Rediscovering Method Confusion in Proposed Security Fixes for Bluetooth

Maximilian von Tschirschnitz*, Ludwig Peuckert*, Moritz Buhl†, Jens Grossklags*
Technical University of Munich
*{firstname}.{lastname}@tum.de, †m.buhl@tum.de

***Abstract*—Previous works have shown that Bluetooth is susceptible to so-called Method Confusion attacks. These attacks manipulate devices into conducting conflicting key establishment methods, leading to compromised keys. An increasing amount of security-sensitive applications, like payment terminals, organizational asset tracking systems and conferencing technologies now rely on the availability of a technology like Bluetooth. It is thus an urgent goal to find and validate a mitigation to these attacks or to provide an appropriate replacement for Bluetooth without introducing additional requirements that exclude device or user groups. Despite recent solution proposals, existing threat models overlook certain attack vectors or dismiss important scenarios and consequently suffer under new variants of Method Confusion.**

We first propose an extended threat model that appreciates the root issue of Method Confusion and also considers multiple pairing attempts and one-sided pairings as security risks. Evaluating existing solution proposals with our threat model, we are able to detect known Method Confusion attacks, and identify new vulnerabilities in previous solution proposals. We demonstrate the viability of these attacks on real-world Bluetooth devices. We further discuss a novel solution approach offering enhanced security, while maintaining compatibility with existing hardware and Bluetooth user behavior. We conduct a formal security proof of our proposal and implement it on commonplace Bluetooth hardware, positioning it as the currently most promising update proposal for Bluetooth.

I. Introduction

In an era where devices become increasingly specialized and mobile, the demand for ad hoc connectivity has grown exponentially. Modern applications often rely on a network of wireless devices from various vendors, replacing traditional hardwired setups. For instance, cash registers are increasingly replaced by simple wireless card readers paired with tablets. In organizations, many more application scenarios in the context of logistics, supply chain, manufacturing and transportation have emerged. Likewise, with the trend towards mobile working and home office the organizational device landscape has substantially broadened.

While new application contexts increase flexibility and trigger innovative uses, they also raise concerns. Applications transfer sensitive data over and expose device interfaces to the wireless medium, leading to strict requirements for secure communication and authentication between those different devices. Meanwhile, threats have become more surgical and persistent, in particular, in the organizational context.

Recognizing this challenge, the need for secure connectivity frameworks, devoid of centralized control, has never been greater. This issue of decentralized ad hoc key establishment is known as the *Ad Hoc Pairing Problem*, posing the challenge on how one may establish a secure key between two devices that share no prior information or root of trust.

Bluetooth is – with an estimated 5.4 billion Bluetooth [1] devices shipped in 2023 – the de facto standard for applications that require ad hoc pairing. We attribute this to its ease of use and support of a large range of device types, even those with limited interfaces. Addressing security, Bluetooth leverages an Out-of-Band (OOB) channel, usually drawing on some form of user interaction, to exchange a small amount of data to facilitate secure key establishment.

Recently, a series of attacks on Bluetooth were uncovered that are collectively known under the umbrella term *Method Confusions (MCs)* [2], [3]. These attacks allow an adversary to Man-in-the-Middle (MitM) a pairing by leading the pairing entities into conducting different pairing protocols, which causes them to mishandle OOB data and eventually accept compromised keys. The potential for MC was overlooked during the design of Bluetooth since it was common practice to analyze pairing protocols in an isolated fashion instead of considering their interaction with other available protocols in the same standard. These attacks thus differ from previous threat scenarios, as they point out a conceptual issue in the design of ad hoc connectivity frameworks. Due to the wide reliance on Bluetooth, it is an urgent goal of the stakeholder research community to develop a fix or replacement to Bluetooth that remains easy to use and does not limit device compatibility, but does not suffer from any kind of MCs. In particular, in the organizational context, leaving the task to recognize a targeted threat such as MC to employees, just serves to further aggravate the burden of the so-called "weakest link" and is unlikely to succeed as prior user studies have shown [2].

Recent work has deepened our understanding of specific MC attacks in the Bluetooth context [3], [4] using amended threat models. Shi et al. [4] also proposed a patch for Bluetooth that aims to mitigate the MC-based attacks that were known at this point. However, despite the urgent need, before deploying any proposal to fix MC into the Bluetooth standard, it must

Network and Distributed System Security (NDSS) Symposium 2025
24-28 February 2025, San Diego, CA, USA
ISBN 979-8-9894372-8-3
https://dx.doi.org/10.14722/ndss.2025.240310
www.ndss-symposium.org

be evaluated carefully whether the issue has been addressed in its entirety.

Our paper demonstrates that while solving specific MC-related attacks, previous works failed to appreciate the greater principles of the MC problem space, and thus missed important attack vectors. For instance, the proposed models do not consider the possibility of an attacker initiating multiple pairings on one device. Furthermore, they do not acknowledge attacks where the attacker succeeds in establishing a connection to just one of the devices (i.e., one-sided pairing) as a threat. We do not see any reason why these scenarios and threats should be dismissed as they are practically feasible, relevant and thus must be expected from sophisticated threat actors in organizational settings.

This motivates our first contribution, the proposal of an extended and more realistic threat model that is based on modeling the fundamental problem behind MC. Drawing on this more holistic threat model, we formally identify multiple novel MC issues that persist despite the previous solution approaches, including the work of Shi et al. [4]. In particular, we uncover multiple new paths for an attacker to gain a MitM position. We demonstrate the feasibility of these new attacks by implementing and evaluating the proposal of Shi et al. [4] as well as our attacks on off-the-shelf Bluetooth hardware.

The shortcomings of existing solution approaches motivate our second contribution, i.e., the proposal of two pairing methods that can be used concurrently without risk of MC and would be a suitable replacement for Bluetooth without requiring new hardware or further user education. We further show that our proposal remains secure even under consideration of the extended threat model, exceeding the security guarantees of previous works in crucial points. Eventually, we provide an implementation of our proposal that runs on commonplace Bluetooth hardware. We hope that our contributions will motivate an improved design for upcoming Bluetooth iterations that plan to address the MC issue.

In summary, our contributions are as follows:

- We extend previous threat models by modeling the root cause of MC instead of focusing only on the known MC-based attacks.
- Using the new threat model, we show that the only previous proposal that even considers MC suffers multiple novel variants of MC attacks and demonstrate these attacks on Bluetooth hardware.
- We design a new connectivity framework that supports the same device and user pool as Bluetooth and reduces user interactions compared to previous proposals.
- We provide a formal proof for the security of this new connectivity framework under our extended threat model, which makes it currently the only secure ad hoc connectivity framework that is a suitable iteration for Bluetooth.
- By implementing our proposal for commonplace Bluetooth hardware, we showcase its practical feasibility.

II. Background

To set the stage, we begin by properly introducing the challenge of ad hoc pairing and explain the methods used by secure connectivity frameworks like Bluetooth, including needed cryptographic primitives. Subsequently, we provide a short overview over previous security models and proofs of existing pairing methods, as well as relevant attacks on them.

A. The Ad Hoc Pairing Problem

An increasing amount of modern applications are centered around the cooperation between physically separate device units. Most of these devices are sold separately by various vendors and are meant to be connected and disconnected an arbitrary number of times [5].

When establishing a secure communication key between such devices, the availability of a central authentication service is not guaranteed or raises privacy issues. Further, any centrally managed identity or ownership management promotes dependency on a single vendor leading to vendor lock-in and isolated ecosystems. Taken together, this results in a situation where these devices cannot rely on any common root of trust.

However, a user who brings such devices into contact can interact with them and support the pairing. Some protocol designs have the user exchange a small amount of information between the devices from which a Long-Term Key (LTK) could be derived (e.g., WPA Passphrase, Bluetooth Passkey). Other approaches require the user to bring the devices into a situation where (only) these devices can simultaneously observe an external signal like surrounding acoustic noise. In any of these scenarios, the user is tasked with creating a situation in which the devices have *exclusive* access to some secure communication channel (OOB channel [6]). In other words, the pairing protocol should assure that only devices that have access to this channel (as selected by the user) know the eventually derived shared key. This specific situation can be referred to as the *Ad Hoc Pairing Problem*. In the following, we will move away from the concept of physical devices and will instead refer to *entities* to allow for the possibility of virtual devices that may share the same physical hardware.

Since the interfaces of entities vary, different methods for ad hoc pairing exist. For instance, an entity that has a display, but offers no option for input, is supported by different pairing methods than a device that only has a keyboard but no output capabilities. Thus, a connectivity framework like Bluetooth must support the coexistence of multiple communication protocols and pairing methods to be widely applicable and useful, which, however, sets the stage for MC attacks.

B. Concept of Out of Band Communication

Next to the public insecure communication path, an OOB channel is a secondary connection between two entities that fulfills certain security properties. Each OOB is characterized by the entities that can access it. In the context of device pairing, we call those entities Legitimate Pairing Partners (LPPs) referring to the partners that were selected by the user. Two security properties of OOBs are relevant in this work.

Confidentiality: Only the LPPs may receive information from a confidential OOB channel. **Authentic OOB Channels**: Only LPPs may enter information into such channels.

Typically, OOB channels are considered to have low capacity and, therefore, cannot be used to directly exchange cryptographic key material. In the remainder of this work and in accordance with the Bluetooth specification and related proofs, we will assume authenticity for every mentioned OOB channel and will point out when confidentiality is additionally assumed. In the figures, we will mark authentic OOB communication as blue dotted arrows. Confidential and authentic OOB communication is highlighted with red dotted arrows.

C. Commitment Schemes

Many ad hoc pairing protocols utilize cryptographic commitment schemes. The concept of commitment schemes generally allows one party to *commit* to a value and publish a commitment without revealing the value itself. At a later point, this party can reveal the committed upon value, which can then be verified by others using the commitment. Commitment schemes generally strive to guarantee two possible properties [7]. **Binding**: After committing to a value, one cannot deviate and reveal a different value without failing the verification of the commitment. **Hiding**: By receiving a commitment, the receiver gains no information about the message. Furthermore, we can say that a commitment scheme is *computationally* Binding or *computationally* Hiding when the Hiding and Binding properties at least hold under the assumption of an adversary that is limited to performing any Probabilistic Polynomial Time (PPT) algorithm [7]. In the following, we will formalize this through a function H that can create the commitment $C = H(N)$ for a value N. This commitment can be shared. When N is revealed, $C = H(N)$ can be verified to assure that no deviation has occurred.

D. PAKE

A Password Authenticated Key Exchange (PAKE) is an interactive two-party or multi-party protocol [8]. To participate in the protocol, each party uses a short value, the password, as input. Through its execution, the same cryptographically strong key is yielded to each party that participated successfully. Parties that did not successfully participate gain no information about the value of that key. A party can only participate successfully in a protocol execution if it uses the same password as the others. Thus, the method assures that only parties that know (or guess) the password learn the derived shared key. Note that an attacker has only one chance to guess a password in this setting as he otherwise fails the protocol participation. Multiple PAKE protocols and implementations have been proven secure [9]–[11]. In this work, we will consider PAKE as a cryptographic primitive that yields a secure key to the parties that use the same password.

E. Bluetooth Pairing

In Bluetooth, LPPs exchange public keys and perform a Diffie-Hellman (DH) key exchange to establish a communication key. To protect this key establishment, Bluetooth offers to verify the exchanged public keys through a so-called *Authentication Stage* [12]. There are two protocols that can act in this stage which claim to protect the pairing from an active MitM attacker, Passkey Entry (PE) and Numeric Comparison (NC). Which of those protocols is used is decided by the capabilities of the devices. Note, those capabilities are advertised in an unencrypted and thus insecure *IOCapabilitiesExchange* at the start of the pairing and can be influenced by the attacker.

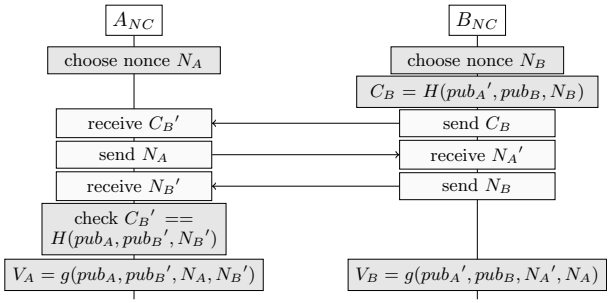


Fig. 1. NC fingerprint calculation phase.

In PE, the two sides of the protocol, the sender S_{PE} and receiver R_{PE}, establish a short secret through the OOB, the *Passkey*. Subsequently, S_{PE} and R_{PE} use the exchanged Passkey to authenticate the public keys through a round-based commitment scheme. This concept assures that the authentication cannot be intercepted by an attacker that does not know the Passkey. A more detailed explanation and diagrams of the protocol can be found in the extended Appendix [13].

In NC, the two sides of the protocol, A_{NC} and B_{NC}, leverage the OOB to compare fingerprints of the public keys. Note that different to the Passkey of PE this fingerprint is not secret. The user compares the fingerprints and confirms on both sides only if they are equal. Only upon confirmation, a device trusts the public keys. The phase calculating the fingerprint is shown in **Fig.** 1. First, A_{NC} and B_{NC} each choose nonces N_A and N_B. Then, B_{NC} publishes a commitment on N_B. When A_{NC} receives the commitment, it publishes N_A upon which B_{NC} reveals N_B, which is subsequently verified by A_{NC} using the commitment. The fingerprint is then calculated by applying a cryptographic hash function to the local and exchanged nonces $(N_A, N_B, N_A{'}, N_B{'})$ and public keys $(pub_A, pub_B, pub_A{'}, pub_B{'})$.

After the Authentication Stage, both sides conduct a so-called *LTK Validation* stage. This is essentially a challenge response exchange to verify that they have indeed derived the same key. We refer to the extended Appendix [13] for a detailed diagram. After successful completion of this stage, the pairing is considered complete, session keys are derived and the connection is encrypted.

III. Related Work

Early attempts of showing security for ad hoc pairing protocols [14]–[19] often disregard that in ad hoc scenarios the LPPs have no basis to authenticate which protocol the other side is actually performing in a pairing. Instead, these proofs

blindly assume that both sides would each correctly perform one side of the same protocol, thereby ignoring the possibility of MC. Most recently, automated proofs that consider MC were developed for the Bluetooth connectivity framework [4], [20], [21] and were able to confirm multiple specific attacks that relate to the MC issue. Note that our work targets the general problem of MC and not certain specific attacks.

A. Known Method Confusion Attacks

The issue of MC was primarily considered by our prior work [2], who show that an attacker is in principle able to trick two pairing Bluetooth devices into conducting different methods since the negotiation of the pairing protocol cannot be authenticated in an ad hoc setting. Furthermore, they argue that it is infeasible for the devices or the user to detect this confusion. They further show that when NC and PE are combined in a pairing the assumptions that usually provide MitM protection in these modes do not hold anymore. Thus, an attacker is able to gain a MitM position between the unsuspecting devices. Follow-up research by Claverie et al. [3] points out that Bluetooth also supports two legacy modes for reasons of backwards compatibility to old devices. As previous works [22], [23] have shown, these legacy methods leak every value they receive or send on the OOB. Claverie et al. demonstrate that when these legacy modes are confused in a pairing with NC or PE, the security guarantees that ordinarily should provide MitM protection are not fulfilled anymore providing the attacker the opportunity to become MitM.

B. Solution Attempts Addressing MC

Since the discovery of MC attacks, there was no connectivity framework available that could be considered secure while supporting a similar device diversity and user landscape as Bluetooth. While in our prior work which discovered MC [2], we proposed initial ideas on how the attack could be mitigated, a detailed analysis of these mitigations by Shi et al. [4] found that those suggestions either required additional hardware functionality, severely reduced device connectivity or added very strong assumptions about user behavior and interfaces.

Shi et al. [4] then made a proposal to extend Bluetooth by the pairing methods Patched NC (PNC) and Patched PE (PPE), thereby creating the first connectivity framework that considered MC and did not increase hardware requirements or reduced connectivity in comparison to Bluetooth. At the start of their execution, those pairing protocols behave identically to NC and PE, but then require another user interaction that aims to detect MC. In the following, we will refer to the protocol sides as A_{PNC} and B_{PNC} for PNC and S_{PPE} and R_{PPE} for PPE, respectively.

As shown in **Fig.** 2, in the second user interaction stage of PPE each pairing partner calculates a 6-digit hash over the Passkey s and messages exchanged earlier to determine the conducted pairing protocol (req, rsp). After the value is transferred by the user, this allows R_{PPE} to verify their LPP is indeed running S_{PPE}.

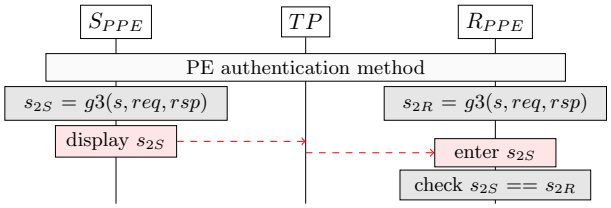


Fig. 2. PPE Method.

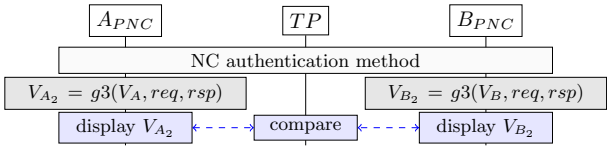


Fig. 3. PNC Method.

Similarly, in the second user interaction stage of PNC (**Fig.** 3), each pairing partner calculates a 6-digit hash over the previously displayed values ($V_{A/B}$) and protocol determining messages (req, rsp). Subsequently, both devices wait for confirmation. Note, that this requires the user to interact twice with each device, opposite to Bluetooth where just one interaction is required, increasing opportunity for user errors. Further note, that if an A_{PNC} or B_{PNC} protocol side is confused with a R_{PPE} protocol side, their model states that the user must always confirm on the PNC side. Their security analysis disregards the threat of such single-sided pairings and is instead only concerned with full MitM compromises.

C. Problems in Related Work Addressing MC

Despite these advances to accommodate the MC issue, the threat models of previous works still overlook an important aspect of ad hoc pairing. All current models assume that an attacker can only confuse two pairings with each other in order to gain a MitM position. Practically though, there is no reason why an attacker could not initiate multiple pairing executions on one entity during an attack. This consideration offers malicious parties a whole variety of new attack paths. In fact, we later demonstrate how this oversight already has serious implications for proposed mitigations of Shi et al. by describing two novel MitM attacks on their protocol. To prevent those oversights in the future and to lay a foundation for this demonstration, we propose a new threat model that takes a broader view on the issue of Ad Hoc Pairing and shows a new way to structure security proofs for Ad Hoc Pairing.

D. General Bluetooth Mitigations

Recent works also suggested general mitigation strategies against Bluetooth attacks. While none of these apply to the case of MC in ad hoc pairing scenarios, we still provide them for purpose of contextual overview. Prior work [24] suggested to introduce a secondary security layer to Bluetooth that relies on a global Public Key Infrastructure (PKI) and a universal and independent certification authority to verify device identities

during or after the pairing. However, this assumes the existence of such infrastructure, which moves beyond the ad hoc pairing context and addresses a different problem space (*cf.* discussion in II-A). Others [25] proposed a stateful monitoring layer to detect 'suspicious' sequences of Bluetooth packets. This does not apply to the issue of MC since there are no indicators for the isolated party that an attack is occurring. We believe, this is why their work does not consider MC in its analysis (see Table 3; [25]).

IV. Ad Hoc Ecosystem Threat Model

We first define the concept of *Ad Hoc Ecosystems*. In an Ad Hoc Ecosystem, the possible behavior of each entity is clearly defined, multiple executions as well as MC are considered, and OOB communication is modeled in a unified and standardized way. We will introduce the core concepts of Ad Hoc Ecosystems and then demonstrate how our approach can help to drastically reduce the required amount of cases that have to be considered in a security analysis. Consequently, this approach allows us and fellow researchers to develop succinct formal security proofs for connectivity frameworks.

First, we say that LPPs that could encounter each other in ad hoc pairing are all considered to be part of the same Ad Hoc Ecosystem. Every possible behavior of these partners (e.g., which protocols they may perform) forms the *Behavioral Set* of this Ad Hoc Ecosystem. Now, every entity in the Ad Hoc Ecosystem must expect their pairing partner to show any combination of behavior that is defined in the Behavioral Set of their Ad Hoc Ecosystem.

The behavior of a LPP is naturally defined by its supported Pairing Protocols. Each Pairing Protocol defines two roles, one for each LPP. We call the full execution of one of these roles a Role Execution (RE). The Behavioral Set is thus populated by the possible REs of the supported pairing methods. We consider REs as atomic and uninterruptible, i.e., once initiated they will run until they decide to terminate (e.g., through pairing failure).

A. Adversary

We model our adversary after the Dolev-Yao concept [26] in which the attacker cannot access or manipulate data or the program flow on the LPPs directly but can alter, delay and read all messages that are transported over unprotected channels (e.g., the public channel), thus potentially causing certain events to be triggered on the LPPs. We assume that the attacker is restricted to the use of PPT algorithms.

While we assume that no two REs can run concurrently on one entity, we must consider that an attacker could cause multiple REs to be performed in sequence. We will make the practical assumption that there is an upper limit to the REs that the attacker can initiate in sequence on an entity before a user becomes suspicious or weary of the pairing process, which we will consider as a terminal failure for the attacker. We call this upper limit the *Frustration Threshold*. In preparation to any security analysis, the Frustration Threshold needs to be chosen with common sense criteria depending on the use case in order to model the capabilities of the attacker. For instance, this value could be chosen based on how long a user is willing to wait for a pairing to complete, or how many OOB interactions the user is willing to perform during the pairing.

This adversary model exceeds the definitions of previous ones, which either did not consider MC at all [14], [15] or did not consider a sequence of REs [3], [4]. We will see that this caused in both cases an oversight of MCs.

B. Security Goal

The primary security goal of every RE is to establish an LTK, if and only if it can verify that this key is exclusively known to the LPPs. The attacker succeeds in breaking the *security of an RE*, if she causes an LTK to be established on the executing entity during the RE for which this does not hold. We say that an RE is secure, if this probability is lower than or equal to some *security factor*, which we denote as μ. In this work, we only consider an Ad Hoc Ecosystem secure to a factor μ if all REs in it are secure to a factor μ. Note, this goal exceeds the definitions of the threat model of Shi et al. [4] which only considers a pairing (and thus an RE) as compromised if the attacker compromises both LPPs. We argue against this restricted view, since even a single-sided pairing may allow an attacker to use the services of the paired device and to extract or insert information (e.g., cached payment information, phone contacts, health data). Possible security implications of single-sided pairing were already discussed in [2]. We further provide step-by-step descriptions of possible single-sided attack vectors in Appendix A-G. In conclusion, we think that a connectivity framework should avoid single-sided pairings by all means.

In sum, to show security for an RE in the Ad Hoc Ecosystem, we need to evaluate all possible stackings of all the possible REs of its Ad Hoc Ecosystem up to the considered Frustration Threshold on both LPPs in which this RE may occur. We then calculate the maximum probability of an attacker to establish a compromised LTK during this RE on the executing entity. The security of an Ad Hoc Ecosystem is then determined by the highest security factor of any of its REs.

C. Simplification of Security Analysis

There are a few practical assumptions which can be made to make it more feasible to perform security proofs in our threat model. We will now explain those assumptions and how they can be used to simplify a security analysis.

First, for our simplification to apply we must be able to assume that when an RE produces an LTK then it is kept secret. Secondly, while an RE may use static information that is inherent to the entity like UIDs or public keys, we assume that no RE reuses data that was left in local memory from previous REs. Note, that all REs used in this work fulfill these assumptions.

Now, consider a security analysis that is supposed to show the security of an RE τ on an entity X pairing with an entity Y (see Section IV-B). We just assumed that the LTK is

securely stored after the pairing. To show that the LTK cannot already be compromised during the RE itself, a security proof must be conducted for τ to determine an upper limit for its security factor μ. These proofs show that the LTK derived in some RE τ is secure if certain data handling rules were adhered to during the execution of the Pairing Protocol; for instance rules on which functions are applied to data or when data is published. Note, that these proofs may make further assumptions (e.g., computational limitations of the attacker), which we will not consider, since they are not concerning the behavior of the LPPs. As discussed, a pairing protocol execution is always distributed among multiple REs on X and Y. Thus, we can partition every RE security proof's data handling assumptions into two categories:

- **A1:** Assumptions about data handling performed by the analyzed RE τ itself (e.g., assuring nonces are uniformly random, nonces are not revealed before some event). Since X is performing τ locally and τ is strictly defined, these assumptions can directly be verified by analysis of the specification of τ.
- **A2:** Assumptions about data handling performed by other REs running on X or Y that interact with τ (e.g., displayed Passkey is a secret, numeric comparison value is a hash).

Furthermore, we can say that for any RE security proof any assumption of type A2 only ever concerns the handling of data that is exchanged between X and Y through OOB interactions. This is generally true, since the public channel can be entirely compromised by an attacker and no proof can ever make any assumptions on how this data has been or is going to be handled after being sent to or received from there. For instance, a proof may be based on the assumption that a received OOB value is the application of a specific hash function to a public key which was received on the public channel. A proof would never make any assumptions about data properties of that unauthentically received public key. A proof may make assumptions for a value to be kept secret by their LPP after it was sent to it over OOB, but would never assume the same for a value that it sent over the public channel. In summary, these proofs never make assumptions about data that is sent or received on the public channel by X or Y.

Since we assumed that no local data is reused between two REs, we can now see that a single RE's security is only affected by other REs if they interact with it (directly or indirectly) over OOB. To consider all relevant RE interactions during analysis of τ, we must consider all possible ways on how τ could be connected to other REs on X or Y through OOB interactions. During analysis of τ we then must make sure that all data handling assumptions of τ's proof are fulfilled in each of these 'puzzled-together' building blocks. Each block can be considered independently of any other REs. Note, that the blocks we want to consider must never exceed the Frustration Threshold on either LPP.

Fig. 4 shows a stack of REs between X and Y where each building block stands for a single RE. The taps on the pieces correlate to the sending of an OOB value whereas the indentations depict an RE that receives a value over the OOB. The REs are interlocking at the OOB interactions. The first piece of X shows multiple OOB interactions in one RE. The pieces are colored according to which REs are interacting with one another: we call this grouping a block. Another block is not allowed in this stack as it would pass the Frustration Threshold.

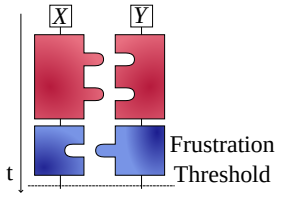


Fig. 4. Two stacked blocks of REs.

To determine the security of an RE in an Ecosystem, we can now consider all possible building blocks up to the Frustration Threshold that it may be part of. The worst possible value for an RE in any block is then the upper limit for the security of this RE. Note, that in most REs there is only a single OOB interaction, which in practice further severely limits the necessary steps for analysis.

D. Third Party

Many pairing protocols require involvement of a locally available third party to aid the pairing process. In most cases the third party is the user that initiates the pairing. We adapt the considerations that are commonly made in previous literature for this third party and include it in our model. Previous works [2]–[4] model the third party to deterministically follow a predefined set of actions. To aid the LPPs the third party can receive or forward OOB data between them, they can evaluate OOB data on their behalf and directly inject the result of this evaluation into the process of an LPP. We call the injection of this result *Feedback*. Note that the third party can only operate on data that is sent over the OOB and has otherwise no insights into the internal processes of the LPPs. While we adopt the assumption that third parties act as deterministic machines, it is worth pointing out that there has been prior work [27] which acknowledges that in practice third parties can make mistakes. This becomes increasingly likely the more data they have to process.

We also adapt the following possible third-party behaviors from the threat models of previous works [2]–[4]:

- **TP1:** If each LPP provides a value and a 'yes'/'no' option for Feedback, then the third party compares the values and feedbacks 'yes' on equality and 'no' otherwise.
- **TP2:** If one LPP provides a value and the other has an input interface, the third party forwards the value and provides, if possible, confirming feedback ('yes') to the displaying side.
- **TP3:** If both LPPs offer input, then the third party generates a random value and forwards it to both sides.

To include these possible behaviors into our building block model, we consider third-party actions to be building blocks that are automatically inserted if they would fit between two REs. Examples of blocks for TP1, TP2 and TP3 can be seen in Appendix A-E.

V. Method Confusion Detection in the Ad Hoc Ecosystem Threat Model

We now demonstrate that by using our Ad Hoc Ecosystem-based threat model it is straightforward to spot MC issues. We illustrate its application procedure by applying it first to the suite of Bluetooth pairing protocols that claim MitM protection. We will further show how our model allows us to discover two novel MC attacks in the connectivity framework proposed by Shi et al. [4]. This contribution demonstrates that there is currently no suitable mitigation for the vulnerable Bluetooth framework, despite an urgent need.

We will further argue that Bluetooth pairing cannot be fixed merely by extending it with additional pairing protocols. Instead, we propose novel pairing methods that are suitable to supersede the current Bluetooth pairing protocols, cleaning up with its current MC issues; all without increasing Bluetooth's requirements on hardware or user behavior. In the following, we will use our threat model to formally proof the security claims of that proposal.

A. Rediscovering Original Method Confusion

Let us now first consider the MitM protecting pairing protocols of Bluetooth as Ad Hoc Ecosystem. In this case, the Behavioral Set of the Ad Hoc Ecosystem would include: S_{PE}, R_{PE}, A_{NC}, and B_{NC} (*cf.* Section II-E). In our threat model a security proof for some RE must hold for every possible matching of OOB interactions to all other possible REs in the Behavioral Set.

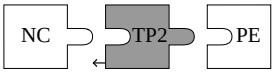


Fig. 5. Combination of NC and PE Role Executions.

Assume now we iterate through the possible matchings between the identified REs. Applying our concept of simplification (*cf.* Section IV-C), we must only consider matchings between REs where OOB data is exchanged. Since all REs have only one OOB interaction in this Behavioral Set we can simplify the analysis (*cf.* Section IV-C) to only $4^2 = 16$ RE pairings.

Eventually, we reach the case where R_{PE} is performed on one LPP, A_{NC} is performed on the other LPP, and their OOB interactions are matched through the TP2 third-party element (see Fig. 5). In this case, R_{PE} receives the display value from A_{NC} as s. Recall, that the security of R_{PE}, relies on the confidentiality of s (*cf.* Section II-E). In this case, however, s could be known by the attacker, since the fingerprint itself is not secret. An attacker could, accordingly, succeed in the round-based commitment scheme and R_{PE} would accept the attackers public key and the resulting DH key (*cf.* Sections III-A and II-E).

Now, we consider the reverse case where A_{NC} is matched with R_{PE} through TP2. In this case TP2 will always respond with confirmatory feedback to A_{NC} since it was able to enter the provided value into the R_{PE} RE. Following the specification of A_{NC}, no matter which value is displayed any public key would be accepted as authentic, and the respective DH key would be used as LTK. Since A_{NC} and B_{NC} are functionally equivalent, the analogous reasoning applies to B_{NC}. Consequently, no reasonable security factor can be proven for neither R_{PE}, A_{NC} or B_{NC} and thus the original Bluetooth Ad Hoc Ecosystem cannot be considered secure. This demonstrates that would our threat model have been applied to Bluetooth during its design phase, the issue of MC would not have gone unnoticed.

B. PNC with PE Method Confusion

Let us now apply our model to the suggested solution proposal of Shi et al. [4]. In this case the Bluetooth Ad Hoc Ecosystem from the previous analysis is extended by A_{PNC} and B_{PNC} from PNC and S_{PPE} and R_{PPE} from PPE. So let us now investigate their security by considering all possible OOB interaction matchings between A_{PNC}, B_{PNC}, S_{PPE} and R_{PPE} and all regular Bluetooth REs.

This includes the case of A_{PNC} being executed on one LPP and two instances of R_{PE} being subsequently executed on the other LPP. In this case each of the OOB interactions of A_{PNC} could be matched to each one of the R_{PE} REs through TP2 elements (see Fig. 6). Again, we can argue that TP2 will

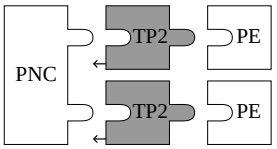


Fig. 6. Combination of PNC and two PE Role Executions.

always and in both interactions give confirming feedback to A_{PNC} no matter which value is displayed. Thus, the attacker can provide her own public key to the A_{PNC} execution and will always succeed in having the resulting DH key accepted as LTK.

We can further observe that each R_{PE} execution receives a value that is known to the attacker. Thus, both executions can be compromised following the reasoning from the just discussed original MC. Meaning, in the case of A_{PNC} matched with two R_{PE} executions both sides can be compromised and no security can be shown. Furthermore, no pairing failure will occur on either side. The attacker effectively gains a full MitM position. Since A_{PNC} and B_{PNC} are functionally equivalent the analogous reasoning applies to B_{PNC}. This is the first and most crucial attack on the work of Shi et al. [4].

C. PNC with PPE Method Confusion

One further possible pairing combination in the Shi et al. extended Ad Hoc Ecosystem is the combination of A_{PNC}

being executed on one LPP and R_{PPE} on the other partner. Both user interaction phases would then again be connected through TP2 elements (see Fig. 7).

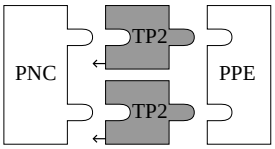


Fig. 7. Combination of PNC and PPE Role Executions.

The first user interaction phase would be equivalent to the original MC and would thus easily be completed by the attacker on both sides as described in the original MC. The second interaction stage would now display another value on the first partner for comparison while the second partner expects input. Again the third party would thus confirm the interaction on the entity running PNC. While the pairing would most likely fail on the PPE partner as it receives a value that does not match the expected hash (*cf.* Section III-B), a single-sided pairing would still be established (see implications discussed in Appendix A-G).

D. Conclusion of the Analysis

Throughout this analysis we notice that a third party will always confirm a comparison request when an input field is presented by R_{PE}. We conclude that no pairing protocol that relies on comparisons (e.g., NC, PNC) can be proven secure while R_{PE} is present in the Behavioral Set of an Ad Hoc Ecosystem. Furthermore, current Bluetooth implementations can be forced into a legacy mode to support pairings between old and new Bluetooth versions. The work of Claverie et al. [3] vividly demonstrated that an entity operating in such legacy mode leaks every value it receives or sends on the OOB to the attacker. So as we consider interaction with legacy mode devices in our threat model, we must also assume that secret Passkey-based methods like PPE would be insecure.

We thus deem it a futile effort to keep adding to Bluetooth's pool of pairing protocols in order to create a secure connectivity framework. Instead we suggest to establish an independent pool of pairing protocols that can be proven immune to MC and can operate on the same device combinations as Bluetooth. This new pool could then either be introduced as a new hardware- and user-compatible version of Bluetooth or be rolled out as completely independent framework. In either case, this would mean that every pairing between patched devices could be considered secure.

Later in this work we will thus propose two methods that are similar to PPE and PNC but require less user interaction and most importantly cannot be confused with each other and as such are able to coexist securely in one Ad Hoc Ecosystem.

VI. Attack Implementation and Evaluation

The work of [4] proposed PPE and PNC only on a theoretical level. To our knowledge, there is still no actual implementation of their proposal. We implemented and published [28] their proposals on actual Bluetooth devices and subsequently verify the attacks described in Sections V-B and V-C using this implementation (see Appendix B). This demonstrates that our attacks are practically feasible and provides a test bed for further research. Since PNC and PPE are proposed as patches of the NC and PE methods, we decided to adapt an open-source Bluetooth stack implementation according to the specifications made by [4]. We opted to use the well-established BTstack [29]. BTstack is implemented in C, supports a variety of hardware platforms, and is well documented. To equip the Bluetooth stack with PNC and PPE operation, we extended the state machine of the BTstack security manager, altering the ordinary operation of NC and PE. For details on the concrete modifications for PNC and PPE we refer to Appendix A-A. We subsequently implemented the MitM attacks described in Sections V-B and V-C using this modified stack. Details on this implementation can be found in Appendix A-B.

A. Practical Evaluation

In our test setup, the victim Initiator and Responder roles run on two Raspberry Pi 3Bs, each equipped with screens and keyboards for user interaction and *Cambridge Silicon Radio, Ltd., Bluetooth USB Dongles* for connectivity. A commonplace Linux Intel(R) Core(TM) i7-8550U laptop with two of the same dongles acts as attacker. The victims ran BTstack implementation samples, which we compiled with either the patched or unpatched BTstack, depending on the attack scenario.

After confirming the pairing functionality, we began to involve the MitM attacker. We launched victim Initiator and Responder instances and triggered the victim Initiator to establish a connection with the MitM's Responder. Previous work [2] has shown that this connection behavior can easily be provoked if the MitM advertises a deceptive name that imitates the true victim Responder while jamming the advertisements of the true victim Responder. Our artifacts contain references for such jamming tools.

When the connection was established, we conducted pairing with both victims as the user model prescribes. When a number was displayed on the victim Initiator while a number was requested for input on the victim Responder, we followed the user model in Section IV-D by transferring the number and confirming it on the displaying side.

In the case of the PNC to twice PE attack, this led both devices into a compromised connection with the MitM. In the case of the PNC to PPE attack, the pairing was only successful on the Initiator side, leading, as predicted, to a one-sided pairing. As such, we were able to verify that both our attacks are feasible in an end-to-end scenario with real-world devices.

VII. Proposal of Novel Pairing Ecosystem

First we introduce a new pairing method which is called the Extended Passkey Entry (XPE) method and is a functionally extended version of the already discussed PE scheme. This will be the first method in our new Ad Hoc Ecosystem.

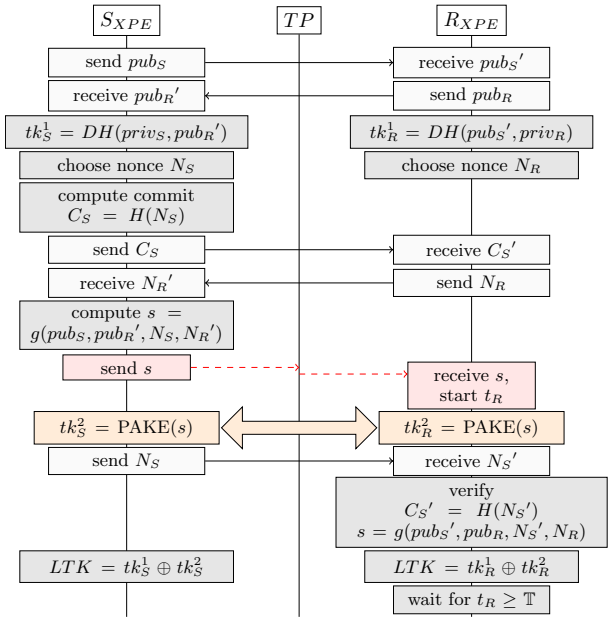


Fig. 8. XPE Method.

Secondly, we transform the general NC scheme into a new pairing method called Extended Numeric Comparison (XNC).

Note, that we will keep the exact choice for the cryptographic primitives and the exact choice for lengths of keys in principle open to the implementer. We will instead make commonplace assumptions for those primitives and show properties of our scheme in a general fashion (with dependency on the length of keys and nonces). It is then the task of the implementer to pick appropriate values and primitives.

a) Properties of g and H: In the following we will use a cryptographic hash function $g : \{0,1\}^m \rightarrow \{0,1\}^l$. In accordance with prior works we will make the assumption that g is a function that maps all distinct input values uniformly onto the output space. This follows the Random Oracle model for approximating cryptographic hash functions [30]. We further assume that the commitment function H fulfills the requirements of a commitment scheme (*cf.* Section II-C).

We will now proceed in explaining the functionality of the new methods. After defining the Behavioral Set of our Ad Hoc Ecosystem we will continue by arguing that any possible combination of these behaviors is secure against any PPT adversary. This will show that the new Ad Hoc Ecosystem consisting of *XNC* and *XPE* is secure even under MC. We will use a global constant $\mathbb{T}$ that will specify a timeout and is globally known to all devices in the Ad Hoc Ecosystem.

A. New XPE Method

In **Fig. 8** we see a diagram of the proposed XPE Method. We now describe the protocol run of XPE for both roles S_{XPE} and R_{XPE}. Like Bluetooth [12, Vol. 2B B.1.1], we assume availability of a local timer function on the R_{XPE} role entity, which is also a requirement of the Bluetooth specification. Before they begin, the S_{XPE} side verifies whether it is able to assume for the OOB to be confidential (recall that an OOB is always authentic in our model). How this decision is made depends heavily on the implementation. For instance, an implementer may trust a NFC connection or the third party to provide confidentiality.

The first interactive step of the protocol is to generate local asymmetric key pairs and to exchange the public key material between S_{XPE} and R_{XPE} (pub_S and pub_R). Note, that this exchange, in fact every exchange over the insecure channel, can be manipulated and eavesdropped. Consequently, we denote a value received over the public channel with V' for the sent value V.

After this public key exchange each side derives a primary temporary key (tk_S^1 and tk_R^1) by performing DH with the received public key and their own private key. Subsequently, they both choose fresh random nonces N_S and N_R. Now S_{XPE} computes a commit C_S on its nonce and publishes it. Only after receiving that commit, R_{XPE} replies with its own nonce and subsequently opens an input interface on the OOB channel to receive a value s. S_{XPE} now calculates and transmits the OOB value $s = g(pub_S, pub_R{}', N_S, N_R{}')$.

When R_{XPE} has received s from the OOB, a local timer t_R is started and R_{XPE} engages in a PAKE step with S_{XPE} using as password the just received s and as asymmetric key material $pub_S{}'$ as well as its own asymmetric key pair pub_R and $priv_R$. It is important that the RE of R_{XPE} does not terminate before a time span of constant $\mathbb{T}$ has passed since the receiving interface was closed, even if the pairing fails. Should R_{XPE} come to the point of opening the input interface, it then effectively assures to occupy the entity until $t_R \geq \mathbb{T}$, which means no other REs can happen during this time.

Simultaneously, after S_{XPE} has completed sending s it awaits an engagement in a PAKE and thus readily participates in the exchange with R_{XPE} using as password s. As discussed earlier, methods of PAKE assure that the participating parties only derive the same key if they prove knowledge about the same secret on the first attempt. They also notify the parties if the key establishment has failed (*cf.* Section II-D).

Consequently, if we assume no tampering or malfunction, S_{XPE} and R_{XPE} each now derive a secondary temporary key tk_S^2 and tk_R^2. S_{XPE} immediately calculates $tk_S^1 \oplus tk_S^2$ and accepts it as its new LTK. It then publishes N_S after which the RE of S_{XPE} terminates successfully. R_{XPE} receives $N_S{}'$ and verifies whether it matches the commit received earlier: $C_S{}' = H(N_S{}')$. If this is the case, R_{XPE} uses the remote $N_S{}'$ and $pub_S{}'$ and the local N_R and pub_R to calculate a comparison value and compare it with s. If this comparison succeeds, R_{XPE} considers $tk_R^1 \oplus tk_R^2$ as the new LTK. Note, an attacker would need to know both tks to derive the LTK.

In any case R_{XPE} waits until $t_R \geq \mathbb{T}$, upon which the RE on R_{XPE} terminates. Note, that secure communication is already possible when the LTK is established. Any remaining cryptographically relevant data is destroyed after the pairing has ended by either side. If any RE derives and accepts a LTK it is put in a secure key storage.

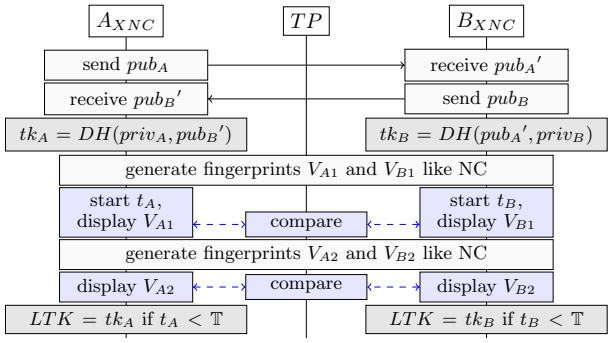


Fig. 9. XNC Method.

B. New XNC Method

We also introduce a new method called XNC to our Ad Hoc Ecosystem accommodating devices that offer no keyboard but just a display and a yes/no option. It is inspired by the proposal of PNC but extends it through timing guidelines which are crucial for the security of this Ad Hoc Ecosystem against MC. For that we assume local timers to be available on the entities. We call the Roles of this Pairing Protocol A_{XNC} and B_{XNC}. **Fig. 9** shows a diagram of the proposed XNC Method.

As in XPE, they first exchange public key material of their fresh local key pairs between A_{XNC} and B_{XNC} (pub_A and pub_B). Again we denote the values received over the public channel differently than their sent counterparts (e.g., $pub_A{}'$, $pub_B{}'$). Each side derives temporary keys, tk_A and tk_B respectively, from the received public and own private keys.

Equivalently to NC then both sides calculate and display a fingerprint as described in Section II-E and shown in **Fig.** 1 to authenticate $pub_A{}'$ and $pub_B{}'$ respectively. As soon as the OOB output interface is opened in that process, a timer is started on either side (t_A and t_B). If that authentication phase fails the pairing is failed and the RE terminates. If it succeeds, another fingerprint is calculated, again like in NC, and displayed for user confirmation, again with $pub_A{}'$ and $pub_B{}'$ as public keys. Should this second phase also succeed, the pairing is successful and the temporary keys tk_A and tk_B are accepted as LTKs. Should either timer t_A or t_B exceed the constant value of $\mathbb{T}$ before the pairing has succeeded the pairing is failed and the RE terminates.

Any remaining cryptographically relevant data is destroyed after the pairing has ended, either successfully or not, by either side. If any RE derives and accepts an LTK, it is put in a secure key storage.

VIII. Security of Novel Ad Hoc Ecosystem

We will prove for this new Ad Hoc Ecosystem that each RE in it is at least *secure* to some security factor μ. Note, that this exceeds the contributions of previous works which do not consider single sided pairing as an issue (see discussion Section IV-B). When analyzing a RE we will always refer to the entity it is executed on as X and to the current LPP of this entity as Y. We demonstrate that the probability of establishing a compromised LTK on X never exceeds μ for any RE. By showing that this μ holds for any RE of the Ad Hoc Ecosystem we show the entire Ecosystem to be secure to factor μ. We will proof security for factor $\mu = \frac{1}{2^{l-1}}$, where l is the bit-length of the OOB value, i.e., Shannon entropy of the OOB value.

We can observe two aspects for this Ad Hoc Ecosystem that will simplify this task. First, any RE in our Ad Hoc Ecosystem key material is stored securely. Second, no data from previous executions is reused on any entity. Thus, our simplification applies (*cf.* Section IV-C) and we must only consider the behavior of REs that are connected over OOB during analysis of one of these REs.

We refer to any data that is ever sent by X or Y either through the public channel or non-confidential OOB as PD_X or PD_Y, respectively. We refer to any data that is ever sent by X or Y over any OOB as O_X^{sent} or O_Y^{sent} respectively and all data ever received by X or Y over their OOB as O_X^{received}, O_Y^{received}, respectively. If there are multiple OOB interactions to be considered, we designate matched interactions with the same subscript (e.g., $O_{X_1}^{\text{sent}}$ and $O_{Y_1}^{\text{received}}$).

a) Role Execution Blocks to Consider: Before we begin, we want to consider how any REs of our Ad Hoc Ecosystem can be combined with each other to make sure we cover all possible blocks that an analyzed RE could occur in. In principle we would have to consider all possible blocks that could be composed of $A_{XNC}, B_{XNC}, S_{XPE}$ and R_{XPE} being executed on two LPPs X and Y. However, it is sufficient to analyze security for A_{NC}, B_{NC}, S_{XPE} and R_{XPE} and generalize the result to A_{XNC} and B_{XNC}. We now argue why this generalization is possible under the assumption one can additionally separately show the special case A_{XNC} and B_{XNC} being secure when connected to R_{XPE}. We will show how our analysis can in fact be simplified by two simple observations. We first notice, that in the upper as well as in the lower authentication phase of A_{XNC} and B_{XNC}, their OOB output value is calculated in the exact same fashion. In both cases the OOB data is generated by applying the NC authentication scheme on the same input data ($pub_A, pub_B, pub_A{}', pub_B{}'$ of the XNC scheme). In fact, for all data that is sent over the OOB from A_{XNC} or B_{XNC} we can assume the same handling as we would assume for any OOB value we would receive from any NC RE (A_{NC} or B_{NC}). If we show that a RE is secure when it is matched with OOB interaction from an A_{NC} or B_{NC} RE, we can also assume it to remain secure if it is matched with the OOB interactions of any A_{XNC} or B_{XNC}. In the following sections we will proceed to show that S_{XPE} and R_{XPE} are secure to some factor if their OOB is matched to A_{NC} or B_{NC}. This then also shows that any RE S_{XPE} and R_{XPE} that is matched with the OOB interaction of either phase of A_{XNC} or B_{XNC} is secure to the same factor no matter which other OOB interaction matchings exist in this block.

Furthermore, we observe that A_{XNC} and B_{XNC} only derive a LTK if both NC authentication phases succeed. Each phase will fail under the exact same conditions as NC would.

If one can show that a RE of NC (A_{NC} or B_{NC}) only continues with probability $\leq \mu$ when under attack, the same also applies for A_{XNC} and B_{XNC}. We will also proceed to show that A_{NC} and B_{NC} are secure to factor μ if matched to A_{NC}, B_{NC} or S_{XPE}. This then also shows that A_{XNC} and B_{XNC} are secure with the same factor when any of their authentication phase's OOB interaction is matched with A_{XNC}, B_{XNC} or S_{XPE}, no matter which other interactions occur in this block. Note, we cannot use this approach for the case of A_{XNC} or B_{XNC} matched with R_{XPE} since we are unable to show that A_{NC} or B_{NC} are secure with R_{XPE}. We will thus proof this case separately at the end of the following argumentation.

Until then our analysis only has to consider OOB interaction blocks consisting of A_{NC}, B_{NC}, S_{XPE} and R_{XPE} which each have only one OOB interaction and from which we can later transfer their positive results to A_{XNC} and B_{XNC}. This means that all independent blocks we must now consider have on either side only one RE. We will simply write $role(X)$ and $role(Y)$ in the following argumentation to express which (single) RE is considered on the X, respective Y side.

A. S Role of XPE

We will start by investigating the cases $role(X) = S_{XPE}$ and $role(Y) \in \{A_{NC}, B_{NC}, S_{XPE}, R_{XPE}\}$. We can observe for the behavior of the role S_{XPE} that the probability of an adversary successfully deriving a shared LTK with S_{XPE} depends on the knowledge of the attacker over both, tk_S^1 and tk_S^2. The likelihood of the attacker knowing tk_S^2 is upper bounded by the probability of guessing s correctly before the PAKE step of S_{XPE} has concluded. This limitation follows from the properties of the PAKE function (*cf.* Section II-D). Thus, we are further interested in how an adversary could narrow down their guess for s before the PAKE step on S_{XPE} expects the guess as input.

First, we recall that an entity in role S_{XPE} only communicates s over an OOB for which confidentiality can be assumed (*cf.* Section VII-A). The only other values that are revealed during the pairing and stand in relationship with s are N_S and C_S. N_S is uniformly chosen and is not revealed until the PAKE step has succeeded. C_S is computationally Hiding N_S through the properties of the commitment scheme. Further, it follows from the uniform distribution of g's mapping and the uniformity of N_S that to an entity which has no prior information about N_S every value for s is equally likely. So effectively, an adversary would receive no information from X with $role(X) = S_{XPE}$ that allows deriving s better than through random guessing, until after the PAKE step has concluded. Considering the properties of PAKE (*cf.* Section II-D) the attacker's probability of success is:

$$Pr[p(PD_S) = s] \leq \frac{1}{2^l}$$

where $p \in PPT$ any *Probabilistic Polynomial Time Machine*.

It remains for us to assess whether data that is sent to Y is kept secret. Primarily, we can observe that in the case of S_{XPE} matching the OOB interaction no information whatsoever is transferred to the S_{XPE} role. Conclusively, S_{XPE} is unable to leak any information about s in this case. In the case of A_{NC} and B_{NC} matching the OOB interaction the data is also not received by the entity itself but by the third party that aids the matching. The third party then compares the received value with a value provided by A_{NC} or B_{NC} and gives feedback about the equality. In those cases the only information that can be revealed through A_{NC} or B_{NC} is whether these values matched (continue pairing) or did not match (pairing failed). By aborting or not aborting, the A_{NC} and B_{NC} roles of NC provide a potential adversary just enough information to eliminate one value from the set of possible OOB values or, respectively, confirm one as the correct one. Similarly, the abortion behavior of the R_{XPE} role of the XPE method, after execution of PAKE allows to exclude or confirm exactly one value from the set of possible OOB values. This is possible, since the attacker can attempt the PAKE with the R_{XPE} role once for one value of their choice. Note, that if the PAKE step fails on Y the received OOB value is discarded and no further information can be gained about it. Consequently, the probability of an adversary correctly guessing the value $s = O_Y^{\text{received}}$ exclusively under the knowledge of data published by the respective receiving entity Y with $role(Y) \in \{A_{NC}, B_{NC}, S_{XPE}, R_{XPE}\}$ can be described as the probability of Y not aborting and thus the value of s being immediately known added to the probability of Y not aborting and thus just reducing the guessable space for s by one element:

$$\begin{aligned} &Pr[p(PD_Y) = s] \leq Pr[\text{“}Y \text{ did not abort”}] + \\ &Pr[\text{“}Y \text{ did abort”}] \cdot Pr[p(PD_Y) = s \mid \text{“}Y \text{ did abort”}] \\ &\qquad = \frac{1}{2^l} + \frac{2^l - 1}{2^l} \cdot \frac{1}{2^l - 1} = \frac{1}{2^{l-1}} \end{aligned}$$

where $p \in PPT$ any *Probabilistic Polynomial Time Machine*.

We can now conclude that the probability of guessing the value of s as input for the PAKE key derivation of S_{XPE} is always less than or equals to $\frac{1}{2^{l-1}}$ in our Ad Hoc Ecosystem. Furthermore, it follows from the properties of PAKE that for any PPT adversary that is unable to MitM the PAKE key establishment, the probability of deriving the established temporary key and thus the eventual LTK is negligible. Conclusively, for a security factor $\mu = \frac{1}{2^{l-1}}$ the $role(X) = S_{XPE}$ is secure.

B. A, B Role of NC and R Role of XPE

We will continue in the next section by investigating the cases $role(X) = \{A_{NC}, B_{NC}, R_{XPE}\}$ and $role(Y) \in \{A_{NC}, B_{NC}, S_{XPE}\}$. This deliberately leaves out the case $role(X) = R_{XPE}$ and $role(Y) = R_{XPE}$. We can observe though that in this case X will never complete pairing since it waits indefinitely to receive a value on the OOB. Thus we can already consider this combination as *secure* since it will never yield a valid pairing key. It further ignores the cases $role(X) = \{A_{NC}, B_{NC}\}$ and $role(Y) = R_{XPE}$ which we will instead address later by directly showing security

$role(X) = \{A_{XNC}, B_{XNC}\}$ and $role(Y) = R_{XPE}$ instead of taking a 'detour' over A_{NC} and B_{NC}.

1) General Properties of Remaining Combinations: We now discuss some properties that hold generally for the remaining pairing combinations of $role(X) \in \{A_{NC}, B_{NC}, R_{XPE}\}$ and $role(Y) \in \{A_{NC}, B_{NC}, S_{XPE}\}$ and then demonstrate how these properties can be used to argue security.

We observe that in all roles $(A_{NC}, B_{NC}, S_{XPE}, R_{XPE})$ the pairing partners (X and Y) pick nonces (N_X, N_Y). N_X and N_Y are always chosen fresh and uniformly at random . Further it holds that X and Y each decide for values of pub_X and pub_Y independently from any external influence. Furthermore, pub_X and pub_Y are not chosen in any dependency to N_X or N_Y and of course N_X and N_Y are also independent ($\perp\!\!\!\perp$) of each other:

$$\forall i, j \in \{pub_X, pub_Y, N_X, N_Y\}, i \neq j : i \perp\!\!\!\perp j \qquad (1)$$

Now, we also observe that for all roles $(A_{NC}, B_{NC}, S_{XPE}, R_{XPE})$ the 'local' nonce value of an entity is *unknown* to remote entities (so only X knows N_X and Y knows N_Y) before the remote nonce and remote public keys ($N_Y{}', pub_Y{}'$ resp. $N_X{}', pub_X{}'$) have been *fixed*. When we say a value is *unknown* to an entity, we express that no PPT entity could guess this value better than with a negligible advantage over ordinary random guessing.

The stated assumption holds for the roles S_{XPE} and B_{NC} since their local nonce is chosen uniformly at random and the only value released before the arrival of the remote public key and remote nonce that is related with the local nonce is a computationally Hiding (*cf.* Section II-C) commitment (C_X resp. C_Y).

For the roles A_{NC} and R_{XPE} it holds that their local nonce will not be revealed before a commit on the remote nonce ($C_Y{}'$ resp. $C_X{}'$) and ($pub_Y{}'$ resp. $pub_X{}'$) have been fixed. Note, that in those cases the local entity is verifying with H whether the later arriving remote nonce is a proper opening to the previously received commitment. The computational Binding property thus implies that when the commit on the remote nonce is fixed the remote nonce itself is also fixed.

Thus, we can say that for every still considered combination ($role(X) \in \{A_{NC}, B_{NC}, R_{XPE}\}$ and $role(Y) \in \{A_{NC}, B_{NC}, S_{XPE}\}$) it holds that

- $pub_X{}', N_X{}'$ must be fixed while N_Y unknown to any entity besides Y
- $pub_Y{}', N_Y{}'$ must be fixed while N_X unknown to any entity besides X

Further we can argue that an adversary who wants to pick $N_X{}'$ or $pub_X{}'$ while under knowledge over the exact value of N_X must first fix $N_Y{}'$ to gain knowledge over that N_X. But since he cannot know $N_Y{}'$ before fixing $N_X{}'$ he then has to fix $N_Y{}'$ while N_Y is unknown to him. Analogously, an adversary that would want to fix $pub_Y{}'$ or $N_Y{}'$ while under knowledge over the exact value of N_Y will have to fix $N_X{}'$ first and do so while the exact value of N_X is unknown to him. Thus we can say that the adversary can choose out of two different limitations (**L1** and **L2**) of knowledge when providing input to the legitimate parties:

- **L1:** Choose to pick $pub_X{}', pub_Y{}', N_X{}', N_Y{}'$ while N_X unknown.
- **L2:** Or, alternatively, pick $pub_X{}', pub_Y{}', N_X{}', N_Y{}'$ while N_Y unknown.

We will use this observation about the limitations of the adversary in a moment.

2) Defining the Successful Attack: For any remaining considered role assignment for X ($role(X) \in \{A_{NC}, B_{NC}, R_{XPE}\}$) and Y ($role(Y) \in \{A_{NC}, B_{NC}, S_{XPE}\}$) we can further observe that eventually Y calculates O_Y^{sent} by applying g to some arrangement of the values $\{pub_X{}', pub_Y, N_X{}', N_Y\}$. X receives this value either itself (in case of R_{XPE}) or it is received by the third party. In either case the received value is compared (by X itself or through help by the third party) with g applied to some arrangement of the values $\{pub_X, pub_Y{}', N_X, N_Y{}'\}$. If this comparison fails X will abort the pairing. We can abstract all possible role assignments by writing that the comparison that is executed here is in any case:

$$g(\sigma_X(\{pub_X, pub_Y{}', N_X, N_Y{}'\})) = g(\sigma_Y(\{pub_X{}', pub_Y, N_X{}', N_Y\})) \qquad (2)$$

where σ_X are all arrangements of parameters to g on X for $role(X) \in \{A_{NC}, B_{NC}, R_{XPE}\}$ and σ_Y are all arrangements of parameters to g on Y for $role(Y) \in \{A_{NC}, B_{NC}, S_{XPE}\}$. The adversary's goal is thus to make this comparison succeed and at the same time he must change at least some of the inputs to inject his own public keys into the pairing, since otherwise the properties of DH would make it infeasible to find the established temporary key ($tk_R^1, tk_{A_{NC}}, tk_{B_{NC}}$ for R_{XPE}, A_{NC} and B_{NC} respectively). This would prevent the attacker in any case from knowing the eventually derived LTK. As we just discussed, the adversary can take two distinct avenues when picking his inputs to the LPPs, and he must achieve that this selection of values fulfills (2). We now want to show that no matter which path he takes, the chances of success for a PPT adversary at this selection process are not greater than $\frac{1}{2^l}$.

3) Upper Bound for Likelihood of Adversary Success: As we discussed the adversary has to decide to have knowledge over either N_X or N_Y when picking any of the inputs to the legitimate entities. If limitation **L1** is picked we can write:

$$\alpha = \{pub_X, pub_Y{}', N_Y{}'\}, r = N_X, \beta = g(\sigma_Y(\{pub_X{}', pub_Y, N_X{}', N_Y\}))$$

If limitation **L2** is picked we can write:

$$\alpha = \{pub_X{}', pub_Y, N_X{}'\}, r = N_Y, \beta = g(\sigma_X(\{pub_X, pub_Y{}', N_X, N_Y{}'\}))$$

We will denote r in the following as bit string of length k: r = $\{0,1\}^k$. Note, that in any case, r is unknown to the

adversary when choosing any $'$-annotated variable. Further, we established earlier (*cf.* (1)) that the remaining variables (not adversary controlled) are also chosen independently of r.

Now, even if we assume a stronger adversary that would be able to pick the entire α and β, we show that he will not succeed to make the equation

$$g(\sigma(\alpha, r)) = \beta : \forall \sigma \in \{\sigma_X, \sigma_Y\} \tag{3}$$

hold more likely a negligible probability, if he has no precise knowledge over r when picking α and β. By showing this, we show that no matter the influence an attacker has on the choice of values for α or β, he is unable to succeed in having (2) fulfilled, no matter which knowledge limitation he picked as his strategy.

A first option by an adversary to choose α and β is to pick random values hoping a value for r will occur fulfilling (3). The probability of a uniformly randomly mapped function to map any random input value (α, r) to a specific output value is the inverse size of the output space, so the attacker's success likelihood with random choice ($\text{Success}_{\text{rand}}$) is

$$Pr[\text{Success}_{\text{rand}}] = \frac{1}{2^l}$$

Since the mapping of g is chosen probabilistically it is of course possible that in practice some output values get hit more often than the expected value would imply. So instead of picking random values for α and β it might be better for the adversary to look at different α values and try multiple possible values for r for those α. This way the adversary could discover that for a certain α more than the expected amount of possible r values hit the same output, which he could then choose. This attack approach is the equivalent to the birthday attack and the advantage gained by the attacker is in fact negligible in face of the effort necessary to find collisions. A detailed analysis of the attackers advantage can be found in Appendix A-F.

C. A and B of XNC versus R of XPE

This leaves us with the RE of A_{XNC} or B_{XNC} matched on any of their authentication phase's OOBs with R_{XPE}. As we have already shown security if A_{XNC} and B_{XNC} are matched in their first authentication phase to A_{XNC}, B_{XNC} or S_{XPE} we have to consider to either have this OOB to not interact with any RE at all, which would cause pairing failure or have it interact with R_{XPE}. This is where the timing component of our protocol proposals becomes important. We consider them to be matched through the TP2 third-party behavior since this is the only behavior that would match the two sides (*cf.* Section IV-D). We then observe that when the third-party feedback arrives at A_{XNC} or B_{XNC}, the R_{XPE} RE has already received their input and blocks its entity (Y) for at least time $\mathbb{T}$. There are no further OOB or third-party interactions possible during this time span. At the same time XNC enforces a timer (t_A and t_B) to start counting when the first value in the first authentication phase is displayed. We can see that those timers will inevitably exceed the time span T before the secondly displayed value can receive any feedback from the third party, and so the pairing on either A_{XNC} or B_{XNC} is aborted. Thus, a matching between A_{XNC} or B_{XNC} with R_{XPE} never establishes an LTK in this Ad Hoc Ecosystem.

Proof Conclusion: We have shown for all REs of the Ad Hoc Ecosystem that they are secure for at least a security factor $\mu = \frac{1}{2^{l-1}}$. This holds true for any Frustration Threshold ($\mathbb{F} \geq 1$). For comparison, the original Bluetooth system was assuming a security factor of $\mu = \frac{1}{2^l}$ but eventually was shown in multiple instances to be insecure. Thus, we deem our proposal to be generally suited for the same applications as Bluetooth for the same value of l (bit length of OOB value). Furthermore, our proposal supports pairing between the same device classes as Bluetooth and can thus be deemed as an improved, seamless and finally secure replacement.

IX. XNC and XPE Implementation

To demonstrate the practical feasibility of our proposal and enable further research, we also implemented and tested XNC and XPE into BTstack. We submitted code and documentation as artifacts (see Appendix B). See Appendix A-C for implementation details.

To demonstrate that our proposal runs on commonplace Bluetooth and IoT hardware, we conducted macro-benchmarks on network overhead, retired instructions and CPU cycle count as well as memory usage. Additionally, we performed micro-benchmarks of the cryptographic functions of each implementation. By using Raspberry Pi 3Bs, which lack crypto acceleration and optimized CPU features, our tests represent real-world hardware and assure that all conclusions of performance differences translate to IoT devices.

Interestingly, we found that XPE actually decreases network load and computational overhead with equivalent memory usage compared to PE/PPE by eliminating the round-based Passkey confirmation process. XNC, when compared to NC/PNC, leads to a modest increase in network payload by 17/34 Bytes (13%/25% increase) on the Initiator/Responder side and a slight increase in computational overhead ($\leq 5\%$) and memory usage ($\leq 1\%$). Detailed results are in Appendix A-D. Since pairing has a one-time cost for connection establishment, is infrequent in a device's lifetime, and since we did not optimize for production, we are confident that current Bluetooth hardware can run XNC/XPE without significant delay or decrease in usability. XNC and XPE require no adjustment in user behavior, user experience, peripherals nor do they require any additional hardware: the implementation only modifies the host portion of the Bluetooth stack and thus no firmware updates are required to deploy our patch.

X. Discussion and Conclusion

This work discusses the persistent issue of MC in the Bluetooth connectivity framework and how even the most recent mitigation proposals are still affected by MC. For that purpose, we propose a threat model that is centered around the core issue of MC and provide an application procedure that is straightforward to execute and comprehensible. With this

model, we rediscover MC issues in Bluetooth and find two new MCs in the currently most prominently discussed solution proposal from Shi et al. [4]. We implement those attacks on off-the-shelf Bluetooth hardware. We thus demonstrate that there is yet no satisfying solution to the issue of MC for ad hoc pairing. In fact, our analysis suggests that a fresh pool of pairing protocols is required in order to purge MC from Bluetooth. Otherwise, it appears infeasible to avoid MC between patched and legacy entities.

Subsequently, as an additional contribution we propose two protocol candidates (XNC and XPE) that are able to fill this gap. We use our model to proof that these protocols cannot be confused with each other. Eventually, we implement our solution proposal on regular Bluetooth hardware. Furthermore, we argue that an Ad Hoc Ecosystem consisting of XNC and XPE supports the exact same device landscape as an Ad Hoc Ecosystem consisting of NC and PE. Our proposed Ad Hoc Ecosystem is currently the only one that supports the same device combinations and user profile as Bluetooth without suffering from MC, thus making it the perfect candidate for either a new iteration of Bluetooth's pairing implementation or alternatively a foundation for a competing framework that seeks to implement secure ad hoc pairing.

Considering the case of applying our solution as iteration for Bluetooth, one could argue that many Bluetooth devices are difficult to update as those devices are often not actively version-managed by their users nor receive regular updates by their vendors. This is a general issue in consumer-device security. Therefore, we believe that organizational environments and security-aware spaces (e.g., payment terminals) would primarily profit from our proposal. In those settings, it is more likely that vendors and users keep devices up-to-date and thus it is more likely that both pairing partners run a secure version.

In the long term, it is certain that the remaining device landscape will also benefit from the introduction of our proposal since more and more old devices are eventually replaced by patched ones. We also suggest to implementers of our solution proposal, organizational managers, as well as vendors, to make patched devices clearly distinguishable from unpatched ones; for instance by applying a logo or seal of approval. This will prevent users from combining old and new devices and motivate users to accelerate the replacement of old devices.

Until now, the work of Shi et al. was the only promising candidate to mitigate MC in Bluetooth while supporting a similar device diversity and user landscape as Bluetooth. However, our findings show multiple flaws in their proposal. Still, their proposal is at this point purely theoretical as we are the first ones to provide an implementation. While currently no actual devices are affected, this might change soon when their proposals are adopted by the Bluetooth committee. Thus our findings should be discussed openly. Consequently, after review and publication of this work we plan to start a dialogue with the respective groups and committees in the Bluetooth standardization space. We deem it crucial that the next iteration of Bluetooth (or connectivity frameworks superseding it) considers our findings in order to grasp the issue of MC at its root. Overall, we hope to contribute to the deeper appreciation and understanding of MC and point out possible solution paths for different scenarios.

Acknowledgment

The authors extend their gratitude to all the supporters, reviewers, and artifact reviewers. Special thanks go to Kai Angnis, Fabian Kilger, Manuel Andreas, Fabian Franzen, and Matthias Ringwald.

References

[1] Bluetooth SIG, "Bluetooth market update 2023," https://www.bluetooth.com/2023-market-update/, 2023.

[2] M. Tschirschnitz, L. Peuckert, F. Franzen, and J. Grossklags, "Method confusion attack on Bluetooth pairing," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 1332–1347. [Online]. Available: https://doi.org/10.1109/SP40001.2021.00013

[3] T. Claverie, G. Avoine, S. Delaune, and J. L. Esteves, "Extended version: Tamarin-based analysis of Bluetooth uncovers two practical pairing confusion attacks," HAL Open Science, Tech. Rep. hal-04079883, Apr. 2023. [Online]. Available: https://hal.science/hal-04079883

[4] M. Shi, J. Chen, K. He, H. Zhao, M. Jia, and R. Du, "Formal analysis and patching of BLE-SC pairing," in *32nd USENIX Security Symposium (USENIX Security)*. USENIX Association, 2023, pp. 37–52. [Online]. Available: https://www.usenix.org/conference/usenixsecurity23/presentation/shi-min

[5] F. Stajano and R. Anderson, "The resurrecting duckling: Security issues for ad-hoc wireless networks," in *Security Protocols*, B. Christianson, B. Crispo, J. A. Malcolm, and M. Roe, Eds. Springer, 2000, pp. 172–182.

[6] A. Kumar, N. Saxena, G. Tsudik, and E. Uzun, "A comparative study of secure device pairing methods," *Pervasive and Mobile Computing*, vol. 5, no. 6, pp. 734–749, 2009.

[7] I. Damgård, "Commitment schemes and zero-knowledge protocols," in *Lectures on Data Security: Modern Cryptology in Theory and Practice*, I. Damgård, Ed. Springer, 1999, pp. 63–86.

[8] S. M. Bellovin and M. Merritt, "Encrypted key exchange: Password-based protocols secure against dictionary attacks," in *1992 IEEE Computer Society Symposium on Research in Security and Privacy (SP)*, 1992, pp. 72–84.

[9] J. Katz, R. Ostrovsky, and M. Yung, "Efficient password-authenticated key exchange using human-memorable passwords," in *Advances in Cryptology – EUROCRYPT 2001*, B. Pfitzmann, Ed. Springer, 2001, pp. 475–494.

[10] R. Gennaro and Y. Lindell, "A framework for password-based authenticated key exchange," in *Advances in Cryptology – EUROCRYPT 2003*, E. Biham, Ed. Springer, 2003, pp. 524–543.

[11] S. Jiang and G. Gong, "Password based key exchange with mutual authentication," in *International Workshop on Selected Areas in Cryptography*, H. Handschuh and M. A. Hasan, Eds. Springer, 2004, pp. 267–279.

[12] *Bluetooth Core Specification 5.3*, Bluetooth Specification Contributors, July 2021.

[13] M. von Tschirschnitz, L. Peuckert, M. Buhl, and J. Grossklags, "Extended appendix," Nov. 2024. [Online]. Available: https://doi.org/10.5281/zenodo.14246308

[14] S. Vaudenay, "Secure communications over insecure channels based on short authenticated strings," in *Advances in Cryptology – CRYPTO 2005*, V. Shoup, Ed. Springer, 2005, pp. 309–326.

[15] Y. Lindell, "Comparison-based key exchange and the security of the numeric comparison mode in Bluetooth v2.1," in *Topics in Cryptology – CT-RSA 2009*, M. Fischlin, Ed. Springer, 2009, pp. 66–83.

[16] M. Troncoso and B. Hale, "The Bluetooth CYBORG: Analysis of the full human-machine passkey entry AKE protocol," Cryptology ePrint Archive, Paper 2021/083, 2021, https://eprint.iacr.org/2021/083.

[17] R. Chang and V. Shmatikov, "Formal analysis of authentication in Bluetooth device pairing," in *Joint Workshop on Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis (FCS-ARSPA)*, 2007, pp. 45–61.

[18] D. Jia and R. Hsu, "Formal modeling and analysis of Bluetooth 4.0 pairing protocol," Stanford University, Working Paper, 2013.

[19] M. Fischlin and O. Sanina, "Cryptographic analysis of the Bluetooth secure connection protocol suite," in *Advances in Cryptology – ASIACRYPT 2021*, M. Tibouchi and H. Wang, Eds. Springer, 2021, pp. 696–725.
[20] M. K. Jangid, Y. Zhang, and Z. Lin, "Extrapolating formal analysis to uncover attacks in Bluetooth passkey entry pairing," in *30th Network and Distributed System Security Symposium (NDSS)*, 2023.
[21] J. Wu, R. Wu, D. Xu, D. J. Tian, and A. Bianchi, "Formal model-driven discovery of Bluetooth protocol design vulnerabilities," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 2285–2303.
[22] Y. Shaked and A. Wool, "Cracking the Bluetooth PIN," in *Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services*. ACM, 2005, pp. 39–50.
[23] T. Rosa, "Bypassing passkey authentication in Bluetooth Low Energy," Cryptology ePrint Archive, Paper 2013/309, 2013. [Online]. Available: https://eprint.iacr.org/2013/309
[24] M. Fischlin and O. Sanina, "Fake it till you make it: Enhancing security of Bluetooth Secure Connections via deferrable authentication," Cryptology ePrint Archive, Paper 2024/874, 2024. [Online]. Available: https://eprint.iacr.org/2024/874
[25] X. Che, Y. He, X. Feng, K. Sun, K. Xu, and Q. Li, "BlueSWAT: A lightweight state-aware security framework for Bluetooth Low Energy," arXiv, Tech. Rep. 2405.17987, 2024. [Online]. Available: https://arxiv.org/abs/2405.17987
[26] 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.
[27] E. Uzun, K. Karvonen, and N. Asokan, "Usability analysis of secure pairing methods," in *International Conference on Financial Cryptography and Data Security*, S. Dietrich and R. Dhamija, Eds. Springer, 2007, pp. 307–324.
[28] M. von Tschirschnitz and M. Buhl, "Advanced Method Confusion and X-Mitigations." [Online]. Available: https://github.com/maxdos64/advanced_mc
[29] BlueKitchen GmbH, "BTstack," http://bluekitchen-gmbh.com/btstack/.
[30] A. Mittelbach and M. Fischlin, *The Theory of Hash Functions and Random Oracles: An Approach to Modern Cryptography*. Springer Nature, 2021.
[31] M. Abdalla, B. Haase, and J. Hesse, "Security analysis of CPace," in *Advances in Cryptology – ASIACRYPT 2021*, M. Tibouchi and H. Wang, Eds. Springer, 2021, pp. 711–741.
[32] H. J. Abdalla Michel, Hasse Björn, "CPace, a balanced composable PAKE," Internet Requests for Comments, IETF Data Tracker, RFC, March 2024. [Online]. Available: https://datatracker.ietf.org/doc/draft-irtf-cfrg-cpace/
[33] M. Raab and A. Steger, ""Balls into bins" – A simple and tight analysis," in *Randomization and Approximation Techniques in Computer Science*, M. Luby, J. D. P. Rolim, and M. Serna, Eds. Springer, 1998, pp. 159–170.
[34] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*. Chapman and Hall/CRC, 2007.
[35] *Assigned Numbers*, Bluetooth Specification Contributors, February 2024, https://www.bluetooth.com/specifications/assigned-numbers/.
[36] R. Zaveri, "Data-extraction-from-fitbit," https://github.com/rz0012/Data-Extraction-from-Fitbit, 2020.
[37] K. Makan, "BlueCrawl," https://github.com/IOActive/BlueCrawl, 2018.

APPENDIX A
ADDITIONAL MATERIALS

A. Patched BTstack

We modified the functions that are called by the Initiator and Responder after the LTK Validation stage has succeeded. Instead of completing the pairing, our modification forces both sides into a new second stage, as specified by [4]. In this second stage, both sides calculate a function $g3$ that takes the Passkey (PE) or the display value (NC) and hashes it with the messages sent and received during the pairing request and response phase. We implement $g3$ using the available cryptographic primitives of BTstack. Subsequently, our modification repurposes the user interaction API of BTstack to let the user either compare or transfer the result values, depending on whether the first stage was based on NC or PE. If a value is transferred, the receiving side compares it with the local result of $g3$. If that comparison is successful or the user has confirmed a comparison, our modification returns to the regular flow of the state machine to complete the pairing.

B. MitM Attack on PNC and PPE Implementations

In the attack from Section V-B, we assume one of the victims (e.g., the Initiator) performs a PNC Role, while the other victim (e.g., the Responder) performs (unpatched) PE.

To implement such an attack, we created a MitM attack framework that can perform pairing with two victims of different patch-level simultaneously, on one side using a patched PNC and on the other an unpatched PE authentication method, For that purpose our framework consists of three memory-independent processes. First, the master process (M) initializes POSIX pipes to coordinate pairing between the processes and exchange data. M then spawns two processes: I (a Bluetooth device implementation compiled an unpatched BTstack implementation) and R (a Bluetooth devices compiled with the patched BTstack implementation). When all processes are initialized, they enter a suspended state, waiting for incoming connections.

When R is engaged in pairing by the initiating victim, it enforces the PNC authentication method by sending carefully chosen capabilities during the IOCapabilitiesExchange. It then sends a signal to I, leading I to engage as displaying device in a PE pairing with the victim Responder. I proceeds with PE until it must choose a Passkey to display, then waits for further data.

As soon as R calculates the first display value, it forwards this value to I, which sets it as its chosen Passkey. Once pairing between the victim Responder and I succeeds, I alerts R, which confirms the comparison with the victim Initiator. Simultaneously, I silently dissolves the established connection with the victim Responder and requests a new PE pairing. Again, I waits for R to report its display value and uses it as the Passkey. As soon as the second PE authentication succeeds, the second confirmation value is subsequently confirmed by R. Note, the victim Initiator is unable to notice that the victim Responder is performing the unpatched PE authentication since they have no direct communication.

Similarly, we implement the attack from Section V-C. Here, the initiating victim performs the (patched) PNC and the responding victim the (patched) PPE authentication method. Thus, this time, both I and R are compiled with a patched version of BTstack. Again, I only engages with the Responder victim when R is engaged. R leads the pairing to a PNC authentication, while I enforces the PPE model, again, with I as the displaying side. As in the other attack, R hands through both values it calculates as display values to I, which uses them as their passkey/display value.

C. XNC and XPE Implementation

1) XNC: For XNC, we modified the state machine of NC to reset its state to the nonce exchange phase when the user

confirms the first displayed value. This leads to a second validation phase, as specified in our proposal. After a second successful user interaction, the pairing is concluded as usual. Additionally, we added a stopwatch that starts when the first value is sent to the display of the device and is checked again during the key validation phase. If the stopwatch exceeds our timespan constant of $\mathbb{T}$, the pairing fails. The constant $\mathbb{T}$, which determines the timestamp, can be set in the BTstack configuration file and defaults to 20 seconds.

2) XPE: XPE required extensive modification of BTstack's state machine of PE. After public keys are exchanged, the partner offering the user input is determined as the Receiver, and the other as the Sender. According to their role definitions, both sides decide on nonces, and the Sender shares a commitment over its nonce. Subsequently, the Receiver reveals its nonce, upon which the Sender calculates a hash over local and remote nonces and public keys. To choose nonces and calculate commits and hashes, we utilized functions that are required in the Bluetooth specification and thus implemented by BTstack. Next, the calculated value is displayed to the user by the Sender and asked to be entered by the Receiver using BTstack's user interaction API. Once this step is completed, the Receiver initiates PAKE.

We opted to use CPACE as the PAKE primitive. CPACE has undergone multiple review cycles in academic [31] and standardization settings [32], and most importantly, it offers a long-standing C implementation based on the portable libsodium library [1]. Note, that this choice of PAKE method is preliminary and targeted at creating this PoC. Nevertheless, production implementations should consider the current availability of PAKE methods and implementations to find the best fit for their use case. Our code is thus structured to accommodate any choice of PAKE primitive. Generally, a PAKE request is first calculated by the chosen implementation on the Receiver using the entered s value as a password. When the request arrives at the Sender, the Sender conducts a PAKE step yielding key tk_S^2 and a response packet. When the response packet arrives at the Receiver, the Receiver can now calculate its key tk_R^2. The primitive ensures that the Receiver and Sender only calculate the same key if they use the same s. To verify that both sides were able to establish the same key, we added a challenge-response exchange between Sender and Receiver that verifies that $tk_S^2 == tk_R^2$. After that verification, the PAKE step of our protocol is concluded and the assumptions of our protocol are fulfilled. Now the Sender can reveal the nonce that was earlier committed, upon which the Receiver verifies the commitment as well as the value of s. Again, we can utilize built-in functions of BTstack for that. Eventually, both sides calculate the LTK as the XORed result of their DH-keys that were established before the authentication phase and their PAKE keys.

To implement the timing requirements of our protocol, the XPE state machine sets a flag when the Receiving side opens a user input. It then uses the system timer to ensure that this Role Execution will not terminate until the timespan $\mathbb{T}$ has passed after user input was received. Note that, for this purpose, we also placed hooks in the error and abortion handlers of BTstack to prevent an attacker from sending failure messages to shorten the targeted wait timespan.

[1] https://github.com/jedisct1/cpace

TABLE I
NETWORK PERFORMANCE COMPARISON OF NC METHODS IN BYTES.

Device	Unpatched NC	PNC	XNC
Initiator	128	128	170
Responder	101	101	122

TABLE II
NETWORK PERFORMANCE COMPARISON OF PE METHODS IN BYTES.

Device	Unpatched PE	PPE	XPE
Initiator	926	905	133
Responder	920	899	132

D. Performance Evaluation

In order to assess the practicability of the proposals of XNC and XPE to replace the current state of Bluetooth without requiring any changes to hardware we conducted performance measurements and compared modified and unmodified Bluetooth stacks. The tests were conducted on two Raspberry Pi 3Bs, each equipped with screens and keyboards for user interaction and *Cambridge Silicon Radio, Ltd., Bluetooth USB Dongles* for connectivity and no crypto acceleration of special instruction sets.

In order to compare the network loads created by the proposals, we collected packet traces using BTstacks built-in packet logging functionality. **Table I** and **Table II** summarize all payload sizes sent and received by the partners during the pairing procedure.

We observe a slight increase in network load when comparing XNC with NC and PNC. We attribute this to the additional exchange of nonces needed for the second comparison. On the other hand, when we look at XPE, PPE, and PE we can see that XPE reduces network load significantly while PPE shows a slight increase. This is the case since XPE replaces the multi-round commit and reveal phase of PE with the much less network intensive CPace exchange, while PPE introduces additional signaling messages to let the displaying device know when to proceed to the next phase.

In order to assess the computational overhead we conducted two types of benchmarks. First, we performed micro-benchmarks by using the *perf* Linux-kernel instrumentation to measure the CPU-cycle and retired instruction count of the cryptographic libraries involved. We assume that cryptographic primitives are the main culprit when it comes to generating computational overhead and profiling just those functions allows us to eliminate overhead caused by network and user interaction delays. In order to assure though that we did not overlook any other sources of overhead we additionally conducted macro-benchmarks that measure CPU and retired

TABLE III
MICRO-BENCHMARK RESULTS FOR NC/PNC/XNC (JUST CRYPTO); RETIRED INSTRUCTIONS (TOP) AND CPU CYCLES (BOTTOM) COUNT IN UNITS OF THOUSANDS.

Device	Unpatched NC	PNC	XNC
Initiator	179,904 (49) 239,404 (148)	180,010 (51) 239,894 (155)	180,335 (48) 240,516 (214)
Responder	135,139 (45) 180,290 (122)	135,252 (45) 180,463 (131)	135,583 (47) 181,172 (282)

TABLE IV
MICRO-BENCHMARK RESULTS PE/PPE/XPE (JUST CRYPTO); RETIRED INSTRUCTIONS (TOP) AND CPU CYCLES (BOTTOM) COUNT IN UNITS OF THOUSANDS.

Device	Unpatched PE	PPE	XPE
Initiator	188,134 (2,449) 258,291 (3,257)	187,492 (54) 257,015 (393)	183,640 (50) 243,300 (137)
Responder	135,139 (45) 180,290,118 (122)	142,806 (66) 197,558 (684)	137,105 (66) 182,130 (115)

instruction count as well as memory usage of the whole stack execution.

All benchmarks were run 1000 times for each protocol implementation (PE/NC, PPE/PNC, and XPE/XNC). The average of each measurement set is posted in the respective table and the standard deviation is included. The micro-benchmark results are in **Table III** and **Table IV**. The macro-benchmark results can be found in **Tables V, VI, VII, VIII**. As mentioned earlier, we cannot find any significant increase in computational overhead when comparing XNC and XPE with the other methods and thus think that our proposed methods will perform well on existing Bluetooth hardware.

TABLE V
MACRO-BENCHMARK RESULTS NC/PNC/XNC (WHOLE STACK); RETIRED INSTRUCTIONS (TOP) AND CPU CYCLES (BOTTOM) COUNT IN UNITS OF THOUSANDS.

Device	Unpatched NC	PNC	XNC
Initiator	145,187 (68) 195,416 (247)	145,304 (59) 196,082 (409)	145,975 (65) 198,014 (250)
Responder	100,750 (44) 136,668 (225)	100,841 (65) 137,408 (377)	101,444 (44) 138,635 (227)

TABLE VI
MACRO-BENCHMARK RESULTS PE/PPE/XPE (WHOLE STACK); RETIRED INSTRUCTIONS (TOP) AND CPU CYCLES (BOTTOM) COUNT IN UNITS OF THOUSANDS.

Device	Unpatched PE	PPE	XPE
Initiator	156,871 (1,992) 219,850 (2,680)	156,105 (64) 218,302 (564)	149,250 (57) 200,151 (184)
Responder	112,390 (54) 160,746 (639)	111,576 (52) 159,212, (526)	104,518 (45) 141,142 (476)

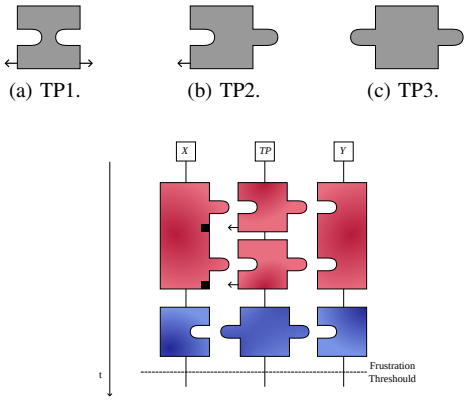
TABLE VII
MEMORY USAGE OF WHOLE STACK OF NC/PNC/XNC IN KILOBYTES.

Device	Unpatched NC	PNC	XNC
Initiator	2, 752, (37)	2, 752, (37)	2, 753, (35)
Responder	2, 739, (37)	2, 742, (34)	2, 741, (35)

TABLE VIII
MEMORY USAGE OF WHOLE STACK OF PE/PPE/XPE IN KILOBYTES.

Device	Unpatched PE	PPE	XPE
Initiator	2, 752, (36)	2, 764, (0)	2, 753, (35)
Responder	2, 741, (35)	2, 703, (63)	2, 742, (34)

E. Third-Party Building Block Model



(a) TP1. (b) TP2. (c) TP3.

(d) Example with TP2 and TP3.

Fig. 10. Third-Party Building Blocks.

Fig. 10 shows the different third-party behaviors described in Section IV-D. It showcases two blocks and how the third party interacts with them. The red block requires two OOB interactions, similar to a PPE block where the X side is the displaying device. The arrows on the TP pieces symbolize injecting the confirmation as feedback into the displaying device. The blue block uses one OOB interaction and requires an input on both devices.

F. Analysis of Advantage of Bruteforce Attacker

We now want to estimate how feasible it would be for a PPT adversary to find an accumulation of collisions of r values for a fixed α in comparison to the advantage an adversary would gain through this effort. First let us consider that the adversary chooses with α a part of the input to g. Since we assumed uniform distribution for g, every possible value for r will cause the output of g to be uniformly distributed in the output space (for any r each possible β is hit with $\frac{1}{2^l}$). The adversary now wants to run this experiment for a fixed α and many different values for r to see if one of the values for β occurs exceedingly often.

This exact experiment has already been discussed extensively as the "balls in bins" problem [33] where balls are

thrown (uniformly) into a set of buckets. From the respective literature it is well known that the maximum load of any of i buckets after j balls have been thrown is bounded through $\Theta(\frac{j}{i})$ for $j \geq i\log(i)$ which we will assume in favor of the adversary using brute force. A note to the interested reader may be that even tighter bounds have been found and extensively proven by Raab and Steger [33] for the same restrictions on j and i. For our purposes though this loose bound is sufficient.

Thus we can say for the maximum accumulation of hits $\mathbb{B}$ for any output value to occur is bounded by: $\mathbb{B} = \Theta(\frac{a}{2^l})$ where a is the amount of possible values for r that were sampled in the brute force effort. We note that in practice $a \leq 2^k$ (recall that k is the bit-length of nonce r) but we can ignore this restriction in the following as this would only limit the attacker's abilities, so we assume a stronger adversary by ignoring it. We can thus now say for the probability of the attacker succeeding within a many brute force steps (Success_a)

$$Pr[\text{Success}_a] = \Theta(\frac{a}{2^l}) \cdot \frac{1}{2^k}$$
$$\implies \exists c > 0 : Pr[\text{Success}_a] \leq \frac{c \cdot a}{2^l} \cdot \frac{1}{2^k} = c \cdot a \frac{1}{2^{(k+l)}}$$

Following the standard practice of security proofs [34, Section 3.11] we now model the effort a that the adversary can feasibly afford as computationally asymptotically. For that we simply say that a is the result of any PPT function applied to some security parameter n. We further define the security parameter dependent on the length of r and β such that $n = k + l$. We can now write that the probability of success of any PPT adversary (Success_{PPT}) is

$$\exists c > 0 \forall p \in PPT : Pr[\text{Success}_{PPT}] \leq c \cdot p(n) \frac{1}{2^n}$$

Now we can further observe that

$$\forall p, p' \in PPT \forall c > 0 \exists N \forall i > N : c \cdot p(i) \frac{1}{2^i} \leq \frac{1}{p'(i)}$$

Thus we can say that $c \cdot p(n)\frac{1}{2^n}$ is *negligible* for all $c > 0$ and for all $p \in PPT$ [34, 3.1.1] and thus Pr['Succ with brute force of PPT adv.'] is negligible for large enough values for n. Since in common implementations (e.g., Bluetooth [12]) $k + l \approx 148$ we assume the security parameter to be sufficiently large.

This finding is intuitively explained by considering that an adversary that follows the brute force approach has to find at least $\frac{m}{2^l}$ collisions in the mapping for it to certainly bring success with probability at least $\frac{m}{2^k}$. When either k or l grow the adversary also has to find exponentially more collisions to maintain a certain probability of success. With growing k or l the PPT adversary is thus eventually overpowered.

We have shown that the best probability of success for the adversary to fulfill (3) (with his own public key injected) is random guessing, since any brute force effort is not computationally feasible.

G. Security Implications of Single-Sided Pairing

To clarify the potential security implications of single-sided pairing, we present two practical scenarios where device X and Y attempt to pair, but pairing only succeeds with X.

Scenario 1: Immediate Detection by the User

1) **Pairing Attempt:** X and Y initiate a pairing process, but only X completes the pairing successfully. Y fails to connect, and the user notices this failure.
2) **User Reaction:** The user, realizing that Y has not paired, decides to disconnect X manually. However, by the time the user reacts (even within seconds), significant damage can already occur.
3) **Attacker Exploitation:** Once X is paired, the connection with the attacker is encrypted and trusted, the attacker gains access to various interfaces (so called Bluetooth Profiles) exposed by X. These Profiles may include sensitive data like the user's call history, text messages (including 2FA codes), phonebook contacts, and even health-related data. Some Profiles also permit to alter the device configuration and also enable keyboard or pointing device input. A list of globally standardized interfaces can be found here: [35]. Often the vendor supplies their own app with the device to specify additional custom interfaces for data transfer between app and device. This information like GPX tracking data or fitness measurements are especially interesting for the attacker. Scripts to discover these custom interfaces and to download this information are already available (e.g., [36], [37]).

Scenario 2: Inability to Detect/React

1) **Hidden Pairing Failure:** Many devices provide limited feedback on pairing success. In this case, X successfully pairs, but the user only notices that the intended application on Y does not work as expected. There is no clear indication that X has completed pairing.
2) **Missing Option to Disconnect/Awareness:** Some devices do not provide clear menus or options for managing Bluetooth connections. In cases where pairing slots are limited, older connections may be silently overwritten instead. The user thus may not be able to manage connections appropriately. We must also consider that the average user is not experienced in interpreting errors of the Bluetooth state machine and thus would not spend effort to disconnect X.
3) **Full MitM:** The user, attempting to resolve the issue, may retry pairing with Y. However, during this second attempt, the attacker could now also establish a single-sided pairing with Y, completing their goal of achieving a full MitM attack.

APPENDIX B
ARTIFACT APPENDIX

The *AdvancedMC* artifact allows readers to test our implementations and attacks on actual Bluetooth hardware, thus confirming that the proposed protocols function as expected on real-world hardware. We further provide tools to conduct performance comparisons between the various implementations. Reproduction requires up to four Bluetooth USB devices that are compatible with the BTstack implementation[2].

A. Description & Requirements

The README file provides instructions on building and running the protocol implementations. We also summarize the necessary steps in the following.

1) Access: The artifact is published and available (see https://doi.org/10.5281/zenodo.14214999).

2) Hardware Dependencies: Any attack setup consists, in principle, of two victim devices and one MitM device. All of them should run a Linux operating system. The victims should each be connected to one Bluetooth chipset (see list in footnote[2]) and the MitM to two devices with a supported Bluetooth chipset.

3) Software Dependencies & Benchmarks: The following dependencies should be pre-installed on the Linux machine(s): *pkg-config, make, libsodium-dev, libusb-dev, tshark*.

B. Artifact Installation & Configuration

Install the software dependencies and connect the USB dongles. Make sure they are recognized by the system using *lsusb*. Peruse the README file (see artifact root directory). The original Bluetooth design supports two separate 'secure' pairing methods. The first is based on comparing numbers (NC), the second relies on transferring numbers (PE). The artifact also contains three improved implementations:

1) MBTstack: Standard Bluetooth stack extended with debug hooks for debugging
2) PatchedMBTstack: MBTstack patched according to Shi et al. [4]; PE replaced with PPE and NC with PNC
3) XMBTstack: MBTstack patched according to our proposal; PE replaced with XPE and NC with XNC

Subdirectories include sample Bluetooth applications: in 'unpatched_victims' for MBTstack; 'patched_victims' for PatchedMBTstack; 'x_victims' for XMBTstack. To build those samples, simply run *make* in the sample directory.

C. Major Claims

- (C1): We implemented PNC and PPE as proposed in [4], which can be verified through experiment E1.
- (C2): We implemented XNC and XPE as we proposed in our paper, which can be verified through experiment E2.
- (C3): We implemented the PNC-on-PE attack and the PNC-on-PPE attack, verifiable through experiment E3.
- (C4): Implementation overheads of XNC and XPE are not unreasonably higher then regular Bluetooth (E4).

[2]https://github.com/bluekitchen/btstack?tab=readme-ov-file#supported-chipsets

D. Evaluation

We will first guide through the steps for evaluating that each of our pairing implementations functions correctly on commodity Bluetooth hardware. Make sure to run all binaries as root user. This is necessary to have the appropriate permissions to access the USB device. To avoid running the examples as root, please make sure to set the udev rules appropriately according to your individual system configuration.

1) Experiment (E1): [PNC and PPE Pairing] [20 human-minutes]: We establish a pairing on real Bluetooth hardware using the PNC and PPE methods as proposed by Shi et al. [4].

[Preparation] Enter the 'patched_victims' directory and build using `make`.

[Execution] Run `sudo ./responder.bin <USB_BUS>:<USB_DEVICE#>` in one terminal (use 'lsusb' to gather the USB information of the dongles). This will initialize the stack and output the Bluetooth address of the ready Responder: e.g. "RESP(GEN): BThack up and running on **01:AA:BB:CC:DD:EE**."

Now, use the second terminal to run the Initiator and start pairing with a running Responder: `sudo ./pnc_initiator.bin <USB_BUS>:<USB_DEVICE#> <TARGET_RESPONDER_MAC_ADDRESS>` using a different USB dongle than for the Responder. Pairing should now initialize; follow the pairing process by acting like a user from the widely accepted Bluetooth user behavior model. That means, whenever you are asked to press yes/no when a number is displayed, you confirm if either the same number is displayed on the other side or an input field is provided. When asked to transfer a number, do so if an input field is supplied on the other side. Subsequently, kill the applications and repeat the experiment using 'ppe_initiator.bin' instead of 'pnc_initiator.bin' on the Initiator side.

[Results] Following the instructions to compare or transfer numbers should assure that PNC and PPE pairing works. You may also try to transfer the wrong numbers or not confirm the comparison, this should block or abort the pairing.

2) Experiment (E2): [XNC and XPE Pairing] [20 human-minutes]: We establish a pairing on real Bluetooth hardware using the XNC and XPE methods as proposed in our work.

[Preparation] Enter the 'x_victims' directory and build using `make`. Open two terminals in this directory and verify that two Bluetooth USB dongles are connected.

[Execution] Repeat the procedures from E1 and only alter the names of the binaries in the commands to 'xnc_initiator.bin' or 'xpe_initiator.bin', respectively.

[Results] Following the instructions to compare or transfer numbers should assure that XNC and XPE pairing works. You may also try to transfer the wrong numbers or not confirm the comparison, this should block or abort the pairing.

3) Experiment (E3): [Attack tests] [30 human-minutes]: We demonstrate how the first attack on the PNC and PPE methods work in practice as we describe them in Section V-B.

[Preparation] Build the examples in the 'patched_victims' and 'unpatched_victims' directories. Open three terminals in the root directory and run `make` to build 'pnc_mitm_pe.bin'

and ‘pnc_mitm_ppe.bin’. Verify that four Bluetooth USB dongles are connected.

[Execution] Run `sudo ./unpatched_victims/responder.bin <USB_BUS>:<USB_DEVICE#>` in one terminal using ‘lsusb’ to gather the USB information of the dongles. This will output the Bluetooth address of the ready Responder.

Now, run in another terminal `sudo ./pnc_mitm_pe.bin <USB_BUS>:<USB_DEVICE#> <USB_BUS2>:<USB_DEVICE#2> <TARGET_RESPONDER_MAC_ADDRESS>` using two of the still unused dongles. The MitM example will then also display its own Responder MAC address. Now, use the remaining terminal to run the Initiator and start pairing with a running MitM-Responder: `sudo ./patched_victims/pnc_initiator.bin <USB_BUS>:<USB_DEVICE#> <MITM_RESPONDER_MAC_ADDRESS>` using the remaining USB dongle. Pairing should now initialize. Follow the pairing process by acting like a user from the widely accepted Bluetooth user behavior model. That means, whenever you are asked to press yes/no when a number is displayed, you confirm if either the same number is displayed on the other side, or if an input field is provided. When you are asked to transfer a number, do so if an input field is supplied on the other side.

Subsequently, kill the applications and repeat the experiment using ‘pnc_mitm_ppe.bin’ instead of ‘pnc_mitm_pe.bin’ and ‘patched_victims/responder.bin’ instead of ‘unpatched_victims/responder.bin’.

[Results] In the first experiment, both sides should successfully establish a pairing as long as user interaction followed the user model described in Section IV-D; demonstrating a full MitM compromise. In the second experiment, the pairing should succeed on the PNC side of the pairing.

4) Experiment (E4): [Performance Measurements] [1.5 human-hours]: We supply two types of performance measurement tool sets. The macro benchmark measures the instruction and cycle count as well as memory overhead of the entire pairing execution. This includes network and user-induced latencies. The micro benchmark measures only cycles and instructions of the cryptographic primitives, thus excluding those latencies. We recommend performing both types of tests with each implementation to confirm that XNC and XPE have no unreasonable overhead in comparison to the original Bluetooth methods or PNC and PPE. Our own results for this experiment can be found in **Tables III–VIII**.

[Preparation] We now give the instructions to profile Bluetooth samples in the directory d, which could be either ‘unpatched_victims’ ‘patched_victims’ or ‘x_victims’.

To profile using the micro benchmark, enter directory d and run: `make clean && MEASURE=1 make`; to profile using the macro benchmark, enter directory d and run: `make clean &&QUIT_ON_SUCCESS=1 make`.

[Execution] Benchmarking can be performed using two Bluetooth devices connected to the same or to separate machines. For that reason, the benchmarking applications operate in a server-client fashion. First, the benchmarking application is launched in server mode, specifying an available TCP port, the USB ID of a Bluetooth device and the Responder binary to be benchmarked. Next, the application is launched in client mode on the same or a separate device, providing the corresponding Initiator binary, USB ID, the server’s port and IP address.

For instance, open two terminals in the root directory of the artifact, one for the Initiator and one for Responder device. Let us assume d=‘unpatched_victims’ and one wants to profile the combination ‘responder.bin’ and ‘pe_initiator.bin’. We **always** begin by conducting one regular pairing between the Initiator and Responder to test functionality and gather the Bluetooth address of the Responder.

Subsequently, a benchmark (e.g., in this case macro) can be run as follows. On the Responder side (from the root directory): `sudo ./measure-macro.py -s <LISTENING_PORT> ./unpatched_victims/responder.bin <USB_BUS>:<USB_DEVICE#>`
And on the Initiator side (root directory): `sudo ./measure-macro.py -c <RESPONDER_DEVICE_IP> <RESPONDER_DEVICE_PORT> ./unpatched_victims/pe_initiator.bin <USB_BUS>:<USB_DEVICE#> <TARGET_RESPONDER_MAC_ADDRESS>`

To perform a micro benchmark, compile the samples accordingly (MEASURE=1) and use the same syntax and procedure, only replacing ‘measure-macro.py’ with ‘measure.py’.

[Results] These scripts perform pairing repeatedly, measuring the overall performance and writing the observations to a ‘.msmt’ file. Run for as many iterations as deemed necessary to confirm validity. The ‘msmt’ files can be evaluated using the scripts in the ‘eval’ folder. Further, our implementations produce network logs (‘*.pklg’) in the ‘/tmp’ directory. The ‘eval’ folder contains a script to evaluate the payload sizes of these logs. Comments in the script detail the packet format and provide further evaluation details. We already collected a full network measurement set in the artifact folder ‘network_measurements’.

E. Notes

Pairing is under normal circumstances a rare process in the lifetime of a device. Performing hundreds of pairings for profiling reasons can cause devices to reset themselves and the new USB address needs to be fetched through *lsusb*.