# An Authentication Mechanism for Remote Keyless Entry Systems in Cars to Prevent Replay and RollJam Attacks

Rohini Poolat Parameswarath[1] and Biplab Sikdar[2]

***Abstract*— Modern cars come with Keyless Entry Systems that can be either Remote Keyless Entry (RKE) systems or Passive Keyless Entry and Start (PKES) systems. In the initial versions of RKE implementation, fixed code was used by the key fob to unlock the car door. However, this method is vulnerable to replay attacks as an adversary may capture and replay the same code later to unlock the car. A rolling code system was introduced to protect RKE systems from such replay attacks. Studies have shown that even the rolling code system is vulnerable to certain attacks. In this work, we investigate the attacks possible on RKE systems and propose an efficient and effective authentication mechanism to defend RKE systems against such attacks with minimal changes to the existing RKE system. The proposed mechanism makes use of hashing and asymmetric cryptographic techniques for the secure transmission of signals from the key fob to the car that cannot be replayed. The security of the proposed mechanism is shown using informal security proof and simulation of the proposed solution is also provided.**

## I. Introduction

With the advancement of technology, modern cars are equipped with keyless entry systems to lock and unlock cars which eliminates the inconvenience of using physical keys [1]. Keyless systems consist of a key fob with the user and a transceiver at the car. There are two types of keyless systems: Remote Keyless Entry (RKE) systems and Passive Keyless Entry and Start (PKES) systems. In the RKE system, the car door is unlocked or locked when the corresponding button on the key fob is pressed. In the PKES system, the user does not have to press the button on the key fob. When the key fob is close to the car and the user presses a button on the car door, the door opens. In this work, we specifically look at the RKE system.

RKE systems are based on unidirectional Radio Frequency (RF) transmission from the key fob to the car [2]. When a button on the key fob is pressed, RF signals in the 315 MHz, 433 MHz, or 868 MHz bands (depending on the geographic location) are generated. They are received by a receiver in the car and based on the command received, a lock or unlock operation is carried out.

The initial versions of RKE systems used a fixed, static code to unlock the car. This technique was highly susceptible to attacks as an adversary could capture the signal transmitted from the key fob to the car and replay it. To prevent this type of replay attacks, rolling codes were introduced. The rolling code is generated from a counter value. When the unlock button on the key fob is pressed, the counter value increases and a new rolling code is generated which can be used only once to unlock the car door [2]. Microchip Technology's KeeLoq and NXP's Hitag-2 rolling code schemes are widely used in RKE systems. The RollJam attack by Samy Kamkar [3] is a special type of replay attack against rolling code-based RKE systems. Through this attack, it has been shown that even a system based on rolling codes is vulnerable to replay attacks by using special types of equipment.

These attacks can be perpetrated with relatively inexpensive hardware and software. Hence, developing a preventive technique against such attacks is an important task. In this work, we propose an authentication mechanism based on hashing and asymmetric cryptography to prevent attacks on RKE systems.

### A. Related Work

Researchers have investigated the vulnerabilities in keyless systems and have come up with multiple solutions for attack detection and defence. The authors of [4] discussed attack surfaces in cars including remote keys and developed a security-by-design framework. The authors of [5] surveyed attacks on connected and autonomous vehicles and the defence techniques available. Remote automotive attack surfaces such as RKE and Bluetooth for different car models were surveyed in [6]. In [7], the authors classified vulnerabilities and discussed the defence methods existing in connected and autonomous vehicles.

In [2], the authors studied vulnerabilities that exist in rolling code-based RKE systems and demonstrated attacks. The authors of [8] discussed the Hitag-2 algorithm-based RKE and showed that keys equivalent to the legitimate key can be extracted from Hitag-2.

While there have been studies on existing vulnerabilities and devising new attacks, researchers have been working on solutions as well. The RF-fingerprinting method was proposed to differentiate legitimate and malicious requests in [1]. The authors of [9] proposed a timestamp-based solution to defend against replay attacks in RKE systems. A solution using time stamping and XOR encoding to protect RKE systems from replay attacks was given in [10]. A protocol for RKE using a symmetric encryption algorithm was proposed in [11]. Ultimate KeeLoq technology [12] is a solution based on a running timer that protects against replay attacks. We propose an authentication mechanism

[1]Rohini Poolat Parameswarath is with Department of Electrical and Computer Engineering, College of Design and Engineering, National University of Singapore rohini.p@nus.edu.sg

[2]Biplab Sikdar is with Department of Electrical and Computer Engineering, College of Design and Engineering, National University of Singapore bsikdar@nus.edu.sg

with minimal changes to the existing RKE system and make use of the current time directly in the proposed solution.

### B. Our Contribution

Even today there are cars in the market that can be easily unlocked by capturing and replaying the signals from the key fob to the car. To counter this vulnerability, we propose a robust mechanism that prevents replay and RollJam attacks. The proposed solution is based on hashing and asymmetric cryptographic techniques for transmitting signals from the key fob to the car. Our contribution can be summarised as follows:

**1. Design of a novel authentication mechanism for the RKE system:** We propose a novel authentication mechanism that enables the car to authenticate a legitimate key fob. The proposed solution involves minimal changes to the existing RKE system. It is built on unidirectional communication between the key fob and the car.

**2. Protection from several attacks:** The proposed mechanism protects RKE systems from replay and RollJam attacks. In addition, the mechanism also provides protection from impersonation attacks where an attacker generates messages to get authenticated to unlock the car.

**3. Informal security proof:** We also provide informal security proof to demonstrate the security of the proposed mechanism.

**4. Simulation of the proposed scheme:** We simulate the proposed scheme using Python to demonstrate its effectiveness.

The rest of the paper is organized as follows. In Section II, we discuss the system model and the preliminaries. In Section III, we present the proposed mechanism. Section IV presents the security analysis of the proposed scheme. Then, we discuss the simulation experiments, results and provide a performance comparison with other existing RKE solutions. Finally, conclusions are given in Section V.

## II. System Model and Preliminaries

In this section, we discuss the system model, the adversary model, and the basics of asymmetric cryptography.

### A. System Model

A key fob is used to lock or unlock the car. There are at least two buttons on the key fob that trigger the lock and unlock operation of the car, respectively. In the RKE system, the RF transmission from the key fob to the car is unidirectional. The system model is illustrated in Figure 1.

### B. Adversary Model

We now describe how an adversary, who has intercepted the signal between the key fob and the car, will be able to use it later to unlock the car. For RKE systems that use static code, if an adversary can intercept the signal once by using specialised hardware and customised software, he/she can replay it and unlock the car anytime. Certain car models have implemented rolling codes that change every time a user presses the key fob. Through the RollJam attack, Samy Kamkar [3] has shown that even the RKE systems based on rolling codes are vulnerable to attacks. When the user presses the key fob, the RollJam device jams the signal so that it will not reach the car while the attacker captures and stores it. Since the first attempt to unlock the car failed, the user attempts to unlock the car again. During the second unlock attempt by the user, the attacker jams the signal and captures it again. At the same time, the attacker sends the signal which was captured during the first unlock attempt to the car. As a result, the attacker has the latest code which can be used to unlock the car.

Fig. 1: Car and Key fob

### C. Asymmetric Cryptography

For sending the signal from the key fob to the car in a secure way, we make use of asymmetric cryptographic techniques. We use the Rivest–Shamir–Adleman (RSA) algorithm for the proposed authentication mechanism. Asymmetric cryptography is based on two related keys: a private key and a public key. Encryption and signing can be achieved with asymmetric cryptography. For encryption, a message is encrypted with the receiver's public key and is sent to the receiver. Upon receiving the message, the receiver decrypts the message with his/her private key. In the signature mechanism, the message is signed with the private key of the sender. Upon receiving the message, the car receiver verifies the signature using the sender's public key.

## III. Proposed Mechanism

We propose a robust authentication mechanism based on hashing and asymmetric cryptographic techniques. The proposed mechanism consists of two phases: setup and authentication.

### A. Assumptions

The assumptions made in this paper are given below:

• The message transmission in the setup phase between the key fob and the car is through a secure channel.

• The clocks of the key fob and the car receiver are synchronized. This assumption is the same as in [9].

• The key fob does not share the private key of the key pair with any other party.

### B. Setup Phase

In the setup phase, the key fob generates a private key and public key pair through RSA key generation. The steps for RSA key generation are given below [13]:

From two random, large prime numbers $p$ and $q$, $N$ is calculated as

$$N = p \times q. \tag{1}$$

The Totient function $\phi(N)$ is given by

$$\phi(N) = \phi(p)\phi(q) = (p-1)(q-1). \tag{2}$$

Choose a random, large integer $d$ such that $gcd(d, \phi(N)) = 1$ where *gcd* is the greatest common divisor. Then, we find $e$, the multiplicative inverse of $d$, such that

$$e \times d \equiv 1 \mod \phi(N). \tag{3}$$

Public key $K_{pu}$ consists of $N$ and $e$, and the private key $K_{pr}$ consists of $N$ and $d$. A seed $S_i$ is also generated. A message $M_1 : \{K_{pu}, S_i\}$ consisting of the public key and the seed is sent to the car. The receiver at the car stores these values. The setup phase must be run at the start of the usage of the key fob. The setup operation can be implemented by providing one more button on the key fob or by pressing an existing button on the key fob consecutively a fixed number of times. The setup phase is depicted in Figure 2.

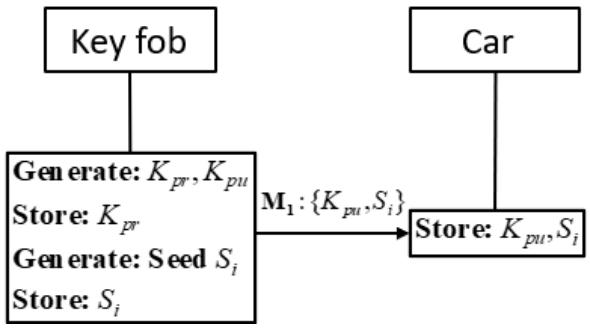


Fig. 2: Setup phase.

### C. Authentication Phase

When the key fob button is pressed to unlock the car, a random number is generated from the seed:

$$R_i = RAND(S_i). \tag{4}$$

The generated random number, date, and current time are concatenated and its hash value *H* is calculated. This will be used as the authentication parameter that the car will use to authenticate the key fob:

$$H = h(R_i \parallel CurrentDate \parallel CurrentTime). \tag{5}$$

The hash value is signed with the key fob's private key using the RSA algorithm to generate the signature:

$$\sigma = H^d \, mod(N). \tag{6}$$

Subsequently, the seed value is incremented by 1.

$$S_i = S_i + 1. \tag{7}$$

A message $M_2 : \{\sigma, cmd\}$ consisting of the signature $\sigma$ and a command value *cmd* indicating lock or unlock operation is sent to the car. Upon receiving the signature from the key fob, it is verified at the receiver with the key fob's public key using RSA algorithm. $H$ is computed from the signature as

$$H = \sigma^e \, mod(N). \tag{8}$$

The receiver generates a random number from its current seed value $S_j$:

$$R_j = RAND(S_j). \tag{9}$$

After that, the receiver calculates the authentication parameter which is the hash value of the generated random number, date, and current time:

$$H^{'} = h(R_j \parallel CurrentDate \parallel CurrentTime). \tag{10}$$

$H$ and $H'$ are compared. If they are equal, the command is executed resulting in the lock or unlock operation. If they are not equal, the seed value is incremented and steps from (9) are repeated. This is to take into consideration the scenario where the user has pressed the key fob accidentally before coming near the car and the seed value has incremented in the key fob. In such a situation, the seed value in the key fob would be ahead of that in the receiver. Hence the receiver will check a fixed number (e.g., 100) of seed values ahead of its current seed value and do the comparison. If any of the corresponding $H'$ is equal to $H$, the command *cmd* is executed. It should be noted that if the receiver checks seed values ahead, the value of the current time will change. Hence, it is important to store the current time at the beginning of the authentication phase and use it for the computation.

After successful operation, the seed value is incremented:

$$S_j = S_j + 1. \tag{11}$$

The details of the authentication phase are depicted in Figure 3.

The $CurrentTime$ is used in the calculation of the authentication parameter. There will be a difference in the time when the key fob calculated $H$ and when the receiver calculates $H^{'}$ due to the propagation and transmission time. This difference will be in the order of tens or hundreds of milliseconds. However, the clock resolution used in this solution is in seconds. Hence, differences due to propagation and transmission time will not be an issue. Note that a clock resolution of a second is sufficient to prevent the replay and the RollJam attacks. This is because the attacker sends the captured 'unlock' signal when the user is not in the vicinity of the car. Hence, after capturing the 'unlock' signal sent by the key fob, sending the captured signal to the car immediately is unlikely. In other words, the probability of capturing a signal and replaying it within one second is negligible.

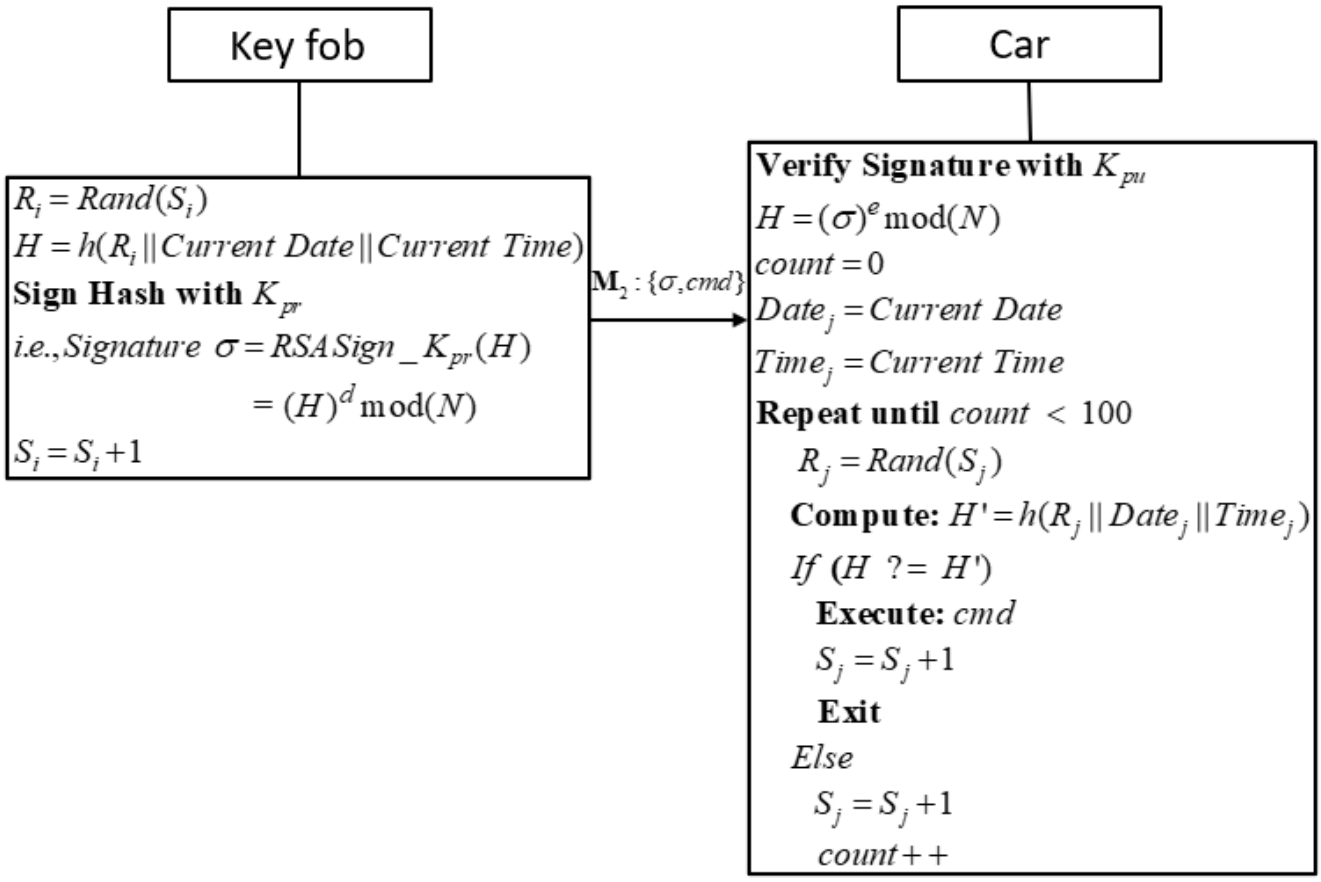


Fig. 3: Authentication phase.

## IV. DISCUSSION

This section presents the informal security analysis, simulation experiments, and the performance evaluation of the proposed authentication mechanism.

### A. Informal Security Analysis

1) **Key fob Authentication:** A legitimate key fob knows the valid parameter $R_i$ and the private key $K_{pr}$ to construct a message $M_2 : \{\sigma, cmd\}$ that the receiver validates and uses to authenticate the key fob. Thus, the proposed scheme provides authentication.

2) **Protection Against Replay Attacks:** The authentication mechanism is designed in such a way that a random number $R_i$, the current date, and the current time are used in constructing the authentication parameter $H = h(R_i \parallel CurrentDate \parallel CurrentTime)$. $R_i$ and $CurrentTime$ will change during each session. This will prevent the scenario where an adversary captures and replays the previously sent message $M_2$ to get authenticated. Thus, the proposed scheme ensures protection against replay attacks.

3) **Protection against RollJam Attacks:** Current time is used in constructing the authentication parameter. This will prevent the RollJam attack where an attacker jams two consecutive signals and replays the first message to get authenticated. When an attacker tries to execute the RollJam attack, since the current time in $M_2$ is different from the current time in the receiver, the receiver will reject $M_2$. Hence, the proposed scheme is resilient against the RollJam attacks.

4) **Protection Against Impersonation Attacks:** If an attacker wants to impersonate a legitimate key fob, he/she needs to send $M_2 : \{\sigma, cmd\}$. However, the attacker does not know the random value $R_i$ to construct the authentication parameter $H = h(R_i \parallel CurrentDate \parallel CurrentTime)$ and the private key $K_{pr}$ to sign it. This ensures protection against impersonation attacks.

5) **Signature Property:** The message is signed by the private key $K_{pr}$ of the key fob which ensures that it is created by the owner of the private key, i.e., the key fob.

6) **Non-repudiation:** The key fob signs the authentication parameter *H* with its private key $K_{pr}$. The receiver verifies the signature using the key fob's public key $K_{pu}$. The private key $K_{pr}$ used to sign the authentication parameter *H* is only known to the key fob. This ensures non-repudiation since the key fob cannot deny having signed the authentication parameter.

### B. Simulation Experiments

In this section, we demonstrate the proposed solution through simulation by coding the process described in

TABLE I: Comparison of security features

| Features | Greene et al. [9] | Greene et al. [10] | Glocker et al. [11] | Proposed Method |
|---|---|---|---|---|
| Authentication | Yes | Yes | Yes | Yes |
| Resilience Against Replay Attack | Yes | Yes | Yes | Yes |
| Resilience Against RollJam Attack | Yes | Yes | No | Yes |
| Signature Property | No | No | No | Yes |
| Non-repudiation | No | No | No | Yes |
| Key is not shared | No | No | No | Yes |
| Informal Security Proof | No | No | No | Yes |

Section III. The code was written in Python using RSA from Crypto.PublicKey module. This process was repeated 100 times, which was kept as the maximum threshold in this simulation. If hash values did not match in 100 iterations, the authentication message was rejected. The experimental results are given below.

**Normal Operation**: The authentication message $M_2$ was sent 100 times with 3-second gap between each message. In all the 100 iterations, the received signature was verified to be correct, and authentication was successful.

**Replay Attack**: To simulate replay attack, the first message sent from the key fob was captured by the attacker. Then, the captured message was sent by the attacker to the car at a later point in time. The replay attack was conducted 100 times and the results show that the proposed solution is robust against the replay attack.

**RollJam Attack**: To simulate the RollJam attack, the first message sent by the key fob was captured by the attacker but prevented from reaching the car. The next message that was sent by the key fob was also captured by the attacker and prevented from reaching the car. However, at this stage, the attacker sent the previously recorded message to the car. The RollJam attack was conducted 100 times and the results show that the proposed solution prevents the RollJam attack as well.

### C. Performance Evaluation

In this section, we compare the performance of the proposed scheme with other existing schemes for key fob authentication. We analyze security properties such as authentication, protection against replay and RollJam attacks, etc. to do the performance comparison. The comparison of the security features is summarised in Table I. Though [9], [10] and [11] cover the key security features, signature property and non-repudiation property are some of the additional properties provided by our proposed solution. In [9], [10] and [11], the key to encrypt the message is shared with the receiver whereas, in our proposed mechanism, the private key is not shared with any other party which provides an additional level of security. Hence, the proposed scheme ensures all the important security properties.

Now, we evaluate the proposed scheme in terms of the computation cost. The setup phase occurs only once. Hence, we focus on the computation cost of the authentication phase. Further, the time taken by hashing operation is not significant compared to the time taken by RSA signature generation and verification. Hence, we look at the execution time taken by the RSA operations. On an Intel P4 2.0 GHz machine with 512MB of RAM, RSA signature generation and RSA signature verification both take 0.01 seconds when the key length is 1024 bytes [14]. The number of operations performed and total execution time taken by the key fob and the receiver during authentication is given in Table II and is plotted in Figure 4.

TABLE II: Number of operations and computation time.

| **Operation** | **Key fob** | **Receiver** |
|---|---|---|
| *RSA Signature Generation* | 1 | 0 |
| *RSA Signature Verification* | 0 | 1 |
| **Computation Time (s)** | 0.01 | 0.01 |

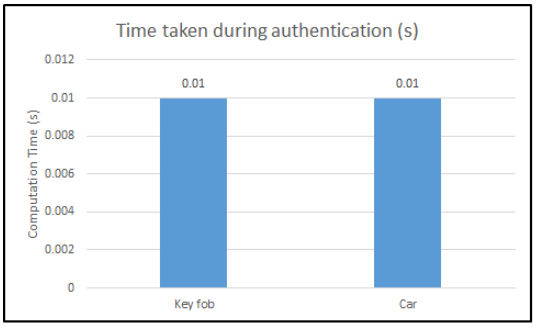


Fig. 4: Time taken during the key fob authentication.

## V. Conclusion

In this paper, we explored the vulnerabilities existing in RKE systems. Then, we proposed a solution that enables authentication of a legitimate key fob which prevents some common attacks. We provided informal security proof for the authentication mechanism. Through simulation, we demonstrated that the proposed authentication mechanism can reliably detect replay and RollJam attacks.

## References


[1] K. Joo, W. Choi, and D. H. Lee, "Hold the door! fingerprinting your car key to prevent keyless entry car theft," in *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society, 2020. [Online]. Available: https://www.ndss-symposium.org/ndss-paper/hold-the-door-fingerprinting-your-car-key-to-prevent-keyless-entry-car-theft/

[2] F. D. Garcia, D. Oswald, T. Kasper, and P. Pavlidès, "Lock it and still lose it—on the (in) security of automotive remote keyless entry systems," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016.

[3] Defcon 23 rolljam attack. [Online]. Available: https://samy.pl/defcon2015/

[4] A. Chattopadhyay, K.-Y. Lam, and Y. Tavva, "Autonomous vehicle: Security by design," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–15, 2020.
[5] M. Pham and K. Xiong, "A survey on security attacks and defense techniques for connected and autonomous vehicles," *Computers & Security*, p. 102269, 2021.
[6] C. Miller and C. Valasek, "A survey of remote automotive attack surfaces," *black hat USA*, vol. 2014, p. 94, 2014.
[7] X. Sun, F. R. Yu, and P. Zhang, "A survey on cyber-security of connected and autonomous vehicles (cavs)," *IEEE Transactions on Intelligent Transportation Systems*, 2021.
[8] R. Benadjila, M. Renard, J. Lopes-Esteves, and C. Kasmi, "One car, two frames: attacks on hitag-2 remote keyless entry systems revisited," in *11th {USENIX} Workshop on Offensive Technologies ({WOOT} 17)*, 2017.
[9] K. Greene, D. Rodgers, H. Dykhuizen, K. McNeil, Q. Niyaz, and K. A. Shamaileh, "Timestamp-based defense mechanism against replay attack in remote keyless entry systems," in *2020 IEEE International Conference on Consumer Electronics (ICCE)*, 2020, pp. 1–4.
[10] K. Greene, D. Rodgers, H. Dykhuizen, Q. Niyaz, K. Al Shamaileh, and V. Devabhaktuni, "A defense mechanism against replay attack in remote keyless entry systems using timestamping and xor logic," *IEEE Consumer Electronics Magazine*, vol. 10, no. 1, pp. 101–108, 2020.
[11] T. Glocker, T. Mantere, and M. Elmusrati, "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)*. IEEE, 2017, pp. 310–315.
[12] Ultimate keeloq. [Online]. Available: http://ww1.microchip.com/downloads/en/AppNotes/00001683A.pdf
[13] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.
[14] N. Jansma and B. Arrendondo, "Performance comparison of elliptic curve and rsa digital signatures," *nicj. net/files*, 2004.