LASER: Lightweight and Secure Remote Keyless Entry Protocol

Vanesa Daza[1,2] and Xavier Salleras[1,2]
[1]*Department of Information and Communication Technologies, Universitat Pompeu Fabra, Barcelona, Spain*
[2]*CYBERCAT - Center for Cybersecurity Research of Catalonia, Spain*

Keywords: Remote Keyless Entry, Wireless Security, Jamming-and-Replay Attack, Relay Attack.

Abstract: Since Remote Keyless Entry (RKE) systems started to be widely used, several vulnerabilities in their protocols have been found. Attacks such as jamming-and-replay attacks and relay attacks are still effective against most recent RKE systems (Ibrahim et al., 2018), even when many secure schemes have been designed. Although they are interesting from a theoretical point of view, the complexity of these solutions is excessive to implement them into a fob (Karani et al., 2016). This paper presents a lightweight and general solution based on a one message protocol, which guarantees the integrity and validity of the authentication in RKE systems, protecting the communication against the well-known jamming-and-replay and relay attacks, without using complex cryptographic schemes. Moreover, we also adapt our protocol for passive RKE (PRKE) systems. Our solution also includes a novel frequency-hopping-based approach which mitigates deny-of-service attacks. Finally, a prototype has been implemented using non-expensive hardware. Obtained results assure scalability, effectiveness and robustness.

1 INTRODUCTION

The usage of RKE systems has been increasing over the years, being them widely used to remotely lock and unlock cars, garage doors, sensors, doorbells or alarms. The first RKE systems used a simple protocol, where a code was sent in plaintext to a receiver which had to execute a command, let us say, unlock a door. However, as sniffing and replaying the code was enough to be able to unlock such a door, a new scheme called *rolling codes* was developed, and it is still widely used nowadays. Such scheme pretends to be secure so the key fob computes and sends a new code each time it is used, and each code is accepted by the receiver just once. Even so, it has been proved that rolling codes are vulnerable to different attacks, and authorities are starting to report[1] criminals taking profit of these vulnerabilities. This fact has led researchers to design new secure schemes (Lv and Xu, 2012) to protect these systems, but their complexity made manufacturers not to implement them, so it would mean to develop key fobs with some disadvantages, i.e. a higher price or a faster draining of the battery. This is due to the fact that many solutions proposed to use cryptographic schemes (Ni et al., 2007) which needed higher computing power than the available in the current fobs. Furthermore, some solutions (Glocker et al., 2017) usually need more than one message to exchange some private information or instruction command, which increases the complexity of the protocol.

Contributions. We provide a secure protocol[2] to be implemented by manufacturers into both RKE and PRKE systems with the only requirement of having a real-time clock, synchronized periodically as detailed in our protocol. Our scheme is robust against both jamming-and-replay attacks and relay attacks; furthermore, it mitigates the effectiveness of jamming-based deny-of-service attacks, thanks to the integration into the protocol of a frequency-hopping approach. Moreover, our solution is a one message protocol for RKE systems and a two messages protocol for PRKE systems, where both approaches are proved to be lightweight. We also demonstrate how our solution can be implemented achieving good results. More details about the presented work can be found in (Daza and Salleras, 2019).

Roadmap. In Section 2 we explain both RKE and PRKE systems along with the common attacks that

[1]https://www.west-midlands.police.uk/news/watch-police-release-footage-relay-crime

[2]The presented solution has been submitted as an invention to be patented with European Patent application number 19382339.0, on May 6th, 2019.

Daza, V. and Salleras, X.
LASER: Lightweight and Secure Remote Keyless Entry Protocol.
DOI: 10.5220/0007830103720377
In *Proceedings of the 16th International Joint Conference on e-Business and Telecommunications (ICETE 2019)*, pages 372-377
ISBN: 978-989-758-378-0

can be performed against them. In Section 3 the state-of-the-art is presented. In Section 4 we explain our solution. The implementation of the proposed solution, along with the experiments and the results derived from them are explained in Section 5. We finally conclude in Section 6.

2 BACKGROUND

Remote Keyless Entry Systems. We call RKE to those systems which are composed of a fob F and a device D. When a button on F is pressed, a radio frequency signal is sent to D, including an instruction command that D will have to execute. These systems are commonly used to lock or unlock cars and open their boots, to open a garage door, to control a temperature sensor, etc. The main protocols used by these systems can be divided as follows:

- *Fixed codes.* This is the simplest scheme. F sends a command *cmd* to D, which is essentially a bit stream referring to an action that D will have to perform.
- *Rolling codes.* Both F and D have previously agreed on a secret key from which derives a sequence of codes $N_1, N_2, \ldots, N_p$. Then, each time a button on F is pressed, the next code c is computed and sent to D, who checks if the received number is equal to a value c that previously it also computed. Apart from c, a command *cmd* is also sent, which is typically a sequence of bits that refers to an action D will have to do, i.e. unlock a car. Each value c can be used only once. In case D may have not received some of the codes sent by F, it commonly checks up to the next 256 generated codes, and when a correct value c is received by D, all the codes behind it cannot be used again. One of the most used rolling codes devices has been KeeLoq (Microchip Technology Inc., 1996).

Passive Remote Keyless Entry. PRKE systems (King, 1998) are a special type of RKE. They do not require the user to manipulate F. Instead, as soon as D receives an external input (i.e. if D is a door, someone pulling the handle), it automatically sends a request to F, which replies with a confirmation. The most used protocol (NXP Semiconductors N.V., 2012) for PRKE systems is the challenge-response protocol, where D sends to F a random value r, the challenge, and waits for a specific response to verify. For instance, F encrypts r using a pre-shared secret key *sk*, and sends the cipher c to D. By means of *sk*, D can decrypt c and verify the identity of F.

Jamming-and-Replay Attack. As depicted in Figure 1, these attacks (Kamkar, 2015) are performed using two transceiver devices. One of them is placed near to D, hidden from the view of the victim V, and jamming the frequency used by the system an attacker A is willing to hack. Then, the other one is close to F, eavesdropping the communications. When V presses the button of F, the signal it sends is jammed by the jamming transceiver J, and V is forced to use an alternative (i.e. a physical key). Meanwhile, A captures the message sent by F, and as D never receives it, A will be able to replay it later. Finally, the jammer can be remotely deactivated by A, as soon as he is sure that V will not try to use F again.

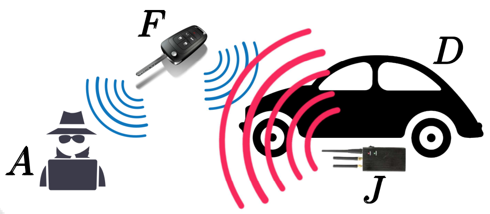

Figure 1: Jamming-and-replay attack.

Relay Attack. These attacks (Francillon et al., 2010) are performed using two transceivers connected through an LTE network or similar, as depicted in Figure 2. One of them is close to D, and the other one to F. Like this, they create a bridge between both endpoints. If the attacked system is a PRKE, when the attacker A_2 pulls the car handle the challenge-response protocol is performed through the bridge created by both attackers. Otherwise, if we are talking about an RKE system, we have to expect that the user may either accidentally press the button on F, or leave it unattended (thus allowing the attacker A_1 to press the button).

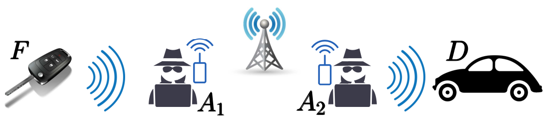

Figure 2: Relay attack.

Deny-of-Service (DoS) Attack. This kind of attack (Thakur and Sankaralingam, 2013) is also based on jamming the frequency used by the protocol, but in this case with the main goal of denying the service. It has a lower impact on the system security as it does not grant access to the system, but it bothers the user, who will require a physical key if he wants to perform the action.

3 RELATED WORK

Regarding the attacks against RKE systems, an important contribution on the topic has been recently done in (Ibrahim et al., 2018). They demonstrate as the jamming-and-relay attacks are nowadays still effective against a wide variety of modern cars, by making use of two units of a radio frequency device called HackRF One[3], one for jamming and the other one for logging data and replaying later. Another important contribution has been done in (Garcia and Oswald, 2016), where an attack allowing an attacker to recover the secret key used in a specific RKE implementation is introduced. As stated in the paper, major manufacturers have sold systems with this vulnerability for over 20 years.

Implementation of the Attacks. By making use of two radio frequency devices called Yardstick One[4] (YS1), a jamming-and-replay attack can be performed by using a python implementation[5] of this attack. This implementation makes use of a library called *rflib*, included in a software used by YS1 called RfCat[6]. That said, one antenna will be jamming while the other will be sniffing the code of the fob. The same implementation is useful for performing just the DoS attack. Moreover, taking this implementation as a starting point, implementing a relay attack is trivial.

Proposed Solutions. Many secure schemes (Lv and Xu, 2012), (Glocker et al., 2017) have been designed to increase the security of RKE and PRKE systems. The main problem they present is their complexity, so they use cryptographic schemes which are hard to implement into cheap key fobs. On the other hand, some schemes (Jeong and So, 2018) have been proved to be both simple and effective against relay attacks. One of them, proposed in (Ranganathan and Capkun, 2018), demonstrates that a protocol calculating the time between message exchanges can determine if a relay attack is being performed against a PRKE or not. This is the main idea behind *LASER*, which also solves the replay vulnerability.

4 OUR SCHEME: LASER

In this section we explain step-by-step our protocol, *LASER*, for both RKE and PRKE systems. We consider a fob F and a generic device D, assuming it to be a car. First, both endpoints have to agree on a randomly generated secret key *sk* large enough to make a brute-force attack hard to accomplish (i.e. a 256-bits key). They also need to agree on a set of commands *cmd*, used for example to lock the car, unlock it, etc. D also has a car identification number ($device_{id}$) known by F.

In both RKE and PRKE systems, both F and D will be required to compute a hash. The hash function used by both devices was required to be lightweight in order to optimize the timings and the resources consumption. For our implementation and analysis we have chosen to use *Blake2*, a hash function proposed in (Aumasson et al., 2013), which guarantees a low power and computing resources consumption. Furthermore, it is proved to be as fast as *MD5*, but solving the security vulnerabilities *MD5* presents. In particular we are interested in using *Blake2s*, a version of *Blake2* optimized for 8-bit platforms, which are the kind of cheap processors commonly used for key fobs. Basing our solution in the usage of a hash function like *Blake2* instead of using some complex cryptographic scheme, we are decreasing the costs of implementing our solution, and also avoiding a fast draining of the battery.

Our solution performs a frequency-hopping protocol where the frequency channel used to transmit the messages changes each period of time p. This means that both D and F must agree on the same channel, and to achieve it they perform the Protocol 4.1.

Protocol 4.1 (Frequency-hopping for LASER). *The frequency-hopping for a specific endpoint, which has a number of available frequency channels N_c, is performed as follows:*

1. *Each period of time p (both F and D have previously agreed on this value) it gets the current datetime in a timestamp form, sums the secret key sk to it and calculates its hash h.*
2. *It calculates the channel ch, which is the modulo N_c of the integer representation of h: $ch \equiv int(h) \pmod{N_c}$.*

Next subsections explain the specific details for either RKE and PRKE systems, and the security analysis of both approaches.

4.1 Protocol Details

LASER Protocol for RKE. In this scheme, D is required to be always listening to a specific channel, so it will be continuously performing the Protocol 4.1. However, F will perform it just before to start the Protocol 4.2. When the owner of D wants to execute a command *cmd* by pressing a button on F, F calculates *ch* by first calculating h, but rounding the timestamp to the previous multiple of p. Then, next protocol is performed (as depicted in Figure 3):

[3]https://greatscottgadgets.com/hackrf/
[4]https://greatscottgadgets.com/yardstickone/
[5]https://github.com/exploitagency/rfcat-rolljam
[6]https://github.com/atlas0fd00m/rfcat

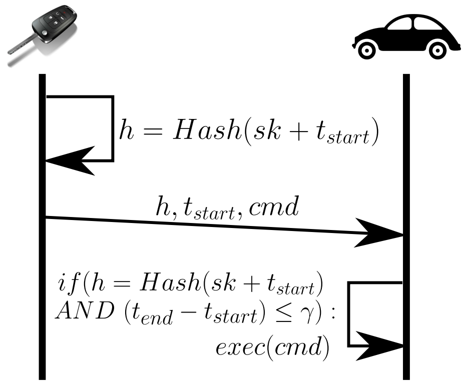
$h = Hash(sk + t_{start})$
h, t_{start}, cmd
$if(h = Hash(sk + t_{start})$
$AND\ (t_{end} - t_{start}) \leq \gamma):$
$exec(cmd)$

Figure 3: LASER for RKE.

Protocol 4.2 (LASER for RKE)**.** *Both D and F follow the next protocol:*

1. *F takes the current timestamp t_{start}, sums it to sk and computes its hash h.*
2. *F sends h over ch along with the real timestamp t_{start} and the command cmd.*
3. *As soon as D receives the message sent in the last step, it gets the current timestamp t_{end}, and checks if the difference between t_{start} and t_{end} is lower than or equal to a threshold γ, previously estimated.*
4. *If the above condition is true, and h is correct, D executes cmd.*

An accurate time synchronization between F and D is crucial, as F has to send an exact timestamp. To overcome this drawback, we propose the usage of the same approach we introduced in our protocol: if F sends a timestamp t_{start} that does not verifies $(t_{end} - t_{start}) \leq \gamma$, D replies with a message h_{sync}, t_{sync}, where t_{sync} is the correct timestamp and $h_{sync} = Hash(sk + t_{sync})$. F updates its real-time clock after verifying h_{sync}. The purpose of sending also a hash here, is to avoid an attacker being able to send messages to F to modify its current time.

LASER Protocol for PRKE. In this scheme, it will be F who is continuously performing the Protocol 4.1. When the owner of D wants to unlock it by pulling the handle, D calculates ch by first calculating h, but rounding the timestamp to the previous multiple of p. Then, next protocol is performed (as depicted in Figure 4):

Protocol 4.3 (LASER for PRKE)**.** *Both D and F follow the next protocol:*

1. *D sends over ch a SYN message to F including the device$_{id}$. At the moment it sends the message, it also starts to calculate a message exchanging time t_e.*

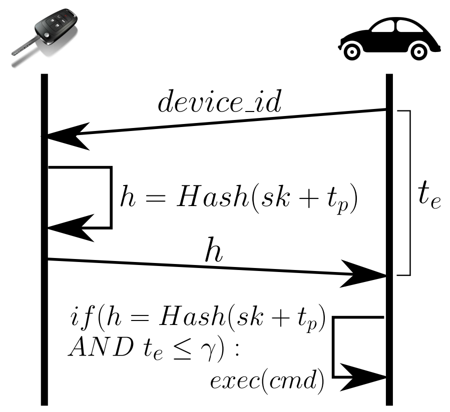

Figure 4: LASER for PRKE.

2. *F computes the hash value of sk plus t_p, and sends the result h to D.*
3. *As soon as D receives the message sent in the last step, it stops the counter of t_e. Like this, now D knows a value t_e which is the time between D sending a message and receiving a response. If the received value h is correct and t_e is lower than or equal to a threshold γ, D executes the desired action.*

In PRKE, if D does not receive a response after sending the first message of the protocol, it can be that t_p on F is incorrect. In this case, D must send h_{sync}, t_{sync} using all the other frequencies, to be able to reach the one used by F, and make it update its current time.

4.2 Security Analysis

Preventing Jamming-and-Replay Attacks. To prevent jamming-and-replay, our solution sends a unique hashed value h of a secret key sk concatenated to a timestamp. Like this, each hash will be unique in time, and will be accepted by the receiver just at that moment. Plus, the fact of concatenating a secret key makes impossible for an attacker A to generate a new hash.

Preventing Relay Attacks. We first need to estimate the threshold γ, which is the maximum amount of time a message should take going from F to D in RKE (in PRKE, is the maximum time it should take for a message to go from D to F, plus the response message to go back to D). In this scenario, if a message took an amount of time $(t_{end} - t_{start})$ higher than γ, we could say that F is placed further from D than what it should be, and that the protocol is performed by means of a relay attack, using an LTE network or similar.

Preventing DoS. Both endpoints have different frequency channels available to perform the frequency-hopping protocol, and they agree on a channel without an attacker being able of knowing it. As they change the transmitting channel each short period of time p (which should be defined by the manufacturer considering the best performance of the device), an attacker willing to perform a DoS against us will have to jam a wide range of frequencies at the same time. It can be done by means of several jamming devices, which is an expensive investment[7].

5 EXPERIMENTS AND RESULTS

Implementation. A prototype[8] of our solution has been developed and tested using Python and RfCat. Our code is composed of a single script, which can be run either for F and D, providing a $device_{id}$ and a sk. The range of frequencies used by the code to perform the frequency-hopping protocol can be provided by the user, where the available range depends on the antenna used, in this case YS1. When we press the key corresponding to the command we want to execute, the protocol keeps being performed while the user retains the key pressed, meaning this that messages are sent continuously till the key is released. Even pressing and releasing the key quickly, our tests demonstrate that around 6 messages are sent on average (in both RKE and PRKE implementations). This has been done on purpose, like is done in regular RKE and PRKE systems, to avoid having to press more than once if the receiver is not able to catch the message the first time, due to random hardware errors.

Estimating the Threshold. For each system RKE and PRKE we have tried to execute a command one thousand times. The success rate has been 100% in both cases, meaning this that the command has been always executed. By logging the timestamps into a dataset, we have found out that the time it takes for a message to go from an endpoint to the other one is never higher than $t_{max} = 136$ ms for the RKE solution, as shown in Table 1. For PRKE systems, where the calculated time is how much it takes D to receive F's reply, the maximum time it took has been $t_{max} = 175$ ms. Choosing this maximum value as the threshold could be dangerous if a relay attack is performed: for the RKE, if the message takes the minimum time $t_{min} = 55$ ms to go to the attacker A_1, and the second attacker A_2 gets to send the relayed message in the same amount of time, it would take 110 ms. Assuming that the attackers will not be able to exchange the relayed message in less than $t_{max} - 110 = 26$ ms is a weak premise. Plus, taking the average amount of time we are compromising the usability of the system, so most of the time the user will have to press the button more than once, as shown in Figure 5. To overcome this, we can use the third quartile of the dataset, which is higher than the average in both RKE and PRKE systems. We can see in Figure 5 that now the effectivity is higher as well. As every time we press the button in the fob we are sending around 6 messages, the probability of failing when trying to execute a command is almost negligible, so the success rate for each message is almost 75%.

Table 1: Information extracted from timestamps of RKE and PRKE systems, expressed in milliseconds.

System	t_{max}	t_{min}	t_{avg}	t_{Q3}
RKE	136	55	71	79
PRKE	175	113	157	164

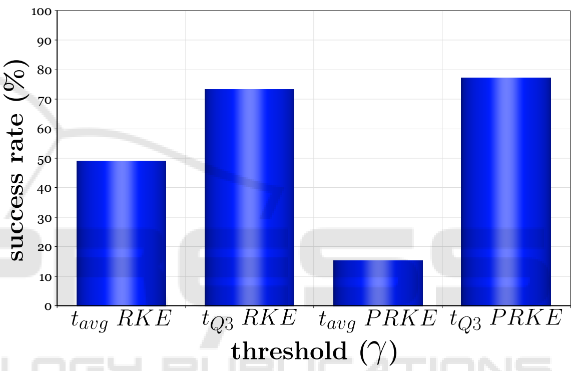

Figure 5: Success rate when trying to execute a command in both RKE and PRKE systems considering different thresholds.

Robustness Against Relay Attacks. Let us have an RKE relay attack scenario as depicted in Figure 6. If the minimum time it can ever take for the user's hardware to send a message from F to D is t_{min}, we can be sure that $t_{FA_1} = t_{min}$ is the minimum value that can be achieved. As such, our scheme is secure as far as the attackers are not able to achieve the following statement:

$$\begin{aligned} t_{FA_1} + t_{A_1A_2} + t_{A_2D} &\le \gamma \\ (t_{A_1A_2} + t_{A_2D}) &\le \gamma - t_{FA_1} \end{aligned} \tag{1}$$

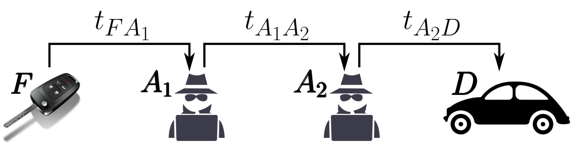

Figure 6: RKE relay attack scenario.

On the other hand, we have a PRKE relay attack scenario as depicted in Figure 7. If the minimum time

[7]https://www.jammer-store.com/hpj16-all-frequencies-jammer.html
[8]https://github.com/xevisalle/laser

it can ever take for the user's hardware to send a message from D to F and send the answer back to D is t_{min}, we can be sure that $(t_{DA_2} + t_{FA_1}) = t_{min}$ is the minimum value that can be achieved. As such, our scheme is secure as far as the attackers are not able to achieve the following statement:

$$\begin{gathered} t_{DA_2} + t_{A_2A_1} + t_{A_1F} + t_{FA_1} + t_{A_1A_2} + t_{A_2D} \leq \gamma \\ (t_{A_2A_1} + t_{A_1F} + t_{A_1A_2} + t_{A_2D}) \leq \gamma - (t_{DA_2} + t_{FA_1}) \end{gathered} \tag{2}$$

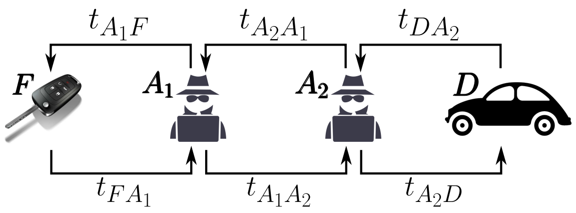

Figure 7: PRKE relay attack scenario.

As detailed in Section 2, the bridge between A_1 and A_2 can be done through an LTE network or similar. Knowing that the average uplink latency in LTE networks is 10.5 *ms* (Amjad et al., 2018), we could assume two attackers getting lower values for $t_{A_1A_2}$ and $t_{A_2A_1}$. Even so, assuming that a relay attack can be successful against *LASER* is a strong premise.

6 CONCLUSION

In this paper we have introduced *LASER*, a lightweight and secure scheme for both RKE and PRKE systems. *LASER* solves the security issues present into these systems, completely avoiding jamming-and-replay and relay attacks without using complex cryptographic schemes. Furthermore, it mitigates DoS attacks thanks to a simple frequency-hopping protocol. *LASER* is easy-to-implement and we demonstrated it by implementing a prototype using non-expensive hardware. Last but not least, we proved the effectiveness and robustness of our solution through different experiments we performed.

ACKNOWLEDGEMENTS

This work was supported by Project RTI2018-102112-B-I00 and by the European Research Council under the H2020 Framework Programme/ERC grant agreement 694974.

REFERENCES

Amjad, Z., Sikora, A., Lauffenburger, J.-P., and Hilt, B. (2018). Latency reduction in narrowband 4g lte networks. In *2018 15th International Symposium on Wireless Communication Systems (ISWCS)*.

Aumasson, J. P., Neves, S., Wilcox-O'Hearn, Z., and Winnerlein, C. (2013). BLAKE2: simpler, smaller, fast as MD5. https://blake2.net/.

Daza, V. and Salleras, X. (2019). LASER: Lightweight And SEcure Remote keyless entry protocol (Extended version). arXiv:1905.05694 [cs.CR].

Francillon, A., Danev, B., and Capkun, S. (2010). Relay attacks on passive keyless entry and start systems in modern cars. https://eprint.iacr.org/2010/332.

Garcia, F. D. and Oswald, D. (2016). Lock It and Still Lose It-On the (In)Security of Automotive Remote Keyless Entry Systems. *25th USENIX Security Symposium (USENIX Security 16)*.

Glocker, T., Mantere, T., and Elmusrati, M. (2017). A protocol for a secure remote keyless entry system applicable in vehicles using symmetric-key cryptography. In *2017 8th International Conference on Information and Communication Systems (ICICS)*, pages 310–315.

Ibrahim, O. A., Hussain, A. M., Oligeri, G., and Di Pietro, R. (2018). Key is in the air: Hacking remote keyless entry systems. In *Proc. of the International Workshop on Cyber Security for Intelligent Transportation Systems (CSITS2018)*.

Jeong, H. and So, J. (2018). Channel correlation-based relay attack avoidance in vehicle keyless-entry systems. *Electronics Letters*, 54(6):395–397.

Kamkar, S. (2015). Drive it like you hacked it: New attacks and tools to wirelessly steal cars. DEFCON 23.

Karani, R., Dhote, S., Khanduri, N., Srinivasan, A., Sawant, R., Gore, G., and Joshi, J. (2016). Implementation and design issues for using bluetooth low energy in passive keyless entry systems. In *2016 IEEE Annual India Conference (INDICON)*.

King, J. D. (1998). *Passive remote keyless entry system*. US6236333B1.

Lv, X. and Xu, L. (2012). AES encryption algorithm keyless entry system. In *2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*, pages 3090–3093.

Microchip Technology Inc. (1996). *TB003 An Introduction to KEELOQ Code Hopping*.

Ni, X., Shi, W., and Fook, V. F. S. (2007). AES security protocol implementation for automobile remote keyless system. *IEEE 65th Vehicular Technology Conference*.

NXP Semiconductors N.V. (2012). NXP keyless entry/go solutions: Advancing keyless entry/go.

Ranganathan, A. and Capkun, S. (2018). Are we really close? verifying proximity in wireless systems. *IEEE Security Privacy*, pages 1–1.

Thakur, N. and Sankaralingam, A. (2013). Introduction to jamming attacks and prevention techniques using honeypots in wireless networks. IRACST - International Journal of Computer Science and Information Technology and Security (IJCSITS).