Securing IR communication from replay attacks

Aisha Kothare and Namruth Reddy

1. Abstract

Infrared (IR) communication is a popular wireless communication method that utilizes infrared light to transmit data between devices. This technology is commonly used in a variety of consumer electronics such as remote controls, mobile devices, and security systems. IR communication offers several advantages over other wireless communication methods, such as low power consumption, immunity to radio frequency interference, and ease of implementation. However, it is vulnerable to replay attacks. Replay attacks can occur when an attacker intercepts and records the IR signals transmitted between devices, and then replays them later to gain unauthorized access to the system. We propose and implement 2 methods to secure this communication using challenge-response authentication and time-stamping. Furthermore, we investigate the performance of different security mechanisms, including computational complexity, energy consumption, and communication overhead, and provide a comparison of their advantages and limitations.

1. Introduction

While traditional IR devices only support simple operations, such as changing TV channels, modern devices often allow for the entry of sensitive information, such as credit card numbers or personal information, via an IR remote. The proliferation of IoT-enabled consumer electronics, such as smart TVs, has increased the risk of sensitive information being exposed to eavesdroppers via infrared (IR) communication. This creates a new problem, as these visual media devices can potentially be used to control and monitor other IoT devices in the home, leading to the exposure of personal information to attackers.

Infrared (IR) communication is a low-cost and low-power wireless communication method that utilizes Infrared Emitting Diodes (IREDs) on the transmission side and IR photodiodes on the reception side. The hardware components required for IR communication are less expensive than those required for other wireless communication methods, such as Wi-Fi, Zigbee, and Bluetooth. Additionally, IR communication consumes less power, making it an attractive option for battery-operated devices. Unlike other wireless communication methods, IR communication uses light waves and doesn’t interfere with radio signals. Therefore, the IR channel can be used without regulation, offering greater flexibility and freedom for users.

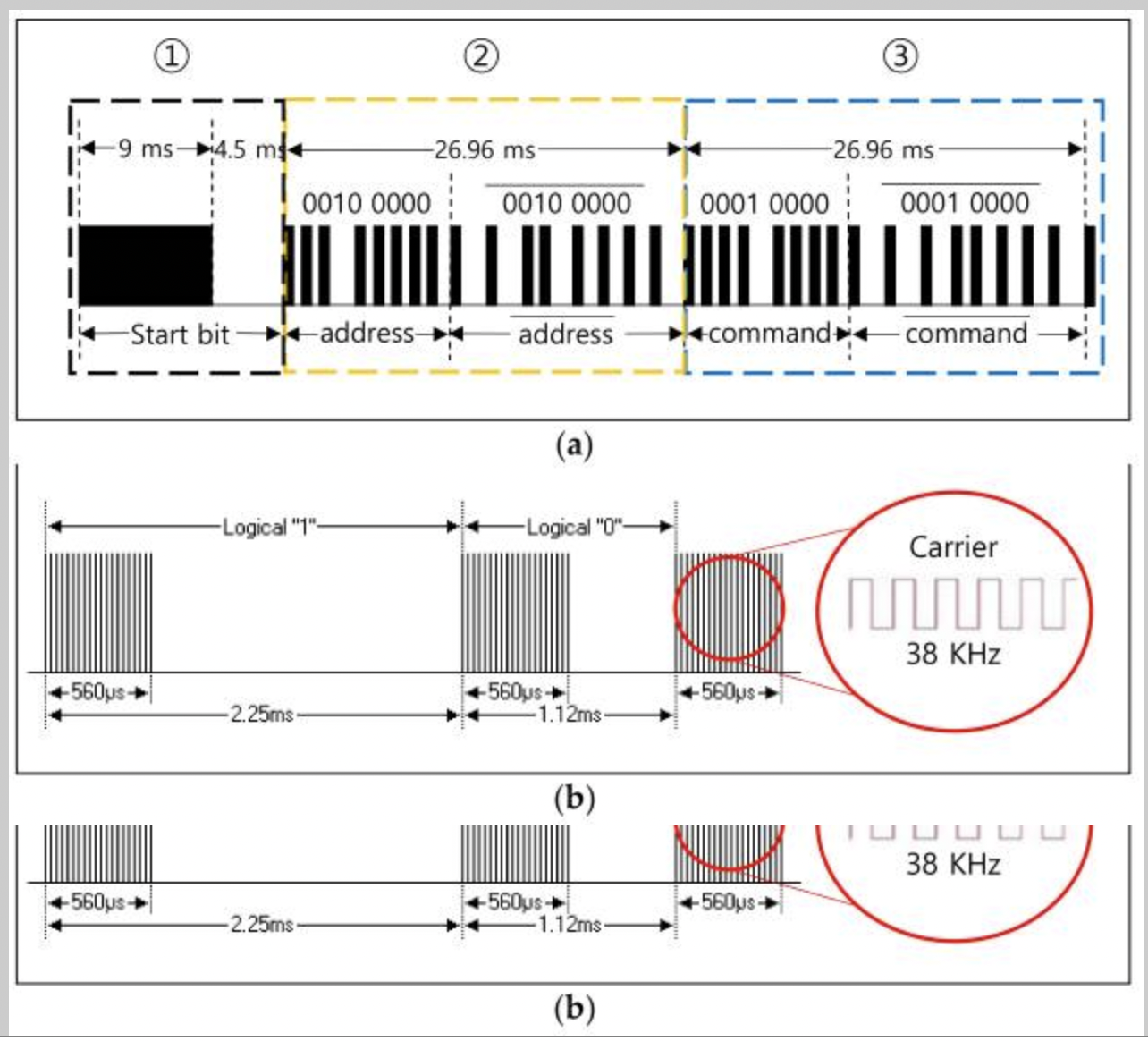
Our work examines the potential risks associated with IR communication with respect to smart home devices and proposes a range of solutions to mitigate these risks, including secure pairing methods, encryption techniques, and time-stamp mechanisms. As a starting point we use relatively simple IR devices to implement these measures, like LED lights, and smart TV. By adopting these measures, users can ensure that their personal information remains protected, even as they take advantage of the convenience and functionality of IoT-enabled devices without hampering the performance or usability of them.

1. Background and Motivation

The process of IR communication between a transmitter and receiver involves the conversion of an electrical signal into a modulated IR signal, which carries the data to be transmitted. The modulated IR signal is transmitted via an IR LED, which emits infrared light that carries the modulated data. On the receiver side, the modulated IR signal is received through an IR receiver module, which detects and demodulates the signal to extract the original data. The receiver device then processes the demodulated data to perform the desired operation, such as turning on a TV or changing the volume of a speaker. IR communication is a line-of-sight communication technology, this implies that the transmitter and receiver must be in direct line of sight with each other without any obstacles. This is because IR light cannot penetrate solid objects such as walls or obstacles, unlike other wireless communication technologies such as Wi-Fi and Bluetooth.

The choice of the protocol used in IR communication depends on several factors such as the type of device being used, the range of communication required, and the data transfer speed required. Protocols such as RC-5 and NEC are ideal for remote controls due to their simplicity and low power consumption. Protocols such as IrDA are designed for high-speed data transfer and are commonly used in mobile devices and computers. The IrDA protocol is designed for short-range wireless data transfer, typically up to 1 meter, and uses a low-power consumption mode to conserve battery life. While ZigBee is designed for longer-range wireless communication and can operate over distances of up to 100 meters. ZigBee uses a mesh network topology, which allows devices to communicate with each other without the need for a centralized control point. This makes it an ideal protocol for use in home automation systems, sensor networks, and industrial control systems. Our work focuses on securing NEC protocol by adding time stamps and thresholds and we use our own IR transmitter and receiver.

NEC protocol:

This Figure shows the frame format of the NEC protocol used for transmitting 32-bits in one frame. The NEC frame consists of a start bit pulse, an 8-bit address, and an 8-bit command, as shown in (a). The start bit is used to identify the beginning of the frame, as the communication is asynchronous. The 8-bit address is used to identify the device, and the 8-bit command carries information about the pressed key on the IR remote. This command and address is transmitted twice with the second occurrence having all bits iinverted and can be used for verification if the entire frame was received. To verify the received information, the inverted address and command are appended to the frame, as depicted in the figure. In case of extended NEC mode, the address bits are doubled, giving 16 bits to allocate for the device address. The extended NEC protocol doesnt reverse the address bits, instead its represented using two bytes (16 bits) - the low byte and the high byte - allowing for a maximum of 65,536 possible addresses. However, each address still handles only 256 different commands, preserving the original command redundancy. 

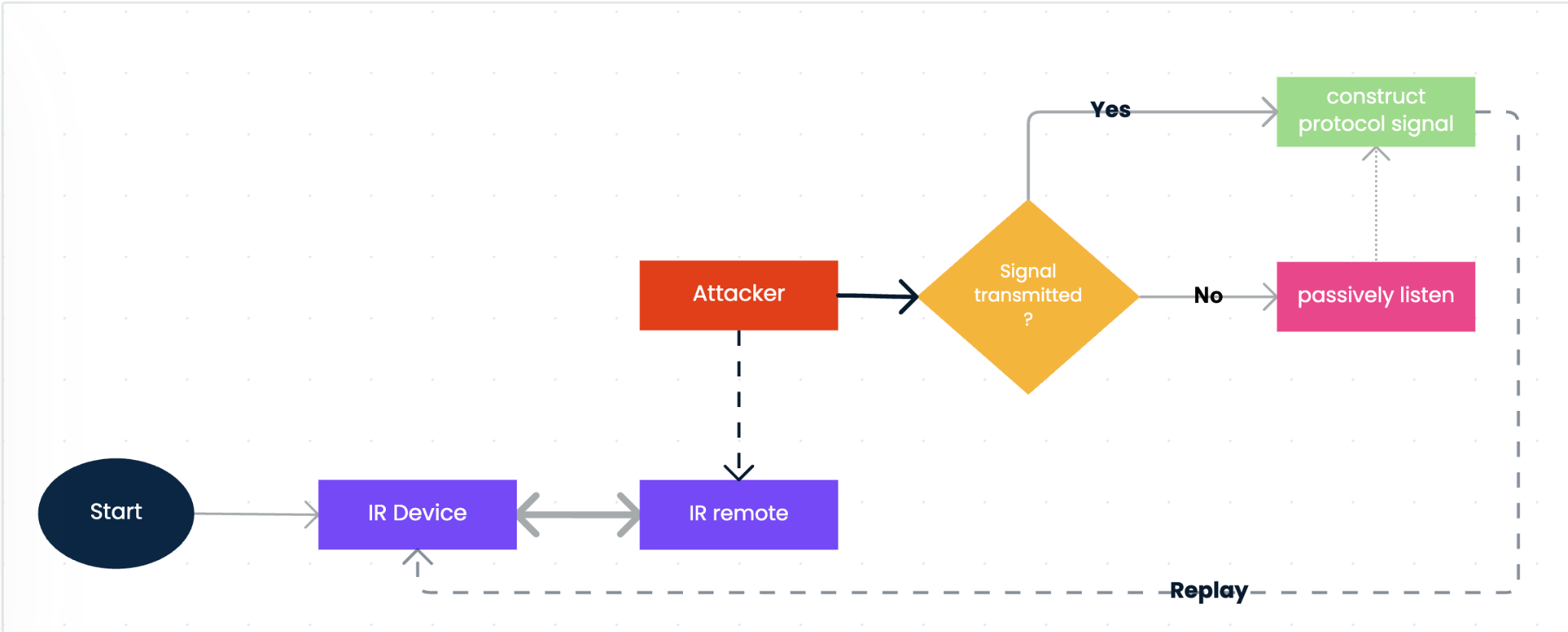
This protocol uses pulse-distance modulation, Figure (b) shows the frame data transmitted using modulation with a 38 KHz carrier. A logical '1' is represented by a 560 μs active-high and a 1690 μs active-low, while a logical '0' is represented by a 560 μs active-high and a 560 μs active-low. These timings enable the receiver to decode the transmitted signal correctly and ensure reliable communication between the transmitting and receiving devices. Overall, this protocol offers a simple and efficient way to transmit data using infrared communication.

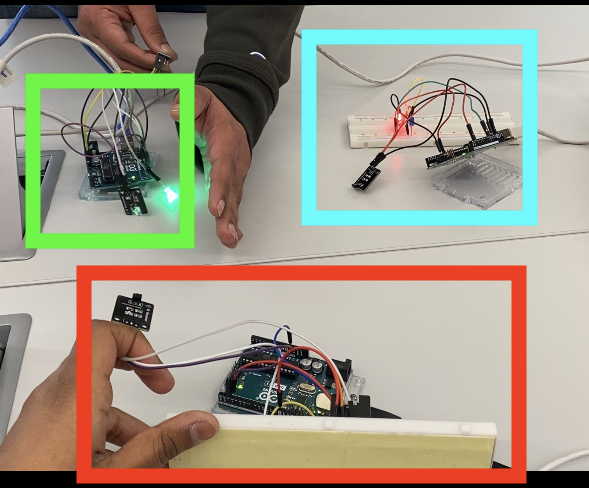
1. Methodology

4.1 Setup

Our setup consists of 3 Arduino Unos and 3 sets of IR transmitters and IR receivers. We use them to simulate an environment containing an IR-transmitting object, an IR receiver, and the attacker. The attacker functions as a separate IR transceiver. The IR transmitter and receiver modules used are Gikfun Digital 38khz Sensor Module Kit for Arduino EK8477. We use LEDs to demonstrate the successful communication exchange, replay attack, and session key transfer. Buttons are used to initiate key transfers [in the case of the session key method] and message exchanges. In order to simulate a real-world environment, the Arduino is connected to a power supply after code is flashed on them, instead of being fed by the computer power source. In the diagram below, green indicates the transmitter setup, blue indicates the receiver and red indicates the attacker.

The flowchart demonstrates the control flow for the entire IR system we set up. The attacker is present passively in the environment to record any IR signals that are being transmitted to the device and replay them consequently. This is done by building a transceiver.

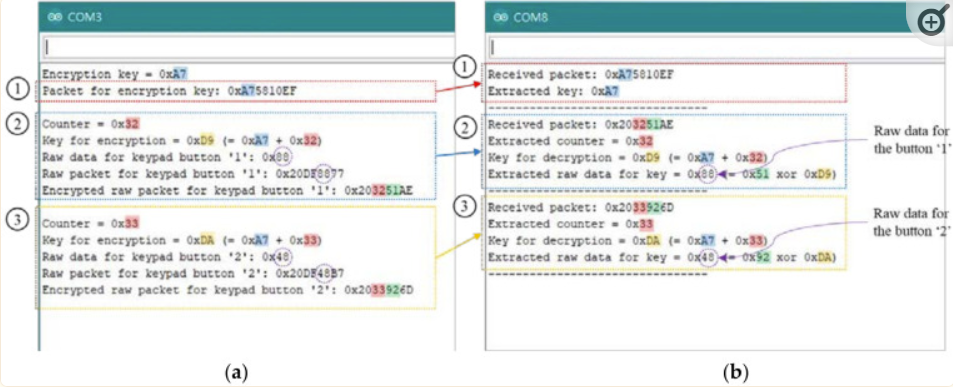


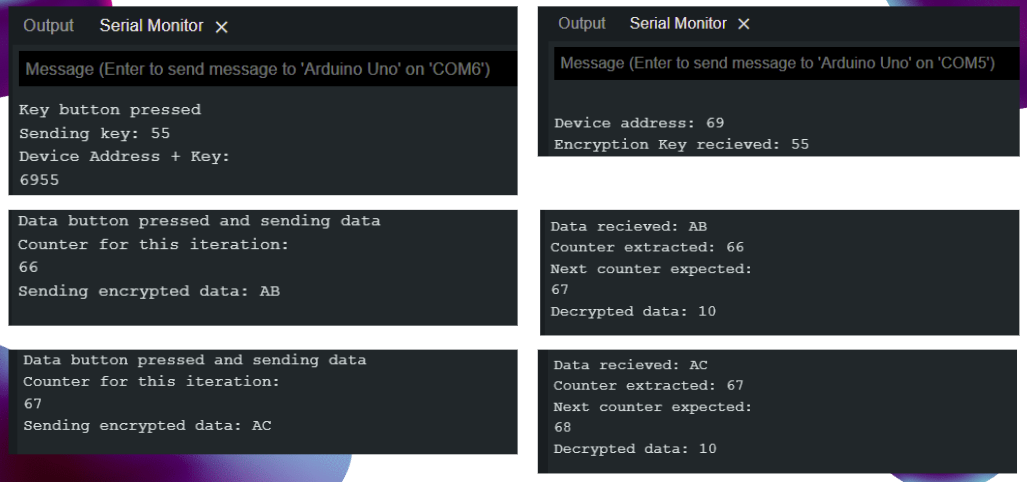


4.2 Secure communication methods:

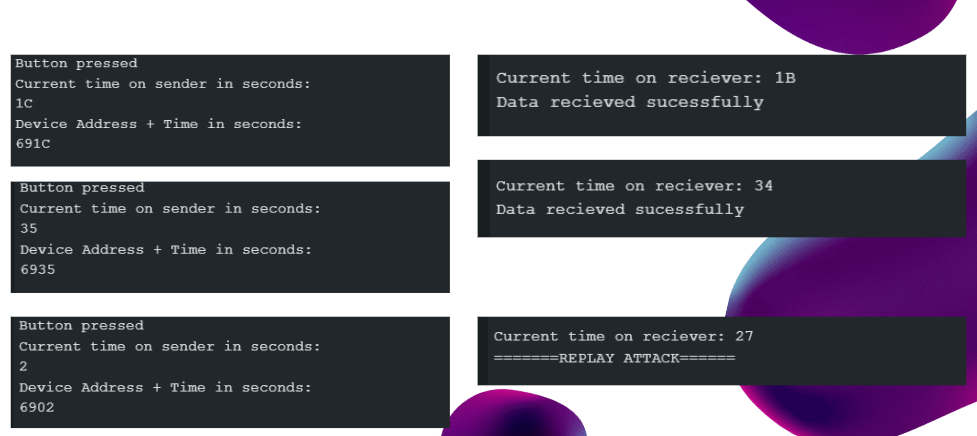
We have implemented two methods that can be used secure IR communication from replay attacks. Both of them will be described below:

**Session Key approach**: This approach involves generating a session key for every new session in the communication between the IR transmitter and receiver. Each session is reset when the receiver is restarted. When a new session starts, the transmitter shares a random session key between 1 and 254 with the receiver. This random key is generated using an 8-bit timer inside the transmitter. We are limiting the key to 8 bits because of the power requirements of the transmitter. Once the transmitter sends the session key, for every communication thereafter, the contents in the packet are encrypted using the session key. To increase security and also prevent replay attacks, we use a pre-determined counter that is added to the session key for each session and then the packets are encrypted and sent across the wireless channel. On the other side of the communication, the receiver decodes the packet using the encryption key it received during the first packet of the session and processes the values. For every next packet, the receiver is expecting a pre-determined counter increment, so if the same packet is sent again by an attacker, that would be invalidated because the expected counter doesn’t match the received counter.





**Time to Live (TTL) approach**: Time to live is another popular approach to solving replay attacks. It is based on the concept that, once a packet is sent by the transmitter, there is a specific time for only which it is valid. This time limit can be set based on the needs of the system and the user. Lower TTL means more accuracy but the same time more false positives and higher TTL means vice-versa. But the basis of the approach involves the transmitter and receiver synchronized beforehand. We are not implementing the synchronization part in this project but once the transmitter and receiver are synchronized, we can use this approach to prevent replay attacks. In this method, the transmitter and receiver are synchornized for up to seconds because of the limitation of the number of bytes that can be transmitted using the NEC protocol. For each packet, the transmitter adds a timestamp in seconds on when the packet was generated, then the packet is sent across the wireless medium to a receiver. The receiver, which is also synchronized with the transmitter then calculates the difference between the current time and the timestamp on the packet. If the difference is more than the pre-agreed TTL, then the packet is discarded else the packet is sent for further processing.



4.3 Strengths and limitations

Each of the methods described above has its own strengths and limitations. A few of the important ones are discussed in this section. The session key approach is easier to implement and doesn’t need any pre-existing synchronization, unlike the TTL method, though easier to implement, assumes that transmitter and receiver can be synchronized somehow before the secure communication can take place. The session key approach also provides encryption capabilities that can be used when critical data is being sent across the wireless medium and eavesdropping attacks along with replay attacks can also be prevented. TTL method doesn’t provide any protection from eavesdropping attacks.

TTL method is not vulnerable to brute force attacks but the session key approach can be when we have an advanced adversary. TTL method needs high-level synchronization to work, in this implementation, synchronization is only done in seconds, advanced adversaries can break this approach if they are able to replay the packets within the given TTL period. Performance impact on communication is negligible when TTL is used since only basic instructions are used in this approach. The session key approach however shows a 10% increase in time taken for the receiver to process the packets because of all the operations involved in calculating expected counters, decrypting data, and more. Advanced attackers can also sniff the session key in the first packet of a session but that can be avoided if keys are hardcoded in the transmitter and receiver when they are manufactured especially for high-security devices.

4.4 Challenges Faced

Some of the challenges faced in implementing the secure communication methods using session keys and TTL are noted below:

* Implementing both methods involved modifying the existing NEC packets to include details such as keys, counter values, TTL values, and more. We had to precisely encode and decode the packets at the sender and receiver. This is because only a limited amount of data can be sent in the NEC protocol.
* Session key approach is slightly complicated since it involves both encryption and integrity check. So coding the algorithm was not a straightforward process.
* TTL method also involves tight synchronization to work and to get the synchronization, we have to power both the transmitter and the receiver Arduino at the same time

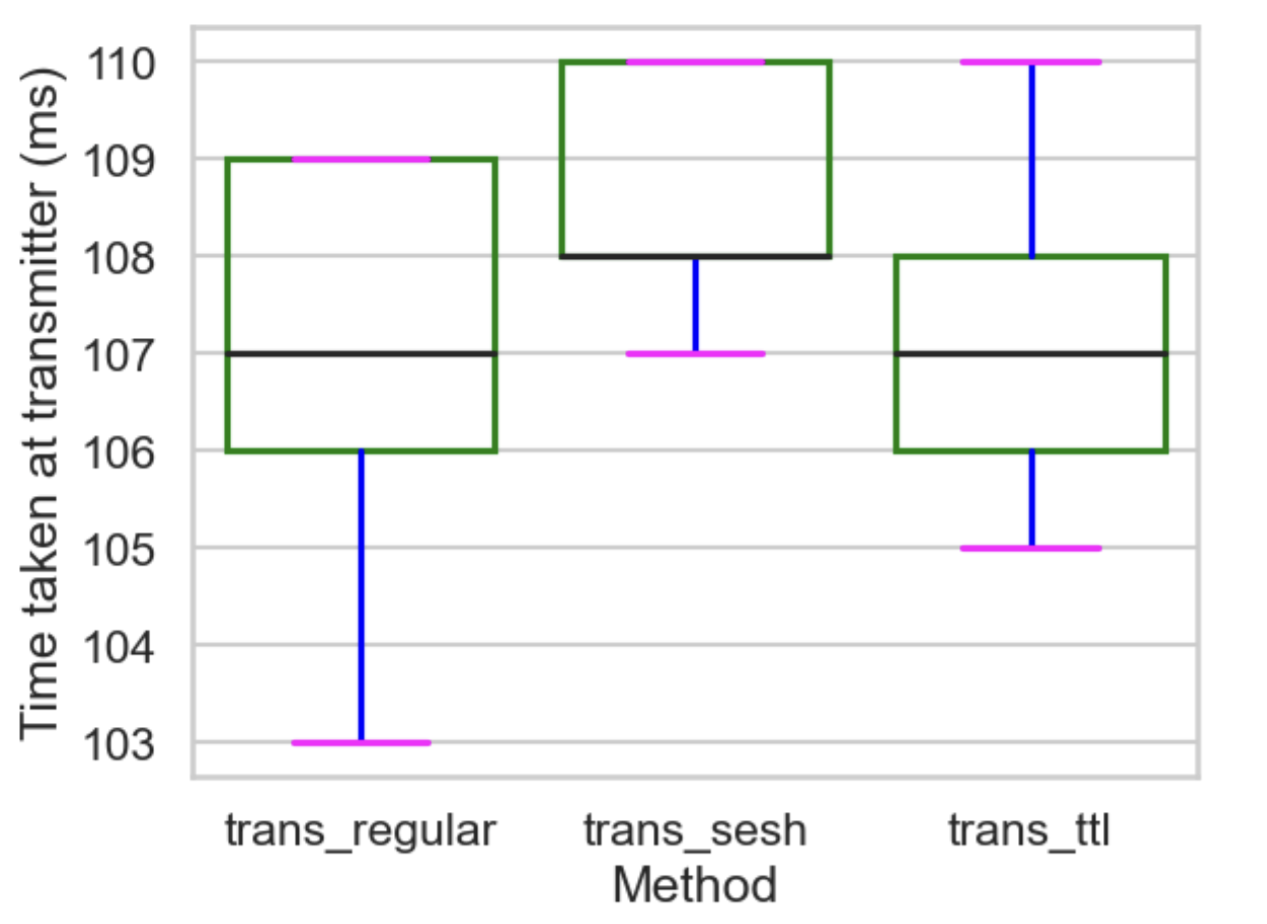
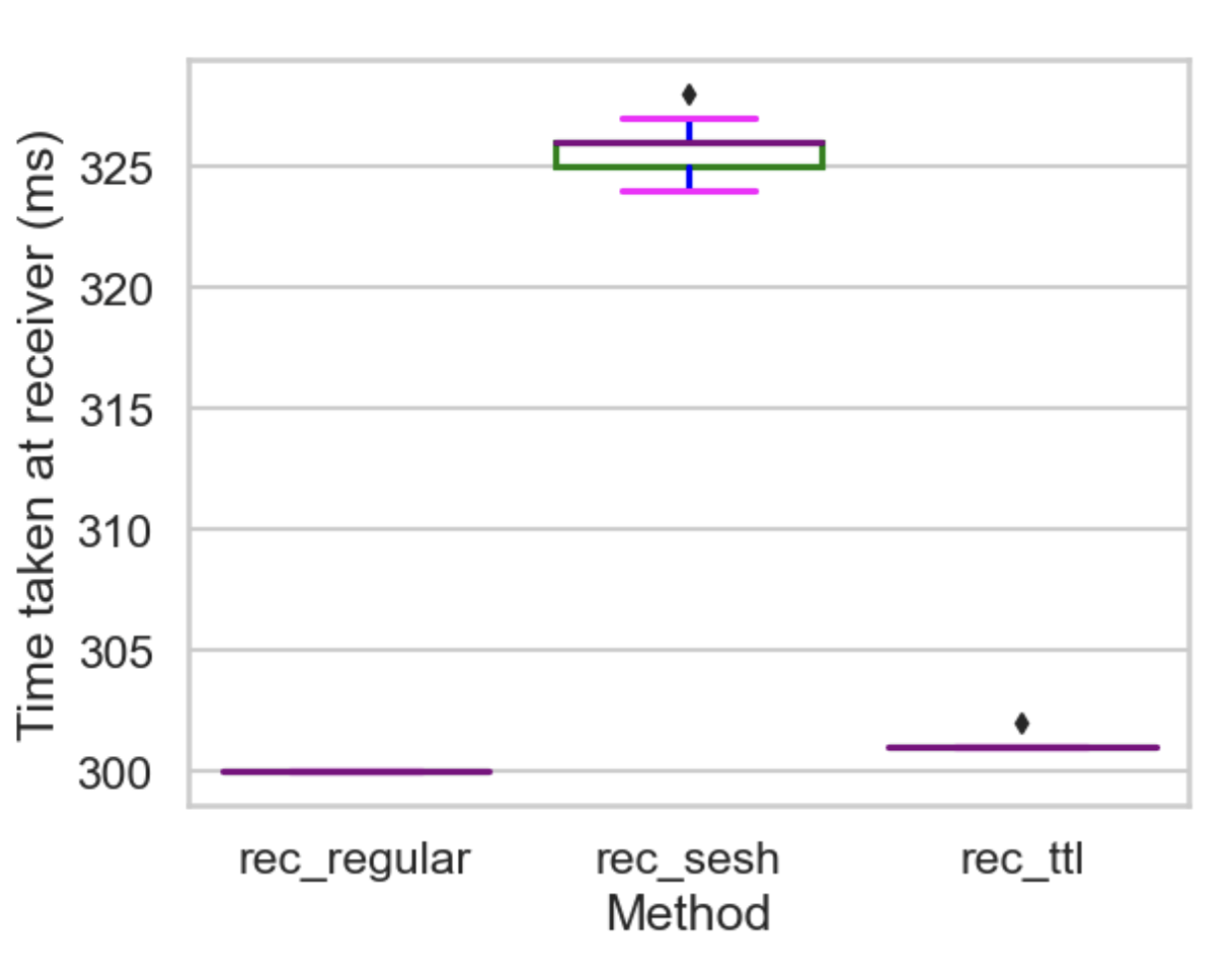
1. Evaluation

For evaluating our 2 security methods against insecure IR communication, we conducted 20 observations for the 3 communication strategies. The experiments were conducted in Line of Sight conditions indoors, in presence of other IR devices. The table below shows the time measurements (in milliseconds) for each of the communication strategies:

|  | **trans\_regular** | **rec\_regular** | **trans\_sesh** | **rec\_sesh** | **trans\_ttl** | **rec\_ttl** |
| --- | --- | --- | --- | --- | --- | --- |
| **0** | 107 | 300 | 108 | 326 | 108 | 301 |
| **1** | 106 | 300 | 108 | 325 | 107 | 302 |
| **2** | 108 | 300 | 109 | 325 | 107 | 301 |
| **3** | 107 | 300 | 107 | 324 | 108 | 301 |
| **4** | 109 | 300 | 108 | 324 | 110 | 301 |
| **5** | 107 | 300 | 107 | 324 | 106 | 301 |
| **6** | 107 | 300 | 107 | 324 | 106 | 301 |
| **7** | 109 | 300 | 108 | 325 | 108 | 301 |
| **8** | 103 | 300 | 108 | 326 | 108 | 301 |
| **9** | 105 | 300 | 108 | 324 | 109 | 301 |
| **10** | 106 | 300 | 108 | 327 | 106 | 301 |
| **11** | 104 | 300 | 110 | 327 | 109 | 301 |
| **12** | 108 | 300 | 110 | 326 | 109 | 301 |
| **13** | 106 | 300 | 110 | 328 | 106 | 301 |
| **14** | 107 | 300 | 110 | 326 | 106 | 301 |
| **15** | 107 | 300 | 110 | 326 | 105 | 301 |
| **16** | 108 | 300 | 108 | 327 | 106 | 301 |
| **17** | 109 | 300 | 109 | 327 | 107 | 301 |
| **18** | 109 | 300 | 109 | 326 | 107 | 301 |
| **19** | 109 | 300 | 110 | 326 | 107 | 301 |
| **20** | 109 | 300 | 110 | 326 | 107 | 301 |

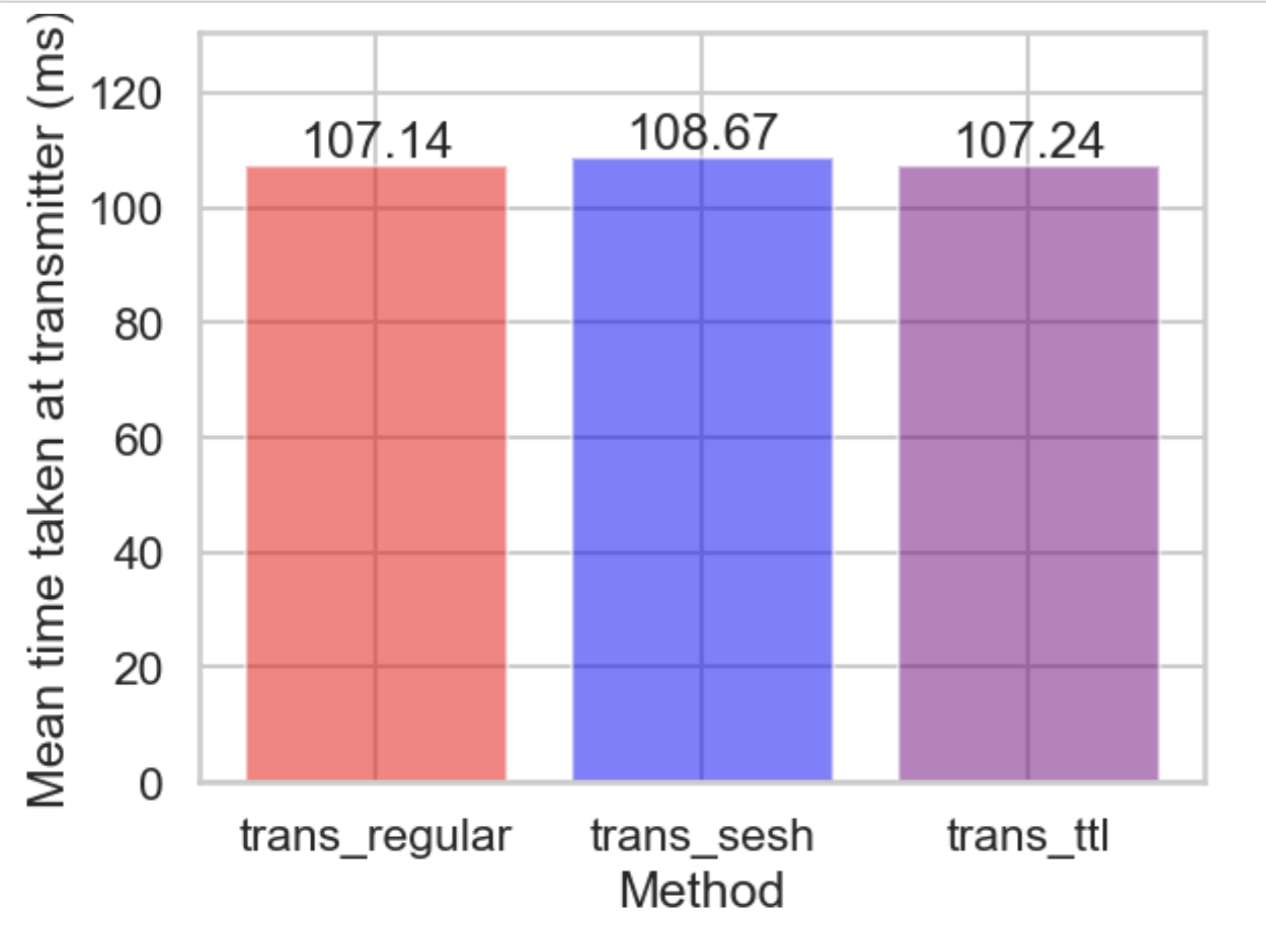
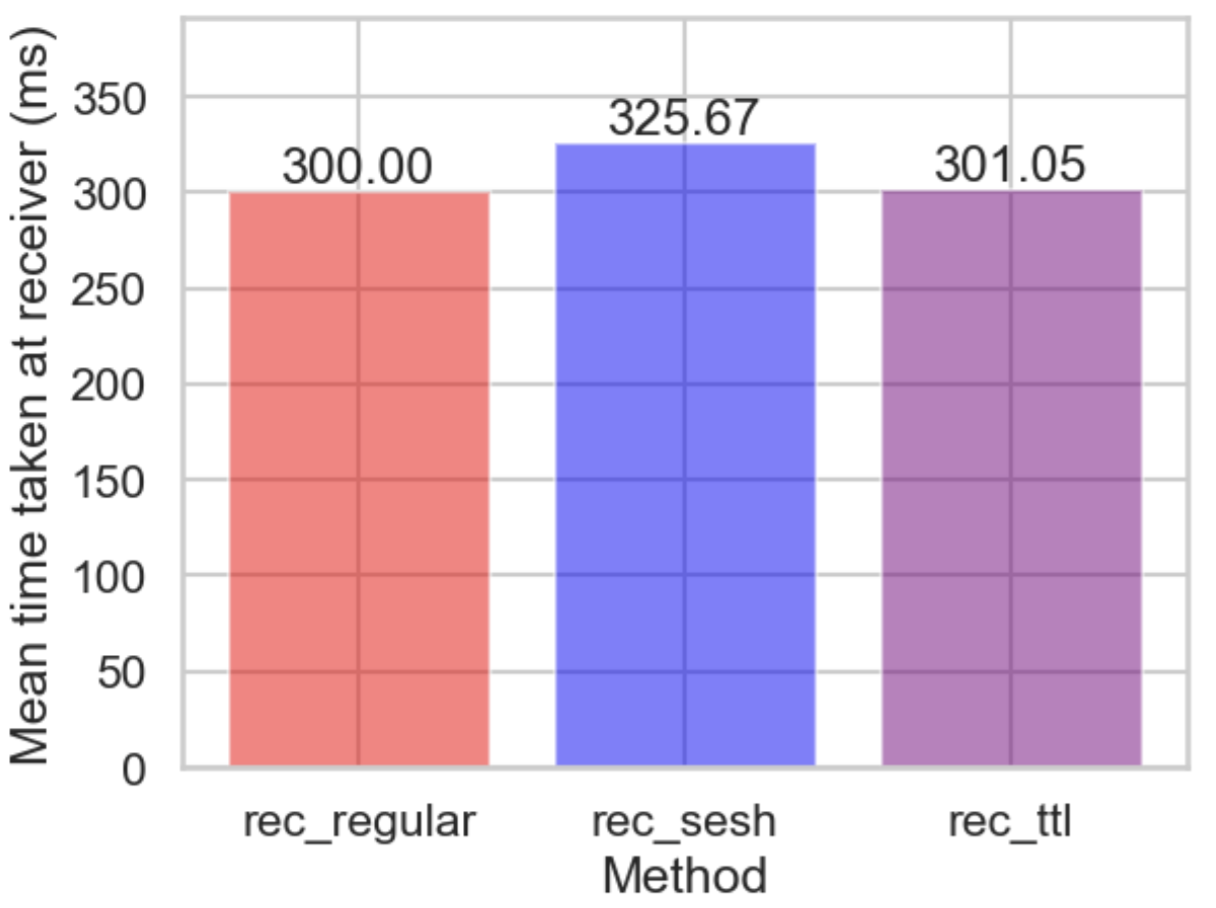
The transmitter time taken for the session key method doesn’t include the time taken for session key exchange. The session key method involves a one-time setup cost to establish the key exchange, but subsequent communications using the key are much faster and more secure. In contrast, the regular method has no setup cost, but each communication is insecure, and the TTL method involves the overhead of repeatedly updating the TTL values. Therefore, when considering the total time taken for multiple communications, the session key method is likely to be the most efficient in terms of both speed and security.

We used a boxplot to observe the median values and interquartile range for the time taken at transmission and reception using 3 methods. They showed that the distribution of total communication time was relatively consistent across the three methods, with no significant outliers or deviations.



The average transmission time for the "TTL" method was found to be the fastest, with an average of 107.24 ms, while the "Regular" and "Session Key" methods had an average transmission time of around 107.14 and 108.67 ms respectively.

The "Regular" method had the lowest average reception time of around 300ms, while the "Session key" and "TTL" methods had a slightly higher average reception time of around 325 and 301.05 ms respectively. With respect to security overheads, the time taken at the transmitter by the session key method is naturally higher consistently than that by the threshold method since it involves the generation of the random session key per session. The threshold method has no such computation on the transmitter side and just involves adding a timestamp. The receiver side can be seen to have “Session key” having the highest mean receiver processing time taken since it involves decrypting the key for the first packet in session and consequently checking the counter for every other packet, the “TTL” method just checks for the incoming packet to have a timestamp in a given time frame and hence doesn’t involve any complexity other than the fact that it requires high synchronization to work.

.

1. Conclusion and Future Work

Our goal, when we started the project, was to understand IR communication better, perform replay attacks on common IR devices such as TVs, string lights, bells, etc and finally build a security layer that can stop these replay attacks. We believe we were able to successfully achieve our goals.

We presented the introduction and background needed to better understand IR communication. We have also explained how the popular IR protocol - NEC works and how can we make use of the extended NEC to add security data such as keys, counters, and TTL in the packet along with the data that can then be used by the receiver to validate the packet received. We have also performed a detailed evaluation of both the session keys and TTL approach to IR communication and observed that the TTL approach took less processing time at the receiver. The strengths and limitations of each approach are also listed in the methodology section of the paper.

Future work for this project can include modifying existing devices to implement these security mechanisms and checking the feasibility. Currently, these methods are implemented and demonstrated using Arduino. There can also be more work done on the synchronization for the TTL method. We are currently assuming that the transmitter and receiver are synchronized and then implementing the TTL approach to discard packets that have expired TTL. Jamming can also be an area of further review to see if IR signals can use friendly jamming in order to hide the receiver from receiving any other signal except the valid transmitter’s signal.

1. References

References for the project include many sources such as papers, youtube videos, GitHub repositories and more.

* [Eavesdropping Vulnerability and Countermeasure in Infrared Communication for IoT Devices by Minchul Kim and Taeweon Suh](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8706134/)
* [Performing replay attacks using Arduino](https://www.youtube.com/watch?v=JL821Ng-6vA)
* IR remote library for Arduino - <https://github.com/Arduino-IRremote/Arduino-IRremote>
* Timing information library for Arduino - <https://github.com/PaulStoffregen/Time>
* NEC protocol information -

[NEC Infrared Transmission Protocol | Online Documentation for Altium Products](https://techdocs.altium.com/display/FPGA/NEC+Infrared+Transmission+Protocol)

<https://www.circuitvalley.com/2013/09/nec-protocol-ir-infrared-remote-control.html>

1. Appendix

Source code used in this project can be found in the Github repo here

<https://github.com/namruth/Secure_IR>