RollBack: A New Time-Agnostic Replay Attack Against the Automotive Remote Keyless Entry Systems

LEVENTE CSIKOR, Institute for Infocomm Research (I^2R), A*STAR, Singapore
HOON WEI LIM, NCS Group, Singapore
JUN WEN WONG, DSBJ Pte. Ltd., Singapore
SOUNDARYA RAMESH, School of Computing, National University of Singapore, Singapore
ROHINI POOLAT PARAMESWARATH, Department of Electrical and Computer Engineering, College of Design and Engineering, National University of Singapore, Singapore
MUN CHOON CHAN, School of Computing, National University of Singapore, Singapore

Automotive Keyless Entry (RKE) systems provide car owners with a degree of convenience, allowing them to lock and unlock their car without using a mechanical key. Today's RKE systems implement disposable rolling codes, making every key fob button press unique, effectively preventing simple replay attacks. However, a prior attack called RollJam was proven to break all rolling code–based systems in general. By a careful sequence of signal jamming, capturing, and replaying, an attacker can become aware of the subsequent valid unlock signal that has not been used yet. RollJam, however, requires continuous deployment indefinitely until it is exploited. Otherwise, the captured signals become invalid if the key fob is used again without RollJam in place.

We introduce RollBack, a new replay-and-resynchronize attack against most of today's RKE systems. In particular, we show that even though the one-time code becomes invalid in rolling code systems, replaying a few previously captured signals consecutively can trigger a rollback-like mechanism in the RKE system. Put differently, the rolling codes become resynchronized back to a previous code used in the past from where all subsequent yet already used signals work again. Moreover, the victim can still use the key fob without noticing any difference before and after the attack.

Unlike RollJam, RollBack does not necessitate jamming at all. In fact, it requires signal capturing only once and can be exploited at any time in the future as many times as desired. This time-agnostic property is particularly attractive to attackers, especially in car-sharing/renting scenarios in which accessing the key fob is straightforward. However, while RollJam defeats virtually any rolling code–based system, vehicles might have additional anti-theft measures against malfunctioning key fobs, hence against RollBack. Our ongoing analysis (with crowd-sourced data) against different vehicle makes and models has revealed that ~ 50% of the

5

This research was supported by the National University of Singapore, NCS Group, and I^2R, A*STAR, Singapore.
Authors' addresses: L, Csikor, Institute for Infocomm Research (I2R), A*STAR, 1 Fusionopolis Way, Connexis South Tower, Singapore 138632; e-mail: csikor_levente@i2r.a-star.edu.sg; H. W. Lim, NCS Group, NCS Hub, 5 Ang Mo Kio Street 62, Singapore 569141; e-mail: hoonwei.lim@ncs.com.sg; J. W. Wong, DSBJ Pte. Ltd., 11 Bishan Street 21, Singapore 573943; e-mail: junwenwong@gmail.com; S. Ramesh, School of Computing, National University of Singapore, Computing 1, Computing Drive, Singapore 117417; e-mail: soundaryaramesh96@gmail.com; R. Poolat Parameswarath, Department of Electrical and Computer Engineering, College of Design and Engineering, National University of Singapore, 4 Engineering Drive 3, Singapore 117583; e-mail: rohini.p@nus.edu.sg; M. C. Chan, School of Computing, National University of Singapore, Computing 1, Computing Drive, Singapore 117417; e-mail: dcscmc@nus.edu.sg.

2378-962X/2024/01-ART5 $15.00
https://doi.org/10.1145/3627827

examined vehicles in the Asian region are vulnerable to RollBack, whereas the impact tends to be smaller in other regions, such as Europe and North America.

CCS Concepts: • **Security and privacy → Vulnerability management;**

Additional Key Words and Phrases: Remote Keyless Entry, rolling code, vulnerability, replay attack, RollJam, `RollBack`, resynchronization

ACM Reference format:
Levente Csikor, Hoon Wei Lim, Jun Wen Wong, Soundarya Ramesh, Rohini Poolat Parameswarath, and Mun Choon Chan. 2024. RollBack: A New Time-Agnostic Replay Attack Against the Automotive Remote Keyless Entry Systems. *ACM Trans. Cyber-Phys. Syst.* 8, 1, Article 5 (January 2024), 25 pages.
https://doi.org/10.1145/3627827

1 INTRODUCTION

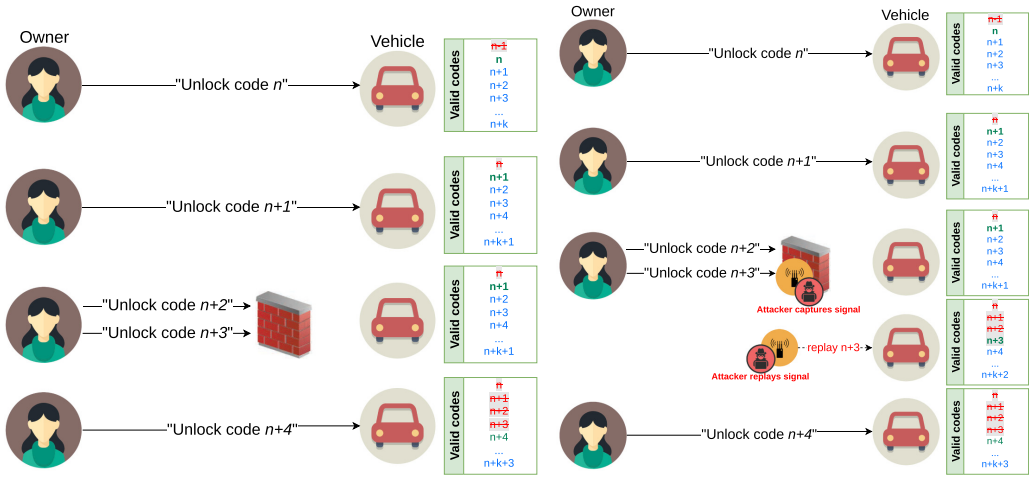
The automotive industry has undergone a tremendous evolution since the first car was made more than a century ago. While the efficiency and versatility have been continuously evolving, since the early 1980s, manufacturers have constantly been squeezing more and more embedded computers, known as Electronic Control Units (ECUs), into our cars to enhance safety [33], stability [4], diagnostics [6], and comfort [22, 45], to name a few [9]. On the one hand, this paradigm shift from the traditional mechanical mechanisms to an all-digital control has been clearly proven beneficial. On the other hand, computerized vehicles open up a broad set of new attack surfaces [7, 14, 17, 19, 27, 30, 43, 44].

One of the earliest comfort-enhancing inventions is the *Remote Keyless Entry (RKE)* system, which eliminates the need for physical keys and allows one to remotely lock and unlock the vehicle[1] merely by using a key fob. Because RKE has already been present in commercial vehicles since the early 1980s [22], it has been (and still is) one of the main targets of the attackers [7, 12, 18, 19, 30]. RKE systems use wireless radio signals; due to the limited number of required commands (e.g., lock, unlock) and, most importantly, the power and resource constraints of the small battery-operated key fobs, the communication between the key fob and the vehicle is designed to be simple. Some deployments may use encryption to avoid eavesdropping (i.e., capture and decoding of signals) or tampering attacks (i.e., "flipping" lock signals to unlocks). However, replaying signals, even if they are encrypted, is straightforward. Today, many RKE systems still implement static codes to control the vehicle from the key fob. Therefore, capturing an encrypted "unlock" signal allows an attacker to replay it and access the vehicle anytime afterward.

To cope with these simple replay attacks, *rolling codes*, i.e., code hopping [24], have been introduced wherein a particular code[2] (e.g., an "unlock" code) is considered disposable, i.e., it is only used once. In a nutshell, every button click on the key fob triggers a counter in the key fob and in the vehicle upon reception to roll, making it valid for subsequent use in the future. Put differently, sent codes that are *used once* are invalidated by the next code, effectively preventing replay attacks (see Figure 1(a)).

Note that a sent code can also be considered *unused* if the key fob has emitted the signal but the vehicle did not receive it. An example of this is when the unlock button is accidentally pressed (i.e., in our pocket, or when our toddler plays with the key fob) outside of the vehicle's vicinity (depicted by "unlock code n+2" and "n+3" in Figure 1(a)). To avoid getting out-of-sync and hence locking ourselves out of our vehicle in such cases, rolling code–based systems provide a safety feature that allows the key fob's counter to be steps ahead compared with the vehicle's counter. This is achieved

[1]In newer models, a key fob can also be used to turn on and off the anti-theft alarms or even start and stop the engine.
[2]In this article, the terms *code* and *signal* are used interchangeably.



(a) Essence of rolling codes: every signal is unique and gets invalidated by the next one.

(b) "Straightforward exploit" of the safety feature in rolling code-based systems

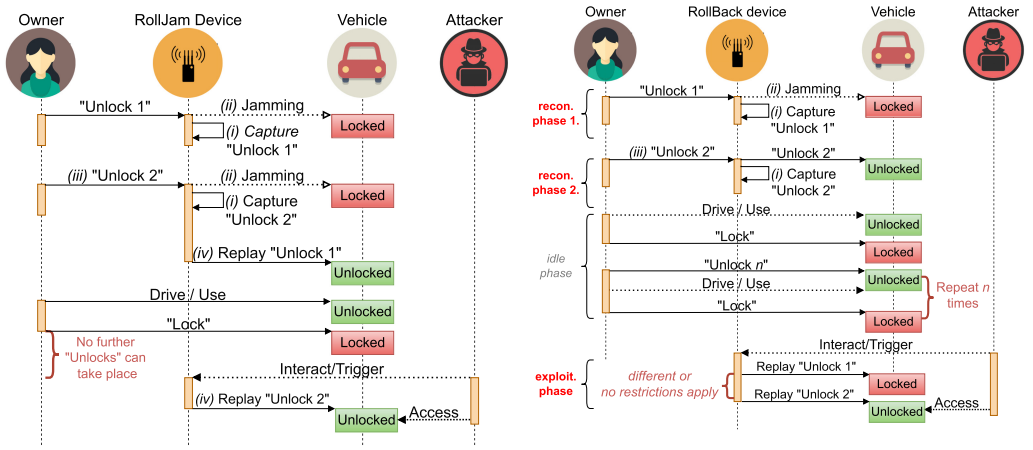
Fig. 1. Rolling code technology in a nutshell and its safety feature exploited.

by having *not one but a set of valid "future codes"* maintained at the vehicle. If the received code from the key fob matches any of these future codes, the vehicle resynchronizes to the code in the last key fob signal and invalidates all previous (but unused) ones from this set (refer to "Unlock code n+4" in Figure 1(a)). Clearly, if an attacker could obtain one of these unused future codes (i.e., capture the signals of the accidental button presses outside of the vicinity of the car), and she can replay it before the owner uses the key fob again, the attacker can get access to the vehicle (see Figure 1(b)). However, obtaining these future codes are extremely difficult in practice, especially if an attacker wants to target a random victim. That is the reason why this safety provisioning is considered a handy feature that makes the key fob use seamless and less troublesome.

In 2015, a somewhat sophisticated attack technique called RollJam [18] proved the rolling code–based key fob systems to be breakable. In a nutshell, by using a careful sequence of signal jamming, capturing, and replaying, RollJam can effectively convert this safety provisioning feature into an exploit.

RollJam is based on four main "principles": *(i)* capturing unlock signals, *(ii)* jamming the frequency band towards the vehicle at the same time to hinder correct signal reception, *(iii)* the owner's second trial as a fail-over mechanism, and, most importantly, *(iv)* timely replay of previously captured signals. To this end, a special-purpose device (hereafter, *rolljam device*) is used as a man-in-the-middle proxy and a signal jammer between the key fob and the vehicle (see Figure 2(a)). Briefly, the victim is lured to *(iii)* press the unlock button in a key fob twice by *(ii)* jamming the first unlock signal. At the same time, both first and second unlock signals are *(i)* captured; however, when the second signal is jammed, the rolljam device quickly *(iv)* replays the one captured the first time. As a result, the vehicle acts as intended, i.e., unlocks, and the victim assumes that the signal reception was poor on the first try. On the other hand, the attacker (i.e., by the rolljam device) becomes aware of the following valid unlock signal (see more details in Section 2.3). Therefore, once the owner stops using the vehicle and leaves it unattended, the attacker can replay this signal to access the vehicle.

RollJam, however, has two main drawbacks. First, suppose that the owner unlocks the vehicle again *without* the rolljam device in action. In this case, the rolling code in the RKE system advances,



(a) RollJam is particularly sensitive to timing; it has to be aware of the next valid unused code.

(b) A RollBack variant using only two captured signals at any time.

Fig. 2. Differences between RollJam and RollBack.

invalidating all previous codes, including the one possessed by the attacker. Consequently, properly suffixing the rolljam device at a hidden spot of the vehicle and replaying the *valid* unlock signal in a timely manner, i.e., step *(iv)*, are the keys to the success of RollJam. Second, similar to the above, if the attacker succeeds in using the captured valid yet unused signal, she cannot use it again; to repeat unlocking the same vehicle in the future, the whole attack must be redone from scratch.

In this article, we present RollBack, a new time-agnostic replay-and-resynchronize attack. Even though a one-time code becomes invalid in rolling code–based systems, replaying a few previously captured (consecutive) signals can trigger a rollback-like mechanism in certain RKE systems, making all former captured (unlock) signals valid again, hence, the name RollBack.[3] At the same time, the rollback-like mechanism involves the execution of the instruction encoded in the signals, e.g., unlocking the vehicle.

Consequently, unlike RollJam, RollBack does not have to keep track of the latest valid yet unused code continuously. In other words, we do not need the long step-sequence $(i) \rightarrow (ii) \rightarrow (iii) \rightarrow (i) \rightarrow (ii) \rightarrow (iv)$ to be repeated, and (iv), every time to eventually access the vehicle (see Figure 2). In general, RollBack does not need step *(iv)* at all, and only requires steps $(i) \rightarrow (ii) \rightarrow (iii) \rightarrow (i)$ once. Then, replaying the captured signals can unlock the victim's vehicle *any time in the future* and *as many times as desired*. This makes RollBack more flexible and time-agnostic, thereby substantially reducing the exertions required by an attacker.

Interestingly, RollBack only necessitates the jamming of the first signal (ii) to acquire the subsequent signal within a relatively brief period by making the victim press the button again. In normal circumstances, however, the time-agnostic nature of RollBack renders it irrelevant whether the captured signals are received by the vehicle; hence, jamming is not required. See further details regarding this characteristic later in Section 3.

During our analysis,[4] we found that not all vulnerable vehicles and RKE systems are equally susceptible to RollBack. Therefore, we derive *five* different variants of RollBack with regard to a

[3]Rollback is a process in database management that involves canceling a (set of) transaction(s) to bring the database to its previous state before those particular transactions would have been performed.

[4]Our crowd-sourced analysis is still ongoing. At the time of this writing, we have already tested around ~55 different vehicle makes, models, and RKE systems.

small set of properties (e.g., number of previously captured signals, sequence of the signals, time frame and pace of replay) required for the successful replay attack. We found that vehicles and RKE systems being the most vulnerable to RollBack can be unlocked with only *two signals captured any time in the past*. Moreover, these two signals do not even have to be strictly consecutive (see our definitions later), i.e., the victim can still use the key fob between the times the attacker manages to capture those two signals. This makes RollBack particularly alarming as, in addition to the aforementioned appealing properties, it further minimizes the required efforts of the attacker.

One notable enhancement in comparison to RollJam is that RollBack is *instruction-agnostic*, rendering it even more perilous. This implies that the nature of the captured signals, whether they correspond to lock or unlock instructions, becomes inconsequential, thereby simplifying the capturing process even further. Only the last captured and replayed signal needs to contain the desired instruction, specifically the unlock command, in order to gain access to the vehicle (see additional details about this property in Section 5.2).

Similar to RollJam and other RKE attacks, permanent mitigation might be cumbersome if RKE ECU firmware cannot be upgraded over-the-air, requiring calling back whole fleets of vehicles to the factory or dealerships. Some precautionary measures can be applied against signal jamming–based attacks, such as RollJam, by assuring proper signal reception by being close to the vehicle and pressing the lock button for the second try if the first unlock signal is not received. In certain scenarios, e.g., car-sharing use cases, risks can be minimized by disabling the RKE system until the vehicle is unlocked through the car-sharing app (see details in Section 9). Nevertheless, since RollBack, in essence, is a passive listener in the signal capturing phase without the need of signal jamming, besides the car-sharing advice, none of the previously mentioned approaches are applicable to mitigate RollBack.

Our main contributions are summarized below:

- After revisiting keyless entry systems and RollJam in more detail (see Section 2), we propose RollBack (see Section 3). In contrast to RollJam, it can unlock a vehicle *indefinitely* at *any time in the future* and *as many times as desired* by merely replaying previously captured (unlock) signals that are already invalid. Hence, RollBack is more effective.
- We delineate a (hidden) property of today's RKE systems that mimics the *modus operandi* of RollBack, hence making it the most relevant candidate to be the root cause of the vulnerability (see Section 8). However, for the time being, we could not ascertain whether our attack exploits an implementation bug or a limitation inherited from the design of the key fob re-synchronization or learning feature.
- Through a currently limited yet ongoing real-world experiment, we scrutinize the effectiveness of RollBack on a variety of popular vehicles[5] from different regions of the world. We show that around 40% of the RKE implementations worldwide are vulnerable to RollBack (see Section 4) and the Asian regions seem to be more affected.
- We propose five different variants of RollBack based on the requirements, e.g., number of different signals to capture and replay, the time frame and pace of replay, and the consecutiveness of the signals.
- We also discuss that, due to the re-synchronization and instruction-agnostic property of RollBack and the typical human behavior, astute attackers can rely on capturing lock signals to either fasten the signal capturing process (without signal jamming) or to cover the tracks by locking the vehicle again (see Section 5).

[5]We used our and our friends' and family members' vehicles with their consent due to responsibility issues.

- While the root cause of the attack is unknown mostly due to the lack of documentation, access to resources, and knowledge, we delineate a key fob learning process, as a potential root cause, that mimics the behavior or `RollBack`.
- Finally, we discuss possible mitigation strategies; some are precautionary measures the vehicle owner can take when `RollBack` requires signal jamming and advice to car-sharing services that are particularly vulnerable to `RollBack` (see Section 9). We also discuss possible practical mitigation, e.g., using timestamps.

2 BACKGROUND AND RELATED WORK

Next, we briefly discuss the evolution of the keyless entry systems. Then, we present the main types of attacks that emerged against this fundamental feature of today's vehicles.

2.1 The Evolution of Keys and Entry Systems

2.1.1 Physical Keys. For several decades after the very first car was made in 1886, vehicles had no key at all [26]. The first key was introduced in 1949 by Chrysler Corporation for ignition and starting the engine [16]. It also acted as a safety precaution to prevent children from accidentally starting and moving the car if left in gear.

2.1.2 Immobilizer. To deter vehicle theft, Honda has made the first keys enhanced with a so-called immobilizer. The immobilizer is a passive device that uses RFID (Radio Frequency IDentification) technology to communicate with the transponder near the keyhole and verifies the legitimacy of the key fob before starting the engine. Without the correct transponder, the keyhole is either mechanically blocked, preventing illegitimate keys from turning, or ECUs will not let the fuel flow and start the ignition. Research conducted in Australia and the European Union have shown that car thefts have been significantly reduced after making immobilizers mandatory [35, 42].

2.1.3 Remote Keyless Entry (RKE). RKE is an unidirectional authentication system. In RKE, besides advanced features that recently became available (e.g., start, stop, panic), the user unlocks or locks the vehicle by pressing the corresponding button on the key fob. When a button is pressed, Radio Frequency (RF) signals are emitted towards the car in the frequency bands of 315 MHz, 433 MHz, or 868 MHz depending on the geographic location. The receiver located in the vehicle receives the RF signals (from even up to hundreds of meters) and carries out the intended action (e.g., lock, unlock).

Note that the selection of the frequency band is primarily aimed at preventing any interference with other services authorized by government regulations. It does not impact the operational functionality of the equipment itself but rather establishes the specific spectrum within which the radio signals are transmitted.

2.1.4 Passive Keyless Entry System (PKES). Unlike RKE, the PKES operates automatically when the user, i.e., the key fob, is near the vehicle. Also, PKES uses bidirectional challenge-response communication for appropriate authentication. PKES allows the owner with the correct key fob to unlock and automatically lock the car by pulling the door handle and when the owner walks away, respectively. PKES key fobs are also integrated with RKE, i.e., it still has buttons as a fail-safe/secondary mechanism or feature for drivers in favor of the "old-fashioned" button-based operation.

While the PKES also uses rolling codes, due to the owner's proximity and the fact that an attacker does not know when the unlock signals are emitted, they are significantly less vulnerable to typical replay attacks that affect RKE systems. However, they are susceptible to relay attacks [10].

In this article, we focus on the RKE systems exclusively.

2.2 Rolling Codes

Next, we briefly discuss the evolution of rolling codes used in RKE systems and define some notations used later in the article. The history of RKE systems reaches back to the 1970s [37], when early motorized garage openers used static codes sent in "plain text" over the air to carry out the intended action (e.g., open, close). However, by merely sniffing and replaying captured signals, attackers were able to easily unlock garage doors. To overcome this issue, rolling codes [24] were introduced, and they have been widely used due to its increased protection (compared with static codes) yet with less computation complexity (compared with the increased protection). The latter property is particularly important as it results in small and simple key fobs with an average battery life of up to four years [46].

There are a few well-known manufacturers providing rolling code–based RKE systems for the automotive industry. For instance, Microchip Technology provides `Classic, Advanced`, and `Ultimate KeeLoq` with publicly available documentation and data sheets. On the other hand, semiconductor companies such as NXP [28], Omron, and Texas Instruments also provide proprietary solutions for vehicle manufacturers. For the technical explanations below, we focus on RKE systems using the `Classic` and `Advanced KeeLoq` technology since their documentations are publicly available. Note, however, in essence, all rolling code–based technologies are conceptually similar.

Applying the rolling code technology means that every key fob signal transmission is unique, i.e., it changes with every individual button press. Uniqueness is achieved by incrementing a 16-bit wide *counter*[6] in the key fob (and in the vehicle upon reception) with each button press. A button press is valid if the counters at each side are in sync. Then, each of the parties increments its counter[7] to be in sync for the following button press. Accordingly, if an attacker captures a valid signal sent from the key fob and received by the vehicle with counter $C_k = n$ and replays it, it will be discarded by the receiver in the vehicle as its counter $C_v > C_k$, i.e., $C_v = (n + k) : k > 0$.

On the other hand, provision is made for cases in which a button is pressed on the key fob while it is out of range of the vehicle, i.e., when using the key fob to lock/unlock the car and $C_k > C_v$. These cases are further divided into two different *operation windows* [25, 40].

2.2.1 Single Window. If $C_{diff} = C_k - C_v$ is small,[8] e.g., $C_{diff} < 16$, counter synchronization takes places immediately at the first button press without the need of any additional steps. Counter synchronization means that the receiver unit in the vehicle invalidates all non-received codes before the one present in the last key fob signal.

2.2.2 Resync/Double Window. If $16 < C_{diff} < 2^{15}$, the receiver temporarily stores the counter $C_k = l$ and waits for a subsequent transmission, i.e., the same button has to be pressed once more. If the subsequent transmission has counter $C_k = l + 1$, the receiver resynchronizes on the last transmission received. Observe that the synchronization requires two button presses and the vehicle acts only upon the reception of the second one when synchronization finishes.

If any of the above fails,[9] the key fob signal received by the vehicle is discarded. Note, furthermore, that due to the underlying encryption mechanisms (e.g., in [40]), the change of even one bit of information (e.g., counter increment) results in a significant change in the final transmitted signal. Hence, it is computationally infeasible for an attacker to infer the next valid, say, unlock signal by capturing the previous one.

[6]Recent advanced implementations, e.g., `Ultimate KeeLoq`, also maintain timestamps to improve security [40]. However, it is not confirmed whether RKE manufacturers have already adopted them.

[7]For simplicity, here, we suppose an integer increment of 1. However, in reality, the next valid counter is generated via cryptographic hash functions.

[8]Note that different manufacturers use different thresholds.

[9]This window is termed *blocked window* [25].

2.3 Related Work: Different Attacks Against RKE Systems

In essence, the design of the rolling code scheme should provide a sufficient level of security. However, the earliest deployments have been proven to be breakable. For instance, `Classic KeeLoq` technology currently primarily used by garage doors only, was broken by cryptoanalysis [1, 3] and side-channel attacks on the key derivation scheme used by the receiver [8, 20]. Subsequently, enhanced `KeeLoq` implementations, i.e., `Advanced KeeLoq` and `Ultimate KeeLoq`, have addressed these issues by using stronger encryption algorithms and longer keys [40].

Another simple yet efficient method criminals use against rolling code–based key fobs is jamming the signals when victims press the lock button to hinder the vehicle from receiving it correctly. If it happens without the victim noticing it, the car is left unlocked. A more sophisticated variant of this attack is "selective jamming and replaying": besides the previously mentioned jamming, the attackers also capture the lock signal. Consequently, if this happens again without the victim noticing it, the criminals can lock the vehicle after stealing all belongings to leave a false impression of the car having been left adequately locked. Note that once a signal is captured, without additional knowledge (e.g., encryption keys, command code table), it is impossible to convert it into another signal, i.e., flipping a lock signal to an unlock is infeasible.

HITAG2 from NXP, another widely used RKE scheme using rolling codes, has been used by many car manufacturers worldwide (e.g., Renault, Ford, Chevrolet, Lancia, Opel). Recently, researchers have demonstrated a correlation-based attack allowing the recovery of the cryptographic key and thus cloning the key fob having captured only four to eight rolling codes [12]. Furthermore, the research also revealed that most VW Group vehicles (e.g., VW, Seat, Audi, Porsche) manufactured since 1995 rely on a few master keys. By recovering these keys from the ECUs, an attacker can effortlessly clone the key fob of any such vehicle by only capturing one unlock signal.

In 2015, Samy Kamkar proved all rolling code-based schemes to be breakable with his RollJam [18] attack. RollJam relies neither on any cryptoanalysis nor side-channel attacks; it converts a safety feature into an exploit. In essence, RollJam is an advanced "selective jamming and replaying" method; with a careful sequence of jamming, capturing, and replaying signals, it allows an attacker to capture an unused signal from the key fob that can be replayed later to unlock the target vehicle without the victim noticing it. As briefly discussed in Section 1, RollJam is based on four principles, *(i)* capturing unlock signals, *(ii)* jamming the frequency band towards the vehicle at the same time to force the owner *(iii)* to retry, and *(iv)* timely replaying of previously captured signals.

The operation of RollJam is summarized in Figure 2(a). When the unlock button is pressed on the key fob, the rolljam device hidden on or near the target vehicle *(i)* captures the signal and, at the same time, *(ii)* jams the frequency band towards the vehicle to hinder correct signal reception. Since the vehicle does not respond, *(iii)* the owner presses the same button again, assuming poor signal reception. This time, however, the rolljam device repeats not only steps *(i) and (ii)*, but also quickly *(iv)* replays the previously captured signal towards the vehicle (without jamming). As a result, the vehicle acts as intended, i.e., unlocks the doors. In addition, the rolljam device becomes aware of the next valid code for the same action, i.e., it knows what signal to send to unlock the car again in the future. However, if the owner uses the key fob to unlock the car again without the rolljam device in action, the signal the attacker possesses will be invalidated, forcing her to redo the whole process. While RollJam, in general, is effective against all rolling code–based RKE systems, it requires careful and continuous attention due to *(iv)*.

Recently,[10] an attack called Rolling-PWN [21] saw the light of day and hit the headlines of several online news sites, e.g., the New York Post [2], The Drive [38], and Security Affairs [31].

[10] Around a month before the Black Hat debut of `RollBack`, i.e., in the beginning of July 2022.

The authors of Rolling-PWN found that Honda vehicles manufactured between 2012 and 2022, implementing rolling code–based RKE systems, are vulnerable to replay attacks. In particular, the authors found a somewhat similar behavior to RollBack;[11] sending the unlock commands in a consecutive sequence to the Honda vehicles will resynchronize the counter. However, the required sequence of codes, exactly how many codes need to be captured and replayed, or any other relevant (hardware-specific) details have not yet been disseminated publicly.

3 ROLLBACK: A NEW TIME-AGNOSTIC REPLAY ATTACK

Next, we propose RollBack, a new time-agnostic replay attack, which by exploiting a hidden property in the RKE systems overcomes the limitations of RollJam. In particular, RollBack can unlock a vehicle by simply capturing and replaying a few already invalidated unlock signals at *any time in the future* and *as many times as desired* without the need of recapturing any further signals later on.[12] In what follows, we describe the threat model of RollBack by using the same setting as shown for RollJam (i.e., by applying signal jamming) to ease the comparison. However, while jamming can fasten the attack process, unlike RollJam, RollBack *does not necessitate signal jamming* at all.

3.1 Threat Model and the Operation of RollBack

The primary goal of the attack is to unlock a vehicle without the victim's authorization (and, potentially, the victim noticing). Like in all RKE attacks, the vehicle becomes unlocked the same way as using the original key fob, leaving the car intact.

In our threat model, the attacker has a device that can capture, jam, and replay signals in the frequency band used by the target vehicle. For simplicity, let us call this device RollBack-device. In particular, let $\mathcal{S}_I^i$ denote a key fob signal sent towards the vehicle with a rolling code counter $i \in \{1, 2, \ldots, 2^{15}\}$ and an instruction $I := \{unlock, lock\}$. For instance, $\mathcal{S}_{unlock}^{534}$ marks an *unlock* signal with rolling code counter $i = 534$. Furthermore, let $Capture_A(\mathcal{S}_I^i)$ and $Jam_A(\mathcal{S}_I^i)$ denote that an attacker A captures the key fob signal $\mathcal{S}_I^i$ and jams the frequency band toward the vehicle, respectively, at the same time, i.e., when $\mathcal{S}_I^i$ was sent by the victim. Finally, let $Send_V(\mathcal{S}_I^i)$ and $Send_A(\mathcal{S}_I^i)$ mark when the victim (V) and the attacker (A) send $\mathcal{S}_I^i$ using the original key fob and using a special-purpose device intended to replay captured signals, respectively.

The functioning of RollBack (see Figure 2(b)) can be categorized into two distinct phases: reconnaissance and exploitation.

3.1.1 Reconnaissance Phase. The attacker places the RollBack-device near the car that is locked and left in public (e.g., in a parking lot). When the victim comes back to his/her car and tries to unlock it via the key fob, i.e., when the victim runs $Send_V(\mathcal{S}_{unlock}^i)$, the RollBack-device *(i)* captures the signal ($Capture_A(\mathcal{S}_{unlock}^i)$), and *(ii)* jams the frequency band ($Jam_A(\mathcal{S}_{unlock}^i)$) to hinder the vehicle from receiving it correctly (recon. phase 1. in Figure 2(b)). As a result, the victim assumes poor reception and *(iii)* presses the same unlock button again, i.e., the victim runs $Send_V(\mathcal{S}_{unlock}^{i+1})$ (recon. phase 2. in Figure 2(b)). This time, the RollBack-device captures the second consecutive unlock signal (i.e., it runs $Capture_A(\mathcal{S}_{unlock}^{i+1})$). *However*, unlike RollJam, it also lets the car receive it, i.e., the attacker *does not* run ($Jam_A(\mathcal{S}_{unlock}^{i+1})$). Accordingly, the vehicle unlocks, and the victim drives away, assuming that no harm has been done (idle phase in Figure 2(b)). Note that since RollBack does not have to keep track of the next valid unlock signal, it is unnecessary to suffix the RollBack-device to (a hidden spot of) the vehicle. Practically speaking, due to the size of

[11] Twitter: https://bit.ly/3wZrCf4

[12] See RollBack in action at Youtube: https://bit.ly/3RB1LSu

the inexpensive elements needed (see later in Section 3.2), such a special-purpose wallet-size [13] `RollBack`-device can be simply thrown below the vehicle. At the end of the reconnaissance phase (after recon. phase 2. in Figure 2(b)), the attacker becomes aware of two consecutive correct unlock signals. Recall, by the rolling code design, that both captured signals are *no longer valid*.

3.1.2 Exploitation Phase. Unlike RollJam, this phase does not have to follow the first phase directly. In other words, the victim can continue to lock, unlock, and use the car as usual as *many times* the victim wants (idle phase in Figure 2(b)). Nevertheless, at any given latter time, once the vehicle is locked, the attacker can unlock the vehicle (without the victim's authorization) by replaying the previously captured two consecutive unlock signals, i.e., by running $Send_A(\mathcal{S}^{(i)}_{unlock})$ and $Send_A(\mathcal{S}^{(i+1)}_{unlock})$ (exploit. phase in Figure 2(b)).

For brevity, our threat model does *not* cover further intentions of the attacker after unlocking the vehicle. The attacker might steal belongings left inside the car or use other attack methods (if necessary) to steal the vehicle itself.

3.2 Essential Hardware

For our comprehensive analysis, we use Software Defined Radio (SDR) devices. In essence, these devices have wireless receivers (and transmitters) that can be fine-tuned via software, for instance, in which frequency domain they should listen to signals. One of the most well-known commercial-off-the-shelf (COTS) devices is HackRF One [11], which is capable of both transmitting and receiving signals, and costs ~300−400 USD at the time of writing. The COTS software, called `gqrx` can be used to easily identify the exact frequency used by the key fob to transmit the signals. On the other hand, since all key fobs operate in the licensed spectrum, they all (must) have a unique registered identifier with the U.S. Federal Communications Commission (FCC). Therefore, one can look up the publicly available details of a key fob by keying in its FCC ID at https://fccid.io/. Once the correct frequency is identified, the other COTS software, called Universal Radio Hacker (URH, [34][13]), can be used to control SDR devices, i.e., to practically capture and replay (the unlock) signals. To jam the frequency using the SDR device, an attacker has a large variety of options. What the attacker chooses depends completely on her appetite and knowledge. For instance, she might use inexpensive programmable development boards and radio transmitters, such as Arduino-based deployments, or even a Raspberry Pi with a full-fledged operating system and RTL-SDR dongles [36] for reception and/or CC1101 transceivers for jamming [39].

Note that, essentially, `RollBack` relies on the exact hardware requirements as RollJam. Moreover, since jamming is not necessarily needed (see Section 3) for the success of `RollBack`, a `RollBack`-device has even less requirements. Therefore, it would cost no more than 20-30 USD [15].

Note, furthermore, that similar to RKE attacks (e.g., RollJam), RollBack does not necessitate execution in an isolated environment. Just as in a regular scenario in which one unlocks a vehicle using RKE in a crowded parking lot, the simultaneous use of other RKE systems has a minimal impact on the success of the attack. The crucial factor lies in the attacker's ability to capture the signals; replaying them can be done easily and repeatedly, if necessary, at any given time. Moreover, during the signal capture process, the capturing device's signal reception bandwidth window is intentionally narrowed down compared with the vehicle's bandwidth windows (see Figure 3). This allows effective jamming of the vehicle's bandwidth window without hindering the attacker's signal capture capabilities.

[13]There are several other publicly available free and/or open-source software packages, e.g., GNURadio and OpenSDR, that can be used for the same purpose.

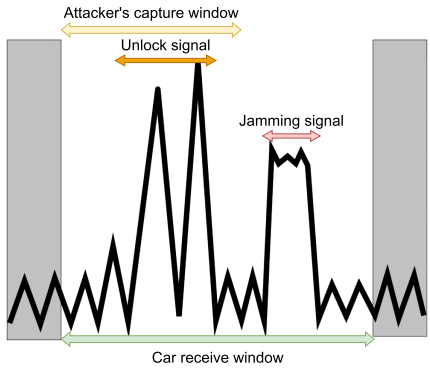


Fig. 3. Signal capturing and jamming. The jamming signal is within the range of the vehicle's receive window. On the other hand, the attacker's capturing device's spectrum is purposely narrowed down to avoid being affected by the jamming.

3.3 Different Variants of RollBack

When we first discovered the vulnerability, we had tested a pretty outdated vehicle, a Nissan Latio from 2009 (see details in Section 4.1). In this case, RollBack had the following properties.

Naturally, first, we identified how many signals we needed to replay. In the case of the Nissan Latio, this number turned out to be only *two*; however, as we will show, other vulnerable systems might require more than that. Accordingly, the first (and most important) property of RollBack is the number of signals (i.e., #SIGNALS) an attacker has to capture (and replay).

The second observation we had is that the attacker strictly has to run $Capture_A(\mathcal{S}^{i}_{unlock})$ and $Capture_A(\mathcal{S}^{i+1}_{unlock})$ and replay them in the same sequence. Put differently, capturing and replaying, for instance, $\mathcal{S}^{i}_{unlock}$ and $\mathcal{S}^{i+k}_{unlock} : k > 1$ does not trigger the expected rollback-like mechanism. Hence, we call the second property SEQUENCE and it can be Strict (as in the case of the Nissan Latio mentioned before), or Loose if it is not required, i.e., when replaying signals in the capturing (i.e., ascending) order is sufficient but there could be further valid and forfeited signals in between.

Furthermore, in the case of the Nissan Latio, we observed that the two consecutive unlock signals have to be replayed *within 5 seconds*; otherwise, RollBack is unsuccessful. We termed the third property TIMEFRAME; it indicates the maximum number of seconds that can elapse between two signals when replayed. We indicate TIMEFRAME as $\bigotimes$ when there is *no limit* on the maximum number of seconds. When TIMEFRAME $\neq \bigotimes$, we confirmed the value of TIMEFRAME, by carefully trimming gaps between the captured signals to exactly $N = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ seconds. Then, we saved the signals, replayed them, and observed whether RollBack succeeds. Note that once the signals are captured, TIMEFRAME can be easily adjusted via the SDR software by cutting or copy-pasting the breaks or background noises between the signals.

During our analysis (detailed later in Section 4), we derived *five* different versions of RollBack regarding the properties mentioned above. The different combinations are summarized in Table 1.

4 EVALUATION

Next, we evaluate RollBack and discuss which vehicles are vulnerable.

Disclaimer. For our experiments, we *did not* carry out any attempts with RollBack in the wild. All tests were executed in an isolated environment, where no other vehicles and/or key fobs were

Table 1. Different Variants of RollBack Derived from Our Analysis

Variant	#SIGNALS	SEQUENCE	TIMEFRAME
$\texttt{RollBack}^{\texttt{Loose}}_{\otimes}(2)$	2	Loose	$\otimes$
$\texttt{RollBack}^{\texttt{Strict}}_{\otimes}(2)$	2	Strict	$\otimes$
$\texttt{RollBack}^{\texttt{Strict}}_{N}(2)$	2	Strict	N sec
$\texttt{RollBack}^{\texttt{Strict}}_{\otimes}(3)$	3	Strict	$\otimes$
$\texttt{RollBack}^{\texttt{Strict}}_{\otimes}(5)$	5	Strict	$\otimes$

Each variant encodes all properties as $\texttt{RollBack}^{\texttt{SEQUENCE}}_{\texttt{TIMEFRAME}}(\texttt{\#SIGNALS})$.

in close proximity. Note, however, as mentioned in Section 3.2, isolated environment is needed to execute RollBack in the wild. All captured signals (for the tests) were stored temporarily only; after capturing the signals and replaying them, the data was removed permanently immediately. We stored two key fob signals for a longer period, i.e., ~ 100 days, to validate RollBack's time-agnostic feature. Afterward, those stored signals were also removed permanently. Note that replaying key fob signals do not cause any harm to the vehicle, the key fob, and the whole electronic ecosystem irrespective of being vulnerable to RollBack. Thus, the tested vehicles continue to work and behave as usual.

This article is the first publicly disseminated, detailed written information about our findings and about RollBack in general. We used the shorter and more condensed preliminary versions of this document during our attempts t initiate disclosure processes with RKE chip manufacturers and AUTO-ISAC members. See more details about the disclosure processes and findings in Section 7.

4.1 Vehicles Evaluated

In the beginning, our examination was confined to a restricted selection of vehicles, primarily focusing on Asian makes and models prevalent in Singapore. However, following the presentation of our initial discovery [5], numerous automotive cybersecurity experts, passionate enthusiasts, and car owners have endeavored to replicate RollBack against their own vehicles. In fact, RollBack has even been successful in different domains, e.g., smart door locks [23]. Nevertheless, many have shared their findings by contributing to our publicly accessible crowd-sourced database.[14] The vehicles examined and their relevant data are detailed in Table 2 and Table 3. Model date means the time frame the actual model was in production, whereas the Mfg. date denotes the actual manufacturing date of the vehicle we tested. Such information can usually be obtained by using the vehicles' identifier, i.e., their vehicle identification numbers (VINs), and publicly available services.[15] The region refers to the geographical region where the vehicle is used and registered, which potentially implies the manufacturing location as well.

Different vehicles and their key fobs use different frequencies. However, since the used frequency did not have an impact on whether the vehicle is vulnerable to RollBack (see Section 2.1.3), we omit the exact frequency bands. We could also obtain the exact RKE manufacturer and chip version and serial number most of the time by manually disassembling the key

[14]The online crowd-sourced database is available here: https://tinyurl.com/2p99vd7c
[15]One can rely on https://vindecoderz.com to check all publicly available basic servicing information about a vehicle by using its VIN number.

Table 2. Details of the Vehicles Tested Against RollBack

Car Make	Model	Model date	Mfg. date	RKE manufacturer	RollBack (variant)	Region
BMW	1 series (e87)	2004–2013	2005	NXP 26A0700	NO	Europe
	X3	2011–2017	2011	NXP F7953	NO	North Am.
Chrysler	Pacifica	2003–2007		NXP F7941	NO	North Am.
Dacia	Spring	2021-	2021	NXP 61M1300	NO	Europe
Fiat	500*	2007–2023	2013		NO	Europe
Honda	Fit (hybrid)	2016–2018	2016	NXP F2951X	$\text{RollBack}^{\text{Strict}}_{\otimes}(5)$	Asia
	Fit	2018	2018	NXP 61X0915	$\text{RollBack}^{\text{Strict}}_{\otimes}(5)$	Asia
	Brio*	2016	2011–2020		$\text{RollBack}^{\text{Strict}}_{\otimes}(5)$	Europe
	City	2017	2017	NXP F2951X	$\text{RollBack}^{\text{Strict}}_{\otimes}(5)$	Asia
	Mobilio RS Navi*	2017	2014-		$\text{RollBack}^{\text{Strict}}_{\otimes}(5)$	Europe
	Vezel	2016–2022	2017	NXP F2951X	$\text{RollBack}^{\text{Strict}}_{\otimes}(5)$	Asia
Hyundai	Elantra	2013–2015	2015	Omron MD-015	$\text{RollBack}^{\text{Loose}}_{\otimes}(2)$	Asia
	Elantra	2012	2012	NXP 32182C[16]	NO	Asia
	Avante	2018–2020	2020	NXP F7936[17]	NO	Asia
	i20*	2014–2020			NO	Europe
	ix20*	2010–2019	2012		$\text{RollBack}^{\text{Strict}}_{\otimes}(2)$	Europe
Kia	Cerato/Forte K3	2016–2018	2017	Omron MD-011	$\text{RollBack}^{\text{Loose}}_{\otimes}(2)$	Asia
	Cerato/Forte K3	2012–2018	2015	Omron MD-011	$\text{RollBack}^{\text{Loose}}_{\otimes}(2)$	Asia
	Ceed	2018-	2022	NXP A1M05	NO	Europe
	Sportage	2021	2021	NXP A1M05	NO	Europe
Mazda	3	2018	2018	NXP A2V25	$\text{RollBack}^{\text{Strict}}_{\otimes}(3)$	Asia
	3*	2009–2013	2009		NO	Europe
	3*	2019–2023	2019		NO	Europe
	2*	2014–2022	2015		NO	Europe
	2 Sedan	2018	2018	NXP F7953	$\text{RollBack}^{\text{Strict}}_{\otimes}(3)$	Asia
	2 Sedan		2017	NXP F7953	$\text{RollBack}^{\text{Strict}}_{\otimes}(3)$	Asia
	2 HB (facelift)	2020	2020	NXP A2V25	$\text{RollBack}^{\text{Strict}}_{\otimes}(3)$	Asia
	Cx-3	2019	2019	NXP A2V25	$\text{RollBack}^{\text{Strict}}_{\otimes}(3)$	Asia
	Cx-5	2018	2018	NXP F7953	$\text{RollBack}^{\text{Strict}}_{\otimes}(3)$	Asia
	6	2002–2005	2004	9861 082	NO	Europe
Mitsubishi	Montero GLS*	2019			NO	Europe
Nissan	Teana	2014	2014	NXP 063168C	NO	Asia
	Latio	2007–2012	2009	Microchip	$\text{RollBack}^{\text{Strict}}_{\$}(2)$	Asia
	Sylphy	2012–2019		NXP F7952	$\text{RollBack}^{\text{Strict}}_{\$}(2)$	Asia
	Navara*	2010			$\text{RollBack}^{\text{Strict}}_{\otimes}(2)$	Asia
Opel	Vivaro	2001–2014	2010	NXP F7946	NO	Europe
	Crossland X	2017-	2018		NO	Europe
	Astra*	2015–2021	2017		NO	Europe
Renault	Megane	2008–2016		NXP F7953	NO	Europe
	Clio*	2005–2013	2008		NO	Europe

For the vehicles for which the release date and manufacturing date are the same, only the manufacturing date was available by using the vehicle's identifier (VIN). The cells for which we could not identify a certain property of the vehicle are intentionally left blank. The data for car models denoted with an asterisk are crowd-sourced.

[16]Inferred from https://bit.ly/3POlZaz.
[17]Inferred from https://bit.ly/3OrwbEV.

Table 3. Continuation of the Details of the Vehicles Tested Against RollBack

Car Make	Model	Model date	Mfg. date	RKE manufacturer	RollBack (variant)	Region
Toyota	Wish	2009–2017			NO	Asia
	Corolla Axio	2015–2017		TI 37143ADN	NO	Asia
	Altis	2005		TI 37200A	NO	Asia
	Prius (hybrid)	2020	2020	TI	NO	North Am.
	Rush*	2017			$\texttt{RollBack}^{\texttt{Strict}}_{\otimes}(2)$	Asia
	Wigo S*	2017			$\texttt{RollBack}^{\texttt{Strict}}_{\otimes}(3)$	Asia
	Hilux Conquest*	2020			NO	Asia
	Fortuner*	2006			NO	Asia
	C-HR*	2016–2023	2020		NO	Asia
	Rav4*	2006–2013	2008		NO	Europe
Volkswagen	T-Cross*	2023			NO	Asia
	T-ROC*	2017–2023			NO	Europe
	Touareg	2006–2010	2008	NXP F7943	NO	Europe

fobs.[18] When disassembling the key fob was either infeasible or the chip(s) on the PCB were obscured (e.g., via black paint), we tried to gather manufacturer information by keying in its FCC ID at https://fccid.io/ or looking for spare key fobs on different retailers' sites. The found chips are detailed in the penultimate column of Table 2 as well as in Table 3. If we could not obtain the RKE manufacturer by any of the above-mentioned ways, we left the corresponding cells in Table 2 intentionally blank.

Finally, the last column indicates whether the vehicle — or, more precisely, the RKE system — is vulnerable to RollBack (indicated by the actual RollBack variant that works).

From the experiment (cf. Table 2 and Table 3,), which is continuously being updated, we can conclude the following.[19] First, ∼ 50% of the examined Asian vehicles were found to be vulnerable to a RollBack variant. However, the examined vehicles in other regions show less vulnerability to RollBack (for now). For instance, we found that Mazda vehicles manufactured in Asia tend to be vulnerable, whereas their European counterparts (even the same model) are not vulnerable to RollBack.

On the other hand, we can observe that the vulnerability is not specific to any sole vehicle, car make, or model. While the age (i.e., model and manufacturing date) does not seem to be a deciding factor (e.g., while Mazda 6 with a model date 2002–2005 is resilient to RollBack, a newer model Mazda Cx-5 with a model date 2018 is prone to the attack), the used RKE system's manufacturers *might be* a telltale sign. Specifically, all tested Korean vehicles (such as Kia and Hyundai) employing RKE systems from Omron were found vulnerable in every instance. Notably, the exploit only necessitates the use of two unlock signals, which could even be captured independently in the past (i.e., SEQUENCE=Loose). Drawing similar conclusions about NXP, however, is not possible since the assessment of multiple vehicles equipped with NXP transponders in their key fobs revealed that some were found to be secure while others were deemed unsafe. Furthermore, we observe that all three tested Toyota vehicles, for which we identified the key fob manufacturer as Texas Instruments, turn out to be immune to RollBack. According to further analyses on Toyota vehicles, however, our database showed that this, in general, does not mean that all Toyota vehicles are protected. In particular, Toyota Wigo S and Toyota Rush have been found to be vulnerable. Accordingly, we cannot conclude at the moment whether vehicles equipped with RKE systems from Texas Instruments are generally resilient to RollBack.

[18]In some cases, the key fob's printed circuit board had an extra plastic cover, which could not be removed without causing permanent damage.

[19]Please contribute to our crowd-sourced database if you have tested RollBack by filling out this form: https://tinyurl.com/4t95jprh

Last, but not least, Microchip RKE systems were probably more ubiquitous in the past. However, their rolling code–based solution can still be found in today's vehicles, which means they might all be vulnerable to RollBack.

We summarize the results on the rest of the vehicles (that either use RKE systems from manufacturers other than NXP/Omron or for which we could not identify the RKE manufacturer) as follows. Fiat 500, Hyundai i20, Mazda 2, Mazda 3, Mazda 6, Mitsubishi Montero GLS, Opel Crossland X, Opel Astra, and Renault Clio have been found to be immune to RollBack. Honda Brio, Honda Mobilio RS Navi, Hyundai ix20, Nissan Latio, and Nissan Navara are prone to RollBack.

According to the latest dataset, 40% of the tested vehicles worldwide turned out to be vulnerable to RollBack. Within this set, 20% do not require the signals to be replayed strictly consecutively, which is particularly alarming. Moreover, in all cases, RollBack requires capturing 2 signals only. Nonetheless, among all vulnerable vehicles, less than 20% require 5 signals to be captured; this number is 38% and 43% for 2 and 3 signals, respectively.

Bear in mind that not the key fob (as it only sends the signals) but its counterpart (i.e., the receiving unit in the car *per se*) seems to be vulnerable. Moreover, the key fob manufacturer usually produces key fobs (i.e., the transponders) *only*, and different original equipment manufacturers (OEMs) supply the receiving units. Yet, our results indicate a strong relationship between the key fob manufacturer and the receiving unit as, with the exception of NXP F7953, we have not found any two RKE systems that use the same transponder chip in their key fobs but react differently to RollBack.

Bear in mind that, like other key fob–based attacks such as RollJam, RollBack targets a specific vehicle. The signals captured during the reconnaissance phase of RollBack are unique to that particular car and cannot be utilized across different vehicles regardless of their make, model, or other distinguishing features. For instance, the key fob signals obtained for our Mazda 2 HB (facelift) vehicle are exclusively applicable to that specific vehicle and cannot be applied to compromise an entire fleet of the same Mazda models produced in the same year.

5 FURTHER APPEALING FEATURES OF ROLLBACK

This section discusses how easily *attackers might hide their tracks* after accessing a vehicle and shows that RollBack, in certain cases, can be successfully launched with even less effort, i.e., *without the need for signal jamming*.

5.1 Re-locking the Vehicle After Access

Recall that due to the counter re-synchronization, if subsequent signals are captured and replayed, they also work as expected straight away afterward. Using the notations defined in Section 3.1, assume that the attacker not only captures consecutive unlock signals (e.g., $Capture_A(\mathcal{S}^{i}_{unlock})$, $Capture_A(\mathcal{S}^{i+1}_{unlock})$ in the case of $\texttt{RollBack}^{\text{Loose}}_{\bigotimes}(2)$) but also captures a following lock signal $\mathcal{S}^{i+2}_{lock}$ (i.e., $Capture_A(\mathcal{S}^{i+2}_{lock})$). In this case, irrespective of whether the victim continues to use the key fob as normal (i.e., whether the last signal received by the car is $\mathcal{S}^{i+2}_{lock}$ or $\mathcal{S}^{i+j}_{(un)lock} : j > 2$), after $Send_A(\mathcal{S}^{i}_{unlock})$ and $Send_A(\mathcal{S}^{i+1}_{unlock})$ (in the case of $\texttt{RollBack}^{\text{Loose}}_{\bigotimes}(2)$), the vehicle unlocks and also re-synchronizes to the counter $(i+1)$. Accordingly, after the attacker has accessed the vehicle, when running $Send_A(\mathcal{S}^{i+2}_{lock})$, the car will lock, giving a false feeling to the owner of having the vehicle left adequately locked.

5.2 RollBack is Instruction-agnostic

To accomplish re-synchronization using RollBack, the specific instructions within the signals are generally insignificant except for the last signal. Particularly, a combination of lock and unlock signals can be replayed to initiate the counter re-synchronization process. However, it is crucial

that the last signal in the sequence is an unlock signal in order to ultimately unlock the vehicle, as the vehicle will respond based on the instructions provided in the final signal. For instance, in the case of $\texttt{RollBack}^{\texttt{Loose}}_{\bigotimes}(2)$, capturing and replaying one lock signal and then an unlock signal is sufficient to unlock the target vehicle. Suppose now that the attacker captures the lock signals emitted when the victim left the vehicle in a parking lot (i.e., $Capture_A(\mathcal{S}^i_{lock})$). Then, the attacker waits for the victim to come back and unlock the vehicle; this time the attacker runs $Capture_A(\mathcal{S}^{i+1}_{unlock})$. Recall that, in the case of $\texttt{RollBack}^{\texttt{Loose}}_{\bigotimes}(2)$, the second signal does not even have to be strictly consecutive, i.e., the attacker can simply capture any following unlock signal (e.g., $Capture_A(\mathcal{S}^{i+k}_{unlock} : k > 1)$) to unlock the vehicle. After replaying these two signals in sequence, the vehicle will be locked and re-synchronized to the counter $(i + 1)$, and the vehicle will react according to the instruction in the last signal, i.e., it unlocks.

This makes RollBack particularly alarming as this signal sequence can be easily captured at once without applying any signal jammer. Moreover, even if the vehicle is susceptible to a RollBack-variant that requires more signals, they can also be captured without jamming due to the following typical human behavior and the vehicles' safety features. For instance, when we leave something worthy unattended (e.g., the vehicle in the parking lot or the main entry door to our home), we usually confirm whether the locking was done adequately. For this reason, most of us still push (down the handle on) the door of our home after locking to double-check whether the lock itself is not malfunctioning. Similarly, it is always worth pressing the lock button on the key fob once more when we leave our vehicle behind since it confirms adequate locking by flashing the emergency signals and/or honking.

Pressing the lock button again (for a third or even more time) afterward, thereby making the vehicle honk, can also become handy afterward. People tend to use this feature in huge parking lots to locate the vehicle *per se.*

On the other hand, vehicles usually implement a safety feature when unlocking the car via the key fob. This feature allows the owner to only unlock the driver's door upon pressing the unlock button for the first time. However, if one does not drive alone, giving access to the other co-riders (e.g., family members), we have to press the unlock button twice to unlock all doors.

These features and usual human factors enable all RollBack-variants to be successfully launched without the need for any signal jammer.

6 CAR-SHARING SERVICES: THE MOST ATTRACTIVE TARGETS OF ROLLBACK

Car sharing has recently been viral, especially in countries where the cost of ownership for a vehicle is extremely high compared with the average. Car sharing, in essence, makes classic car renting much more accessible, more convenient, and much cheaper. Instead of renting a vehicle for at least a day, doing a lot of paperwork in person, and getting lost among the different insurance policies and waivers, car-sharing costs are significantly lower due to the non-necessity of staff, hour or minute-based conditions, and the convenience of using a mobile application to access and lock the vehicle in the beginning and at the end of the rental, respectively.

The typical steps of car-sharing are as follows. Users (already registered for the service) can use the mobile app to book a car (for a certain period). Once the booking timeslot starts, the user can unlock the vehicle by instructing the mobile application to do so. In the background, the car-sharing company's service remotely unlocks the vehicle utilizing additional ECUs added to the car for this specific purchase. Once the vehicle unlocks, the user will find the original key fob at a hidden spot in the car (usually in the glove compartment); then,the user can start driving. Note that, typically, there are further steps the car-sharing company might require (e.g., photo-taking, damage checking, and petrol level checking) However, from our attacker's point of view, they are not relevant. After the user returns the vehicle to a designated parking lot, the individual has to

put back the key fob in the hidden spot where it was found in the beginning. To finish the rental, the user has to get out of the vehicle, close all doors, carry out any aforementioned additional steps required by the car-sharing company, and use the application to lock the vehicle.[20]

An attacker can easily use the key fob to capture the required number of unlocking signals during the rental phase. Since the attacker temporarily owns the vehicle, she might even carry out further tests (e.g., checking which `RollBack`-variant works and how many signals are required accordingly). Once she returns the vehicle, the rental process officially ends. During that period, the attacker took care of the vehicle well and no harm was done. Later, other users will use the car. An attacker, most of the time, does not even need any effort (e.g., physically following the car, installing a GPS tracker) to keep track of the vehicle. The car-sharing service gives all the necessary information to the attacker. In particular, in point-A-to-A car-sharing, in which each vehicle has a single dedicated lot it has to be returned to in order to finish its rental, the given vehicle's status and booking schedules are usually available upfront. In the case of point-A-to-B car-sharing, i.e., in which vehicles can be picked up and returned to different places, individual booking schedules might not be available. However, information required for a seamless booking experience (e.g., license plate numbers of nearby vehicles, only showing currently available vehicles) is available through the application. This means that attackers can easily implement crawling scripts to obtain the necessary location information about the target vehicle.

Utilizing such information, the attacker can significantly reduce suspiciousness by waiting for the vehicle to be booked (and used) by several other users. Once there is a time slot when the vehicle is available, the attacker can launch `RollBack` to access and steal the vehicle (since the key fob is inside the car). Note that since car-sharing companies usually install GPS trackers to keep track of their fleet, stealing the vehicle might be less appealing or requires more effort (e.g., GPS signal jamming). Yet, using the same availability information, the attacker can check when a particular vehicle will be booked in the future. Then, she can approach the vehicle before the scheduled booking starts, wait for the victim to rent the vehicle, and follow the victim until the vehicle is temporarily left, i.e., when it is locked but not returned, for instance, during shopping. The attacker can then use `RollBack` to unlock the vehicle and steal the belongings left behind.

While one can quickly come up with countless different ways how and when to exploit `RollBack` and what an attacker might do afterward, due to the simplicity and little effort needed, `RollBack` is particularly alarming for car-sharing (and classic car-renting) companies, as attackers can do much harm to the rental companies' user bases, thus, eventually to the companies' reputations.

7 RESPONSIBLE DISCLOSURE PROCESS

In this section, we describe our responsible disclosure process, particularly, how we started, what obstacles we bumped into, and eventually, what take-aways we had.

It was not immediately clear to us at the outset whom we should contact with respect to our initial findings. That is, after finding one car make and model vulnerable, should we contact the car manufacturer, e.g., Hyundai, straight away? They would probably ask first: which specific *vehicles* are vulnerable? Are they the newest models or older ones? Is there any other model from the same make that was found vulnerable? Are all Hyundai vehicles vulnerable? We would have not been able to answer many of these questions due to our limited experiment.

Therefore, we kept experimenting with different vehicles we could have access to until we reached a certain point when 2 to 3 RKE systems using different key fob transponder chips from the same key fob vendor were found vulnerable, irrespective of the vehicle itself.

[20]Some advanced car-sharing companies have already gone completely keyless, i.e., there is no key in the vehicle at all, and even temporarily locking the vehicle in a parking lot (without returning the car) is done through the mobile app.

This led us to two key fob manufacturers: NXP and Omron (see Table 2). While Omron did not have a specific website for reporting vulnerabilities, we have tried to reach out to them through their contact forms found on their international[21] and local[22] (i.e., Singapore) sites. We did not receive any response from Omron. NXP, on the other hand, takes vulnerability disclosure processes very seriously. Vulnerabilities can be reported to their PSIRT (Product Security Incident Response Team), for which all necessary information is provided on their website.[23]

We had a virtual session with NXP in March 2022. We concluded that the vulnerability that we found is indeed a vulnerability and there is no such feature that exactly works the same way as `RollBack`. However, the vulnerability is in the receiver side of the RKE system, which manages the rolling codes, and verifies the validity of each code received; the key fob only sends the signals expected by the vehicle.

On the other hand, it is somewhat known that vendors producing key fobs *only produce* the transponders, and car manufacturers obtain the receiving parts from other OEMs. Accordingly, it is very likely that vehicles using key fobs from other vendors might have the same type of vulnerability due to the supply chain for the receiving units. The key fob manufacturers are (likely) not responsible for the receiving unit, which seems to be the component vulnerable to `RollBack`.

NXP then kindly assisted us to reach out to the affected car manufacturers via the Automotive Information Sharing and Analysis Center (Auto-ISAC[24]). Auto-ISAC is a United States–based industry-driven community that shares and analyzes intelligence about emerging cybersecurity risks to the vehicle and collectively enhances vehicle cybersecurity capabilities across the global automotive industry. The Auto-ISAC members comprise the majority of car and OEM manufacturers worldwide. From our engagement with the relevant car manufacturers, we ended up having two main take-aways from the disclosure process. First, the Auto-ISAC members acknowledged the vulnerability as well as our intention to present our findings (with or without limitations on the context) at Black Hat USA 2022. Second, since our attack targets one specific vehicle (not a fleet of vehicles in general) and has to be redone from scratch for other vehicles (even from the same make/model), it might not be particularly alarming for the car manufacturers.[25] Roughly speaking, there is not much difference between breaking the windows/lock-picking the doors of the target vehicle to steal belongings and doing a more sophisticated and unnoticeable attack such as `RollBack` to achieve the same result. Both approaches always need to pick the target, find the right timing, and carry out the attack. Furthermore, `RollBack` on its own does not allow an attacker to steal the vehicle itself.

We found that through the recently revealed vulnerability (Rolling-PWN [21]), the reaction of Honda [41] has somewhat underpinned our above-mentioned conclusions drawn.

8 TOWARDS FINDING THE ROOT CAUSE

According to the normal operation (discussed in Section 2.2.1 and Section 2.2.2), since the counter value C_k of the key fob signals replayed by `RollBack` is smaller than C_v, they should be discarded. Thus, when we first discovered this vulnerability, we immediately thought that the phenomenon belongs to some sort of key fob re-synchronization, which is required when a new transmitter (i.e., a key fob) is learned to the receiver (i.e., the vehicle's RKE system) or when the battery is replaced

[21] https://bit.ly/3yXGElG
[22] https://bit.ly/3ooVr42
[23] https://bit.ly/3BeMLF1
[24] Their website can be found at https://automotiveisac.com/
[25] Note that this conclusion is utterly our opinion on the subject and it does not reflect any statements from any car manufacturers.

in the key fob and it might lose its last counter values.[26] *However, currently, we cannot confirm the root cause of this vulnerability* for several reasons. First, datasheets with an explanation on how the system architecture works (including the described learning process) are only available for Microchip offerings [25, 40]. Therefore, we discuss the key fob learning process in Microchip KeeLoq systems in detail and point out the critical steps that are not (completely) in line with the operation of `RollBack`.

In the KeeLoq system [25], the typical learning process is as follows (see Figure 4). After entering into the learning mode, when a button on the new key fob is pressed, the first signal is sent to the vehicle. The signal has an unencrypted part containing the key fob's serial number and an encrypted part containing the rest of the data, e.g., rolling code counter, discrimination bits, and button pressed.[27] Using the master key added during manufacturing, the receiver in the vehicle generates the correct encryption/decryption key[28] for the key fob using its serial number emitted unencrypted in the first signal. Then, after decrypting the packet using the freshly generated key, the receiver authenticates the signal. Briefly, authentication involves validating the correct key use via the discrimination bits and buffering the counter value $C_k = n$. Then, the receiver waits for the second signal, i.e., for the second button press on the key fob. When the second signal is received (and authenticated), the receiver checks whether the transmission is indeed the second one, i.e., whether the second counter $C_k = n + 1$. The receiver stores the key fob's serial number, current synchronization counter, and appropriate decryption key upon successful completion of this process. Finally, the system exits from the learning mode. After this point, whenever the freshly added key fob is used in the future, this decryption key is retrieved from the memory along with the stored synchronization counter.

Clearly, the operation of the above-mentioned learning process mimics the operation of `RollBack`. However, there are *five* key observations we have to consider, as they are not elaborated sufficiently and they might undermine such a claim accordingly.

8.1 Learn Mode

Observe that the learning sequence starts with the step `Enter Learn Mode`. Depending on the make, model, and build-year, different vehicles implement different yet intricate approaches to put the receiver in the car into learn mode. In other words, to avoid accidentally entering into learn mode, the vehicle (i.e., the RKE system) requires a very uncommon sequence of actions that would not be carried out during normal use. For instance, some Toyota vehicles require the key to be turned in the ignition from OFF to ON and repeat within 5 seconds [29].[29] However, `RollBack` does not require entering into this mode explicitly.

On the other hand, upon a successful learning process, the system should exit from this mode by default (see `Exit` step in Figure 4). This means that the vehicles found vulnerable to `RollBack` (see details in Section 4) are either always in a learn mode (i.e., do not exit) or do not have this initial step at all, i.e., synchronizing a new key fob to the vehicle is oversimplified.

8.2 Time Frame

As discussed in Section 3.3, some RKE implementations require the captured signals to be replayed within a certain time frame (e.g., $\texttt{RollBack}_N^{\texttt{Strict}}(2)$), while others have no such requirement. This

[26] Note that one can easily find a third-party tutorial (video) on how to learn a new key fob to a certain vehicle make and model. However, these tutorials neither reveal which manufacturer's RKE system they configure nor why the learning process works in that way.

[27] For more details about the basic packet formats, refer to [25].

[28] The KeeLoq algorithm uses a symmetrical block cipher; hence the encryption and decryption keys are identical.

[29] One can easily find several tutorial videos online on how to learn a new key fob to a vehicle.

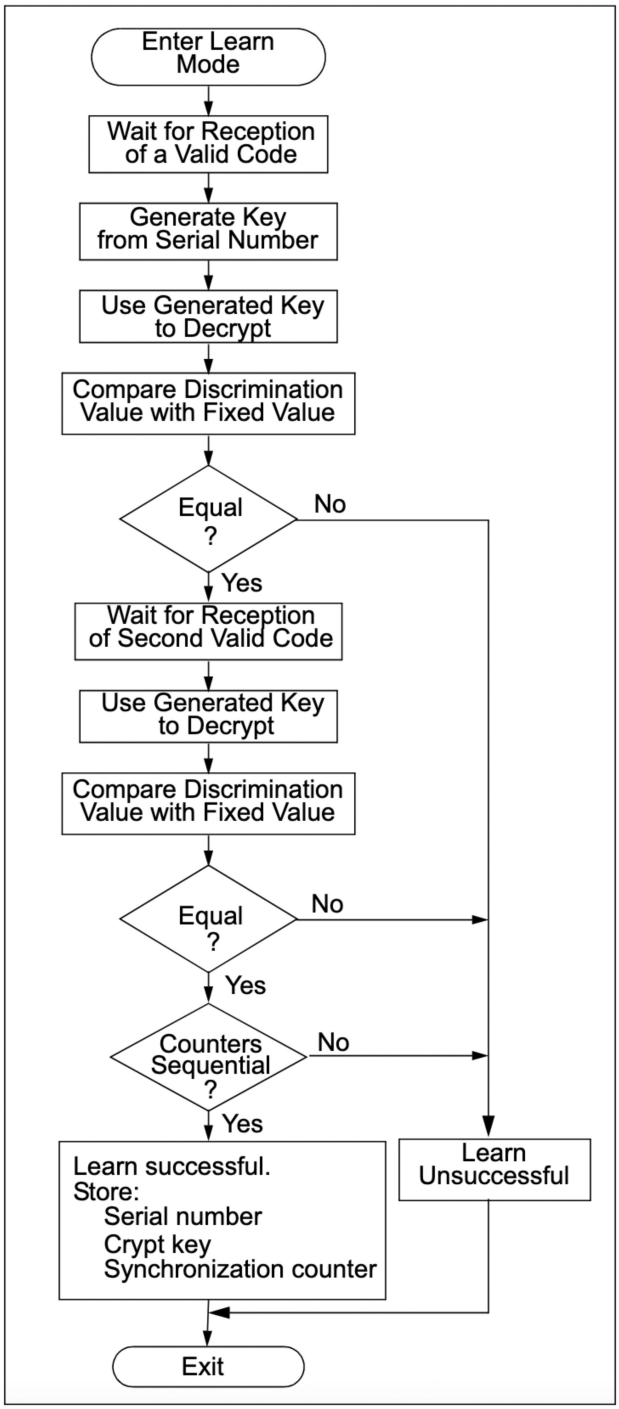


Fig. 4. Typical learning sequence in KeeLoq HCS200/HCS300 RKE systems [25].

property is not defined in the available documentation, e.g., in [25, 40]. However, even [25] claims that the method describes a typical implementation; real-world deployments might be altered to fit other needs.

8.3 Number of Signals and Their Sequence

While the learning process requires the key fob to be pressed two times in a sequence, several RollBack variants we derived work differently. For instance, $\texttt{RollBack}^{\texttt{Loose}}_{\otimes}(2)$ does not require strictly consecutive signals, whereas other variants, e.g., $\texttt{RollBack}^{\texttt{Strict}}_{\otimes}(5)$, need more than two signals. Recall that the learning process described in Figure 4 applies to Microchip's solutions; however, the previously mentioned RollBack variants work against other RKE manufacturers (see details in Section 4).

8.4 Vehicle's Reaction

Another missing piece from the puzzle is to describe which *(i)* actual button (and its instructed action) should be pressed, and *(ii)* whether the same button has to be pressed for the second time. However, since only the key fob's serial number and the discrimination bits matter during the learning process, pressing two different buttons and sending two different signals *(i) to (ii)* accordingly should have no impact on the learning process. Put differently, sending a lock signal and an unlock signal should be sufficient to learn a new key fob to the vehicle.

Nevertheless, at the end of the learning process (see Learn Successful in Figure 4), there is no indication of whether the vehicle should react to the second button press with the intended action (e.g., lock the doors if lock button was pressed). However, in the case of RollBack, the intended action in the last signal (e.g., unlock) is always materialized.

8.5 Relearning the Same Old Key Fob

There is no information available about what happens if an already learned key fob (e.g., the original key fob) is being re-added to the system. One of the vital steps in the learning process is to save the serial number of the key fob and the accompanying crypt key in memory. Thus, the vehicle can have this information straight away from memory in the future, when the new key fob is used. During the learning process, however, there is no step involved in checking whether the serial number of the key fob is already known (before adding it to the memory). Due to this missing check of the key fob's serial number as well as the lack of indication how the vehicle should react (cf. Section 8.4), it is unclear whether re-adding an already known key fob is silently ignored (i.e., leaving the system still in learning mode waiting for a new key fob to be added) or re-added as new.

8.6 Out-of-sync Counters

Finally, observe that during the learning process, the counters of the key fob are buffered for the first signal and only stored upon success. However, the counter's value C_k is not checked (against the counter at the vehicle C_v. This, on the other hand, is somewhat expected. Normally, a new key fob cannot be in sync with the vehicle, hence, the learning process. Furthermore, synchronizing the new key fob's counters to the counters of the actual key fob we use every day would make no sense at all. The different key fobs are always going to be out of sync due to using one of them at a time; hence, the vehicle's receiver stores a separate synchronization counter for all key fobs learned. This can be the reason why consecutive but out-of-sync old counters are always accepted without further validations.

While the learning process is the only action we identified in the RKE system that somewhat mimics the operation of RollBack, according to our arguments above, we cannot state with confidence whether RollBack indeed exploits this feature. Nevertheless, if the found exploit is in the

learning process, then the vulnerable vehicles are *probably* unintentionally left in a *"forever" learn mode* (Section 8.1), which allows re-adding an already learned key fob (Section 8.5) by simply replaying old consecutive signals (Section 8.6), and the vehicle will react accordingly (Section 8.4).

9 MITIGATION

To identify and propose proper mitigation strategies or patches, the root cause of the vulnerability must be identified first. However, as mentioned in Section 8, for the time being, we were not able to pinpoint the root cause with confidence. Accordingly, in this section, we devise different types of mitigation strategies: general advice for an owner to be vigilant and avoid being the target of RKE attacks mostly relying on jamming (e.g., RollJam) in the case of astute attackers (see Section 5) and advice for car-sharing/rental scenarios.

9.1 General Advices

Since `RollBack`, just like other replay-based attack techniques (e.g., RollJam [18]), can utilize jamming to speed up the whole process, a user should be vigilant for the possibility of exposure to signal jamming. The most important thing is always to be close enough to the vehicle to avoid poor signal reception. Thus, if the first button press was not realized by the vehicle (but the second was[30]), then there is a high chance of the first signal being jammed (and captured). In such circumstances, the owner may press the lock and unlock buttons interchangeably until *(i)* both of the two last button presses were correctly received, and *(ii)* the vehicle acts as intended. If only *(i)* holds, the owner might still be exposed to continuous attacks such as RollJam, which jams the latest signal and replays a previously captured one. However, with *(ii)*, the owner can definitely rule out the possibility of such attacks taking place.

Additionally, advanced rolling code implementations having precise timestamps besides the counters (e.g., in `Ultimate KeeLoq` [40]) avoid any practical replay attacks because of the time difference between the vehicle and the key fob's signal.

Note that `RollBack` does not require jamming at all. Accordingly, since in essence it works as a passive listener during the reconnaissance phase (see Section 3.1.1), there is no way to realize whether one is a victim of `RollBack`.

9.2 The Problem of Instruction Agnosticism

While having one rolling code per each learned key fob simplifies the design and reduces the resource requirements, implementing different rolling codes for each instruction will easily evade the problem discussed in Section 5. In particular, by replaying lock signals and hence re-synchronizing its counters, only the further yet invalid *lock* signals would work. On the other hand, the rolling codes of the unlock instructions would remain intact, still preventing the replay of a single unlock signal to open the vehicle (after re-synchronizing the lock instruction's counter). This would significantly reduce the easiness of `RollBack`, requiring signal jamming in almost all cases. As mentioned in Section 9.1, once signal jamming is taking place, a vigilant user can identify it.

9.3 Car-Sharing Scenarios

Car-sharing companies require additional ECUs to enable their users to unlock and lock their vehicles using the mobile application. There are several options to implement such behavior, e.g., using Internet and API calls and mobile SMS. However, most of the time, that function works independently of the other ECUs in the vehicle. This means that even if the vehicle is locked through this ECU (i.e., via the mobile app), the original RKE system can still be used to unlock the

[30]This can also justify that the battery has sufficient charge in the key fob.

vehicle; hence, it is still vulnerable to RollBack. Therefore, for car-sharing companies, it would be worth "connecting" this additional mobile app–related ECU to the rest of the system and enabling the RKE system only if the vehicle is unlocked through the app (and disabling otherwise). However, this only protects the vehicle after it is returned. When someone renting the vehicle temporarily leaves it in a parking lot adequately locked via the key fob but the rental is still ongoing, RollBack can still be launched.

9.4 Using Timestamps as a Countermeasure

Similar to RollJam, RollBack is a special case of replay attack. One possible solution to prevent such attacks is to make use of the current time, i.e., actual timestamps, in the signal sent from the key fob to the receiver in the car. The authors of [32] proposed an authentication protocol based on the timestamp and asymmetric cryptographic techniques. There are two phases in the proposed protocol: *setup* and *authentication*. The setup phase is executed only once, in the beginning before starting to use the key fob. In the setup phase, a private–public key pair and a seed value are generated at the key fob. The public key and the seed are shared with the receiver in the car. Whenever the user presses the key fob button to unlock the car, the authentication phase takes place. In this phase, a random value is generated from the seed and it is appended to the timestamp. Then, the key fob signs the resultant string with its private key. The signature and the instruction (e.g., unlock) are sent to the car. Since the car receiver has the same parameters (i.e., public key and seed), it can verify the received signature. If the signature verification fails, the instruction will not be executed. When an attacker replays the message (in RollJam or RollBack), the timestamp in the replayed message will be different from the actual timestamp in the receiver, making the signature verification fail. Hence, the attacker's attempt fails. Note that for such a timestamp-based solution to work, the clocks on the key fob and the car receiver must be synchronized. However, time synchronization–related matters (e.g., clock skews) are out of scope of [32].

10 CONCLUSION

Remote Keyless Entry (RKE) systems have been the target of attackers for a long time. Attacks such as jamming, tampering, and replaying captured key fob signals have been quite common. Thus, since the late 1990s, deployments have implemented rolling code technology that, by invalidating all previous codes every time a button is pressed on the key fob, renders the attackers' job much more difficult. However, in 2015, RollJam was proven to break, in general, all rolling code–based systems. By carefully jamming, capturing, and replaying key fob signals, RollJam can always be one step ahead of the original key fob, letting an attacker unlock any vehicle. However, if the owner uses the key fob without the RollJam device being in operation (which requires careful placement to hidden spots on the vehicle, continuous control, etc.), the next (unlock) code the attacker possesses becomes invalidated thanks to the rolling codes.

Here, we developed RollBack, a new time-agnostic replay-and-resynchronize attack against most current RKE systems. We showed that even though the one-time code becomes invalid in rolling code systems, replaying a few previously captured signals consecutively can trigger a rollback-like mechanism in the RKE system. RollBack is instruction agnostic, meaning that any captured signals (irrespective of belonging to an unlock or lock instruction) can trigger the same behavior. Therefore, in a typical use case, RollBack does not require signal jamming at all. Furthermore, it is time agnostic; signals have to be captured only once and can be replayed any time in the future as many times as desired.

We derived *five* different variants of RollBack with regard to the required number of signals to be captured, sequence, and time frame of the replay. Our ongoing analysis revealed that ∼ 40% of all vehicles tested are vulnerable to a variant of RollBack, while vehicles manufactured in

Asia tend to be more prone to this vulnerability. We also crowd-source a database of the (non-)vulnerable vehicles; anyone can contribute to it by filling out the form available at https://tinyurl.com/2p99vd7c.

As a countermeasure, we proposed general advice for the vehicle owners on how they could possibly avoid all types of signal jamming-based RKE attacks in different scenarios, including car-sharing use cases that are the most attractive targets for `RollBack`. However, since `RollBack` does not necessitate jamming and the root cause of the vulnerability is yet to be identified, adequate countermeasures and patches could not be rolled out easily for the time being.

ACKNOWLEDGMENTS

The authors would like to thank Xu Jia for his comments.

REFERENCES

[1] Wim Aerts, Eli Biham, Dieter De Moitié, Elke De Mulder, Orr Dunkelman, Sebastiaan Indesteege, Nathan Keller, Bart Preneel, Guy A. E. Vandenbosch, and Ingrid Verbauwhede. 2012. A practical attack on KeeLoq. *J. Cryptol.* 25, 1 (Jan. 2012), 136–157. https://doi.org/10.1007/s00145-010-9091-9

[2] Thomas Barrabi. 2022. Honda key fob hack could leave all vehicle models since 2012 vulnerable: reports. New York Post [Online], https://bit.ly/3b4364x Accessed: July 2022.

[3] Andrey Bogdanov. 2007. Attacks on the KeeLoq Block Cipher and Authentication Systems. (2007), 13 pages.

[4] Bosch. 2022. Electronic Power Steering (EPS). https://bit.ly/2ZJNI7k Accessed: July 2022.

[5] Levente Csikor, Hoon Wei Lim, Jun Wen Wong, Soundarya Ramesh, Rohini Poolat Parameswarath, and Chan Mun Choon. 2022. RollBack — A New Time-Agnostic Replay Attack Against the Automotive Remote Keyless Entry Systems. Presentation at BlackHat, https://tinyurl.com/522sm5mw

[6] CSS Electronics. 2021. OBD2 Explained — A Simple Intro (2021). Online, https://bit.ly/3pZeyn6 Accessed: July 2022.

[7] Thomas Eisenbarth, Timo Kasper, Amir Moradi, Christof Paar, Mahmoud Salmasizadeh, and M. T. Manzuri. 2008. Physical cryptanalysis of KeeLog code hopping applications. *IACR Cryptology ePrint Archive* 2008 (01 2008), 58.

[8] Thomas Eisenbarth, Timo Kasper, Amir Moradi, Christof Paar, Mahmoud Salmasizadeh, and Mohammad T. Manzuri Shalmani. 2008. On the power of power analysis in the real world: A complete break of the Keeloq code hopping scheme. In *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008 (Lecture Notes in Computer Science)*, Vol. 5157. Springer, Berlin, 203–220. https://doi.org/10.1007/978-3-540-85174-5_12

[9] Embitel. 2017. Electronic Control Unit is at the Core of All Automotive Innovations: Know How the Story Unfolded. Blog post, https://bit.ly/3DNRCei Accessed: July 2022.

[10] Aurélien Francillon, Boris Danev, and Srdjan Capkun. 2010. Relay attacks on passive keyless entry and start systems in modern cars. *IACR Cryptology ePrint Archive* 2010 (01 2010), 332.

[11] Scotts Gadgets. n.d. HackRF One. Online, https://bit.ly/3DCNmy6 Accessed: July 2022.

[12] Flavio D. Garcia, David Oswald, Timo Kasper, and Pierre Pavlidès. 2016. Lock it and still lose it — on the (in)security of automotive remote keyless entry systems. In *25th USENIX Security Symposium (USENIX Security 16)*. USENIX Association, Austin, TX, 929–944. https://bit.ly/3pwZKvV

[13] Dan Goodin. 2015. Meet RollJam, the $30 device that jimmies car and garage doors. Blog post, https://bit.ly/2YKvmD8 Accessed: July 2022.

[14] A. Greenberg. 2015. Hackers Remotely Kill a Jeep on the Highway—With Me in It. WIRED article, https://bit.ly/3AJLhjn Accessed: July 2022.

[15] A. Greenberg. 2015. This Hacker's Tiny Device Unlocks Cars and Opens Garages. WIRED article, https://bit.ly/3EedD6d Accessed: July 2022.

[16] Lozier Herbert. 1964. 90 firsts in American automotive history. In *Popular Science*. Bonnier Corporation, 81–83. https://bit.ly/3BIxee4

[17] Jmaxxz. 2019. You Car is My Car. Presentation at DEFCON 27, https://bit.ly/3peIzPu

[18] S. Kamkar. 2015. Drive It Like You Hacked It: New Attacks and Tools to Wirelessly Steal Cars. Presentation at DEFCON 23, https://bit.ly/3j0NZKc

[19] Kunal Karnik, Manandeep, Saurabh Kale, and Ajinkya Medhekar. 2020. On vehicular security for RKE and cryptographic algorithms: A survey. *International Journal of Engineering Research and Technology* 9 (2020), 911–915.

[20] Markus Kasper, Timo Kasper, Amir Moradi, and Christof Paar. 2009. Breaking KeeLoq in a flash: On extracting keys at lightning speed. In *Proceedings of the 2nd International Conference on Cryptology in Africa: Progress in Cryptology (AFRICACRYPT '09)*. Springer-Verlag, Berlin, 403–420. https://doi.org/10.1007/978-3-642-02384-2_25

[21] Kevin2600 and Wesley Li. 2022. Rolling Pwn Attack. [Online], https://bit.ly/3czwTCw Accessed: Jul 2022.
[22] Matt Lake. 2001. How It Works; Remote Keyless Entry: Staying a Step Ahead of Car Thieves. New York Times post, https://nyti.ms/3DLnSyS Accessed: July 2022.
[23] Seungjoon Lee, Kwonyoup Kim, and Seokhie Hong. 2023. Grand Theft House: RF Lock Pick Tool to Unlock Smart Door Lock. Presentation at BlackHat Asia 2023, https://tinyurl.com/48bm6rvc
[24] Kobus Marneweck. 1996. An introduction to KeeLoq™ code hopping. Microchip App Notes, https://bit.ly/3BVV5qs Accessed: July 2022.
[25] Microchip. 2011. KeeLoq™ Code Hopping Encoder. Microchip HCS200. https://bit.ly/3GqCl5c. Accessed: Jul 2022.
[26] Jardine Motors. n.d. The History of Car Technology. https://bit.ly/3lFTHCK Accessed: November 2022.
[27] Sen Nie, Ling Liu, and Yuefeng Du. 2017. Free-fall: Hacking Tesla from Wireless to Can Bus. (2017).
[28] NXP. 2013. Advancing keyless entry/go. NXP solutions brochure. https://bit.ly/3LGJyjB. Accessed: Jul 2022.
[29] Oak Lawn Toyota. n.d. How to Program a Toyota Key Fob. Online, https://bit.ly/3mxmnhH Accessed: July 2022.
[30] David F. Oswald. 2016. Wireless attacks on automotive remote keyless entry systems. In *Proceedings of the 6th International Workshop on Trustworthy Embedded Devices (TrustED '16)*. Association for Computing Machinery, New York, NY, USA, 43–44. https://doi.org/10.1145/2995289.2995297
[31] Pierluigi Paganini. 2022. Experts demonstrate how to unlock several Honda models via Rolling-PWN attack. Security Affairs [Online], https://bit.ly/3RVusdZ. Accessed: July 2022.
[32] Rohini Poolat Parameswarath and Biplab Sikdar. 2022. An authentication mechanism for remote keyless entry systems in cars to prevent replay and RollJam attacks. In *2022 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, Aachen, Germany, 1725–1730. https://doi.org/10.1109/IV51971.2022.9827256
[33] A. Paul, R. Chauhan, R. Srivastava, and M. Baruah. 2016. Advanced Driver Assistance Systems. SAE Technical Paper 2016-28-0223, https://bit.ly/3aJUEUz. Accessed: July 2022.
[34] Johannes Pohl and Andreas Noack. 2018. Universal radio hacker: A suite for analyzing and attacking stateful wireless protocols. In *12th USENIX Workshop on Offensive Technologies (WOOT 18)*. USENIX Association, Baltimore, MD, 14. https://bit.ly/3p56MYG
[35] Robert Potter and Paul Thomas. 2001. Engine ImmobiliIsers: How Effective Are They? https://bit.ly/3aC75l4. Accessed: 2021-08-10.
[36] RTL-SDR.com. n.d. Quick Start Guide. Online, https://bit.ly/3vJu1Zk. Accessed: July 2022.
[37] Brian Santo. 2019. The Consumer Electronics Hall of Fame: LiftMaster Garage Door Opener. IEEE Spectrum, https://bit.ly/3BJ2Z6t. Accessed: July 2022.
[38] Rob Stumpf. 2022. I Tried the Honda Key Fob Hack on My Own Car. It Totally Worked. Security Affairs [Online], https://bit.ly/3Os6dRt. Accessed: July 2022.
[39] Texas Instruments. 2021. CC1101 - Low-Power Sub-1 GHz RF Transceiver. Datasheet, https://bit.ly/3H7dYK7. Accessed: July 2022.
[40] Cristian Toma. 2014. Introduction to Ultimate KeeLoq™ Technology. Microchip App Notes, https://bit.ly/3jjz79W. Accessed July 2022.
[41] Bill Toulas. 2022. Hackers can unlock Honda cars remotely in Rolling-PWN attacks. BleepingComputer News, https://bit.ly/3otJ8U4. Accessed: July 2022.
[42] Jan C. van Ours and Ben Vollaard. 2016. The engine immobiliser: A non-starter for car thieves. *The Economic Journal* 126, 593 (2016), 1264–1291. https://doi.org/10.2139/ssrn.2214895
[43] Roel Verdult, Flavio D. Garcia, and Josep Balasch. 2012. Gone in 360 seconds: Hijacking with Hitag2. In *21st USENIX Security Symposium (USENIX Security 12)*. USENIX Association, Bellevue, WA, 237–252. https://bit.ly/3maFaPO
[44] Roel Verdult, Flavio D. Garcia, and Baris Ege. 2015. Dismantling Megamos Crypto: Wirelessly lockpicking a vehicle immobilizer. In *24th USENIX Security Symposium (USENIX Security 15)*. USENIX Association, Washington, D.C., 703–718. https://bit.ly/3m6Elrb
[45] Jian Wang, Yameng Shao, Yuming Ge, and Rundong Yu. 2019. A survey of vehicle to everything (V2X) testing. *Sensors* 19, 2 (2019), 20. https://doi.org/10.3390/s19020334
[46] YourMechanic. 2016. How Long Does a Key Fob Battery Last? AutoBlog post. https://bit.ly/2T4oSw2. Accessed: July 2022.

Received 23 November 2022; revised 28 July 2023; accepted 6 October 2023