lemmas are introduced from the perspective of A that involve the session variables $data_A, data_B$ and key.

Authentication. The lemma for authenticity is made up of three constraints (CNs) as shown below:

/***** For all data received at the legitimate entity B *****/

1. $\forall\ \textbf{Commit}_{\mathbf{B}}(U, A, B, data, Key)\ @i_B \wedge \neg(\exists\ \textbf{MakeIntruder}(U)@i_I) \Rightarrow$

/***** the data should have been sent by a legitimate entity A *****/

/***** before receiving at B, from the device owned by user U *****/

2. $\exists\ \textbf{Running}_{\mathbf{A}}(U, A, B, data, Key)\ @\ i_A \wedge i_A < i_B$

/*****and no other entity should receive the same data or use the same key *****/

3. $\wedge \neg(\exists\ \textbf{Commit}_{\mathbf{B}}(U_x, A_x, B_x, data_x, Key)\ @i_x \wedge i_x \neq i_B)$
4. $\wedge \neg(\exists\ \textbf{Commit}_{\mathbf{B}}(U_x, A_x, B_x, data, Key_x)\ @i_x \wedge i_x \neq i_B)$

For all instances of $\mathsf{Commit}_{\mathrm{B}}$ in legitimate B, there exists a running instance of $\mathsf{Running}_{\mathrm{A}}$ before it, which is tagged with the same entities and session variables (**CN-I at line 1-2**); For all instances of $\mathsf{Commit}_{\mathrm{B}}$ at the legitimate B, there is no other instance of $\mathsf{Commit}_{\mathrm{B}}$ with the same data (**CN-II at line 3**) and; For all instances of $\mathsf{Commit}_{\mathrm{B}}$ at the legitimate B, there is no other instance of $\mathsf{Commit}_{\mathrm{B}}$ with the same key (**CN-III at line 4**); Combining all constraints together, the lemma requires that all instances of $\mathsf{Commit}_{\mathrm{B}}$ be unique and for each of its instances, there exists a $\mathsf{Running}_{\mathrm{A}}$ instance with the same entities and session variables before it. That is, for any unique received data instances on the legitimate device B, there exists a legitimate device A that sends the same data, which are encrypted by the same key, to it, and the two communicating devices belong to the same user.

We now further explain how this encoding implies authenticity. In particular, consider the cases where the above FOL logic fails. For all unique instances of $\mathsf{Commit}_{\mathrm{B}}$,

- **Failure of CN-I.** No instance of $\mathsf{Running}_{\mathrm{A}}$ exists before or after the $\mathsf{Commit}_{\mathrm{B}}$ instance: indicates that a legitimate device B receives data from an unknown source (the intruder); All instances of $\mathsf{Running}_{\mathrm{A}}$ occur after $\mathsf{Commit}_{\mathrm{B}}$, which also indicates that the first legitimate $\mathsf{Commit}_{\mathrm{B}}$ receives data from an unknown source.
- **Failure of CN-II.** Two or more instances of $\mathsf{Commit}_{\mathrm{B}}$ occur with the same data $data_B$: indicates that the intruder has managed to use the same data in two possibly distinct sessions with A and B (e.g., replay attacks).
- **Failure of CN-III.** Two or more instances of $\mathsf{Commit}_{\mathrm{B}}$ occur with the same key: indicates that the intruder has managed to use the same key in other sessions. At a high level, the session variables $data_B$ and key are derived from fresh terms. Therefore, in protocols, the execution among legitimate devices must always produce unique and distinct values of $data_B$ and key in each session.

In all the above failure cases, the legitimate B received data from an unintended entity: the intruder, and thus implies that from the perspective of B' the intended other device A is not authentic. Also note that in all failure cases occur

with respect to the Commit_B action. Here, the uniqueness of Running_A action is not implied. Therefore, the above lemma authenticates A for B but not the other way around. To complete the authentication of B for A, similar lemmas are queries with Running_B and Commit_A actions.

To optimize to model runtime, standard authentication lemmas can be reduced to lemmas with fewer constraints. For example, the standard authenticity lemmas in the Lowe's hierarchy of authentication [30] involve many sub-constraints. However, data-steal and data-inject lemmas involve fewer constraints as illustrated below:

/***** Data Steal from Entity A *****/

$$\forall\ \textbf{Running}_\textbf{A}(U_A, A, B, data, Key)\ @i_A \wedge \neg(\exists\ \textbf{MakeIntruder}(U_A)@i_I) \wedge \textbf{Commit}_\textbf{B}(U_B, A, B, data, Key)\ @\ i_B \Rightarrow \neg(\exists\ \textbf{MakeIntruder}(U_B)@i_I)$$

/***** Data Inject at Entity A *****/

$$\forall\ \textbf{Commit}_\textbf{A}(U_A, A, B, data, Key)\ @\ i_A \wedge \neg(\exists\ \textbf{MakeIntruder}(U_A)@i_I) \wedge \textbf{Running}_\textbf{B}(U_B, A, B, data, Key)\ @i_B \Rightarrow \neg(\exists\ \textbf{MakeIntruder}(U_B)@i_I)$$

Here, the $\textbf{Running}_\textbf{X}$ and $\textbf{Commit}_\textbf{X}$ action labels roughly correspond to sending and receiving the encrypted data actions at entity X in the protocol model. A failure in these variety of the constraint-reduced lemmas also reveals a possible practical flaw in the protocol. Most of the attacks in our model were derived as a result of the failure of these constraint-reduced lemmas. Additionally, once the constraint-reduced lemmas are proved within manageable time, it can be used to prove the standard lemmas faster.

APPENDIX B
SUPPLEMENTARY FIGURES AND CODE

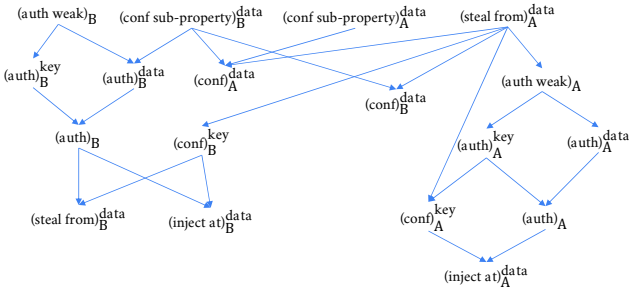

Fig. 10: Lemma reuse mapping graph. All the lemmas used in the verification of the patched model.

Tamarin Code 1: Equational theories and restrictions

```
functions: MAC/2, h/1, split1/1, split2/1, merge/2,
           dhs/1, dhp/1, dha/2, dhb/2

equations: merge(split1(v), split2(v)) = v,  // bitsplit
↪  equation theory
           dha(a, dhp(b)) = dhb(b, dhp(a)) // Diffie Hellman
           ↪  equation theory

           // dhp(x):  derive DH public key using DH private
           ↪  random parameter x

           // dha(): derive shared key at initiator device A
           // dhb(): derive shared key at responder device B

restriction equality: "All x y #i. Eq(x, y) @ i ==> x = y"
restriction inequality: "All x y #i. Neq(x, y) @ i ==> not(x
↪  = y)"

// PATCH: Static Passocde Attack and Group Guessing Attack
↪  -- enforce different PE passcode (Kpe) in each session
restriction random_kpe_in_each_session:
"
 (All Kpe #i #j.  KpeValue_A(Kpe)@i & KpeValue_A(Kpe)@j ==>
 ↪  #i=#j)
 & (All Kpe #i #j.  KpeValue_B(Kpe)@i & KpeValue_B(Kpe)@j
 ↪  ==> #i=#j)
"

// single rule instatiation is sufficient
restriction single_MakeIntruder_perUID_is_enough:
" All uid #i #j.  MakeIntruder(uid)@i & MakeIntruder(uid)@j
↪  ==> #i=#j"
```

Tamarin Code 2: Key Tamarin *Rules*

```
// Instantiate users IDs
rule GenUserIDs [color=#E5E8E8]:
[Fr(~u)]
--[GenUserIDs(~u)]->
[!User(~u)]


// Instantiate devices IDs
rule GenDeviceAddr [color=#E5E8E8]:
[!User(u), Fr(~addrX)]
--[GenDeviceAddr(u, ~addrX)]->
[!UserDeviceAddr(u, ~addrX),
 Out(~addrX) // Address are public and can be spoofed
 ]


// Device possession to Intruder is by leaking user ID to
↪  public channel
rule MakeIntruder [color=#CD5C5C]:
[!User(u)]
--[MakeIntruder(u)]->
[Out(<u>)]


// Generate random passcode for Show Passcode step
rule GenKpe [color=#E5E8E8]:
let keyOut = senc{~Kpe, 'SetPasscode'}u
in
[!User(u), Fr(~Kpe)]
--[GenKpe(u, ~Kpe)]->
[Out(keyOut)]
```

APPENDIX C
TAMARIN TRACE FOR METHOD CONFUSION ATTACK

The attacks discovered in our model are detailed and clean. As discussed in §VI, the attack traces generated by Tamarin are very large. Therefore, we illustrate the precision of the Tamarin-produced attack by providing a portion snapshot of the model trace in Figure 11.

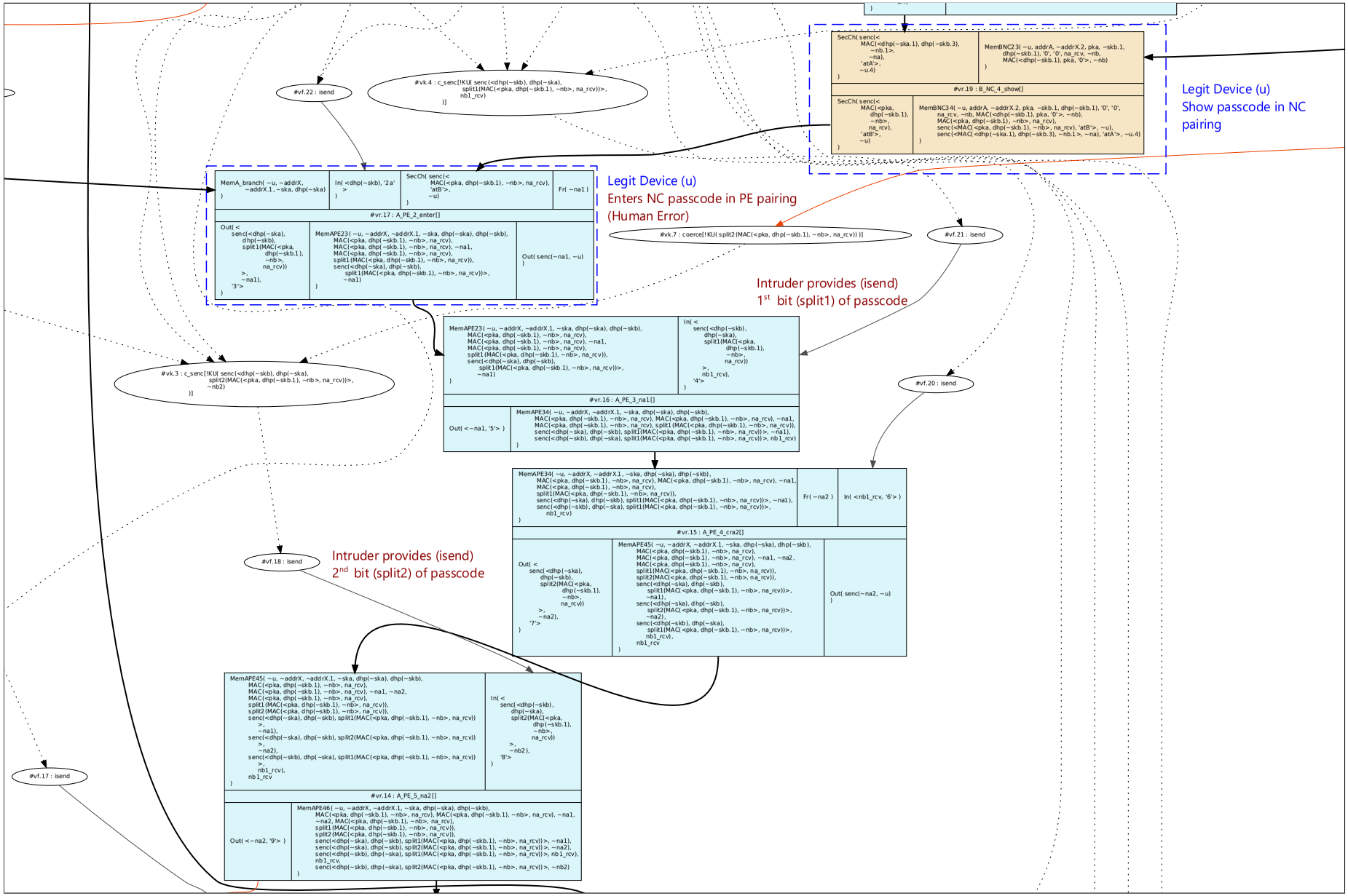

Fig. 11: A portion snapshots of Tamarin produced method confusion attack trace. The complete trace can be accessed in the provided Tamarin code folder.