Formal Analysis and Patching of BLE-SC Pairing

Min Shi, Jing Chen, Kun He, Haoran Zhao, Meng Jia, and Ruiying Du, *Wuhan University*

https://www.usenix.org/conference/usenixsecurity23/presentation/shi-min

This paper is included in the Proceedings of the 32nd USENIX Security Symposium.

August 9–11, 2023 • Anaheim, CA, USA

978-1-939133-37-3

Formal Analysis and Patching of BLE-SC Pairing

Min Shi, Jing Chen, Kun He, Haoran Zhao, Meng Jia, and Ruiying Du

School of Cyber Science and Engineering, Wuhan University
{itachi, chenjing, hekun, haoran, jiameng, duraying}@whu.edu.cn

Abstract

Bluetooth Low Energy (BLE) is the mainstream Bluetooth standard and BLE Secure Connections (BLC-SC) pairing is a protocol that authenticates two Bluetooth devices and derives a shared secret key between them. Although BLE-SC pairing employs well-studied cryptographic primitives to guarantee its security, a recent study revealed a logic flaw in the protocol.

In this paper, we develop the first comprehensive formal model of the BLE-SC pairing protocol. Our model is compliant with the latest Bluetooth specification version 5.3 and covers all association models in the specification to discover attacks caused by the interplay between different association models. We also partly loosen the perfect cryptography assumption in traditional symbolic analysis approaches by designing a low-entropy key oracle to detect attacks caused by the poorly derived keys. Our analysis confirms two existing attacks and discloses a new attack. We propose a countermeasure to fix the flaws found in the BLE-SC pairing protocol and discuss the backward compatibility. Moreover, we extend our model to verify the countermeasure, and the results demonstrate its effectiveness in our extended model.

1 Introduction

Bluetooth is a wireless technology standard for exchanging data between devices in short-range. Because of requirements evolution, the standard was divided into Bluetooth Classic (BC) and Bluetooth Low Energy (BLE) in 2010 [14]. Compared with BC, BLE is more prevalent and is widely used in daily life to connect devices, such as Bluetooth keyboards and sports watches, with minimal energy overhead. The Bluetooth Special Interest Group (Bluetooth SIG) predicts that 95% of Bluetooth-enabled devices will support BLE by 2026, where about 42% BLE devices support BLE single-mode and 53% devices support BLE + BC dual-mode [1].

The Bluetooth standard adopts *pairing* protocols to establish a trusted connection between Bluetooth devices. Roughly speaking, a pairing protocol enables two devices to authenticate each other and derive a shared secret key to protect subsequent messages. The latest pairing protocols in BLE and BC are BLE Secure Connections (BLE-SC) and Secure Simple Pairing with Secure Connections (SSP-SC), respectively [16]. These two protocols are almost the same except for key derivation and Input/Output (IO) support (see Section 2.5 for details).

Despite the widespread use of BLE, the security of the BLE-SC pairing protocol has not been systematically studied. Recently, Tschirschnitz et al. [36] disclosed the method confusion attack, which abuses the design flaws of the BLE-SC pairing protocol in the Bluetooth specification version 5.2 [15] to perform Man-in-the-Middle (MITM) attacks. Although they proposed several countermeasures to fix those design flaws, the Bluetooth SIG does not accept their countermeasures because of the backward compatibility issue, which leaves BLE devices still at risk. Moreover, their work just disclosed a specific kind of attack to witness design flaws rather than systematically analyzing the security properties of the BLE-SC pairing protocol. This situation raises the following two questions:

1. *How to formally analyze the security of the BLE-SC pairing protocol and disclose the design flaws?*
2. *How to fix the found design flaws while maintaining backward compatibility?*

In this paper, we aim to use Tamarin Prover [33] to analyze the BLE-SC pairing protocol in the latest Bluetooth specification version 5.3 [16]. Tamarin Prover is one of the most preeminent symbolic tools based on formal methods. It has achieved great success in analyzing real-world security protocols such as TLS 1.3 [18, 19], 5G-AKA [8, 17], EMV [9, 10], WPA2 [21], and Distance Bounded protocols [31, 32].

Challenges. We face the following three challenges in formally analyzing the BLE-SC pairing protocol.

First, the flow of the BLE-SC pairing protocol is complicated. The protocol consists of three phases: pairing feature exchange, Long Term Key (LTK) generation, and transport specific key distribution. In the LTK generation phase, there are four association models: Just Work (JW), Numeric Comparison (NC), Passkey Entry (PE), and Out-of-Band (OOB). These make it difficult to model the whole BLE-SC pairing protocol formally. Note that existing studies [12,26,29,34,35] only consider the LTK generation phase and usually analyze at most one association model. Besides, the BLE-SC pairing protocol not only exchanges messages between devices but also involves interactions between the device and the user. Therefore, we need to model the user's capabilities and the interface between the device and the user.

Second, existing symbolic models do not fully capture the adversary capability in the BLE-SC pairing protocol. Specifically, symbolic models employ the *perfect cryptography* assumption, in which encryption schemes are black boxes such that an adversary cannot learn anything from encrypted messages without possessing the key. However, LTK is resized to not exceed the shorter of the initiating and responding devices' maximum encryption key sizes in the BLE-SC pairing protocol. The resize operation may result in a low-entropy key and provide additional capabilities to the adversary.

Third, there are various pairing situations in practice. According to the IO capabilities (whether to use the defined IO capabilities or undefined IO capabilities, called OOB capabilities) and security configurations (does this connection need to be resistant to MITM attacks), we have 29 different pairing models. This means that even if we model the BLE-SC pairing protocol, we need to derive a different model for each pairing situation, which is a tedious and error-prone process.
Contributions. Our main contributions are as follows.

- We develop the first comprehensive formal model of the BLE-SC pairing protocol that is compliant with the latest Bluetooth specification [16]. Our model covers all phases and available association models. In addition, we model the user and the interactions between the user and the devices.

- We extend the adversary capability in symbolic models to capture attacks caused by low-entropy keys, such as the key negotiation downgrade attack [5]. Specifically, we design an oracle that allows an adversary to know the secret key from an encrypted message if the entropy of that key is low.

- We perform a full-scale study of 29 pairing situations by instantiating the device in the general model with different device configurations. Our analysis confirms two existing attacks. Moreover, we disclose a new attack called keysize confusion attack.

- We propose a backward-compatible countermeasure to resist the attacks found in the BLE-SC pairing protocol and provide computer checked security proofs for the patched protocol. The proofs show that our countermeasure is sufficient to prevent the found attacks.

Responsible Disclosures: We have reported our findings and countermeasure to the Bluetooth Special Interest Group (SIG).

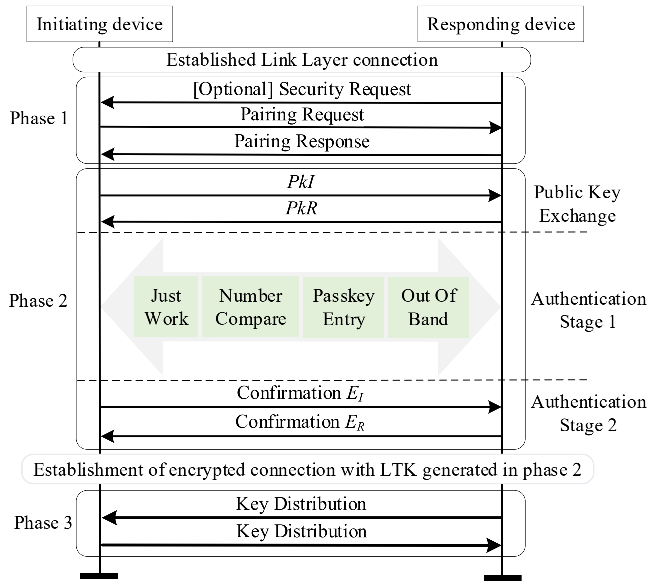


Figure 1: Overview of BLE-SC Pairing

The Bluetooth SIG has acknowledged our findings. They also claim that manufacturers are free to adopt our fixes during implementation, but it is atypical to be recommended or mandatory in the core specification since our fixes require additional steps from a user.

2 Bluetooth Pairing

We review the pairing protocols in the Bluetooth specification version 5.3 [16] and refer the interested readers to the specification for further details. We focus on the BLE-SC pairing protocol but also describe the difference between BLE-SC pairing and the SSP-SC pairing.

2.1 Overview of BLE-SC Pairing

BLE-SC pairing aims to establish keys to encrypt a link and distribute specific keys in the encrypted link to verify signed data and random address resolution. The BLE-SC pairing protocol is performed when two devices do not share a link key (for example, they are connected for the first time) or when the stored keys cannot satisfy the security requirements. The BLE-SC pairing protocol consists of the following three phases, as shown in Figure 1.

1. *Pairing Feature Exchange (Phase 1).* The devices exchange their authentication requirements and capabilities.

2. *Long Term Key (LTK) Generation (Phase 2).* The devices derive an LTK based on their authentication requirements and capabilities.

3. *Transport Specific Key Distribution (Phase 3).* The devices distribute specific keys in the link encrypted by the LTK.

Table 1: Specification-defined IO capabilities. The input and output capabilities are mapped to a single IO capability.

	Numeric Output①	No Output②
Keyboard Input③	KeyboardDisplay	KeyboardOnly
Yes-No Input④	DisplayYesNo	NoInputNoOutput
No Input⑤	DisplayOnly	NoInputNoOutput

①: Device has the ability to display a 6-digit number. ②: Device does not have the ability to display a 6-digit number. ③: Device has a numeric keyboard that can input the digits ‘0’ to ‘9’ and a confirmation. ④: Device has the ability to indicate “yes” and “no”. ⑤: Device does not have the ability to indicate “yes” or “no”.

2.2 Pairing Feature Exchange

This phase aims to determine the parameters used in the subsequent phases. It can be invoked by the initiating device sending a pairing request or the responding device sending a security request. The following four fields are important to our model. They are sent by the initiating/responding device in the pairing request/response.

- `IOCap`: This field indicates the specification-defined IO capability used in the LTK generation phase. The Bluetooth specification defines five IO capabilities of the device, as shown in Table 1.
- `OOB`: This field indicates whether the device has received authentication data using Out-Of-Band (OOB) capabilities, i.e., IO capabilities that are not defined in the specification.
- `MITM`: This field indicates whether the device requires preventing Man-In-The-Middle (MITM) attacks.
- `KeySize`: This field indicates the maximum encryption key size that the device can support, from 7 to 16 bytes.

2.3 LTK Generation

This phase aims to generate an LTK to establish an encrypted link between the initiating and the responding devices. The LTK phase consists of three stages, as shown in Figure 1.

2.3.1 Public Key Exchange

In this stage, the two devices run an Elliptic Curve Diffie-Hellman (ECDH) key exchange protocol to derive a Diffie-Hellman (DH) key. Specifically, suppose (PkI, SkI) and (PkR, SkR) are the public-private key pairs of the initiating and responding devices. After exchanging their public keys, the initiating and responding devices calculate their DH keys $DHKey_I = PkR^{SkI}$ and $DHKey_R = PkI^{SkR}$, respectively.

Algorithm 1 Association Model Selection.

```
Input:
    Fields $\texttt{OOB}_\texttt{I}$, $\texttt{MITM}_\texttt{I}$, $\texttt{IOCap}_\texttt{I}$ in pairing request
    Fields $\texttt{OOB}_\texttt{R}$, $\texttt{MITM}_\texttt{R}$, $\texttt{IOCap}_\texttt{R}$ in pairing response
Output:
    An association model $\mathcal{M}$
1: if $\texttt{OOB}_\texttt{I} == 1$ or $\texttt{OOB}_\texttt{R} == 1$ then
2:     $\mathcal{M}$ = OOB
3: else if $\texttt{MITM}_\texttt{I} == 0$ and $\texttt{MITM}_\texttt{R} == 0$ then
4:     $\mathcal{M}$ = JW
5: else
6:     $\mathcal{M}$ = mapIOCap2AssModel($\texttt{IOCap}_\texttt{I}$,$\texttt{IOCap}_\texttt{R}$)
```

2.3.2 Authentication Stage 1

In this stage, the two devices authenticate the public keys to determine whether the derived DH key should be used. This authentication process is controlled by association models and may involve the user’s participation.

Association Models. The Bluetooth specification defines four association models: Numeric Comparison (NC), Just Work (JW), Passkey Entry (PE), and Out-of-Band (OOB). The user needs to interact with the devices in the NC and PE association models. In the NC association model, the user compares two 6-digit numbers displayed on two devices and indicates “Yes” on the devices if the two numbers are equal. In the PE association model, the user inputs a self-chosen 6-digit number to the two devices respectively or inputs the number displayed on one device to another. The JW and OOB association models do not need the user to interact with the devices. However, the OOB association model relies on the two devices’ OOB channel to exchange authentication information. The OOB channel, such as an NFC connection, is always deemed as a secure channel that the adversary cannot access. The sequence diagrams of the four association models are shown in Appendix A.

The association model is chosen by Algorithm 1 during the execution of the pairing protocol. The function mapIOCap2AssModel in Algorithm 1 maps IO capabilities to an association model according to Table 2.

2.3.3 Authentication Stage 2

This stage confirms that both devices have successfully completed authentication stage 1. First, each device computes two keys, $MacKey$ and LTK, as shown in Eq. (1), where f5 is a pseudorandom function, $DHKey$ is either $DHKey_I$ or $DHKey_R$, and N_I and N_R are two random string exchanged in authentication stage 1. `I` and `R` are the device addresses of the initiating and responding devices, which are exchanged in the head of the Bluetooth packet.

$$MacKey||LTK = \mathsf{f5}(DHKey, N_I, N_R, \texttt{I}, \texttt{R}) \quad (1)$$

Table 2: Definition of mapIOCap2AssModel that maps the IO capabilities to an association model

$\texttt{IOCap}_\texttt{R}$ \ $\texttt{IOCap}_\texttt{I}$	DisplayOnly	DisplayYesNo	KeyboardOnly	NoInputNoOutput	KeyboardDisplay
DisplayOnly	JW	JW	PE	JW	PE
DisplayYesNo	JW	NC	PE	JW	NC
KeyboardOnly	PE	PE	PE	JW	PE
NoInputNoOutput	JW	JW	JW	JW	JW
KeyboardDisplay	PE	NC	PE	JW	NC

Then, two devices exchange their confirmations and verify the received confirmation by *MacKey*. The confirmation E_I of initiating device is computed by Eq. (2) and the confirmation E_R of the responding device is computed by Eq. (3), where f6 is a pseudorandom function and *ra* and *rb* are two numbers determined by the used association model.

$$E_I = \mathsf{f6}(MacKey, N_I, N_R, ra, \texttt{IOCap}_\texttt{I}, \texttt{I}, \texttt{R}) \tag{2}$$

$$E_R = \mathsf{f6}(MacKey, N_R, N_I, rb, \texttt{IOCap}_\texttt{R}, \texttt{R}, \texttt{I}) \tag{3}$$

Finally, LTK is resized to not exceed the shorter of the initiating and responding devices' `KeySize`.

2.4 Transport Specific Key Distribution

In this phase, the link is encrypted by LTK. The initiating and responding devices may distribute specific keys such as Identity Resolving Key (IRK) and Connection Signature Resolving Key (CSRK) in the encrypted link.

2.5 Overview of SSP-SC Pairing

The protocol flow of SSP-SC pairing and BLE-SC pairing are almost the same. The SSP-SC pairing also uses Algorithm 1 to choose the association model, but it does not support the KeyboardDisplay IO capability. In addition, the function mapIOCap2AssModel of SSP-SC is the same as Table 2 except that the last row and column are removed. As a result, the number of the pairing scenarios of SSP-SC pairing is fewer than BLE-SC pairing. The sequence diagrams of the association models in SSP-SC pairing are the same as that of the BLE-SC pairing, but some cryptographic functions are different. More detailed differences between the two pairing protocols are referred to the Bluetooth specification 5.3 [16].

3 Protocol Modeling

We first briefly introduce Tamarin Prover, the automatic symbolic tool used in this paper and then describe the assumptions and threat model. Finally, we present our formal model for the BLE-SC pairing protocol. We provide our models and results on our git repository [2].

3.1 Tamarin Prover

Tamarin Prover [33] is a powerful tool for symbolic modeling and analysis of security protocols. It has been successfully applied to real-world protocol analysis [8–10, 17–19, 21, 31, 32].

Tamarin Prover takes as input the specifications of the *protocol*, the *adversary*, and the desired *properties*. It can then automatically construct a proof that, even if arbitrarily many instances of the protocol's roles are interleaved in parallel, together with the adversary's actions, the protocol satisfies its specified properties.

Specify Protocol and Adversary. Tamarin Prover uses multiset rewriting rules to specify the concurrent execution of the protocol and the adversary as a transition system, where the state of the transition system is a multiset of facts and the initial state is an empty multiset. Multiset rewriting rules define how the system transitions from the current state to a new one. In Tamarin Prover, a rule consists of a name and three respective parts, each of which is a sequence of facts: *left-hand side*, *actions*, and *right-hand side*. The rule can only be executed if all the facts on its left-hand side are available in the current state. When a rule is executed, it will consume the facts on the left-hand side, i.e., removing them from the state, and produce facts on the right-hand side, i.e., adding them to the state. Note that facts are either linear or persistent. The linear fact can only be consumed once but the persistent fact can be consumed any number of times. Actions specify observable facts in every trace and are used to express security properties. There are three particular types of facts built in Tamarin Prover as follows.

- `Fr`: This fact means producing a fresh (random) value and is implicitly available in all states in Tamarin Prover.
- `Out`: This fact means sending a message to the public channel. The message will extend the adversary's knowledge.
- `In`: This fact means receiving a message from the public channel. The message is constructed by the adversary based on his/her knowledge.

Example 1. Tamarin Prover adopts the following two rules to model a step of a protocol and an adversary. In this step, a participant encrypts a message m under a key k with a symmetric encryption function senc and sends the ciphertext to the public channel, as defined in the following rule.

$$[\,\texttt{Fr}(\sim k),\texttt{Fr}(\sim m)\,]-[\,\texttt{Send}(\sim m)\,]\rightarrow[\,\texttt{Out}(\mathsf{senc}(\sim m,\sim k))\,]$$

Then, the adversary obtains that ciphertext from the public channel, as defined in the following rule.

$$[\,\texttt{Out}(x)\,]-[\,\texttt{K}(x)\,]\rightarrow[\,\texttt{!KD}(x)\,]$$

The fact KD, which has the persistent symbol '!' as the prefix, indicates that the adversary knows x. The actions Send and K are two observable facts that respectively indicate that a participant sends a message and the adversary knows x.

Specify Trace Properties. Trace properties, such as secrecy or variants of authentication, express that something never occurs on any trace of a protocol. Tamarin Prover specifies these properties as guarded first-order logic formulas with time points and uses actions to specify the formulas.

Example 2. The definition of secrecy requires that there is no trace where an adversary could know the secret message m. This property can be encoded as the following lemma, which means that if a participant sent a message m at time i, then there does not exist a time j at which the adversary knows that message.

```
All m #i. Send(m)@#i ==> not Ex #j. K(m)@#j
```

3.2 Assumptions and Threat Model

We make the following assumptions in our model.

- The OOB channel is a secure channel that provides confidentiality and integrity to the transferred data.
- The devices in the user's hand are honest. Furthermore, the devices always support the field KeySize that provides high entropy, such as 16 bytes.
- The user is honest and only has the following capabilities.
  - When given two numbers a and b, the user pushes "Yes" on the two devices in turn if $a = b$.
  - When given a number a and an input field, the user inputs a to the input field and pushes "Yes" if there is a confirm button on the display device.
  - When given two input fields, the user generates a fresh number a and inputs a to the two input fields in turn.

We consider an extension of the Dolev-Yao (DY) adversary [23] in the Bluetooth channel as our threat model. The adversary can intercept, modify, and falsify any message transferred in the channel. The two devices are paired in a trusted environment, which means that an adversary cannot see the display or input field. We do not consider untrusted environment [24] because the PE association model's security depends on the secrecy of the *passkey*. We abstract the field KeySize into two values: "low entropy" means that the adversary can implement brute-force attacks, and "high entropy" means that the adversary cannot. We provide an adversary with the capability of brute-force attacks on low entropy LTKs by defining an oracle. The oracle, modeled by the following rule, allows the adversary to know LTK from an encrypted message if the field KeySize is "low entropy".

$$\text{let } LTK = \mathsf{resize}(\_, \text{“low entropy”}) \text{ in}$$
$$[\,\texttt{In}(\mathsf{senc}(m, LTK))\,]-[\,]\rightarrow[\,\texttt{Out}(LTK)\,]$$

3.3 Comprehensive Formal Model

We give a comprehensive formal model of the BLE-SC pairing protocol. Our model covers all phases and association models in the specification and can capture sophisticated attacks such as the method confusion attack. We first give a general formal model which captures all possible executions of two devices by parameterizing their configuration. Then, we instantiate the device parameters to generate specific models for all pairing situations. Note that we only specify the protocol and the adversary in this section and leave the desired properties of the BLE-SC pairing protocol in Section 4.

3.3.1 General Model

The scale of the general model is huge, which makes the modeling task time-consuming and error-prone. To simplify the modeling task, we adopt the M4 macro processor [27]. Then, we can define a macro and expand this macro with different parameters to generate multiple rules.

For instance, we define the following macro for all branches that the responding device selects the NC association model.

```
define(Gen_NC_ResDCmt,
<!rule NC_ResDCommitment_$1_$2:
  let
    ...
    C_R = f4('g'^skR,DHpkI,~N_R,'0')
    IOCapabilityI = '$1'
    IOCapabilityR = '$2'
  in
  [ ResDDHKey(..., IOCapabilityI, ...,
    IOCapabilityR, ...),
    Fr(~N_R) ]
--[]->
  [ Out(<MacAddR,MacAddI,C_R>),
    NC_State_Res_Sent_Commitment(...,
    IOCapabilityI, ..., IOCapabilityR, ...)
  ]!>)
```

The two parameters $1 and $2 of the macro are the IO capabilities of the two devices from the view of the responding device and mapIOCap2AssModel(IOCapabilityI, `IOCapabilityR`) should be NC. The fact `ResDDHKey(..., IOCapabilityI, ..., IOCapabilityR, ...)` represents that the responding device has derived a DH key. This macro generates the following rule. First, the responding device generates a fresh value N_R. Then, the responding device calculates a commitment C_R using the function `f4`, sends C_R to the initiating device through the Bluetooth channel, and transfers to the next state. These steps are illustrated in Figure 3 in Appendix A.

By expanding the above macro with IO capabilities pairs which result in the NC association model, we can obtain four rules to model all branches in which the responding device selects the NC association model.

Interactions. There are interactions between the user and the devices in the NC and PE association models. Based on our assumptions (see Section 3.2), we model the interactions as data transferred in a secure channel that offers confidentiality and integrity. In other words, the 6-digit number is honestly displayed and can only be seen by the user; the user's input is faithfully and secretly received by the devices.

To model the interactions, we use facts `Out_S` and `In_S` to send and receive messages from the secure channel. For example, $\texttt{Out\_S}(\langle\text{“Display”, “Device”, “User”}\rangle, D, U, V)$ means that the device D displays a number V to the user U and $\texttt{In\_S}(\langle\text{“Display”, “Device”, “User”}\rangle, D, U, V)$ means that the user U sees the displayed number V from the device D.

Users. We abstract the user's behaviors by a state machine, as shown in Appendix B. We use six multiset rewriting rules to specify the state machine.

The following rule models the user's actions when the devices select the NC association model.

$$\begin{aligned}
&[\,\texttt{In\_S}(\langle\text{“DisYesNo”},\text{“Device”},\text{“User”}\rangle, D1, U, m),\\
&\ \ \texttt{In\_S}(\langle\text{“DisYesNo”},\text{“Device”},\text{“User”}\rangle, D2, U, m)\,]\\
&-[\,]\!\!\rightarrow\\
&[\,\texttt{Out\_S}(\langle\text{“Confirm”},\text{“User”},\text{“Device”}\rangle, U, D1, \text{‘Y’}),\\
&\ \ \texttt{Out\_S}(\langle\text{“Confirm”},\text{“User”},\text{“Device”}\rangle, U, D2, \text{‘Y’})\,]
\end{aligned}$$

In this rule, a user U is pairing his/her two devices $D1$ and $D2$. Both devices display the same number m and offer a “Yes” button and a “No” button. The user confirms by pushing the “Yes” buttons on the two devices.

The following two rules model the user's actions when the devices select the PE association model.

$$\begin{aligned}
&[\,\texttt{In\_S}(\langle\text{“AskInput”},\text{“Device”},\text{“User’}\rangle, D1, U, \text{“Input”}),\\
&\ \ \texttt{In\_S}(\langle\text{“AskInput”},\text{“Device”},\text{“User’}\rangle, D2, U, \text{“Input”}),\\
&\ \ \texttt{Fr}(\sim\! passkey)\,]\\
&-[\,]\!\!\rightarrow\\
&[\,\texttt{Out\_S}(\langle\text{“Input”},\text{“User”},\text{“Device”}\rangle, U, D1, \sim\! passkey),\\
&\ \ \texttt{Out\_S}(\langle\text{“Input”},\text{“User”},\text{“Device”}\rangle, U, D2, \sim\! passkey)\,]
\end{aligned}$$

$$\begin{aligned}
&[\,\texttt{In\_S}(\langle\text{“Display”},\text{“Device”},\text{“User’}\rangle, D1, U, passkey),\\
&\ \ \texttt{In\_S}(\langle\text{“AskInput”},\text{“Device”},\text{“User’}\rangle, D2, U, \text{“Input”})\,]\\
&-[\,]\!\!\rightarrow\\
&[\,\texttt{Out\_S}(\langle\text{“Input”},\text{“User”},\text{“Device”}\rangle, U, D2, passkey)\,]
\end{aligned}$$

In the first rule, the user is given two input fields. He/She generates a fresh number *passkey* and inputs it to the two devices. In the second rule, the device $D1$ displays a number *passkey*, and the device $D2$ gives the user an input field. Then, the user inputs *passkey* to the device $D2$. The second rule has a symmetric rule which models the situation that the device $D2$ displays a number *passkey* and the device $D1$ gives the user an input field.

We also model the user's Unexpected but Realistic (UR) behavior as follows.

$$\begin{aligned}
&[\,\texttt{In\_S}(\langle\text{“DisYesNo”},\text{“Device”},\text{“User’}\rangle, D1, U, passkey),\\
&\ \ \texttt{In\_S}(\langle\text{“AskInput”},\text{“Device”},\text{“User’}\rangle, D2, U, \text{“Input”})\,]\\
&-[\,]\!\!\rightarrow\\
&[\,\texttt{Out\_S}(\langle\text{“Confirm”},\text{“User”},\text{“Device”}\rangle, U, D1, \text{‘Y’}),\\
&\ \ \texttt{Out\_S}(\langle\text{“Input”},\text{“User”},\text{“Device”}\rangle, U, D2, passkey)\,]
\end{aligned}$$

One device displays a number *passkey* with a “Yes” button and a “No” button, and the other gives the user an input field. The user inputs *passkey* to the input field and pushes the “Yes” button. This rule also has a symmetric rule. Despite this behavior being unexpected in the specification, the user indeed performs this behavior when faced with this scenario according to the user study [36]. Basin et al. [11] model human misbehavior in security protocols that involve humans as endpoints; however, we model a human who aids the two devices to authenticate mutually.

OOB Channel. The devices transfer their authentication data through an OOB channel in the OOB association model. To verify the security of the OOB association model, we consider the OOB channel as a secure channel that resists MITM attacks (see Section 3.2). This assumption is inferred from the following statement in the Bluetooth specification [16].

> [Vol 3, Part H, 2.3.5.4] **Out of Band**: If the OOB communication is resistant to MITM attacks, then this association method is also resistant to MITM attacks.

Key Size. The key size of LTK is stated in the Bluetooth specification by the following two declarative sentences.

> [Vol 3, Part H, 2.3.4] **Encryption key size**: The shorter of the initiating and responding devices' maximum encryption key length parameters shall be used as the encryption key size.

> [Vol 3, Part H, 2.4.4.2] **Encryption setup using LTK**: The generated LTK size must not be longer than the negotiated encryption key size and its size may need to be shortened.

Table 3: Device configuration and expected association model

Number of Configurations	Device Configuration	Expected Association Model
25	$[\langle 0,0,0,0,1,1,\textbf{InitDIOCap},\textbf{ResDIOCap}\rangle]^{\dagger}$	$\textbf{AS}^{\ddagger}$
3	$\langle 1,0,0,1,-,-,\text{NoInputNoOutput},\text{NoInputNoOutput}\rangle$	OOB
	$\langle 0,1,1,0,-,-,\text{NoInputNoOutput},\text{NoInputNoOutput}\rangle$	OOB
	$\langle 1,1,1,1,-,-,\text{NoInputNoOutput},\text{NoInputNoOutput}\rangle$	OOB
1	$\langle 0,0,0,0,0,0,\text{KeyboardDisplay},\text{KeyboardDisplay}\rangle$	JW

$0,1,-$: False, true, false or true.
†: A list of length 25. **InitDIOCap**, **ResDIOCap** is a Cartesian product of the set of the defined IO capabilities.
‡: **AS** = mapIOCap2AssModel(**InitDIOCap**, **ResDIOCap**), where mapIOCap2AssModel is defined in Table 2.

In our model, the maximum encryption key size is either "low entropy" or "high entropy". When the two devices send pairing request/response messages, they set the `KeySize` field to "high entropy". Note that if a device only supports a "low-entropy" value, an attacker can obtain the LTK by brute-force attacks regardless of flaws in the protocol design. Since our goal is to find design flaws, we assume that both honest devices support a "high-entropy" value. Once all the devices' maximum encryption key sizes in our model are configured as "high entropy" (see Section 3.2), the encryption key size of a device can be directly set to the peer device's maximum encryption key size. In other words, we resize the LTK of a device as $\mathsf{resize}(\_, \texttt{KeySize}_{Peer})$.

3.3.2 Specific Model

To verify the security of specific pairing situations, we use device configurations to instantiate the general model and obtain a specific model for each pairing situation. A device configuration is in the form ⟨ InitDOOBOutCap, InitDOOBInCap, ResDOOBOutCap, ResDOOBInCap, MITMofInitiator, MITMofResponder, InitDIOCap, ResDIOCap ⟩.

- InitDOOBOutCap (resp., ResDOOBOutCap): Whether the initiating (resp., responding) device has the capability to send data to an OOB channel.
- InitDOOBInCap (resp., ResDOOBInCap): Whether the initiating (resp., responding) device has the capability to receive data from an OOB channel.
- MITMofInitiator (resp., MITMofResponder): Whether the initiating (resp., responding) device needs the result connection resistant to MITM attacks.
- InitDIOCap (resp., ResDIOCap): The IO capability of the initiating (resp., responding) device.

As a result, we generate 29 specific models from the general model with parameters shown in Table 3. These specific models cover all pairing situations in the specification. Each specific model corresponds to one specific expected association model and comprises 112 rules.

4 Property Specifying

We specify trace properties desired in our model, including executability property and security properties. Executability property guarantees that our model behaves as expected in the specification without adversary involvement. Specifically, in each specific model, we assess whether two devices execute the expected association model and finish the pairing protocol.

We extract and interpret the security properties that the BLE-SC pairing protocol should satisfy. Note that the Bluetooth specification [16] only explicitly requires Man-In-The-Middle (MITM) protection, as shown below.

> [Vol 3, Part H, 2.3.1] **Security Properties**: In LE Secure Connections pairing, Authenticated man-in-the-middle (MITM) protection is obtained by using the passkey entry pairing method or the numeric comparison method or may be obtained using the out-of-band pairing method.

4.1 Authentication

The authentication property in the symbolic model is generally formalized by corresponding properties while formalized as session matching in the computational model. We specify the authentication property as the *non-injective agreement* in Lowe's hierarchy [30]. In the original non-injective agreement, two facts `Commit` and `Running` are introduced. The `Commit` fact represents an agent's belief about its communication peer's local state, whereas `Running` represents the peer's actual state. The non-injective agreement property is formalized by the corresponding of the two facts. The following lemma is used to specify the non-injective agreement in Tamarin, which means that if an agent `I` believes that a message `t` was sent by an agent `R`, then `t` was indeed sent by `R`.

```
All I R t #i. Commit(I,R,t) @#i
```

```
    ==> (Ex #j. Running(R,I,t) @#j)
```

The authentication property in the BLE-SC pairing protocol is somewhat different from the original one. The Bluetooth specification does not require the device which performs the JW association model to satisfy the non-injective agreement. Informally, the authentication property is represented as follows. If a device believes it has agreed on a key with the peer, then the peer runs the protocol with the same key, unless the device performs the JW association model. The following lemma refers to agreement of the key `K` with device `R` from the device `I`'s perspective.

```
(not Ex #t. IJW()@#t) ==>
  All I R K #i. Commit(I,R,K) @#i
    ==> (Ex #j. Running(R,I,K) @#j)
```

To verify the authentication of the BLE-SC pairing protocol, we specify the authentication of $DHKey$ and LTK from the initiating and responding devices' perspectives respectively. Specifically, the predicate **P1**(I,R,$DHKey$) (resp., **P2**(R,I,$DHKey$)) refers to the agreement of the $DHKey$ with responding device R (resp., initiating device I) from the initiating device I's (resp., responding device R's) perspective, and **P3**(I,R,LTK) (resp., **P4**(R,I,LTK)) refers to the agreement of the LTK with responding device R (resp., initiating device I) from the initiating device I's (resp., responding device R's) respective. We list the lemmas of the four authentication properties (**A1 - A4**) in Appendix C.

4.2 MITM Protection

In MITM attacks, an adversary intercepts and modifies messages sent between honest devices executing the protocol and makes each honest device derive and share an LTK with him/her. The Bluetooth specification [16] describes the MITM attack by the following sentences.

> [Vol 1, Part A, 5.2.3] **Man-in-the-middle protection**: ❶A man-in-the-middle (MITM) attack occurs when a user wants to connect two devices but instead of connecting directly with each other they unknowingly connect to a third (attacking) device that plays the role of the device they are attempting to pair with. ❷The third device then relays information between the two devices giving the illusion that they are directly connected.

The sentence ❶ means that the two devices are not agreement of the session key, representing as $\neg\text{P3}(\text{I},\text{R},LTKI) \wedge \neg\text{P4}(\text{R},\text{I},LTKR)$. The sentence ❷ means that the adversary knows the session key of the two devices, representing as $\texttt{K}(LTKI) \wedge \texttt{K}(LTKR)$. We represent the MITM protection property between I and R as MITMP(I,R). The appearance of MITM attacks means the violation of the MITMP property, representing as $\neg$MITMP(I,R). The MITM protection in the Bluetooth specification can be formally described as

$$\neg\text{MITMP}(\text{I},\text{R}) \Leftrightarrow \texttt{Ex LTKI LTKR \#i \#j.}\\ \neg\text{P3}(\text{I},\text{R},LTKI) \wedge \neg\text{P4}(\text{R},\text{I},LTKR)\\ \wedge \texttt{K}(LTKI)@i \wedge \texttt{K}(LTKR)@j;$$

or equivalently

$$\text{MITMP}(\text{I},\text{R}) \Leftrightarrow not(\texttt{Ex LTKI LTKR \#i \#j.}\\ \neg\text{P3}(\text{I},\text{R},LTKI) \wedge \neg\text{P4}(\text{R},\text{I},LTKR)\\ \wedge \texttt{K}(LTKI)@i \wedge \texttt{K}(LTKR)@j).$$

This means that the MITM protection property is satisfied iff there is not exist a case that sentences 1 and 2 simultaneously occur. We list the guarded first-order logic formula of the MITMP property in Appendix C.

4.3 LTK Confusion Protection

Zhang et al. [38] have described a LTK confusion attack, which their called Denial of Service (DoS) attack. In LTK confusion attacks, an adversary induces the two devices to derive different LTKs $(\neg\text{P3}(\text{I},\text{R},LTKI) \wedge \neg\text{P4}(\text{R},\text{I},LTKR))$, but he/she cannot know both LTKs $(\neg(\texttt{K}(LTKI) \wedge \texttt{K}(LTKR)))$. In this case, the adversary cannot perform MITM attacks. Both devices treat this pairing as a successful pairing. However, the two devices cannot mutually communicate directly or indirectly despite they have completed the protocol, which leads to a DoS attack. To resume the communication blocked by this kind of DoS attack, the user needs to delete a previously stored association and pair again to re-establish a pairing session between devices, when the attacker is out of range. We represent the two devices I and R confuse the session keys $LTKI$ and $LTKR$ as LTKCP(I,R,$LTKI$,$LTKR$). The appearance of the LTK confusion attack means the violation of the LTKCP property, representing as $\neg$LTKCP(I,R,$LTKI$,$LTKR$). We formally specify the LTK confusion protection property as

$$\neg\text{LTKCP}(\text{I},\text{R},LTKI,LTKR) \Leftrightarrow\\ \neg\text{P3}(\text{I},\text{R},LTKI) \wedge \neg\text{P4}(\text{R},\text{I},LTKR)\\ \wedge \neg(\texttt{Ex \#i \#j.}\texttt{K}(LTKI)@i \wedge \texttt{K}(LTKR)@j);$$

or equivalently

$$\text{LTKCP}(\text{I},\text{R},LTKI,LTKR) \Leftrightarrow\\ \text{P3}(\text{I},\text{R},LTKI) \vee \text{P4}(\text{R},\text{I},LTKR)\\ \vee (\texttt{Ex \#i \#j.}\texttt{K}(LTKI)@i \wedge \texttt{K}(LTKR)@j).$$

This means that the LTK confusion protection property is satisfied iff the adversary knows both LTKs of I and R or agreement of the key must hold from I's or R's perspective. We list the guarded first-order logic formula of the LTKCP property in Appendix C.

4.4 Secrecy of Authenticated LTK

Although the specification does not claim this property, the authenticated key should be kept secret from the adversary; otherwise, the adversary can obtain all messages in the encrypted link. We define agreement of the LTK with responding device R (resp., initiating device I) from the initiating device I's (resp., responding device R's) perspective as **P5**(I, R,LTK) (resp., **P6**(R,I,LTK)). We formally define the secrecy property of authenticated LTK between I and R, representing as AUTHSEC$(\mathrm{I},\mathrm{R},LTK)$, as

$$\mathrm{AUTHSEC}(\mathrm{I},\mathrm{R},LTK) \Leftrightarrow$$
$$\mathbf{P5}(\mathrm{I},\mathrm{R},LTK)\ \&\ \mathbf{P6}(\mathrm{R},\mathrm{I},LTK)\ \Rightarrow\ \neg(\texttt{Ex \#i.K}(LTK)@i).$$

This formula means that if LTK is authenticated, then it cannot be known by the adversary. We list the guarded first-order logic formula of the secrecy property in Appendix C.

5 Results

We perform a full-scale study of the pairing protocol by analyzing 29 specific models on a server running Ubuntu 18.04 with two Intel(R) Xeon(R) Gold 5218 @2.30GHz CPUs and 320GB RAM. We summarize the results in Table 4. We find that 7 pairing situations violate the MITM protection property. The counter-example traces show the *method confusion attack*. All pairing situations except situations which perform JW association model violate the LTK confusion protection property and all pairing situations violate the secrecy of authenticated LTK property. The counter-example traces that violate the former show the *keysize confusion attack*, while the counter-example traces that violate the latter show the *key negotiation downgrade attack*.

Method Confusion Attack. In an ideal BLE-SC pairing protocol, the initiating and responding devices choose the same association model. If the NC or PE association models are chosen, the connection is MITM protection. However, an adversary may manipulate the pairing request and response to force the devices to execute different association models. This kind of MITM attack is called method confusion attack and was first found by Tschirschnitz et al. [36].

Our model automatically finds method confusion attacks in 7 pairing situations as summarized in Table 5. The results demonstrate that Tschirschnitz et al. [36] have found all affected situations. The counter-example traces of MCA-PN indicate that the adversary changes the `IOCap` field in the pairing request to force the responding device to choose the NC association model and impersonates the initiating device to interact with the responding device. Correspondingly, the adversary tampers the pairing response to force the initiating device to select the PE association model and impersonates the responding device. The responding device displays a number with a "Yes" button and a "No" button; the initiating device asks the user to input a number. The user inputs the displayed number in the responding device to the initiating device as the *passkey* in the PE association model and pushes "Yes" on the responding device. Finally, the adversary shares two LTKs with the two devices.

Key Negotiation Downgrade Attack. All devices in our model support 16 bytes of maximum encryption key size. Ideally, the size of the result LTK should be 16 bytes. However, an adversary can downgrade the size of the result LTK to 7 bytes. Antonioli et al. [5] first disclose this attack and call it the key negotiation downgrade attack.

Our model automatically discovers the key negotiation downgrade attack, which violates the secrecy of authenticated LTK. The counter-example traces indicate that the adversary manipulates the field `KeySize` to 7 bytes when two devices exchange messages in the pairing feature exchange phase. In the LTK generation phase, the two devices reduce the size of LTK to 7 bytes. This low-entropy LTK offers limited security assurance and is vulnerable to brute force attacks.

Keysize Confusion Attack. We discover this new attack in our model, in which the adversary induces the two honest devices to derive the LTKs with different key sizes. In this attack, the adversary only modifies the `KeySize` fields on the pairing request/response messages in the pairing feature exchange phase. The security mechanism in the BLE-SC pairing protocol cannot detect the disparity of the key sizes. The two honest devices derive the different LTKs after finishing the pairing protocol. Since the adversary cannot know the LTKs of the honest devices, the confidentiality of the communication is not compromised. However, this attack results in an invalid bonding without notification to the user. Comparing to other DoS attacks in which the adversary blocks/jams messages, the invalid bonding cannot self-correct without the user's action. The user needs to delete a previously stored association in the system setting of both devices and pair again to re-establish a pairing session between peers when the adversary is absent.

6 Patching

We first review existing countermeasures and then give our countermeasure that can resist the found attacks. We formally verify the proposed countermeasure to prove the security. Finally, we discuss the countermeasure.

6.1 Countermeasure Review

We introduce the countermeasures that have been proposed to mitigate the method confusion attack and the key negotiation downgrade attack. The patch of Bluetooth should consider the *backward compatibility*, which means the countermeasures should take into account the pairing of a patched device and an unpatched device. Unfortunately, the countermeasures proposed by Tschirschnitz et al. [36] and Antonioli et al. [5] require a significant change in protocol or the device interfaces

Table 4: Analysis Results of BLE-SC Pairing

No.	Pairing Situation	Executability	A1	A2	A3	A4	P1	P2	C
1	DisplayYesNo-DisplayYesNo	NC	✓	✓	✗	✗	✓	✗	✗
2	DisplayYesNo-KeyboardDisplay	NC	✗	✗	✗	✗	✗	✗	✗
3	KeyboardDisplay-KeyboardDisplay	NC	✗	✗	✗	✗	✗	✗	✗
4	KeyboardDisplay-DisplayYesNo	NC	✗	✗	✗	✗	✗	✗	✗
5	DisplayOnly-KeyboardOnly	PE	✓	✓	✗	✗	✓	✗	✗
6	DisplayOnly-KeyboardDisplay	PE	✓	✓	✗	✗	✓	✗	✗
7	DisplayYesNo-KeyboardOnly	PE	✗	✗	✗	✗	✗	✗	✗
8	KeyboardOnly-DisplayOnly	PE	✓	✓	✗	✗	✓	✗	✗
9	KeyboardOnly-DisplayYesNo	PE	✗	✗	✗	✗	✗	✗	✗
10	KeyboardOnly-KeyboardOnly	PE	✓	✓	✗	✗	✓	✗	✗
11	KeyboardOnly-KeyboardDisplay	PE	✗	✗	✗	✗	✗	✗	✗
12	KeyboardDisplay-DisplayOnly	PE	✓	✓	✗	✗	✓	✗	✗
13	KeyboardDisplay-KeyboardOnly	PE	✗	✗	✗	✗	✗	✗	✗
14-25	Other Pairs (12)	JW	✓	✓	✓	✓	✓	✓	✗
26	No-OOB-No-MITM	JW	✓	✓	✓	✓	✓	✓	✗
27	Uni-direction OOB ($\mathtt{I} \rightarrow \mathtt{R}$)	OOB	✓	✓	✗	✗	✓	✗	✗
28	Uni-direction OOB ($\mathtt{I} \leftarrow \mathtt{R}$)	OOB	✓	✓	✗	✗	✓	✗	✗
29	Bi-direction OOB ($\mathtt{I} \leftrightarrow \mathtt{R}$)	OOB	✓	✓	✗	✗	✓	✗	✗

No. 1-25: the two devices cannot communicate OOB data, and at least one device needs resistance to MITM attacks.
No. 26: the two devices cannot communicate OOB data, and both do not need resistance to MITM attacks.
No. 27-29: the two devices can communicate OOB data.
✓: property verified ✗: property falsified A1-A4: authentication properties
P1: MITM protection. P2: LTK confusion protection. C: secrecy of authenticated LTK.

Table 5: Method Confusion Attacks

$\mathtt{IOCap}_\mathtt{I}$ \ $\mathtt{IOCap}_\mathtt{R}$	Display YesNo	Keyboard Only	Display Keyboard
DisplayYesNo	N/A	MCA-PN	MCA-PN
KeyboardOnly	MCA-NP	N/A	MCA-NP
KeyboardDisplay	MCA-NP	MCA-PN	MCA-NP*

MCA-NP: method confusion attack, by letting the initiating device performs the NC association model and the responding device performs the PE association model.
MCA-PN: method confusion attack, by letting the initiating device performs the PE association model and the responding device performs the NC association model.
∗: There is a symmetrical attack MCA-PN, but Tamarin Prover only throws one attack trace.

and affect backward compatibility significantly. Therefore, the Bluetooth SIG does not incorporate those countermeasures into the specification.

6.1.1 Countermeasures for Method Confusion Attack

Tschirschnitz et al. [36] propose countermeasures to the protocol itself by letting a *passkey* used by the PE association model have to be distinctly distinguishable to a number displayed in the NC association model. For instance, the devices use the Latin letters for *passkey* in the PE association model but 6-digit numbers for the NC association model. They also propose three countermeasures that do not require changes to the BLE-SC pairing protocol.

- Enforcing Pairing Method: setting the devices' IO capabilities to the pairing situations deemed secure. Only specific Bluetooth devices can implement this countermeasure.
- User Interface Design Hotfix: designing a UI to warn the user against misusing the number presented. This countermeasure heavily relies on the display capability of the device, which needs to display not only a 6-digit number but also a character string.
- Authenticating Association Model: relying on the user to be aware of the association model and confirm that the two de-

vices use the same association model. This countermeasure reduces the user's experience since the user needs to not only compare the numbers or input numbers but also recognize and compare the association models of both devices. This countermeasure makes a strong security assumption that the user has a strong security awareness.

6.1.2 Countermeasures for Key Negotiation Downgrade Attack

We review the countermeasures proposed by Antonioli et al. [5] to fix the key negotiation downgrade attack.

- Higher Minimum Entropy: setting the minimum entropy, which is used to restrict the minimum entropy of the encryption key, to a higher value. For example, the devices set their minimum entropy to 16 bytes. The connection will be rejected once the entropy of the resulting LTK is lower than 16 bytes. However, if one device's maximum encryption key size is lower than the other device's minimum entropy, the two devices cannot be connected.

- Remove Entropy Negotiation: removing entropy negotiation from the pairing protocol. The LTK's entropy does not rely on the two devices' maximum encryption key size but is set to a fixed size such as 16 bytes. This countermeasure affects backward compatibility.

6.2 Patched BLE-SC pairing

The root cause of the found three attacks is that there is no mechanism to authenticate the pairing request and response messages. As a result, the adversary can modify the fields `IOCap` or `KeySize` to perform attacks. Redesigning association models from scratch will affect the backward compatibility of the protocol. Therefore, we add a patch to the current association models to fix the security flaws and maintain backward compatibility. The OOB association model can be easily patched by authenticating the field `KeySize` through the OOB channel. Thus, we do not propose a countermeasure to patch the OOB association model.

6.2.1 Patched NC

We describe the patched NC association model and show its sequence diagram in Figure 2a.

1. The two devices perform the standard NC association model. Specifically, they displayed numbers V_I and V_R (refer to Figure 3, in Appendix A) with a "Yes" button and a "No" button. The user pushes the "Yes" button in the two devices if $V_I = V_R$.

2. The initiating device computes $H_I = \mathsf{g3}(V_I||req||rsp)$, where *req* and *rsp* are pairing request and response messages and g3 is a function that outputs a 6-digit number, and displays H_I with a "Yes" button and a "No" button.

3. The responding device computes $H_R = \mathsf{g3}(V_R||req||rsp)$ and displays H_R with a "Yes" button and a "No" button.

4. The user compares H_I and H_R, which are two 6-digit numbers, then pushes the "Yes" button if $H_I = H_R$.

6.2.2 Patched PE

The PE association model has two pairing situations.
Display-Input situation. In this situation, one device (suppose the responding device) displays a number *passkey* while the other (suppose the initiating device) asks the user to input the displayed number. We describe the patched PE association model and show its sequence diagram in Figure 2b.

1. The user inputs *passkey* and the standard PE association model is executed (refer to Figure 4, in Appendix A).

2. If the standard PE association model is success, the responding device computes $D_R = \mathsf{g3}(passkey||req||rsp)$ and displays D_R.

3. In the meanwhile, the initiating device computes $H = \mathsf{g3}$ $(passkey||req||rsp)$ and asks the user to inputs D_R.

4. After inputting, the initiating device checks if $H = D_R$.

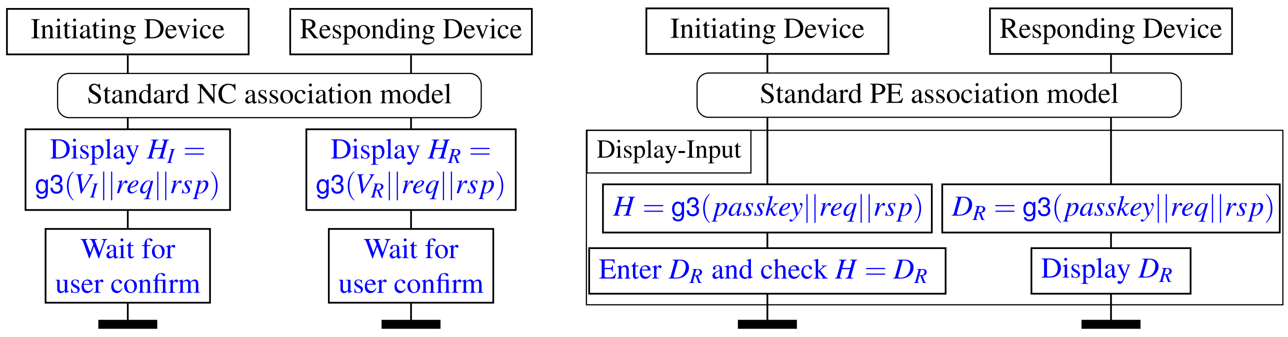
Input-Input situation. In this situation, both two devices ask the user to input a number. The user generates a *passkey* and inputs it to the input fields of both devices. The two devices then execute the standard PE association model.

6.3 Analysis of Patched BLE-SC Pairing

We formally model our patched BLE-SC pairing protocol and verify the model using Tamarin Prover. This model make the same assumptions as Sec. 3.2. We show the results in Table 6.

The patched NC association model satisfies the MITM protection, LTK confusion protection, and secrecy of authenticated LTK properties. These security properties are satisfied in the display-input situation for the patched PE association model. In the pairing situation that the two devices ask for input, the MITM resistance property is satisfied, but the DoS resistance and the secrecy of the authenticated LTK properties are violated. In other words, an adversary can still perform the keysize confusion attack and the key negotiation downgrade attack when two devices with KeyboardOnly as their IO capability are running the patched BLE-SC pairing protocol. To complement our countermeasure, we recommend dividing more fine-grained security levels for LTK and no longer trust LTK negotiated by performing the PE association model in the situation that the two devices ask for input.

The violations of authentication properties A1, A2, A3, and A4 only induce the one-side pairing. In this attack, one device is pairing successfully while the other fails. Users can easily detect the exception and then restart pairing the two devices. Furthermore, resistance to the one-side pairing is not the security goal of the Bluetooth specification.


Initiating Device
Responding Device
Standard NC association model
Display $H_I =$ g3$(V_I||req||rsp)$
Display $H_R =$ g3$(V_R||req||rsp)$
Wait for user confirm
Wait for user confirm
Initiating Device
Responding Device
Standard PE association model
Display-Input
$H =$ g3$(passkey||req||rsp)$
$D_R =$ g3$(passkey||req||rsp)$
Enter D_R and check $H = D_R$
Display D_R

(a) Patched NC (b) Patched PE

Figure 2: Patched BLE-SC pairing

Table 6: Analysis Results of Patched BLE-SC Pairing

No.	IOCap-Pair(I-R)	Executability	A1	A2	A3	A4	P1	P2	C
1	DisplayYesNo-DisplayYesNo	Patched NC	✓	✓	✓	✓	✓	✓	✓
2	DisplayYesNo-KeyboardDisplay	Patched NC	✗	✓	✗	✓	✓	✓	✓
3	KeyboardDisplay-DisplayYesNo	Patched NC	✓	✗	✓	✗	✓	✓	✓
4	KeyboardDisplay-KeyboardDisplay	Patched NC	✗	✗	✗	✗	✓	✓	✓
5	DisplayOnly-KeyboardOnly	Patched PE	✓	✓	✓	✓	✓	✓	✓
6	DisplayOnly-KeyboardDisplay	Patched PE	✓	✓	✓	✓	✓	✓	✓
7	DisplayYesNo-KeyboardOnly	Patched PE	✗	✗	✗	✗	✓	✓	✓
8	KeyboardOnly-DisplayOnly	Patched PE	✓	✓	✓	✗	✓	✓	✓
9	KeyboardOnly-DisplayYesNo	Patched PE	✗	✗	✗	✗	✓	✓	✓
10	KeyboardOnly-KeyboardOnly	Patched PE	✓	✓	✗	✗	✓	✗	✗
11	KeyboardOnly-KeyboardDisplay	Patched PE	✗	✗	✗	✗	✓	✓	✓
12	KeyboardDisplay-DisplayOnly	Patched PE	✓	✓	✓	✓	✓	✓	✓
13	KeyboardDisplay-KeyboardOnly	Patched PE	✗	✗	✗	✗	✓	✓	✓

✓:property verified ✗:property falsified A1-A4: authentication properties
R1: MITM protection. R2: LTK confusion protection. C: secrecy of authenticated LTK.

6.4 Discussion of the Patch

User Experience. The patched BLE-SC pairing improves the security but requires more interactions which may degrade the user experience. The user interacts with the devices only once in the standard NC and PE association models. However, in the patched NC and PE association models, the devices ask the user to interact with them twice. We deem that the reasonable additional interactions are acceptable. The two devices only perform the pairing protocol almost when they connect for the first time. Once the pairing is complete, the devices store the LTK to construct a secure channel when the devices are connected next time. The user rarely needs to involve in the pairing process again.

Backward Compatible. The backward compatibility cannot increase the security of the standard association models but offers the connection of the two devices.

- **Numeric Comparison**: When the device $D1$ is patched and the other device $D2$ is unpatched, the device $D1$ computes H_1 and displays H_1 with a "Yes" button and a "No" button but the device $D2$ does not display anymore. In this situation, the user is aware the device $D1$ is pairing with an unpatched device and can push the "Yes" button to continue the pairing or push the "No" button to interrupt the pairing. When the user continues the pairing process, the security level is downgraded to the standard NC association model.

- **Passkey Entry**: There are three backward compatibility cases of the PE association model, shown as follows.
  - Pairing a patched input device with an unpatched display device. After the devices finish the standard PE association model, the input device computes H and asks the user to input a number displayed on the display device. However, the unpatched display device no longer

computes and displays D_R. The user can cancel the connection or input a reserved number (e.g., "999999") to infer the input device to skip the comparison.

– Pairing an unpatched input device with a patched display device. After finishing the standard PE association model, the display device computes and displays D_R. The input device starts authentication stage 2 and sends its confirmation E_I. The display device escapes waiting for the user's input once it has received the confirmation.

– Pairing a patched input device with an unpatched input device. In this situation, the two devices execute the standard PE association model.

Summary. Our patched BLE-SC pairing sacrifices the user experience to enhance the security of the pairing process while guaranteeing the compatibility of old devices. Our countermeasure only assumes that the user has capabilities to compare and input numbers. In addition, we do not need the devices to have the capability to display any suggestive words. In conclusion, the patched NC and PE association models could be incorporated into the current Bluetooth specification.

7 Related Work

We briefly introduce the formal and informal works on the security of Bluetooth protocols.

Formal Works. Chang et al. [13] present the first automatic formal analysis for the NC association model using ProVerif [12] and Jia et al. [26] conduct the analysis with the Murphi model checker [22]. Lindell [29] also analyzes the security of the NC association model but in the computational model. Sethi et al. [34] enhance the previous model of the Bluetooth SSP to verify the NC and OOB association models. However, the two association models are modeled severally. Cremers et al. [20] also reduce the "perfect cryptography" assumption, but their work is orthogonal to ours since they model an algebraic attack while we model brute-force attacks. Troncoso et al. [35] computationally analyze the PE association model of SSP-SC. Unfortunately, the above formal studies consider only the individual association model. Their works cannot cover the attacks which interplay between the different association models.

A recent independent study by Wu et al. [37] is the first model that considers all available association models. However, their model does not cover all phases of the pairing process. In contrast, we give the first comprehensive formal model covering all available association models and all phases of the BLE-SC pairing protocol. As a result, we find the method confusion attack in 7 pairing situations but Wu's model can only detect the method confusion attack when the PE and NC association models interplay. Our analysis discloses the key negotiation downgrade attack and the keysize confusion attack caused by adversaries tampering with the field `KeySize` in the pairing feature exchange phase, while Wu's model does not find these attacks. In addition, we propose a countermeasure and formally verify it.

Fischlin et al. [25] prove that the Bluetooth connection is indeed secure if the pairing protocol is not attacked. Our work investigates the hypotheses of their study.

Informal Works. The attacks against pairing protocols found by the informal works [7, 28, 38] rely on the reused *passkey* or compromised device, which is beyond our assumptions. The method confusion attack [36] and the key negotiation downgrade attack [5] are disclosed by Tschirschnitz et al. and Antonioli et al. respectively. They also propose countermeasures to prevent the attacks. However, Bluetooth SIG does not accept their countermeasures because of the backward compatibility. Our model automatically discovers these two attacks. We also propose a countermeasure that is backward compatible and analyze the security.

Antonioli et al. disclose and exploit the flaws of the secure connection establishment process in Bluetooth Classic. Key Negotiation Of Bluetooth (KNOB) attacks [3] target the session key negotiation and allow an adversary to control the entropy of the session key. Bluetooth Impersonation AttackS (BIAS) [4] target the link key authentication and allow an adversary to impersonate a legal device without possessing the link key. These studies aim at the protocols after pairing protocols and are beyond the scope of our research. Antonioli et al. [6] also find vulnerabilities in cross-transport key derivation (CTKD), which enables devices to establish BT and BLE security keys by pairing just once. The CTKD is deprecated in the last Bluetooth Specification v5.3 [16].

8 Discussion and Conclusion

Discussion on SSP-SC. Our model of the BLE-SC pairing protocol can be easily adjusted to model the SSP-SC pairing protocol. We also provide the models of the SSP-SC pairing protocol in our anonymous website [2]. There are only 20 specific models for the pairing situations in the SSP-SC pairing protocol. Our analysis of the SSP-SC pairing protocol shows that two pairing situations are vulnerable to method confusion attacks. We do not describe the details of the models and the results of the SSP-SC pairing protocol in this paper. The patched PE and NC association models can apply to the SSP-SC pairing protocol to improve security.

Conclusion. We present the first comprehensive formal model, which covers all phases and available association models of the BLE-SC pairing protocol, to analyze the BLE-SC pairing protocol. Using Tamarin Prover, we automatically find many security flaws, leading to method confusion attack, key negotiation downgrade attack, and keysize confusion attack. We propose a backward compatibility countermeasure to fix the security flaws. Moreover, we provide computer checked security proofs for the countermeasure. We hope that our work will improve the security of the BLE-SC pairing protocol.

Acknowledgments

This research was supported in part by the National Key R&D Program of China under grant No. 2021YFB2700200, the Fundamental Research Funds for the Central Universities under grants No. 2042022kf1195, 2042022kf0046, the National Natural Science Foundation of China under grants No. 62172303, 62076187, the Key R&D Program of Hubei Province under grant No. 2022BAA039, and Key R&D Program of Shandong Province under grant No. 2022CXPT055. The corresponding authors are Jing Chen and Kun He.

References

[1] Bluetooth Market Update, 2022. https://www.bluetooth.com/2022-market-update/.

[2] Models and Results, 2023. https://github.com/luojiazhishu/BLE-SC-Pairing-Model.

[3] Daniele Antonioli, Nils Ole Tippenhauer, and Kasper Rasmussen. The KNOB is broken: Exploiting low entropy in the encryption key negotiation of bluetooth BR/EDR. In *Proc. of USENIX Security Symposium*, 2019.

[4] Daniele Antonioli, Nils Ole Tippenhauer, and Kasper Rasmussen. BIAS: bluetooth impersonation attacks. In *Proc. of S&P*, 2020.

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

[6] Daniele Antonioli, Nils Ole Tippenhauer, Kasper Rasmussen, and Mathias Payer. Blurtooth: Exploiting cross-transport key derivation in bluetooth classic and bluetooth low energy. In *Proc. of AsiaCCS*, 2022.

[7] Johannes Barnickel, Jian Wang, and Ulrike Meyer. Implementing an attack on bluetooth 2.1+ secure simple pairing in passkey entry mode. In *TrustCom*, 2012.

[8] David Basin, Jannik Dreier, Lucca Hirschi, Saša Radomirovic, Ralf Sasse, and Vincent Stettler. A formal analysis of 5g authentication. In *Proc. of CCS*, 2018.

[9] David Basin, Ralf Sasse, and Jorge Toro-Pozo. Card brand mixup attack: Bypassing the PIN in non-visa cards by using them for visa transactions. In *Proc. of USENIX Security Symposium*, 2021.

[10] David Basin, Ralf Sasse, and Jorge Toro-Pozo. The EMV standard: Break, fix, verify. In *Proc. of S&P*, 2021.

[11] David A. Basin, Sasa Radomirovic, and Lara Schmid. Modeling human errors in security protocols. In *IEEE 29th Computer Security Foundations Symposium, CSF 2016*, 2016.

[12] Bruno Blanchet. Automatic proof of strong secrecy for security protocols. In *Proc. of S&P*, 2004.

[13] Richard Chang and Vitaly Shmatikov. Formal analysis of authentication in bluetooth device pairing. *FCS-ARSPA'07*, 45, 2007.

[14] Bluetooth Specification Contributors. Bluetooth Core Specification 4.0, June 2010.

[15] Bluetooth Specification Contributors. Bluetooth Core Specification 5.2, December 2019.

[16] Bluetooth Specification Contributors. Bluetooth Core Specification 5.3, July 2021.

[17] Cas Cremers and Martin Dehnel-Wild. Component-based formal analysis of 5g-aka: Channel assumptions and session confusion. In *Proc. of NDSS*, 2019.

[18] Cas Cremers, Marko Horvat, Jonathan Hoyland, Sam Scott, and Thyla van der Merwe. A comprehensive symbolic analysis of tls 1.3. In *Proc. of CCS*, 2017.

[19] Cas Cremers, Marko Horvat, Sam Scott, and Thyla van der Merwe. Automated analysis and verification of tls 1.3: 0-rtt, resumption and delayed authentication. In *Proc. of S&P*, 2016.

[20] Cas Cremers and Dennis Jackson. Prime, order please! revisiting small subgroup and invalid curve attacks on protocols using diffie-hellman. In *IEEE Computer Security Foundations Symposium, CSF 2019*, 2019.

[21] Cas Cremers, Benjamin Kiesl, and Niklas Medinger. A formal analysis of IEEE 802.11's WPA2: countering the kracks caused by cracking the counters. In *Proc. of USENIX Security Symposium*, 2020.

[22] David L. Dill. The mur*phi* verification system. In *Computer Aided Verification*, 1996.

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

[24] Benjamin Dowling and Britta Hale. Secure messaging authentication against active man-in-the-middle attacks. In *Proc. of EuroS&P*, 2021.

[25] Marc Fischlin and Olga Sanina. Cryptographic analysis of the bluetooth secure connection protocol suite. In *Proc. of ASIACRYPT*, 2021.

[26] David Jia and Richard Hsu. Formal modeling and analysis of bluetooth 4.0 pairing protocol. 2013.

[27] Brian W. Kernighan and Dennis M. Ritchie. *The M4 macro processor*. 1977.

[28] Andrew Y. Lindell. Attacks on the pairing protocol of bluetooth v2.1. In *Black Hat USA*, 2008.

[29] Andrew Y. Lindell. Comparison-based key exchange and the security of the numeric comparison mode in bluetooth v2.1. In *CT-RSA*, volume 5473, pages 66–83, 2009.

[30] Gavin Lowe. A hierarchy of authentication specifications. In *Proc. 10th CSFW*, pages 31–43, 1997.

[31] Sjouke Mauw, Zach Smith, Jorge Toro-Pozo, and Rolando Trujillo-Rasua. Distance-bounding protocols: Verification without time and location. In *Proc. of S&P*, 2018.

[32] Sjouke Mauw, Zach Smith, Jorge Toro-Pozo, and Rolando Trujillo-Rasua. Post-collusion security and distance bounding. In *Proc. of CCS*, 2019.

[33] Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. The TAMARIN Prover for the Symbolic Analysis of Security Protocols. In *Computer Aided Verification*, 2013.

[34] Mohit Sethi, Aleksi Peltonen, and Tuomas Aura. Misbinding attacks on secure device pairing and bootstrapping. In *Proc. of AsiaCCS*, 2019.

[35] Michael Troncoso and Britta Hale. The Bluetooth CYBORG: Analysis of the Full Human-Machine Passkey Entry AKE Protocol. In *Proc. of NDSS*, 2021.

[36] Maximilian von Tschirschnitz, Ludwig Peuckert, Fabian Franzen, and Jens Grossklags. Method confusion attack on bluetooth pairing. In *Proc. of S&P*, 2021.

[37] Jianliang Wu, Ruoyu Wu, Dongyan Xu, Dave Jing Tian, and Antonio Bianchi. Formal model-driven discovery of bluetooth protocol design vulnerabilities. In *Proc. of S&P*, 2022.

[38] Yue Zhang, Jian Weng, Rajib Dey, Yier Jin, Zhiqiang Lin, and Xinwen Fu. Breaking Secure Pairing of Bluetooth Low Energy Using Downgrade Attacks. In *Proc. of USENIX Security Symposium*, 2020.

A Sequence Diagrams of Association Models

- Numeric Comparison (NC): The sequence diagram of NC is shown in Figure 3, where N_I and N_R are two random strings, f4 is a pseudorandom function, and g2 is a function that outputs a 6-digit number. The user only needs to indicate "Yes" on the two devices if $V_I = V_R$.

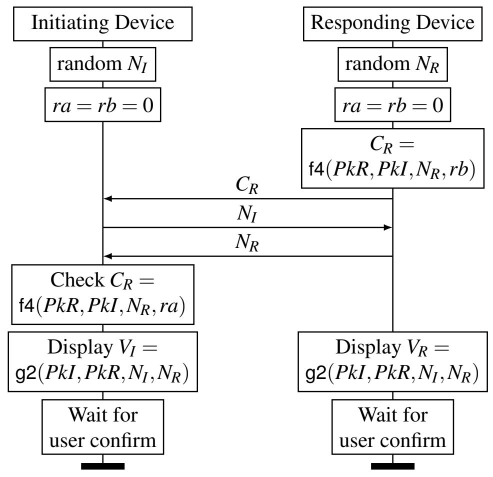

Initiating Device
Responding Device
random N_I
random N_R
$ra = rb = 0$
$ra = rb = 0$
$C_R = \mathsf{f4}(PkR, PkI, N_R, rb)$
C_R
N_I
N_R
Check $C_R = \mathsf{f4}(PkR, PkI, N_R, ra)$
Display $V_I = \mathsf{g2}(PkI, PkR, N_I, N_R)$
Display $V_R = \mathsf{g2}(PkI, PkR, N_I, N_R)$
Wait for user confirm
Wait for user confirm

Figure 3: Sequence Diagram of Numeric Comparison

- Just Work (JW): The sequence diagram of JW is almost the same as NC, except that the display of a 6-digit number and waiting for user confirmation are omitted.

- Passkey Entry (PE): The sequence diagram of PE is shown in Figure 4. The number *passkey* is generated and displayed on one device and entered by the user on the other one. Alternately, the user can input an identical *passkey* into both devices. The variables *rai* and *rbi* are *i*-th bit of *passkey*. Finally, the N_I and N_R are set to N_I^{20} and N_R^{20} respectively.

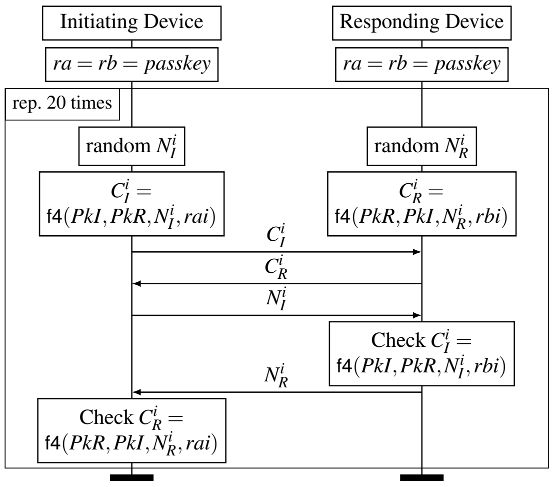

Initiating Device
Responding Device
$ra = rb = passkey$
$ra = rb = passkey$
rep. 20 times
random N_I^i
random N_R^i
$C_I^i = \mathsf{f4}(PkI, PkR, N_I^i, rai)$
$C_R^i = \mathsf{f4}(PkR, PkI, N_R^i, rbi)$
C_I^i
C_R^i
N_I^i
Check $C_I^i = \mathsf{f4}(PkI, PkR, N_I^i, rbi)$
N_R^i
Check $C_R^i = \mathsf{f4}(PkR, PkI, N_R^i, rai)$

Figure 4: Sequence Diagram of Passkey Entry

- Out-of-Band (OOB): The sequence diagram of OOB is shown in Figure 5, where r_I and r_R are two random numbers, I and R are the addresses of initiating and responding devices used during pairing. The OOB channel such as an NFC connection is deemed as a secure channel, in which the exchanged messages can be bi-direction or uni-direction.

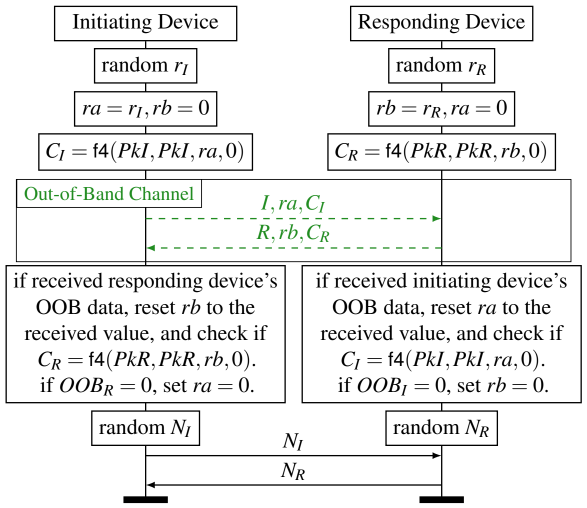

Initiating Device
Responding Device
random r_I
random r_R
$ra = r_I, rb = 0$
$rb = r_R, ra = 0$
$C_I = \mathsf{f4}(PkI, PkI, ra, 0)$
$C_R = \mathsf{f4}(PkR, PkR, rb, 0)$
Out-of-Band Channel
I, ra, C_I
R, rb, C_R
if received responding device's OOB data, reset rb to the received value, and check if $C_R = \mathsf{f4}(PkR, PkR, rb, 0)$. if $OOB_R = 0$, set $ra = 0$.
if received initiating device's OOB data, reset ra to the received value, and check if $C_I = \mathsf{f4}(PkI, PkI, ra, 0)$. if $OOB_I = 0$, set $rb = 0$.
random N_I
random N_R
N_I
N_R

Figure 5: Sequence Diagram of Out Of Band

B State Machine for User Modeling

The state machine of the user is determined by our assumptions (refer to Section 3.2).

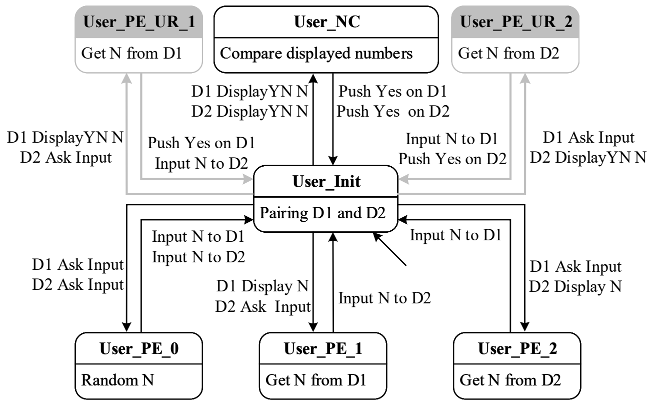


Figure 6: State Machine for User Modeling. (Display: display a 6-digit decimal number. DisplayYN: display a 6-digit decimal number along with a "Yes" button and a "No" button. Ask Input: ask user to input a 6-digit decimal number.)

C Guarded First-Order Logic Formulas

```
predicates: P1(a,b,dhk) <=>
  (All #i.Commit_Init(a,b,<'DHKey',dhk>)@i
  ==>
  (Ex #j.Running_Res(b,a,<'DHKey',dhk>)@j)
  | (Ex #k. InitDJW()@k))
predicates: P2(a,b,dhk) <=>
  (All #i.Commit_Res(a,b,<'DHKey',dhk>)@i
  ==>
  (Ex #j.Running_Init(b,a,<'DHKey',dhk>)@j)
  | (Ex #k. ResDJW()@k))
predicates: P3(a,b,ltk) <=>
  (All #i.Commit_Init(a,b,<'LTK',ltk>)@i
  ==>
  (Ex #j.Running_Res(b,a,<'LTK',ltk>)@j)
  | (Ex #k.InitDJW()@k))
predicates: P4(a,b,ltk) <=>
  (All #i.Commit_Res(a,b,<'LTK',ltk>)@i
  ==>
  (Ex #j.Running_Init(b,a,<'LTK',ltk>)@j)
  | (Ex #k. ResDJW()@k))
lemma A1:
  "All I R DHKey. P1(I,R,DHKey)"
lemma A2:
  "All I R DHKey. P2(R,I,DHKey)"
lemma A3:
  "All I R LTK. P3(I,R,LTK)"
lemma A4:
  "All I R LTK. P4(I,R,LTK)"
lemma MITMP:
  "All I R #i. MP(I,R)@i
  ==>
  not(Ex LTKI LTKR #t1 #t2.
       not(P3(I,R,LTKI)) & not(P4(R,I,LTKR))
       & K(LTKI)@t1 & K(LTKR)@t2)"
lemma LTKCP:
  "All I R LTKI LTKR #i #j.
    LTK(I,R,LTKI)@i & LTK(R,I,LTKR)@j
  ==>
  not(not(P3(I,R,LTKI)) & not(P4(R,I,LTKR))
    & not(Ex #k #l. K(LTKI)@k & K(LTKR)@l))"
predicates: P5(a,b,ltk) <=>
  (All #i.Commit_Init(a,b,<'LTK',ltk>)@i
  ==>
  (Ex #j.Running_Res(b,a,<'LTK',ltk>)@j))
predicates: P6(a,b,ltk) <=>
  (All #i.Commit_Res(a,b,<'LTK',ltk>)@i
  ==>
  (Ex #j.Running_Init(b,a,<'LTK',ltk>)@j))
lemma SecAuthLTK:
  "All I R LTK #i #j.
      FSecAuthLTK(I,R,LTK)@i
    & FSecAuthLTK(R,I,LTK)@j
  ==>
  (P5(I,R,LTK) & P6(R,I,LTK)
    ==> not(Ex #k. K(LTK)@k))"
```