Samir Farhat Dominguez - safarhat@bu.edu
Harshit Agrawal - harshit@bu.edu
Gianna Iafrate - giafrate@bu.edu
03/04/2022
EC544: Project Proposal

Secure Wireless Communication over RF for Data Acquisition

I. Introduction

The purpose of this proposal is to establish the semester project vision, direction, resource needs, risk assessment, and milestones. The use of wireless radio links is advantageous when communication is required in areas where fixed bus networks cannot be used. In particular, free RF bands 314MHz, 433MHz, 866MHz, and 915MHz are often utilized by most commonly used home appliance devices such as various types of remote controls, sensors, and smart-home solutions. The data transmission is always one-way, from sensor probe with a transmitter to receiver. Such a data transfer is the free bandwidth, and it is not an issue to acquire and read sent data. We say a communication exchange is secure when two entities are communicating and a third party is not able to listen in. To accomplish this, they need to communicate in a way that is not susceptible to eavesdropping or interception. Possible attacks on such a data transfer are trivial, and data can be easily captured, modified, or resent. It is not possible to verify the origin of the data on the receiver node.

While standard secrecy methods, such as cryptography, protect the contents of the message from being accessed by unauthorized users, covert communication conceals the existence of the communication to prevent unauthorized users to detect the communication. Secure communication includes means by which people can share information with varying degrees of certainty that third parties cannot intercept what was said and must be secure against replay attack.

The main issue we wish to address is the possibility of eavesdropping even though the message is encrypted and signed. Let's say if an eavesdropper (Eve) intercepts the communication and retransmits it anytime in future, the receiver (Bob) will think the message is signed by transmitter (Alice) and authenticated. For that, a possible prevention method is to timestamp all signals.

II. Resources

This project is not a sequel of our previous projects. While certain specifics aren't completely defined yet, namely whether the medium of the network/transport layer will be implemented through RF or traditional networking, e.g TCP packets built and sent through IP, the list of materials will be comprehensive and encompass all resources that could form a part of the final product.

Hardware	**Software**
Raspberry Pi 3B+ A - Ethernet Cable - USB-C Power Supply Cable - Micro HDMI Cable - Other stuff Alice may need Raspberry Pi 3B+ B - Ethernet Cable - USB-C Power Supply Cable - Micro HDMI Cable - Ethernet Cable - Other stuff Bob may need HackRF - Power Supply - ANT500 Antenna BladeRF - Power Supply - Tri-Band Antenna	GNU Radio 3.7+ - grc flowgraph - ZeroMQ Python - Python 3.9 - pip 22.03 - Libraries - Pseudo random function generators C++/C - gcc 9.3.0 - GNU Make 4.2.1 - Libraries - Pseudo random function generators Github - Code repository and version control Middleware: Twitter API - Credentials set - Account

III. Risk Management

We have identified 6 major areas of potential risk in our design. They are briefly listed below, along with the planned mitigation strategy for each

Risk: Learning curve for eliminating noise from the transmission and reception may be steep
Mitigation: Dedicate one team member to learn and test simple transmission and reception techniques from week one. Leave our implementation flexible enough to allow changes once the transmission abilities are identified.
Worst case scenario: switch to ZeroMQ Server using GNURadio implementing "Radio over IP".

Risk: GNU-Radio not performing responsively on the Raspberry Pi.
Mitigation: Transition to running GNU-Radio in a well-resourced Linux environment machine on one of our team member's laptops.

Risk: Connecting all components with the initial setup on Raspberry Pi not working as expected
Mitigation: Plan to test this connectivity by week three so any changes can be addressed quickly and changes can be made easily.

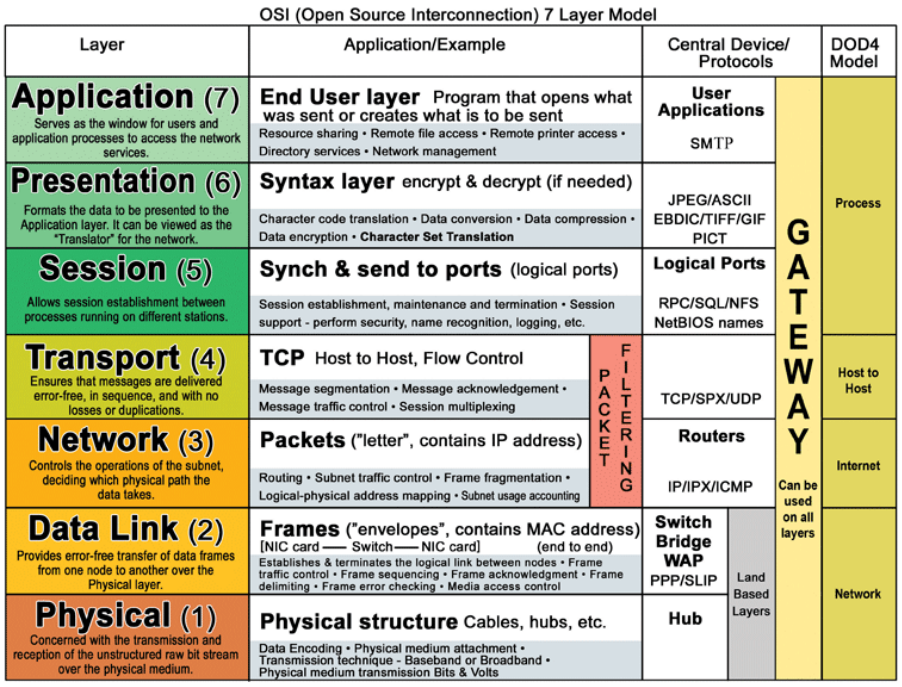
Risk: The learning curve for implementing secure cryptographic encryption with suitable signature and timestamping may be steep and we may underestimate the amount of work required to implement it.
Mitigation: Dedicate one team member to learn and investigate the Cryptographic algorithm and secure MAC from week one. Leaves our implementation flexible enough to switch to a different encryption algorithm if it proves more complicated than expected.

Risk: Hardware failure for either the Pi's, BladeRF, HackRF and otherwise.
Mitigation: Specifically, in regards to the Pi's failing, we know that there are several services that allow it to interact with remote or virtual Raspberry Pi's. Depending on our final transport decision, the RF board breaking down may be somewhat catastrophic, as the boards are expensive and hard to come by with the current shortage. However, depending on the time constraint we may be able to redirect the transport of encoded RF through traditional networking.

Risk: Testing and troubleshooting phase takes longer than anticipated, and project cannot come to fruition.
Mitigation: Ensure the team completes the weekly tasks through check-ins and updating the schedule realistically when applicable. Try to connect all the modules together early on so that there is enough time leftover to test and troubleshoot.

IV. Technical Approach

The project can be divided into 3 major modules: Application layer where user interacts, Cryptographic layer where encryption and signing take place, Communication layer where the ciphertext is modulated and transmitted via RF. These modules can come together to form a pseudo OSI-7-layer model for networking.



OSI (Open Source Interconnection) 7 Layer Model

Layer	Application/Example		Central Device/ Protocols			DOD4 Model
Application (7) Serves as the window for users and application processes to access the network services.	**End User layer** Program that opens what was sent or creates what is to be sent Resource sharing • Remote file access • Remote printer access • Directory services • Network management		**User Applications** SMTP		**GATEWAY** Can be used on all layers	Process
Presentation (6) Formats the data to be presented to the Application layer. It can be viewed as the "Translator" for the network.	**Syntax layer** encrypt & decrypt (if needed) Character code translation • Data conversion • Data compression • Data encryption • **Character Set Translation**		JPEG/ASCII EBDIC/TIFF/GIF PICT			
Session (5) Allows session establishment between processes running on different stations.	**Synch & send to ports** (logical ports) Session establishment, maintenance and termination • Session support - perform security, name recognition, logging, etc.		**Logical Ports** RPC/SQL/NFS NetBIOS names			
Transport (4) Ensures that messages are delivered error-free, in sequence, and with no losses or duplications.	**TCP** Host to Host, Flow Control Message segmentation • Message acknowledgement • Message traffic control • Session multiplexing	PACKET FILTERING	TCP/SPX/UDP			Host to Host
Network (3) Controls the operations of the subnet, deciding which physical path the data takes.	**Packets** ("letter", contains IP address) Routing • Subnet traffic control • Frame fragmentation • Logical-physical address mapping • Subnet usage accounting		**Routers** IP/IPX/ICMP			Internet
Data Link (2) Provides error-free transfer of data frames from one node to another over the Physical layer.	**Frames** ("envelopes", contains MAC address) [NIC card —— Switch —— NIC card] (end to end) Establishes & terminates the logical link between nodes • Frame traffic control • Frame sequencing • Frame acknowledgment • Frame delimiting • Frame error checking • Media access control		**Switch Bridge WAP** PPP/SLIP	Land Based Layers		Network
Physical (1) Concerned with the transmission and reception of the unstructured raw bit stream over the physical medium.	**Physical structure** Cables, hubs, etc. Data Encoding • Physical medium attachment • Transmission technique - Baseband or Broadband • Physical medium transmission Bits & Volts		**Hub**			

Application Layer: User Interface
The application layer is key to ensuring the user interface is working properly and sending the correct message. The user interface will utilize both Python and the terminal window to pull data from an online API source, such as Twitter. Python script will be written to pull data from Twitter and retrieve texts of information. When running the code, the user will need to authenticate oneself through a Twitter API key and credentials to ensure the correct person is accessing this information. From there, the user will pull specific information desired to be sent, and the information will be communicated to the cryptography layer.

Presentation and Session Layer: Cryptography
Cryptography is the science that creates strategies for utilizing complex mathematical transformations to transmit data through conveyance channels in a frame that no one but authorized persons can get.
Cryptography is another main building block of our work. At every stage of development, we will be considering vulnerabilities, such as caching IP mappings and monitoring Ping requests. Taking the cryptographic component, our priorities are to achieve the following between the Raspberry Pi's.

1. Authentication: This entails that at the very least the sender and receiver have the means to establish that they are engaging with the entity they think they are communicating with. Our work will

specifically target message signatures to establish authenticity and will be further elaborated on later.

2. Confidentiality in the presence of a 3rd party: This goal entails ensuring that under a set of conditions, principally not having private key access, a third-party listener with oracle access is unable to establish any part of the plaintext from its corresponding ciphertext.
3. Integrity in the presence of a 3rd party: This goal relates to establishing a protocol where a user is unable to intercept or manipulate the transmission of data between Alice and Bob.

There are three generalized types of cryptographic algorithms; hash functions, symmetric key encryption, and asymmetric key encryption. While asymmetric key encryption can be useful, in the spirit of embedded systems, IoT, and networking, we would like to focus on reducing overhead and computational complexity. Asymmetric key cryptography incurs a large overhead due to the inevitability of having to have higher complexity and key-lengths for public keys. This means that generation, encryption, and decryption algorithms occur more frequently and are far slower to execute.

On the other hand, symmetric encryption and hash functions can be combined to achieve the 3 goals we have outlined. Our plan is to use the Diffie-Hellman Key Exchange Protocol in combination with MACs and AES. Alice and Bob will establish a prime multiplicative group p and base g. They will then perform the necessary generation and message sends to produce a key as per the below diagram. Moreover, pseudo-random functions that mimic uniformly random functions will be used to generate all these variables and keys. This will run thrice in order to generate a signature key, an encryption/decryption key, and a final block key. It is already proven that Diffie-Hellman is computationally infeasible to compromise if performed correctly, even if a malicious party sees every single transmitted message over the public channel.

$$\textcircled{1} = (g^a \bmod p)^b \bmod p$$

$$\textcircled{2} = A^b \bmod p$$

$$\textcircled{3} = (g^b \bmod p)^a \bmod p$$

$$\textcircled{4} = B^a \bmod p$$

$$\textcircled{5} = (g^a)^b \bmod p$$

$$\textcircled{6} = (g^b)^a \bmod p$$

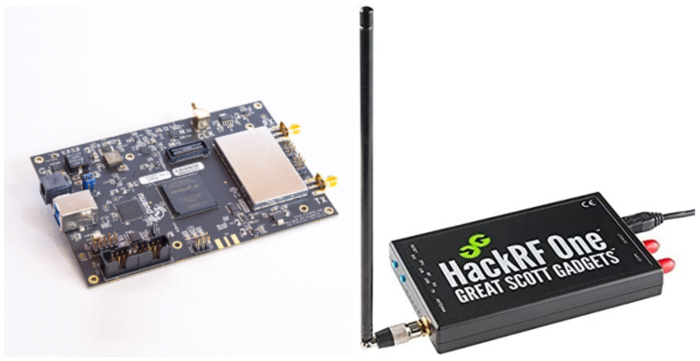
Having established our keys over an insecure channel in a safe manner, we can now discuss the encryption and decryption schemes. We combine AES and MAC to produce a modified CBC-MAC for variable-length message encryption/decryption. This modification involves including length-prepending and last-block encryption to ensure variable-length message indistinguishability under CPA. CBC-MAC involves encrypting a message with the AES block cipher to create contiguous blocks such that each one depends on the encryption of the former. The variable-length consideration comes into play at the final block, where our final block, already signed with the MAC key, is encrypted with the 3rd final-block key.

Communication Layer: RF Signals

The secure communication layer is implemented into the RF communication system. Before the implementation itself, the requirements for the transmitter and receiver sides are as follows:

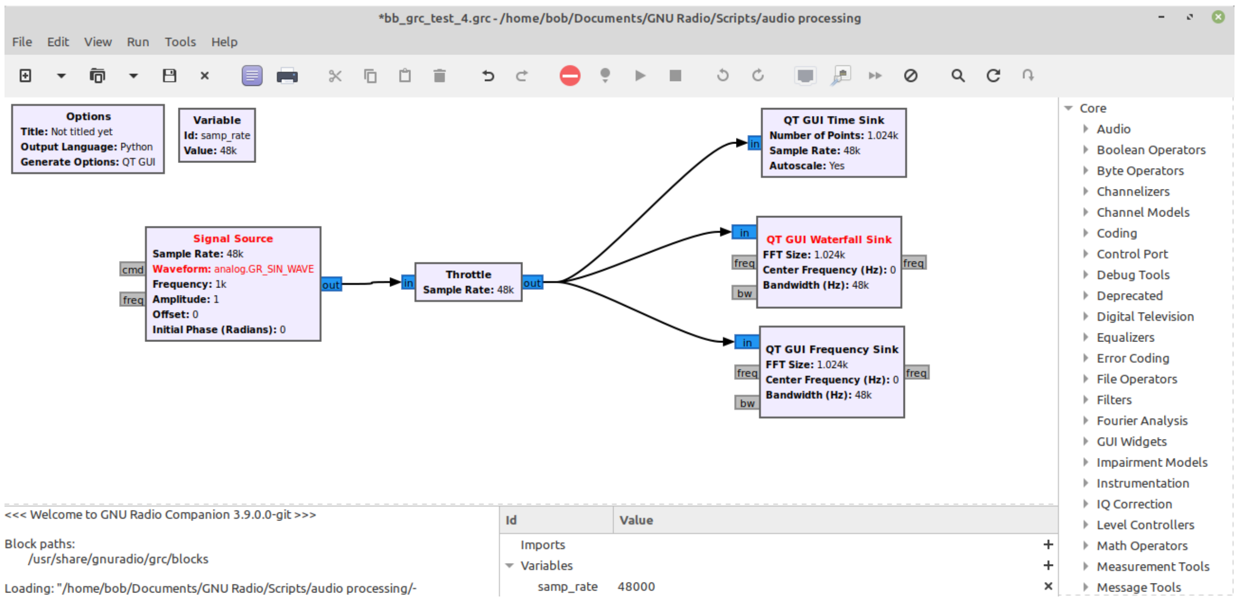
1. Transmitting side: When sending data, ciphertext is different from the previous transmission for the same message to secure against replay attack.
2. Receiving side: Based on the known format of the received raw signal, it will be able to demodulate and decrypt with the correct key.

For transmission, we will be using BladeRF 2.0 Micro, that is a wideband MIMO transceiver that covers 47MHz to 6GHz with up to 56 Mhz of bandwidth. It has 2x2 MIMO, Host connection is via USB 3.0, GNU Radio integration is provided via gr-osmosdr while another device we are going to use as receiver is HackRF One which is a half-duplex transceiver peripheral, operating from 1 MHz to 6 GHz, with a Hi-Speed USB 2.0 connection. It is bus-powered, portable, and has a maximum quadrature sample rate of 20 Msps. GNU Radio integration is provided via gr-osmosdr.



The fundamental issue with using SDR is that there's no uniform standard for transmit frequency, modulation, and data protection, although some applications use signal encryption but aren't always getting a lot of interest.

We'll be using GNU-radio, as it's open-source software development toolkit and provides signal processing blocks to implement software radios. To process digital signals there are several stages (filtering, correction, analysis, detection...) as processing blocks, which can be connected using simple flow-indicating arrows to different blocks in GNUradio, which is C++ and Python code in backen.



*bb_grc_test_4.grc - /home/bob/Documents/GNU Radio/Scripts/audio processing

Options
Title: Not titled yet
Output Language: Python
Generate Options: QT GUI

Variable
Id: samp_rate
Value: 48k

Signal Source
Sample Rate: 48k
Waveform: analog.GR_SIN_WAVE
Frequency: 1k
Amplitude: 1
Offset: 0
Initial Phase (Radians): 0

Throttle
Sample Rate: 48k

QT GUI Time Sink
Number of Points: 1.024k
Sample Rate: 48k
Autoscale: Yes

QT GUI Waterfall Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 48k

QT GUI Frequency Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 48k

```
<<< Welcome to GNU Radio Companion 3.9.0.0-git >>>

Block paths:
    /usr/share/gnuradio/grc/blocks

Loading: "/home/bob/Documents/GNU Radio/Scripts/audio processing/-
```

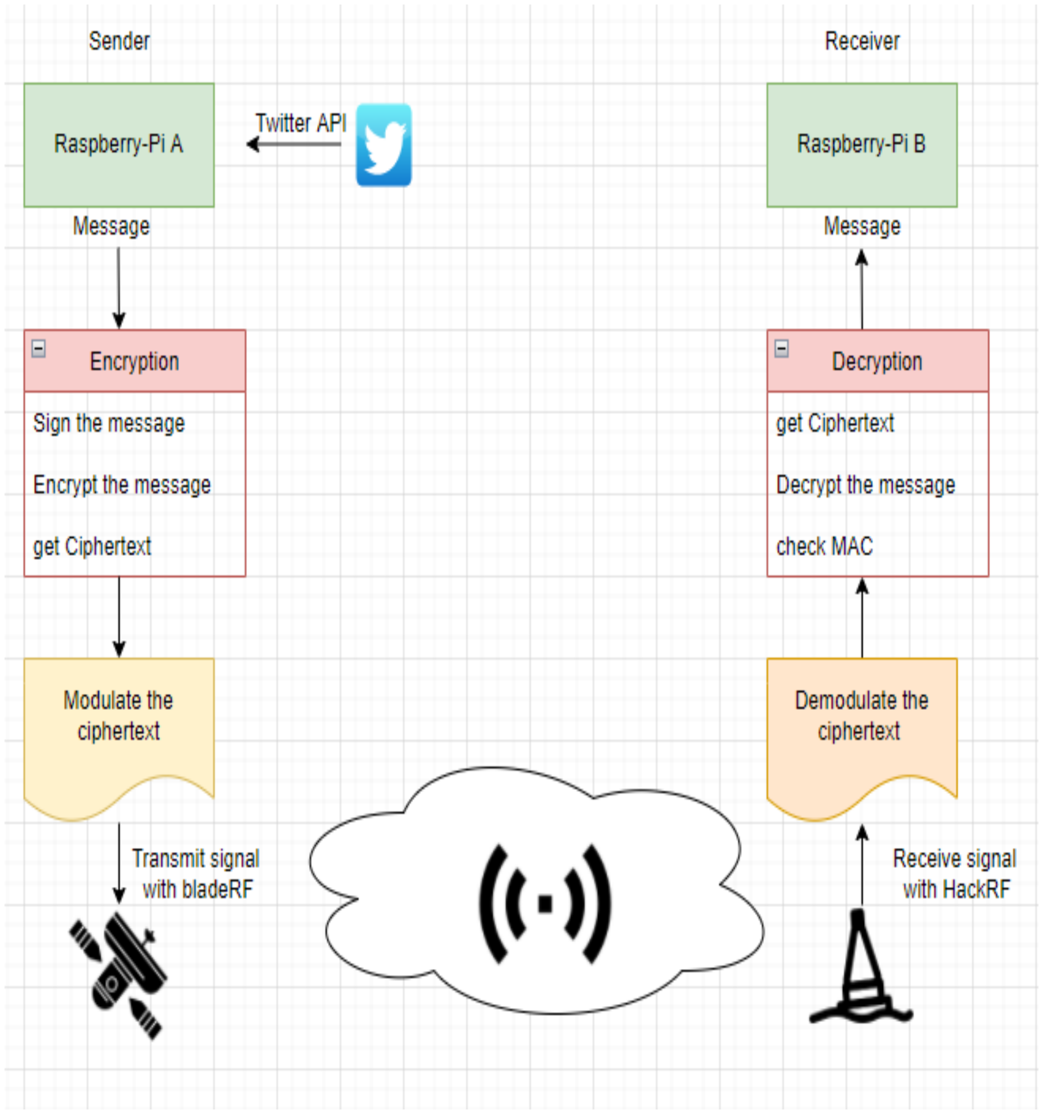
Id	Value
Imports	
Variables	
samp_rate	48000

If the above method fails to transmit data with efficiency or there’s any hardware failures (risk mitigation), we’ll use RF signals over IP, and not over the air with antennas. The use of IP-based transport will eliminate the complexity of deploying actual radios and transmitting RF over the air. In particular, the ‘raw’ RF signals will be tunneled via ZeroMQ and can be received with traditional RF tools, as if they’re coming from an actual SDR. GNU-Radio supports ZeroMQ source blocks natively.

Responsibilities

As for responsibilities, Samir has more experience with cryptography and networking, therefore designing the system for secure encryption. Gianna is familiar with python and front-end development, and will be leading the user interface and integrating modules. Harshit is more oriented towards wireless communication, and he will be leading the Physical layer part. We’ll make sure everyone participates in everything so everyone learns about everything. If our project turns out to be easy, we will add more parts to it.

System Diagram



V. Milestones

Date	**Harshit**	**Samir**	**Gianna**
3/14	- Agree on project vision and direction - Research and decide on appropriate components: ○ GNUradio, ZeroMQ, Cryptography, DSP, Twitter API - Draft proposal & Distribute tasks		
3/21	- Research encoding techniques - Install GNUradio and verify working	- Look into AES specification - Design cryptographic model	- Generate Twitter API key and authentication
3/28	- Test transmit and receive with sdr - Run basic encoding techniques and check for packet loss while transmission	- Run basic code on Raspberry Pi - Write key generation and exchange scripts	- Develop code to pull Twitter information - Work with user interface to draw on specific information to retrieve
4/4	- Working towards converting ciphertext to modulated carrier signal wave - Implement gnuradio flow graph	- Automate key generation/exchange protocol from TLS suite - Integrate encryption/decryption algorithm and run on Pi	- Run tests on sending the specific tweets to Samir's module - Work on user interface for receiving the message
4/11	- Connect the system to check integrity and response rate	- Automate full model on every Alice and Bob interaction - Perform sanity check by interconnecting Gianna's and Harshit's modules	- Connect with Samir's module and test connectivity of encrypting and decrypting the tweets
4/18	- Run several more general tests to make sure the entire system is robust and intact		
4/25	- A detailed manual shall be uploaded to Github, with all the procedures and steps to install our software. We will prepare for the final presentation (if any) and write the final report. - This part is not as hardcore as before, but time-consuming. Therefore the Last week was planned as such.		

References

1. Dudak, Juraj, Gabriel Gaspar, and Pavol Tanuska. "Implementation of Secure Communication via the RF Module for Data Acquisition." Journal of Sensors 2019 (2019).
2. Babak Kia. "CE In a Connected World Project Proposal Grading Criteria." Blackboard: EC544 (2022).
3. Toliupa, Serhii, et al. "RF Signals Encryption with AES in WDID." IT&I Workshops. 2020. (2020)
4. Cameron MacDonald Surman, Hairuo Sun, Yi-Wei Chen. “Satellite Project Final Report.” EC544 Networking the Physical World (2020)
5. Zimmermann, Hubert. "OSI reference model-the ISO model of architecture for open systems interconnection." IEEE Transactions on communications 28.4 (1980): 425-432.
6. Agrawal, Monika, and Pradeep Mishra. "A comparative survey on symmetric key encryption techniques." International Journal on Computer Science and Engineering 4.5 (2012): 877.
7. Yassein, Muneer Bani, et al. "Comprehensive study of symmetric key and asymmetric key encryption algorithms." 2017 international conference on engineering and technology (ICET). IEEE, 2017.
8. Blumenthal, Uri, Fabio Maino, and Keith McCloghrie. "The advanced encryption standard (AES) cipher algorithm in the SNMP user-based security model." Internet proposed standard RFC 3826 (2004).
9. Petrank, Erez, and Charles Rackoff. "CBC MAC for real-time data sources." Journal of Cryptology 13.3 (2000): 315-338.