# Attacking Automotive RKE Security: How Smart are your 'Smart' Keys?

Ritul Satish
*Ashoka University*
ritulsatish@gmail.com

Alfred Daimari
*Ashoka University*
alfred.daimari@alumni.ashoka.edu.in

Argha Chakrabarty
*Ashoka University*
argha.chakrabarty@ashoka.edu.in

Kahaan Shah
*Ashoka University*
kahaan.shah_asp25@ashoka.edu.in

Debayan Gupta
*Ashoka University*
debayan.gupta@ashoka.edu.in

***Abstract*—Remote Keyless Entry (RKE) systems are ubiquitous in modern day automobiles, providing convenience for vehicle owners - occasionally at the cost of security. Most automobile companies have proprietary implementations of RKE; these are sometimes built on insecure algorithms and authentication mechanisms. This paper presents a comprehensive study conducted on the RKE systems of multiple cars from four automobile manufacturers not previously explored.**

**Specifically, we analyze the design, implementation, and security levels of 7 different cars manufactured by Honda, Maruti-Suzuki, Toyota, and Mahindra. We also do a deep dive into the RKE system of a particular Honda model.**

**We evaluate the susceptibility of these systems to known vulnerabilities (such as RollJam [1] and RollBack [2] attacks). This is accomplished using a novel tool – 'Puck-py', that helps analyze RKE protocols. Our tool automates several aspects of the protocol analysis process, reducing time and logistical constraints in RKE research; we provide standardized protocols to execute various attacks using our Puck-Py tool. We find that, despite having a long period of time to fix security issues, several popular automobiles remain susceptible to attacks, including the basic RollJam attack.**

***Index Terms*—RKE security, automotive security, SDR**

## 1. Introduction

Modern cars have adopted several electronic mechanisms to improve user convenience and vehicle safety. In particular, the development of cheap wireless communication technologies like Bluetooth, Radio Frequency Identification (RFID), and Wi-Fi, has allowed automotive manufacturers to create a variety of features.

One such feature is the Remote Keyless Entry (RKE) system using RFID technology. This allows users to control door and boot locks by pressing buttons on a key fob. RKE systems come as a standard feature in most cars today - for many of us, mechanical turnkeys to open and lock cars are a thing of the past. Another important feature in this space is Keyless Ignition (KI) systems, where the key is equipped with a transponder that emits low-frequency signals to a receiver unit in the car to validate itself (usually via a challenge-response mechanism). Once validated, the car activates a start/stop button that can be pushed by the user to start the engine [3]. Keys equipped with RKE and KI systems are often called *Smart Keys*.

The security of RKE systems is critical: a compromise of such a system could lead to vehicle theft and might even jeopardize passenger safety. Smart keys have proved to be vulnerable to several classes of attacks such as Rolljam, Rollback, and Relay [4], [5]. While prior work has explored RKE and Passive Keyless Entry (PKE) schemes, this has been limited to a small set of automotive brands [3], [6]–[8].

Governments and regulators across the world have recognized the grave threat posed by weak automotive RKE systems. This has led to instructions for automotive manufacturers and organizations like the National Highway Traffic Safety Administration (NHTSA) to address RKE system safety hazards [9]. We investigate the RKE and KI schemes of four automotive manufacturers Maruti-Suzuki, Honda, Toyota, and Mahindra [10], providing a comprehensive understanding of the security of smart key systems in real-world use. Our analysis serves as a useful benchmark for assessing the industry's overall vulnerability; we also hope that this will aid in identifying new risks and help manufacturers improve security.

### 1.1. Our Contributions

1) We analyze the RKE schemes of four major automotive manufacturers: Honda, Toyota, Maruti-Suzuki, and Mahindra. Our analysis includes testing the schemes for two major vulnerabilities from previous research: RollJam and RollBack (§2.6.1 & §2.6.2).
   The models tested during our analysis represent approximately 7 million units that were on the road as of 2020 (refer to Table 1). We find that a large number of these cars are vulnerable to even basic attacks, despite these attacks being known for many years.
2) We conduct a detailed analysis of a single RKE system (a Honda; see Section §4). The system we test is currently being used by around 2 million Honda cars worldwide (Refer to Table 1).

3) We design a tool, Puck-py, which streamlines data analysis and processing of radio frequency (RF) packets used in various RKE schemes. It is designed to efficiently capture, demodulate, save, and transmit packets at the desired frequency. It is also capable of carrying out jamming of RF signals. By automating the protocol analysis process, Puck-py enables researchers to conduct in-depth investigations while reducing both logistical and time constraints (Refer to Section §5).

TABLE 1. UNITS SOLD FOR CAR MODELS WE SURVEYED

| Manufacturer | Model | Year | Cars Sold |
|---|---|---|---|
| Maruti Suzuki | Dzire | 2020 | 2,196,046 [11] |
| | Baleno | 2016 | 1,212,137 [12], [13] |
| Honda | Mobilio | 2015 | 245,637 |
| | Jazz | 2016 | 465,230 [14], [15] |
| | Brio | 2016 | 592,792 [16] |
| Toyota | Innova | 2014 | 2,309,174 [17] |
| Mahindra | Scorpio | 2016 | 580,000 [18] |

## 2. Technical Background and Related work

### 2.1. Rf Signals & Modulation & Encodings in Context Of Automobiles

RF signals are used widely in modern automobile RKE systems to make opening vehicle doors from a distance convenient. The frequency of RF signals used in automobiles operates at or around 433 MHz. The smart key transmits an RF signal when the keyholder presses the close or open button. The vehicle's body control module (BCM) then authenticates the signal. Based on this process it either unlocks the door, locks the door or rejects the signal. There are different types of modulations used in vehicle RKE systems of which the most commonly utilized techniques in RF key fobs are Amplitude Shift Keying (ASK) and Frequency Shift Keying (FSK) [3].

Apart from modulation techniques, an encoding technique is applied to decrease the probability of the signal being affected by noise or interference. RKE systems most commonly use Manchester encoding and Differential Manchester encoding.

### 2.2. Components of Smart Key

A typical car keyfob's printed ciruit board (PCB) consists of a central micro-controller ($\mu$C). The $\mu$C are generally made specifically for automotive purposes. These generally include a transponder circuit, programmable controller, ROM, RAM and an EEPROM (Electrically Erasable Programmable Read-only Memory). If the manufacturer deems it necessary, a separate EEPROM IC is also present, mostly in older models, to hold additional data. The keys also contain a coin-cell battery, and an antenna usually embedded inside the PCB. The transponder circuit mentioned earlier can be embedded in the $\mu$C, or they are a separate unit independent of the key's $\mu$C. The term transponder is a portmanteau of transmitter and responder. Here, ts primary function is to emit a response signal upon receiving a signal: this is used mainly in car immobilizers to prevent theft.

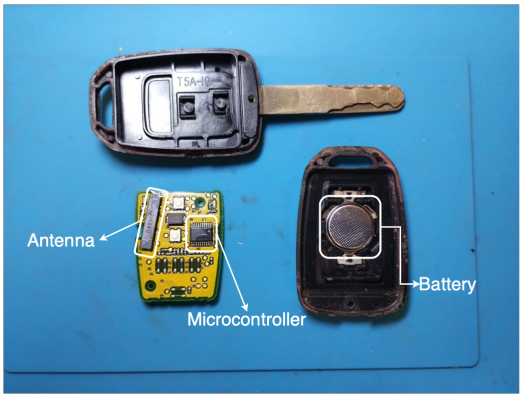


Figure 1. The Components of the disassembled Honda key, the main components are the Coin Cell Battery, the Micro-Controller and the Antenna.

### 2.3. Keyless Ignition or Push Button Start System

This feature enables the car's ignition system to detect a smart key if it is within a distance defined by the manufacturer. This range typically does not exceed the body of the car. The smart key includes a transponder that emits low frequency RF signals to wake the ignition system and authenticate itself. Once authenticated the car's immobilizer is deactivated and the 'Start/Stop' button is activated, allowing the user to start the car.

### 2.4. Rolling Code Scheme & Encryption Algorithms

'Rolling code' or 'Code hopping' schemes are cryptographic schemes used to secure RKE systems against replay attacks, where the attacker simply records and replays the unlock signal from a car key to gain access to the car.

Rolling code schemes prevent such attacks by using an encrypted counter value (or an encrypted random number). *I.e.*, the plaintext in such an encrypted rolling code scheme is a counter value (or a random number), along with other data such as a UID and button press. If a random number is being used, the pseudorandom number generator (PRNG) is known to the smart key and the vehicle. The latter decrypts the code received. If the UID is known, the current counter value is compared to the last counter value. If it is a random number, the vehicle generates the 'next' random number using the common PRNG and compares it to the received value. In both cases, if the value received is between $i+1$ and $i+n$ ( here $i$ is last valid received value and $n$ is the maximum number of missed values), the value is deemed valid and accepted.

The comparison is usually done up to the next 256 numbers, to ensure missed values (such as button presses away from the car) do not impact the system. Using a counter or random value ensures the RKE system is secured from replay attacks as each value can only be used once. A code is no longer valid if it has already been received. Some cars make special allowances to prevent false negatives – if there is a large difference between the code received and the car's current counter, the car waits to see if the subsequent code received is also next in the

sequence. If so, the car resets its counter in sync with the last code received.

A widely used rolling code scheme is KEELOQ [19]. The 'KEELOQ Hopping Codes' use a 16-bit secret counter to synchronize the sending and receiving units. The KEELOQ block cipher has been broken by several techniques like slide-determine [20] and meet-in-the-middle [21] as far back as 2012.

Hitag2 [22] and DST40 [23] are common encryption algorithms used in the authentication protocols of various automotive RKE systems. However, prior work has long since revealed vulnerabilities in these algorithms, specifically dictionary and key-recovery attacks [8], [24]–[26].

## 2.5. Software Defined Radio (SDR)

Software defined radios (SDRs) are radio systems where traditional analog radio components have been replaced by software. The replaced radio components include modulators, demodulators, and tuners. This design allows SDRs to be configurable, enabling them to handle a wide range of protocols, modulation schemes and frequencies [27]. SDRs have two components:

- **An RF frontend** that receives the RF signals and converts them into its equivalent digital representation. (*E.g.*, RTL-SDR [28] and LimeSDR [29].)
- **A software backend** that performs the signal processing the task such as demodulation and decoding. GNU radio, rtl_433 and Universal Radio Hacker are examples of software backends [30], [31]. Software backends are usually run on a computer connected to the RF frontend.

Due to their flexibility and direct integration with computers, SDRs are a popular tool used to attack cars.

## 2.6. Established attacks: RollJam, RollBack, Replay

**2.6.1. RollJam Attack.** RollJam is a modified version of a replay attack that exploits the iterative counter feature of rolling code schemes [1]. Every car has a listening frequency (smart key frequency) range. The difference between upper and lower bounds of this listening range is usually within $0.3 - 0.4$ MHz. RollJam attacks exploit this range. The attack comprises three actions - jamming, interception, and replay.

1) **Jamming step:** Jamming of all communication between the vehicle and the smart key is achieved by sending dummy packets to the target vehicle at a frequency slightly above or below the smart key's frequency but still within the vehicle's listening range.
2) **Interception step:** Two unique RF signals are eavesdropped. The victim is forced to press the smart key's open button twice (since the car does not unlock due to jamming). Both RF signals are stored in memory. To be able to jam the car's reciever while still being able to record the packets transmitted by the key on the attack device the jamming frequency and listening frequncy should not interfere with each other. At the same time, both jamming and listening frequencies should be within the vehicle's upper and lower bounds of the frequency range. Thus, the greater the listening range, the more likely it is RollJam will work. Refer to Figure 2 to see the difference in jamming and listening frequencies.
3) **Replay step:** Once two signals are stored, the first intercepted RF signal (oldest signal) is replayed. The car unlocks but the attacker now has one valid unused rolling code, which can be used to gain access to the vehicle.

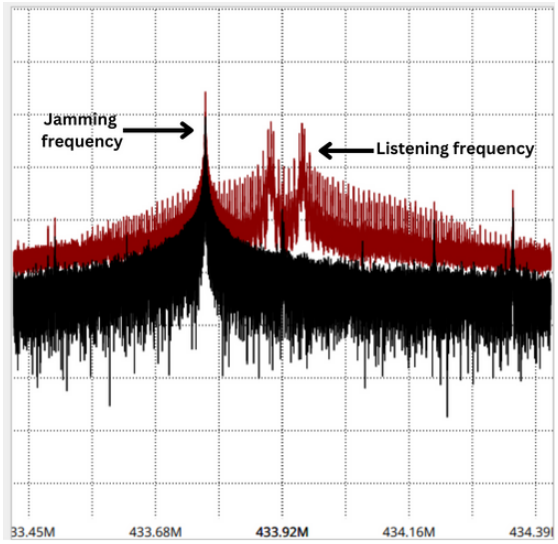


Figure 2. Maruti Suzuki Baleno's listening range accommodates a jamming and listening frequency such that they do not interfere with each other. Baleno uses 2-FSK. The Y-axis measures relative gain.

**2.6.2. RollBack Attack.** RollBack is a time-agnostic replay attack that exploits the re-syncing system in cars with rolling code encryption [2]. It has the same three phases as RollJam - jamming, interception, and replay. The jamming and interception steps are similar to RollJam, where the attacker captures two rolling codes from the smart key. After capturing two rolling codes, jamming stops, and the smart key's RF signals start reaching the vehicle.

The victim can then lock and unlock the car multiple times (unlike with RollJam). The attacker then replays the oldest captured rolling code to reset the rolling codes' counter value or the pseudo random number generator in the vehicle. The second captured rolling key code is then replayed. Since this is the next code in the sequence the counter value is reset and the attacker gains access to the target vehicle [2]. While RollJam relies on replaying the eavesdropped signal before the car hears another unlock signal, RollBack is time agnostic, as long as enough lock and unlock signals have been played between the interception and replay step.

**2.6.3. Relay Attack.** A relay attack involves signal interception and 'relaying' the signal between two rogue devices that can communicate over long distances [32].

In a typical relay attack, an attacker places two rogue devices - one near the system being attacked (*i.e.*, the vehicle), and another near the victim's device (*i.e.*, the smart key). By doing this, the attacker fools the distance

constraint set on the victim's device. Each rogue device intercepts the signal from the victim's device and the target system. The intercepted signal is then shared between the rogue devices [33].

By relaying the signals, the rogue devices trick the target system into believing that the victim's device is within range and is communicating with the system directly. This can allow the attacker to gain access to the target system and in some cases start the vehicle without physical access to the smart key. With the addition of SDRs to the system, attackers may also transmit these systems using digital communication methods with ease, increasing the range and effectiveness of relay attacks.

TABLE 2. POSSIBLE SMART KEY RF ATTACKS

| Key Tech | RF Attacks |
|---|---|
| Immobilizer | None |
| RKE | Rolljam, Rollback |
| RKE and KI | Rolljam, Rollback, Relay |
| PRKE and PKI | Relay |

### 2.7. Similar Works

There exists significant prior work about the security of smart keys. Cryptanalytical methods have been used extensively to identify vulnerabilities in car security systems, exposing weaknesses in various encryption algorithms used within the industry.

In 2012, the Hitag2 security system, which has been in use since 1996 in car immobilizers, was shown to be vulnerable to a host of cryptographic methods [8]. The KeeLoq rolling code scheme was also broken using meet-in-the-middle attacks [21]. Similarly, in 2016 an intense analysis of VolksWagen cars revealed vulnerabilities in both the RKE and immobilizer systems [3]. It was shown that the keys of VolksWagen cars could be cloned, and their 'master keys' recovered to break into cars and start their engines. In 2021, the firmware update platform of the Tesla Model X was revealed as a potential attack surface which enabled the pairing of malicious keys to the car [7].

It is clear that cryptanalytical methods have revealed major gaps in the security system of modern vehicles. However, these methods often require extensive reverse engineering for different manufacturers along with expertise (to execute an attack). Further, cryptanalytical methods have been theoretically explored to a large extent - as such, we focus on 'practical' attacks in this paper.

The RollJam attack was proposed in 2015 as an alternative non-cryptographic method to break into RKE systems (using simply record, jamming, and replay [1] at the protocol level). This was then built upon in 2022 by the RollBack attack which proposed a time-agnostic expansion on RollJam [2]. Critically, these eavesdropping based attacks required far lesser cryptographic knowledge to be executed successfully. They marked a cheaper, easier way to gain unauthorized access to a car.

Of the research described above, only the RollBack attack was extensively tested on various car models in a research setting. Some of the cryptographic attacks were tested on chipsets, but not practically on entire vehicles. *E.g.*, attacks on KeeLoq were tested against simulations and data generated by a HCS410 KeeLoq transponder [21]. Through our research, we seek to assess the effectiveness of RollJam and RollBack on vehicles across various manufacturers and model years. We also propose manufacturer-agnostic tools that will enable researchers to analyze, quantify and compare the security of vehicles with minimal reverse engineering.

## 3. Our Exploration of RKE systems

We explore the RKE systems of four different automotive manufacturers: Honda, Toyota, Maruti-Suzuki, and Mahindra. The RKE schemes of these companies have not been explored by previous research.

Smart keys for the manufacturers were procured by going to local second-hand or spare car markets and enquiring about both fresh and second-hand keys. After extracting PCBs from the keys, we found that a typical key consists of a ($\mu C$) sometimes an added EEPROM, a transmitter/receiver unit, and an antenna in the PCB itself.

### 3.1. IC analysis of various key-fobs

TABLE 3. SMART KEY ICS *(Baleno & Scorpio owners didn't permit us to open the keys to inspect them)*

| Model | IC |
|---|---|
| Dzire | Microchip 12F635 |
| Baleno | NA |
| Mobilio | NXP PCF7941 |
| Jazz | NXP PCF7941 |
| Brio | NXP PCF7941 |
| Innova | Microchip 362T |
| Scorpio | NA |

Both the Suzuki and the Honda key have a single $\mu$C that controls the whole key; the Honda key has a PCF7941. It has a 4 Kbyte ROM, 64 Byte RAM, a 384 bit EEPROM and an 8-bit RISC Architecture (MRK-II) and uses Hitag2 cipher. The Toyota key on the other hand uses a HCS362T IC with a KeeLoq code hopping encoder. The IC itself has EEPROM, a 32-bit shift register, and a Programmable 64-bit (cryptographic) key with read protection. However, unlike the Honda IC, it does not have a transponder built in.

Unlike the Honda and the Toyota key, both of which are manufactured for automotive applications, the Microchip 12F635 used in the Maruti-Suzuki, is a generic IC with a RISC CPU. It also has a KeeLoq compatible hardware cryptographic module and is capable of sending and receiving low-frequency signals. This leads us to believe that the $\mu$C is used as a transponder as well. Refer to Table 3.

### 3.2. RKE Packet Analysis

We utilized two RF demodulation software tools, Universal Radio Hacker (URH) and RTL_433 for extracting RF information from our set of captured Smart Key RF packets. RTL_433 was utilized for receiving and demodulating packets at frequencies of 433.92 MHz, and 315 MHz. It supports 245 RF protocols which enabled us

to handle a wide range of protocols. URH supported common SDRs such as RTL-SDR, HackRF and made it easy to investigate wireless protocols.

The captured set included lock and unlock key presses of multiple smart keys of 4 different manufacturers: Maruti-Suzuki, Honda, Toyota and Mahindra. On pressing the smart key, usually more than one RF packet gets released. Through cropping and replaying the packets to the vehicle, we removed redundant packets to isolate exact packet sizes.

By comparing multiple lock and unlock RF packets for each smart key separately using URH, we were able to identify the preamble, static code and encrypted code for each key. On closely analysing the static code, we identified the bits that contained information about which smart key button was pressed. Each packet has three major segments:

- **The preamble** is a fixed-length code which is at the start of the packet. Its main purpose is to distinguish the signal from other packets in the vicinity and informs the receiver on how to synchronize its clock with the smart key's clock so that it can correctly identify the start of each bit.
- **The static code** consists of important details such as the smart key ID and command type (lock or unlock).
- **The encrypted code** consists of the rolling code.

TABLE 4. MODULATION AND OPTIMAL JAMMING & LISTENING FREQUENCY OF THE SURVEYED CARS

| Model | Mod | Jam(MHz) | Listen(MHz) |
|---|---|---|---|
| Dzire | FSK | 433.5 | 433.9 |
| Baleno | FSK | 433.6 | 433.8 |
| Mobilio | FSK | 433.5 | 433.9 |
| Jazz | FSK | 433.5 | 433.9 |
| Brio | FSK | 433.6 | 433.9 |
| Innova | ASK | 433.3 | 434 |
| Scorpio | ASK | 433.6 | 433.9 |

Specific packet information for the tested smart keys can be referred to in Table 4.

### 3.3. Response to RollJam and RollBack Attacks

We performed RollJam and RollBack attacks using URH and RfCat. We used URH to find the jamming and listening frequency (Refer to Figure 3 to see the jamming and listening frequency for Scorpio). We used URH to capture the first two RF packets from the smart key and saved it on disk. After the storing it on disk, Rfcat stops jamming and allows the smart key and vehicle to synchronize their rolling code counters once again. We then used Rfcat (sub GHz analysis tool) to send the bits at a later time to reset the rolling code counter as explained for each attack protocol [34].

We attempted RollBack attacks with 2, 5, 10 and 15 key presses, however it was not successful on any of the cars. It is possible that the key press count was not high enough for the vulnerability to be exploited. Refer to Table 5 to see the success rate of RollJam and Rollback attacks on the vehicles we tested.

TABLE 5. ROLLJAM AND ROLLBACK ATTACKS

| Model | RollJam | RollBack |
|---|---|---|
| Dzire | ✓ | × |
| Baleno | ✓ | × |
| Mobilio | ✓ | × |
| Jazz | ✓ | × |
| Brio | ✓ | × |
| Innova | ✓ | × |
| Scorpio | ✓ | × |

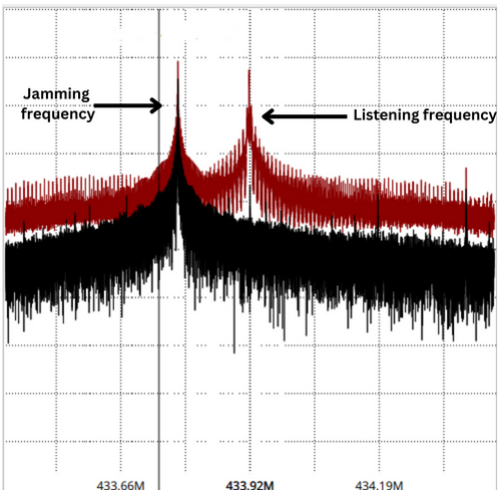


Figure 3. Mahindra Scorpio Jamming and Listening Frequency. Unlike Maruti Suzuki Baleno, Scorpio uses 1-ASK modulation. The Y-axis measures relative gain.

### 3.4. Evaluating Attack Susceptibility

A helpful addition to research and development would be a metric to benchmark vehicle susceptibility to RF based eavesdropping attacks.

Such a metric would have to be based around the range of frequencies available for exploitation for a given car. It would have to take into account hardware related properties of the car's antenna as well as the attacker's antenna. Such a metric would also be influenced by other factors like distance of the attacker from the car and wear and tear of the devices.

In §A) we propose a potential metric to analyse the susceptibility of vehicles to eavesdropping attacks. Although this depends on a variety of hardware related properties of the car's receiver, we attempt to lay out the basic structure for a universal metric to benchmark RKE security levels.

## 4. Comprehensive Analysis of the Honda RKE System

We investigate the Honda RKE system, examining its components and analyzing it to gain a comprehensive understanding of its inner workings. We procured old Honda Smart keys (programmed) and a BCM of a 2016 Honda Mobilio. It is worth noting that several other Honda models, including the Amaze, Brio, and Jazz, utilize the same parts, and thus the findings of this study are applicable to those models as well.

### 4.1. Exploration of the Honda Key fob

As mentioned in section §3, we managed to procure two Honda keys (2012 Brio and 2015 Mobilio) from local scrapyards. The Mobilio key had a PCF7941 Security Transponder and RISC controller [35]. This is a single chip Keyless Entry controller, specifically designed by Phillips Semiconductor for vehicular applications. It has a 4 KB ROM, 64 Byte RAM, a 384 bit EEPROM, and an 8-bit RISC Architecture (MRK-II).

The Brio used a NXP $\mu C$ (61x0915), but due to the unavailability of any documentation regarding its working, extracting anything usable from the IC was a difficult task. There are some methods to re-construct the binary via targeted microscopic images of the exposed ROM and then using image processing to identify each individual section, but we did not have access to the required equipment.

Some key models purposefully hide the printing on the IC which makes such reverse engineering difficult.

### 4.2. Exploration of the Honda Body Control Module (BCM)

Apart from the key, we also obtained a Honda BCM for Honda Mobilio, year 2015 - 2016, several steps were taken to examine its internal components. The process involved cleaning the BCM, removing connectors, and uncovering the underlying PCB. Notable components identified on the PCB include:

- **CAN Transceiver:** The NXP AU5790 CAN Transceiver, which is responsible for providing interface between a CAN data link controller and a single wire physical bus line [36].
- **Multiplexer:** NXP 74HC15D 8-input multiplexer was also identified on the PCB [37]. This component facilitates the selection and routing of multiple input signals to the desired output, contributing to the functionality of the BCM
- **Power MOSFET:** MOSFETs are commonly used for switching and amplifying signals in electronic circuits. Its presence suggests its role in signal control and amplification withing the BCM.
- **EEPROM:** Among the components the Seiko S93A56A stood out due to available documentation about the IC [38]. This IC is a Serial EEPROM (Electrically Erasable Programmable Read-Only Memory) with a capacity of $128 \times 16$ bits. EEPROMs are non-volatile that can store data even when power is removed. The S93A56A is commonly used in automotive applications for storing critical information such as configuration settings, calibration data, and other relevant data. It is notable for being easily readable programmable using relatively commonly available programmers since it lacks built-in read or write protection mechanisms.

Using a XGecu T48 (EEPROM programmer), extracted the raw binary data from the IC and performed static analysis. The extracted firmware itself was 512 bytes in size. Normally, the size of such extracted files range from 0.5 to 2 Mega Bytes, and the binary is usually the

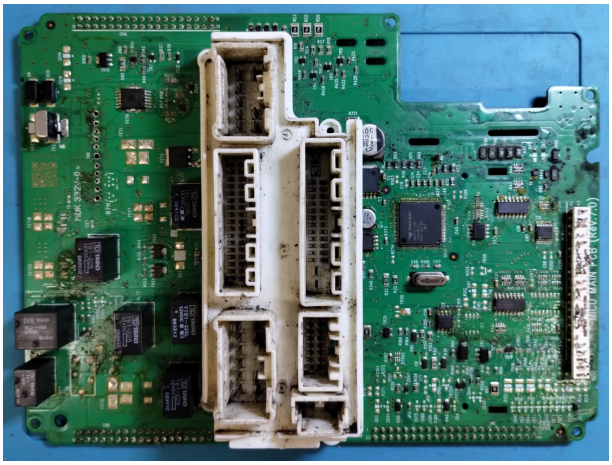

Figure 4. Disassembled Honda BCM: The main components are: (1) NXP 74HC151D (8-input multiplexer), (2) Seiko S93A56A (EEPROM), (3) YAZAKI HMC712

device firmware stored on the EEPROM, as the firmware doesn't need to update on a regular basis.

In our case, the file size itself indicated that this was not the firmware of the device. A hex dump of the file showed many random bits, and converting them to strings also yielded a similar result. Passing the firmware in reverse engineering tools like Ghidra and Binary Ninja also led to the same result. Several possible explanations can account for these observations:

- **Non-Firmware Data:** The Binary Data may not be the actual firmware, but could represent some other form of data stored in the IC. The absence of recognizable patterns or structures by Ghidra or binwalk suggests that it may not contain executable code or human-readable information.
- **Secret Key or Other Relevant Data:** It is possible that binary data represents a secret key used by the BCM for purposes such as authenticating a key fob or managing the immobilizer. This data might be stored in an external storage to prevent unauthorized access or for safekeeping [39].
- **Diagnostic Logs:** Another possibility is that the binary data is stored in the IC's EEPROM could be log files maintained by the BCM for diagnostic purposes [40]. These logs might contain information related to the functioning of the BCM and could be used for troubleshooting or analysis.
- **Damaged EEPROM:** Given that the BCM was obtained from second-hand dealers, and its prior conditions are unknown, it is also plausible that the EEPROM itself is damaged. This could explain the lack of meaningful data extraction or analysis results.

## 5. Our RKE Protocol Analyzer: Puck-py

Research in this area lacks a tool that can perform vehicle manufacturer based packet identification and real-time packet segmentation. To our knowledge there is also no device that can perform RollJam attacks outside a research setting. Such an attack scenario should allow for the victim to use their smart key arbitrarily. This

requires a device that can run for long periods of time, and also store and replay the necessary packets whenever the victim uses their smart key. To tackle such problems, we have developed a portable tool which we call 'Puck-py' that can handle real-time packet identification and packet segmentation.

## 5.1. Build of the Tool

As described earlier, a RollJam attack has three components — smart key signal jamming, smart key signal capture and smart key signal replay. Our tool performs the first two components without any user intervention. To replay a captured signal, the user only needs to input the 'send' command in a terminal program.

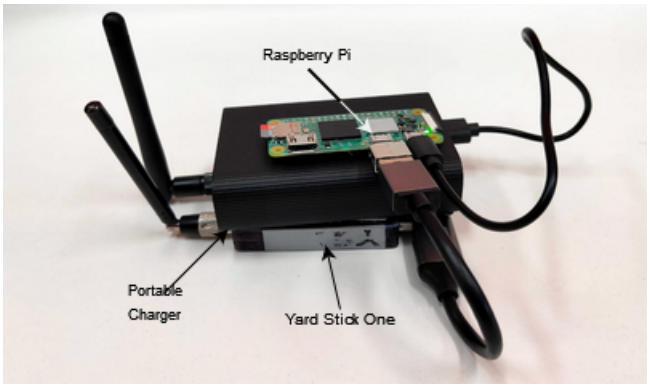


Figure 5. Our RKE Protocol Analyzer: Puck-py; Made with a Raspberry Pi Zero 2 w, a power bank, a yard stick and an RTL-SDR

**5.1.1. Hardware.** The device is small enough to be easily attached to a vehicle. It consists of a Raspberry Pi zero 2w, connected to an RTL-SDR and Yardstick-One using its ports, with a 5000mA power-bank for portability. The YARD Stick One can transmit or receive digital wireless signals at frequencies below 1 GHz: we use this for jamming [41]. The RTL-SDR is an ultra cheap software defined radio based on DVB-T TV tuners. We use this as the eavesdropping device for RF signal capture.

**5.1.2. Software.** Our software comprises two components: packet detection and the transmission.

The packet detection component has been forked from rtl_433 repository. We have added more features to the program. After a signal is detected, the packets go through a preamble detection step. This checks whether the detected packets match the preamble of the rf signal we want. If our program detects the wanted preamble, it sends out the bits in text format to our sending component using Linux dbus.

The signal transmission component is a python program. This component has complete control over the yardstick. Its main role is to perform the jamming and transmission of smart key RF packets. On receiving packets from the packet detection component, it maintains a queue-like data structure to store all the packets received. If more than two packets are received, it replays the oldest packet using the Yardstick one. This component also listens on a separate thread for commands, if it receives a send command, it sends out all packets stored in the data structure. Refer to GitHub links of both the components here: packet detection component, transmission component.

## 5.2. Setup & Execution

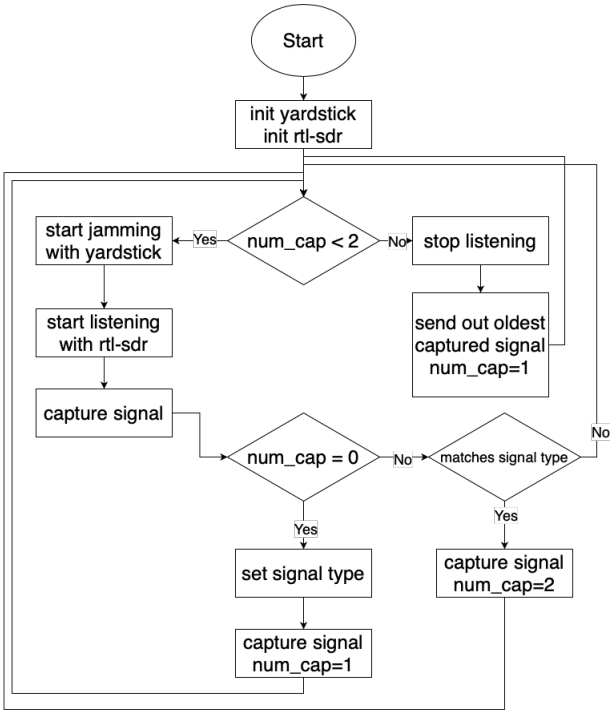


Figure 6. Puck-py program runtime flowchart

For the device to attack a vehicle, a few preliminary steps are required. The smart key listening frequency, jamming frequency and packet information of the smart key's RF packet have to be determined and fed in as program inputs. These are easily determined due to the similarity in listening ranges for various cars. The packet information is needed to create a packet detection function for the vehicle (documentation for how to write this function and add it to the device is on our Github page).

When the device is started, its first action is jamming the vehicle. On detecting an RF signal, our device demodulates the signal into bits and then performs checks on the preamble of the signal. If the preamble matches any pre-stored schema, the device is setup to perform an attack on the corresponding vehicle model.

Every captured signal that is an 'unlock' signal is then stored onto a queue. If the queue, which contains unlock signals, has captured two signals, it releases the oldest unlock signal to open the car. The victim needs to press the unlock button at least two times to open the vehicle's door the first time. After the first signal replay, every key press will unlock the car. The device also prevents a 'lock' signal from reaching the car to prevent the stored 'unlock' key press from getting invalidated as the rolling codes counter used, are the same in both 'lock' and 'unlock' key presses.

To send the stored 'unlock' key press in the puck, the attacker has to be within the puck's proximity, ensure that the puck is connected to the attacker's Wi-Fi, and then perform SSH. The attacker runs a python script that communicates with the 'Puck-py' program, which can command it to replay the stored 'unlock' RF signal. While sending the stored signal, the jamming and listening activity on the device is paused. See Figure 6 for a flowchart on how the device works.

### 5.3. Puck-py Attack Scenario

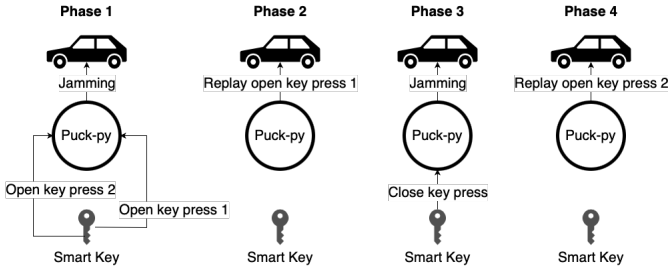


Figure 7. Puck-py Attack Phases

**Phase 1:** We latch our device onto the victim's car and run our software Puck-py (a C program called puckpy_433 and a python program called puckpy). The victim then approaches the vehicle, when they press the open button on their smart key for the first time, the vehicle's door remains locked.

**Phase 2:** After the signal from the second key press is read by Puck-py, it replays the first one. The car deems the signal from the first key press as valid.

**Phase 3:** After driving to their destination, the victim clicks on the close button, but the door does not close. Our device forces the victim to close the door manually by continually jamming all close signals.

**Phase 4:** The attacker now has access to one valid open signal and can go to the vehicle's location, use the Puck-py terminal program (a python program called puckpy_cli) to replay the second captured signal and open the vehicle.

## 6. Conclusion

We conclude that vehicle owners need to be more cognizant of the security-convenience tradeoffs involved in the use of smart keys. The implications of this study extend beyond individual vehicle owners, as the widespread use of insecure RKE systems poses a broader concern for public safety and security. We show that major automotive manufacturers have used insecure RKE systems despite prior work repeatedly raising vulnerability issues. Our novel device **'Puck-py'** helps streamline RKE security evaluation process by significantly reducing manual effort in multiple stages of the process, offering researchers enhanced efficiency and ease of use.

The continued use of vulnerable RKE systems, as documented in this study, underscores the inadequacy of automotive security levels of vehicles currently on the road.

### 6.1. Future Work

- **Explore the RKE systems of more manufacturers:** Extend the investigation of RKE systems beyond the passenger car market to include other sectors in the automotive industry.
- **Improve our RKE protocol analyzer, 'Puck-py':** We would like to add the ability to carry out relay attacks. This will greatly benefit research in PKE systems. Automating the process of identifying and isolating the appropriate frequency for blocking and capturing signals will eliminate the need for manual input.
- **Explore a general metric for safety benchmarking:** We would like to extend our work in to devise a general metric to benchmark safety from RF based attacks.

Our work will facilitate further research by others and make it easy to test unexplored RKE systems.

## References


[1] S. Kamkar, "Drive it like you hacked it," 2015.

[2] L. Csikor, H. Wei Lim, J. Wen Wong, S. Ramesh, R. P. Parameswarath, and M. Choon Chan, "Rollback: A new time-agnostic replay attack against the automotive remote keyless entry systems," sep 2022.

[3] F. D. Garcia, D. Oswald, T. Kasper, and P. Pavlidès, "Lock it and still lose it - on the (in)security of automotive remote keyless entry systems," in *USENIX Security 2016* (T. Holz and S. Savage, eds.), USENIX Association, Aug. 2016.

[4] R. P. Parameswarath and B. Sikdar, "An authentication mechanism for remote keyless entry systems in cars to prevent replay and rolljam attacks," in *2022 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1725–1730, IEEE, 2022.

[5] T. Yang, L. Kong, W. Xin, J. Hu, and Z. Chen, "Resisting relay attacks on vehicular passive keyless entry and start systems," in *2012 9th International Conference on Fuzzy Systems and Knowledge Discovery*, pp. 2232–2236, IEEE, 2012.

[6] R. Benadjila, M. Renard, J. Lopes-Esteves, and C. Kasmi, "One car, two frames: Attacks on hitag-2 remote keyless entry systems revisited.," in *WOOT*, 2017.

[7] L. Wouters, B. Gierlichs, and B. Preneel, "My other car is your car: compromising the tesla model X keyless entry system," *IACR TCHES*, vol. 2021, no. 4, pp. 149–172, 2021. https://tches.iacr.org/index.php/TCHES/article/view/9063.

[8] R. Verdult, F. D. Garcia, and J. Balasch, "Gone in 360 seconds: Hijacking with Hitag2," in *USENIX Security 2012* (T. Kohno, ed.), pp. 237–252, USENIX Association, Aug. 2012.

[9] SRS, "Congress forces nhtsa to address keyless safety hazards." https://www.safetyresearch.net/congress-forces-nhtsa-to-address-keyless-safety-hazards-one-automakers-approach-shows-why-regulations-matter/, Sep 2022.

[10] oica, "World ranking of manufacturers - www.oica.net," 2017.

[11] S. M, "Top 15 most sold cars of 2015 in india - sales analysis," Dec 2016.

[12] S. M, "Top 10 best selling cars of 2019, dzire on top - january to december," Jan 2020.

[13] T. A. Punditz, "Cy2022 - top 30 selling cars analysis," Jan 2023.

[14] M. Gasnier, "Indonesia 2006: Toyota avanza ends 28 year-reign of toyota kijang/innova," Jul 2019.

[15] T. A. Punditz, "Oemwise and modelwise car sales figures for calendar year 2020," Feb 2021.

[16] focus2move, "focus2move: India car sales," Feb 2013.

[17] M. Gasnier, "Vietnam 2006: Toyota innova starts with a bang," Sep 2011.

[18] L. Philip, "15 years and still counting: Scorpio most successful suv of mahindra and mahindra," jul 2017.

[19] C. Toma, "Introduction to ultimate keeloq technology," *Microchip App Notes*, 2014.

[20] N. T. Courtois and G. V. Bard, "Random permutation statistics and an improved slide-determine attack on keeloq," *Cryptography and Security: From Theory to Applications: Essays Dedicated to Jean-Jacques Quisquater on the Occasion of His 65th Birthday*, pp. 35–54, 2012.

[21] W. Aerts, E. Biham, D. De Moitie, E. De Mulder, O. Dunkelman, S. Indesteege, N. Keller, B. Preneel, G. A. E. Vandenbosch, and I. Verbauwhede, "A practical attack on KeeLoq," *Journal of Cryptology*, vol. 25, pp. 136–157, Jan. 2012.

[22] N. Semiconductors, "Hitag 2 transponder ic," 2014.

[23] S. Bono, M. Green, A. Stubblefield, A. Juels, A. D. Rubin, and M. Szydlo, "Security analysis of a cryptographically-enabled RFID device," in *USENIX Security 2005* (P. D. McDaniel, ed.), USENIX Association, July / Aug. 2005.

[24] N. Courtois, S. O'Neil, and J.-J. Quisquater, "Practical algebraic attacks on the Hitag2 stream cipher," in *ISC 2009* (P. Samarati, M. Yung, F. Martinelli, and C. A. Ardagna, eds.), vol. 5735 of *LNCS*, pp. 167–176, Springer, Heidelberg, Sept. 2009.

[25] S. B. M. G. A. Stubblefield and A. J. A. R. M. Szydlo, "Security analysis of a cryptographically-enabled rfid device," 2005.

[26] L. Wouters, J. Van den Herrewegen, F. D. Garcia, D. Oswald, B. Gierlichs, and B. Preneel, "Dismantling DST80-based immobiliser systems," *IACR TCHES*, vol. 2020, no. 2, pp. 99–127, 2020. https://tches.iacr.org/index.php/TCHES/article/view/8546.

[27] A. L. G. Reis, A. F. Barros, K. G. Lenzi, L. G. P. Meloni, and S. E. Barbin, "Introduction to the software-defined radio approach," *IEEE Latin America Transactions*, vol. 10, no. 1, pp. 1156–1161, 2012.

[28] Osmocom, "rtl-sdr," 2023.

[29] L. Microsystems, "Limesdr."

[30] J. Pohl and A. Noack, "Universal radio hacker: A suite for analyzing and attacking stateful wireless protocols," in *12th {USENIX} Workshop on Offensive Technologies ({WOOT} 18)*, 2018.

[31] Merbananr, "Merbananrtl433: Program to decode radio transmissions from devices on the ism bands," Nov 2022.

[32] L. Wouters, E. Marin, T. Ashur, B. Gierlichs, and B. Preneel, "Fast, furious and insecure: Passive keyless entry and start systems in modern supercars," *IACR TCHES*, vol. 2019, no. 3, pp. 66–85, 2019. https://tches.iacr.org/index.php/TCHES/article/view/8289.

[33] A. Francillon, B. Danev, and S. Capkun, "Relay attacks on passive keyless entry and start systems in modern cars," in *NDSS 2011*, The Internet Society, Feb. 2011.

[34] atlas0fd00m, "rfcat." https://github.com/atlas0fd00m/rfcat, 2012.

[35] Philips, "Philips-pcf7941," Nov 1999.

[36] P. Semiconductors, "Datasheet: Au5790 single wire can transceiver," May 2001.

[37] nexperia, "74hc151; 74hct151 8-input multiplexer," Oct. 2022.

[38] S. I. Inc., "Cmos serial e2prom s-93a46a/56a/66a," 2014.

[39] Michael, "What is eeprom," 2011.

[40] I. T. Committee, "Iso 22901-1:2008: Road vehicles - open diagnostic data exchange (odx)," nov 2008.

[41] G. S. Gadgets, "Yard stick one - great scott gadgets," 2009.

[42] R. Sheldon, "What is electric field strength and how is it measured?," Sep 2022.

# Appendix A.
# Eavesdropping RF Attack Sufficiency Value ($\delta$)

Recall that eavesdropping RF attacks, such as RollJam and RollBack require the attacker to successfully jam the vehicle's receiver unit while simultaneously capturing the signal from the smart key. The harder this is, the lower the chances of a successful attack. Cars listen for RF signals at a frequency range. The upper bound ($F_u$) and the lower bound ($F_l$) of this frequency range varies from car to car. We introduce a value called 'attack sufficiency value' (ASV), notated as $\delta$ which serves as metric to identify how vulnerable a particular car is to eavesdropping attacks.

As explained in §2.6.1 the greater the listening range, the easier it is to execute an eavesdropping attack. This is further constrained by $F_k$, since the jamming frequency must be chosen somewhere between $F_k$ and either $F_u$ or $F_l$. So the possible ranges to choose the jamming frequency are $F_k - F_l$ and $F_u - F_k$.

The ability to successfully execute the attack is also impacted by the antenna being used. The ability of the antenna to successfully receive signals is related to the distance from the transmitter using the notion of field strength. Thus, antenna's field strength ($A_{fs}$) can be calculated using the field strength formula [42]:

$$V/M = \frac{\sqrt{30 \cdot watts \cdot 10^{(\text{Gain}_{\text{dBi}}/10)}}}{\text{Distance(m)}}$$

To relate the constraints on executing an eavesdropping attack, ASV ($\delta$) is calculated using the formula below:

$$\delta = A_{fs} \times \max(F_k - F_l, F_u - F_k)$$

where:

$F_k$ is the transmitting frequency of key
$F_l$ is the lowest listening frequency
$F_u$ is the highest listening frequency
$A_{fs}$ is the field strength of listening antenna

This takes into account the largest possible range within which the jamming frequency may be chosen and the ability of the attacker's antenna to eavesdrop on the transmitted packets from a given antenna and distance. Thus we see that the vulnerability of the car to eavesdropping attacks $\propto \delta$.

The $A_{fs}$ for our antenna was calculated using the field strength formula [42] with a distance of 0.5 meters:

$$\begin{aligned} V/M &= \frac{\sqrt{30 \cdot watts \cdot 10^{(\text{Gain}_{\text{dBi}}/10)}}}{\text{Distance(m)}} \\ &= \frac{\sqrt{30 \cdot 0.01 \cdot 10^{(7/10)}}}{0.5} \\ &= 2.45 \end{aligned}$$

We have calculated $\delta$ values for three different car models: Baleno, Dzire and Innova. Refer to Table 6 for $\delta$ results. Intriguingly, we observed divergent $\delta$ values when comparing two Innova cars of identical make, specifications, and year of manufacture. One plausible explanation for this discrepancy, could be the varying impact of wear

and tear on the listening units' efficiency in the respective cars.

TABLE 6. ASV CALCULATION TABLE

| Model | $F_u$ | $F_l$ | $F_k$ | $\delta$ |
|---|---|---|---|---|
| Dzire | 434.3 | 433.5 | 433.9 | 0.98 |
| Innova-1 | 434.7 | 433.3 | 434 | 1.715 |
| Innova-2 | 434.6 | 433.4 | 434 | 1.47 |
| Baleno | 434 | 433.6 | 433.8 | 0.49 |